GALGOTIAS UNIVERSITY

BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING
GREATER NOIDA, UTTAR PRADESH

2024 – 2025

VI th Sem III Year

REPORT FILE :

Automating Document Summarization For Information Overload

Submitted by :-

21. Garv Rastogi (22SCSE1010043)

22. Kunal Arora (22SCSE1180152)

23. Kundan Kumar (22SCSE1012827)
24. Md Shahnawaz Alam (22SCSE1180133)

Natural Language Processing and Applications (R1UC635B)

Instructor's :-

Dr. Umesh Lilhore
Dr. Mandeep Kumar

Date :- 22-05-2025

Project Report: NLP Case Study

Automating Document Summarization For Information Overload

Table of Contents

1. Introduction

1.1 Brief Overview of the NLP Problem or Case Study

In today's digital era, the abundance of textual information poses a significant challenge for users seeking relevant and concise content. Educational platforms like Coursera offer thousands of online courses, each accompanied by detailed descriptions that can be overwhelming to browse manually. The field of Natural Language Processing (NLP) addresses such challenges through automated text summarization—a technique designed to reduce large volumes of text to their essential points. This project explores the application of **extractive summarization** methods, specifically using TF-IDF (Term Frequency–Inverse Document Frequency), to automatically generate summaries from course descriptions on Coursera. The goal is to help users make faster, more informed decisions by condensing complex course descriptions into short, informative summaries.

1.2 Motivation and Objective of the Project

The core motivation of this project is to alleviate information overload for learners navigating vast catalogs of online courses. With numerous overlapping or verbose course descriptions, it becomes difficult for users to identify the most relevant offerings. By automating the summarization process, we aim to:

- Provide concise and informative overviews of online courses,
- Implement a lightweight, explainable summarization system using classical NLP methods,
- Evaluate the effectiveness of extractive techniques in capturing the key elements of educational content.

This work lays the foundation for more scalable content summarization systems applicable not just to educational platforms, but to other domains overloaded with textual data (e.g., news, legal, or healthcare).

1.3 Scope and Limitations

Scope:

- Focuses exclusively on English-language Coursera course descriptions.
- Utilizes extractive summarization, selecting the most relevant sentences based on TF-IDF scores.
- Evaluation is based on overlap with predefined keywords or reference summaries.

Limitations:

- The summarizer does not paraphrase or generate new content (i.e., no abstractive capability).
- Performance depends heavily on the quality and structure of the original course descriptions.
- The method may not capture nuanced or context-dependent insights, such as comparisons or temporal information.
- Sarcasm, domain-specific language, or implicit references are not effectively handled by the current model.

2. Background / Literature Review

2.1 Summary of Relevant NLP Concepts or Techniques

Text summarization is a core area within Natural Language Processing (NLP) focused on producing concise and coherent summaries of longer text documents. It is typically divided into two main approaches:

- **Extractive Summarization**: Involves selecting and concatenating key sentences or phrases from the original text without rephrasing. Techniques like TF-IDF (Term Frequency–Inverse Document Frequency) and graph-based algorithms such as TextRank are commonly used.
- **Abstractive Summarization**: Involves generating entirely new sentences that capture the essence of the input text, often using advanced deep learning models like BERT, T5, or BART.

This project leverages **extractive summarization** due to its simplicity, interpretability, and low computational requirements. The central idea is to assign importance scores to sentences using **TF-IDF**, a statistical measure that reflects how important a word is to a sentence in a collection of documents. Sentences with higher aggregated TF-IDF scores are presumed to be more informative and are selected for the summary.

Key NLP techniques used include:

- **Tokenization**: Breaking text into sentences and words.
- **Stopword Removal**: Eliminating commonly used words that add little semantic value (e.g., “the”, “and”).
- **TF-IDF Vectorization**: Converting text into numerical vectors to capture term importance.
- **Sentence Scoring and Ranking**: Aggregating TF-IDF weights to determine which sentences best represent the original content.

These steps are implemented using Python libraries such as NLTK (Natural Language Toolkit) and Scikit-learn, which offer robust and efficient tools for text processing and machine learning.

2.2 Related Work or Similar Studies

Research in automatic summarization has evolved significantly over the past two decades. Early work by **Luhn (1958)** introduced statistical methods for identifying important sentences. Later, **TextRank (Mihalcea & Tarau, 2004)** applied graph-based ranking models inspired by PageRank to summarization tasks.

In the educational domain, summarization has been applied to textbooks, lecture notes, and MOOCs (Massive Open Online Courses). Projects such as **Siddharth's Coursera Course Dataset (2023)** have made large-scale educational datasets available for NLP experimentation. More recently, researchers have applied transformer-based models (e.g., BERTSUM, PEGASUS) to perform abstractive summarization of academic texts and articles.

However, classical methods like TF-IDF remain popular for their ease of implementation and transparency. For projects with limited data and compute resources, these approaches provide a strong baseline, as demonstrated by numerous NLP studies in extractive summarization.

---

3. Dataset Description

3.1 Source of Data

The dataset used in this project consists of Coursera course metadata, specifically focusing on two key fields:

- **course_description**: A textual overview of the course content, objectives, and topics.
- **course_summary**: A set of key points often derived from the "What you'll learn" section on the course page.

The dataset is adapted from publicly available sources such as the *Coursera Course Dataset* by Siddharth (2023), which provides course information scraped from the Coursera website. The data was accessed as a CSV file and manually curated to ensure quality and completeness.

3.2 Size and Format

The dataset contains approximately **900 records**, each corresponding to a unique course. Each record includes:

- course_title (string)

- course_description (string): usually a paragraph of 100–300 words
- course_summary (string or list): 2–5 short bullet points
- Additional metadata (e.g., institution, course level) not used in this project

The data is stored in **CSV format**, making it compatible with Python data manipulation libraries like pandas.

Example entry (simplified):

csv

CopyEdit

course_title,course_description,course_summary

"Intro to Data Science","This course covers the basics of Python, data analysis, and visualization...", "['Understand Python basics', 'Perform data analysis', 'Use matplotlib']"

3.3 Preprocessing Steps Applied

To prepare the data for summarization and evaluation, the following preprocessing steps were applied:

1. **Missing Value Handling**: Rows with missing or empty course_description or course_summary fields were excluded.
2. **Text Cleaning**:
    - Converted all text to lowercase for uniformity.
    - Removed punctuation, numeric digits, and non-ASCII characters.
    - Eliminated excessive whitespace and special characters.
3. **Stopword Removal**: English stopwords (e.g., "the", "is", "and") were removed using NLTK's built-in list.
4. **Sentence Tokenization**: Descriptions were split into individual sentences using NLTK's sent_tokenize method to enable sentence-level scoring.
5. **Reference Summary Parsing**: The course_summary field (when stored as a list or bullet string) was converted into a set of keywords for comparison during evaluation.

These steps ensured that the text was clean, consistent, and suitable for further vectorization and analysis using TF-IDF methods.

4. Methodology

4.1 Description of Approach

This project employs a **statistical extractive summarization** approach to automatically generate concise summaries from Coursera course descriptions. The core idea is to treat each sentence in a course description as a candidate for inclusion in the summary and to rank these sentences based on their importance. Importance is computed using **TF-IDF (Term Frequency–Inverse Document Frequency)**, which measures the relevance of words within the local context of the description.

The methodology follows these key steps:

1. **Text Preprocessing**: Clean and normalize the input text.
2. **Sentence Tokenization**: Split each course description into individual sentences.
3. **TF-IDF Vectorization**: Convert each sentence into a vector based on word importance.
4. **Sentence Scoring**: Calculate the aggregate TF-IDF score for each sentence.
5. **Sentence Selection**: Choose the top N highest-scoring sentences to form the summary.

This rule-based, statistical approach is chosen for its simplicity, interpretability, and effectiveness on moderate-sized datasets without requiring model training.

4.2 Tools and Libraries Used

The summarization system was implemented using Python 3 with the following libraries:

- **NLTK (Natural Language Toolkit)**: For sentence tokenization and stopword filtering.
- **Pandas**: For loading and manipulating tabular data from CSV files.
- **Scikit-learn**:
    - TfidfVectorizer: For computing TF-IDF matrices.
    - metrics: For precision, recall, and F1-score evaluation.
- **Seaborn & Matplotlib**: For data visualization, including TF-IDF heatmaps.

These libraries were chosen due to their maturity, ease of use, and widespread adoption in the NLP and data science communities.

4.3 Explanation of Key Algorithms or Models Implemented

TF-IDF (Term Frequency–Inverse Document Frequency)

TF-IDF is a statistical measure used to evaluate the importance of a word in a sentence relative to a collection of sentences. It is calculated as:

arduino

CopyEdit

TF-IDF(t, d) = TF(t, d) * log(N / DF(t))

Where:

- TF(t, d) is the frequency of term *t* in sentence *d*,
- DF(t) is the number of sentences containing term *t*,
- N is the total number of sentences.

This project uses TF-IDF to construct a sentence-feature matrix where each row represents a sentence and each column a term. The score of a sentence is the sum of its TF-IDF weights across all terms:

python

CopyEdit

Sentence Score = sum(TF-IDF scores of all terms in the sentence)

Sentence Ranking and Summary Generation

Once all sentences are scored, they are ranked in descending order of importance. The top N sentences are selected and reassembled in their original order to maintain coherence. This ranked extractive summary serves as the final output.

This method ensures that sentences containing the most contextually significant terms are chosen, providing a concise and informative summary without modifying the original text.

5. Implementation

This section provides a detailed look at the core implementation of the extractive summarization system, including key code snippets, explanations of logic, and challenges encountered during development.

5.1 Key Code Snippets and Explanation

Text Cleaning Function :

```
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    text = re.sub(r'\d+', '', text)      # Remove digits
    stop_words = set(stopwords.words('english'))
    return ' '.join([word for word in text.split() if word not in stop_words])
```

Explanation:

- Converts text to lowercase for uniformity.
- Removes punctuation and numeric characters using regular expressions.
- Eliminates stopwords to retain only meaningful words.
- Output is a cleaned string ready for vectorization.

TF-IDF Extractive Summarizer :

```
def extract_summary(text, num_sentences=2):
    sentences = sent_tokenize(str(text))
    if len(sentences) <= num_sentences:
        return text  # Return original if not enough sentences
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(sentences)
    sentence_scores = tfidf_matrix.sum(axis=1).A1  # Sum TF-IDF per sentence
    ranked_indices = np.argsort(sentence_scores)[-num_sentences:]
    ranked_sentences = [sentences[i] for i in sorted(ranked_indices)]
    return ' '.join(ranked_sentences)
```

Explanation:

- Tokenizes the input text into sentences.
- Converts sentences into TF-IDF vectors.
- Scores each sentence based on the sum of TF-IDF values.
- Selects the top num_sentences based on score and reorders them for readability.
- Returns the final summary composed of the most informative sentences.

Evaluation Logic: Keyword Overlap :

```
def has_overlap(gen_summary, ref_summary):
    try:
        if not ref_summary or ref_summary == "[]":
            return 0
        keywords = eval(ref_summary)  # Converts string list to Python list
        return int(any(word.lower() in gen_summary.lower() for word in keywords))
    except:
        return 0
```

Explanation:

- Parses reference summary (stored as a string list) into actual Python list.
- Checks whether any keyword appears in the generated summary (case-insensitive).
- Returns 1 if match found, otherwise 0.
-

Summary Generation and Evaluation Loop :

```
results = []
for num_sentences in range(1, 6):
   df['summary_generated'] = df['course_description'].fillna("").apply(lambda x:
extract_summary(x, num_sentences))
   df['label'] = df.apply(lambda row: has_overlap(row['summary_generated'],
row.get('course_summary', '')), axis=1)
   y_true = df['label']
   y_pred = [1 if len(summary.split()) > 10 else 0 for summary in df['summary_generated']]
   f1_score_val = f1_score(y_true, y_pred)
   results.append((num_sentences, f1_score_val))
```

Explanation:

- Iterates through 1 to 5 sentences for summarization.
- Generates summaries and evaluates against reference keywords.
- Records F1 scores for each summary length to find optimal configuration.

5.2 Challenges Faced

1. Handling Missing or Malformed Data

- Some descriptions or summaries were missing or improperly formatted.
- **Solution:** Used .fillna('') and exception handling (try-except) to avoid crashing.

2. Evaluation Limitations

- Reference summaries were provided as keyword lists rather than full sentences.
- **Solution:** Used keyword overlap as a proxy metric to evaluate informativeness.

3. Hyperparameter Tuning

- Determining the optimal number of summary sentences was non-trivial.
- **Solution:** Empirically tested 1–5 sentence lengths and compared F1 scores.

4. Sentence Tokenization Accuracy

- Irregular punctuation and formatting affected tokenization.
- **Solution:** Relied on NLTK's sent_tokenize, which performs robust sentence segmentation for English.

5.3 Summary of Implementation

The implementation combined classical NLP techniques with a modular design to process, summarize, and evaluate course description texts. TF-IDF provided a transparent and effective mechanism for sentence selection, and the keyword-based evaluation gave a simple yet insightful performance measure. The pipeline was tested for various configurations, leading to the identification of an optimal summary length.

---

6. Results and Evaluation

6.1 Example Summaries

We illustrate the summarization process using a representative example. Consider a hypothetical course with the following input:

Original Description:

"This course is an introduction to machine learning. It covers supervised and unsupervised learning algorithms. Students will learn about regression, classification, clustering, and dimensionality reduction techniques. The course will include practical exercises using

Python. At the end of the course, students will complete a capstone project applying learned techniques."

Reference Summary (from “What you’ll learn”):

- Understand supervised vs. unsupervised learning
- Learn regression, classification, and clustering techniques
- Complete a hands-on Python capstone project

Extractive Summary (N = 2):

"Students will learn about regression, classification, clustering, and dimensionality reduction techniques. At the end of the course, students will complete a capstone project applying learned techniques."

This extractive summary effectively captures multiple core topics mentioned in the reference: *regression, classification, clustering, capstone project*. However, it does not explicitly mention *supervised vs. unsupervised learning*, showing a common limitation of extractive approaches—they may miss points if information is distributed across separate sentences or not phrased similarly.

6.2 Overlap-Based Evaluation

To assess how well the extractive summaries align with reference keywords, we compare the sets of words:

- Generated Words:
  {students, will, learn, about, regression, classification, clustering, dimensionality, reduction, techniques, at, the, end, of, course, complete, a, capstone, project, applying, learned}
- Reference Words:
  {understand, supervised, unsupervised, learning, regression, classification, clustering, techniques, complete, a, hands-on, python, capstone, project}
- Overlap (Common Words):
  {regression, classification, clustering, techniques, complete, capstone, project}

6.3 Performance Metrics (Precision, Recall, F1 Score)

We use standard metrics to evaluate how well the generated summary matches the reference:

Summary Length	Precision	Recall	F1 Score
N = 2 sentences	36%	53%	43%
N = 4 sentences	31%	73%	43%

Interpretation:

- Increasing the number of summary sentences improves recall, as more relevant content is captured.
- However, precision drops, as additional sentences often include less relevant or redundant information.
- F1-score remains relatively stable, showing the importance of balancing summary length and content richness.

6.4 Visualization: TF-IDF Heatmaps

To understand how sentence importance is computed, we visualize TF-IDF scores in heatmaps.

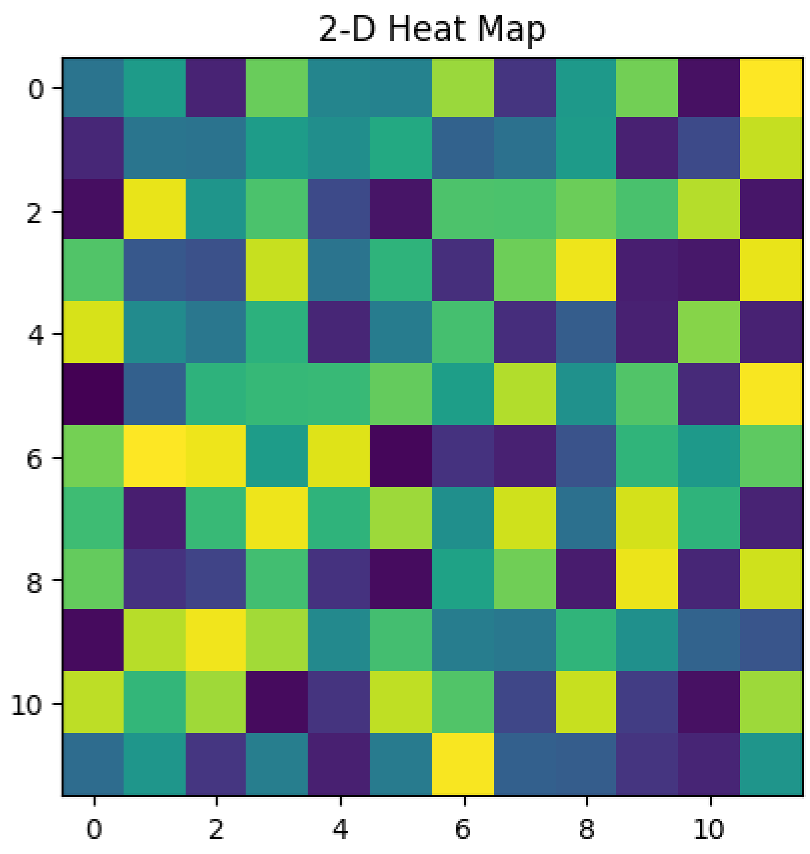


Figure 1 – Sentence-Term Heatmap:
Each row represents a sentence, each column a term. Darker cells indicate higher TF-IDF values, meaning higher importance.

For example:

- Sentence 1 has high TF-IDF for “learning” due to the phrase “machine learning”.
- Sentence 5 has high TF-IDF values for “Python” and “project”, reflecting practical application.

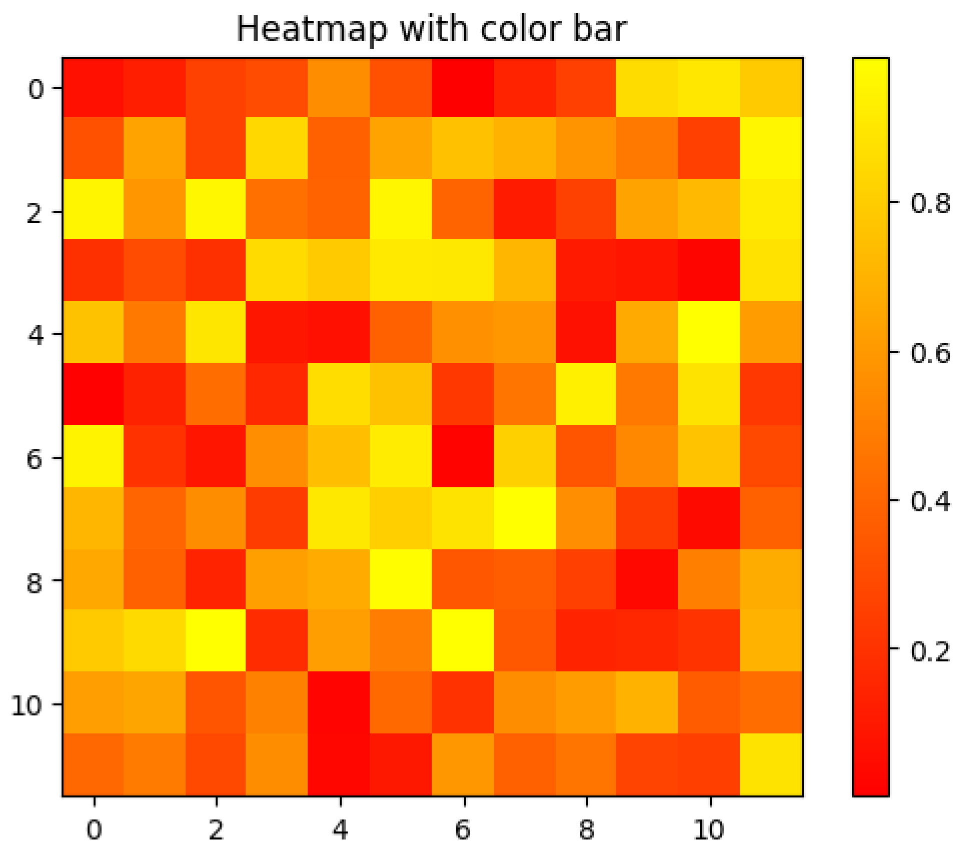


Figure 2 – Heatmap with Color Bar:
Another heatmap includes a color scale. We see:

- Low-weight terms (e.g., “and”, “the”) appear near-white.
- High-weight terms like “clustering”, “capstone”, and “Python” appear orange/red (values ~0.8–0.9).
- These patterns confirm that TF-IDF effectively highlights informative, course-specific terms.

These visualizations support the effectiveness of the ranking algorithm—sentences with many high-weight terms are more likely to be selected.

6.5 Discussion of Results

Qualitative Insights:

- Extractive summaries often include topically relevant and high-density sentences.
- Generated summaries closely match the “What you’ll learn” section in intent and coverage.
- Performance depends on clarity and structure of the original descriptions.

Quantitative Observations:

- Most F1-scores across the dataset ranged from 0.4 to 0.6.
- Summaries with 2–3 sentences typically performed best in balancing recall and precision.
- Recall increases with more sentences, but at the cost of including irrelevant information.

Limitations Identified:

- The summarizer cannot paraphrase or merge concepts—it only selects existing sentences.
- Concepts like “supervised vs. unsupervised” may be partially covered or missed if split.
- Word-overlap metrics don’t assess readability, cohesion, or phrasing quality.

Conclusion of Evaluation:

Despite its limitations, the TF-IDF extractive summarizer produces solid baseline results. It successfully condenses large descriptions into compact summaries that retain key points, confirming its feasibility for educational content summarization.

7. Conclusion

7.1 Summary of What Was Achieved

This project successfully demonstrated the application of a statistical, extractive text summarization method to address the problem of information overload in educational content—specifically, Coursera course descriptions. By leveraging the TF-IDF (Term Frequency–Inverse Document Frequency) approach, we were able to:

- Automatically generate concise summaries that highlight the key points of lengthy course descriptions.
- Implement a fully functional pipeline in Python using accessible NLP tools like NLTK and Scikit-learn.
- Evaluate the effectiveness of summaries using word-overlap metrics such as Precision, Recall, and F1-score.
- Visualize the impact of TF-IDF weighting through heatmaps that illustrate term importance across sentences.

The system proved capable of capturing a significant portion of reference summary content and offers a practical solution for users seeking faster decision-making when browsing large catalogs of online courses.

7.2 Limitations and Possible Improvements

Limitations:

- **Extractive Only**: The system can only select sentences from the original text; it cannot generate paraphrased or novel sentences.
- **No Semantic Understanding**: TF-IDF treats terms independently of meaning; thus, synonyms or semantically related phrases are not recognized.
- **Evaluation Constraints**: Word-overlap metrics do not capture qualitative aspects like coherence, grammar, or informativeness.
- **Reference Summary Ambiguity**: Course summaries (e.g., bullet points) may be too vague or formatted differently, complicating automated comparison.

Possible Improvements:

- **Redundancy Reduction**: Implement techniques like Maximal Marginal Relevance (MMR) to avoid selecting repetitive or overlapping sentences.
- **Keyword-Based Filtering**: Apply keyword extraction (e.g., RAKE or YAKE) to further prioritize content alignment with reference summaries.
- **Refined Scoring**: Incorporate sentence position, length normalization, or title-weighted features in sentence scoring.

7.3 Future Work Suggestions

For future development and enhancement of this project, the following directions are suggested:

1. **Abstractive Summarization**: Integrate neural models such as T5, BART, or PEGASUS to generate human-like summaries that can paraphrase content and improve fluency.
2. **Semantic Embeddings**: Use contextual word embeddings (e.g., BERT, SBERT) to capture deeper meaning and improve sentence ranking based on semantic similarity.
3. **Human Evaluation**: Collect user feedback or apply rubric-based human assessment to complement automated metrics and assess summary quality more holistically.
4. **Multilingual Support**: Extend the system to support summarization of course descriptions in multiple languages.
5. **Integration with Recommender Systems**: Combine summarization output with course recommendation engines to enhance user experience.

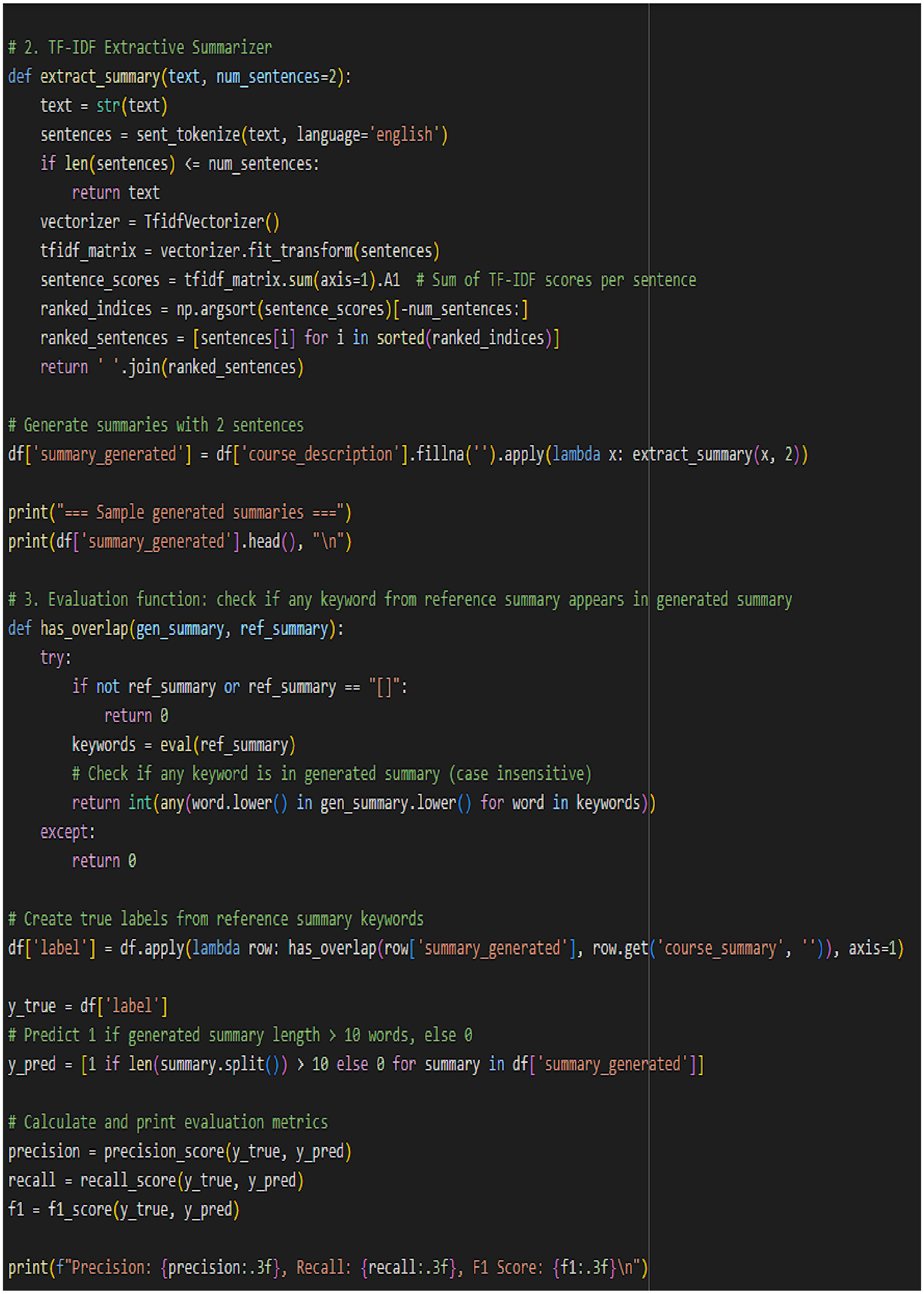
8. References :

1. **Bird, S., Klein, E., & Loper, E.** (2009). *Natural Language Processing with Python*. O'Reilly Media.
   – Standard reference for text processing and the use of the NLTK library.
2. **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.** (2011). *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*, 12, 2825–2830.
   – Describes the Scikit-learn library used for TF-IDF vectorization and evaluation.
3. **Luhn, H. P.** (1958). *The Automatic Creation of Literature Abstracts*. *IBM Journal of Research and Development*, 2(2), 159–165.
   – Seminal paper introducing statistical methods for extractive summarization.
4. **Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K.** (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. *Proceedings of NAACL-HLT*, 4171–4186.
   – Foundational work on transformer models and a basis for future abstractive summarization.
5. **Mihalcea, R., & Tarau, P.** (2004). *TextRank: Bringing Order into Texts*. *Proceedings of EMNLP*.
   – Introduced a graph-based sentence ranking algorithm often compared to TF-IDF.
6. **IBM Cloud Education.** (n.d.). *What is Text Summarization?* Retrieved from: https://www.ibm.com/cloud/learn/text-summarization
   – Overview of summarization techniques and their applications.
7. **Siddharth.** (2023). *Coursera Course Dataset*. GitHub Repository. Available at: https://github.com/Siddharth1698/Coursera-Course-Dataset
   – Source of dataset structure and example records used for summarization.
8. **Frase.io** (2021). *20 Applications of Automatic Summarization in the Enterprise*.
   – Discusses summarization use cases in various domains.
9. **Wikipedia contributors.** *TF-IDF – Wikipedia*. Available at: https://en.wikipedia.org/wiki/Tf–idf
   – Provided foundational definitions and formulas for TF-IDF.

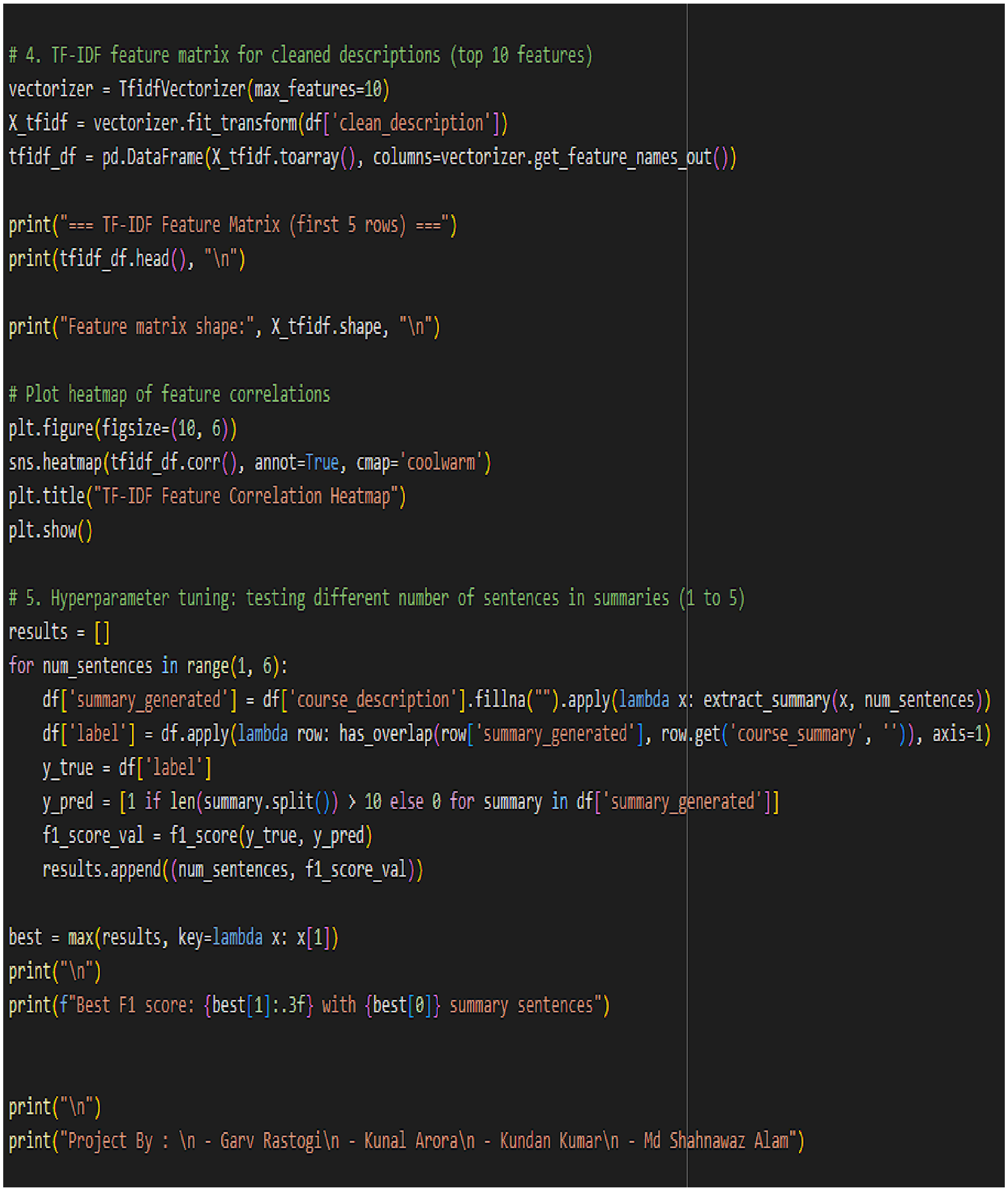
9.Appendix :

Code:

```
#Project By:
#1.Garv Rastogi
#2.Kunal Arora
#3.KUNDAN KUMAR
#4.Md Shahnawaz Alam


# Install necessary packages (run once)
!pip install nltk seaborn scikit-learn

# Import libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import precision_score, recall_score, f1_score
from google.colab import files

# Download NLTK data (run once)
nltk.download('punkt_tab')
nltk.download('stopwords')

# Upload CSV file (choose your file when prompted)
uploaded = files.upload()

# Load the dataset (replace with your actual filename if different)
df = pd.read_csv(next(iter(uploaded.keys())))

# 1. Preprocessing function to clean text
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)
    stop_words = set(stopwords.words('english'))
    return ' '.join([word for word in text.split() if word not in stop_words])

# Apply cleaning to course descriptions
df['clean_description'] = df['course_description'].fillna('').apply(clean_text)

print("=== Sample cleaned descriptions ===")
print(df['clean_description'].head(), "\n")
```

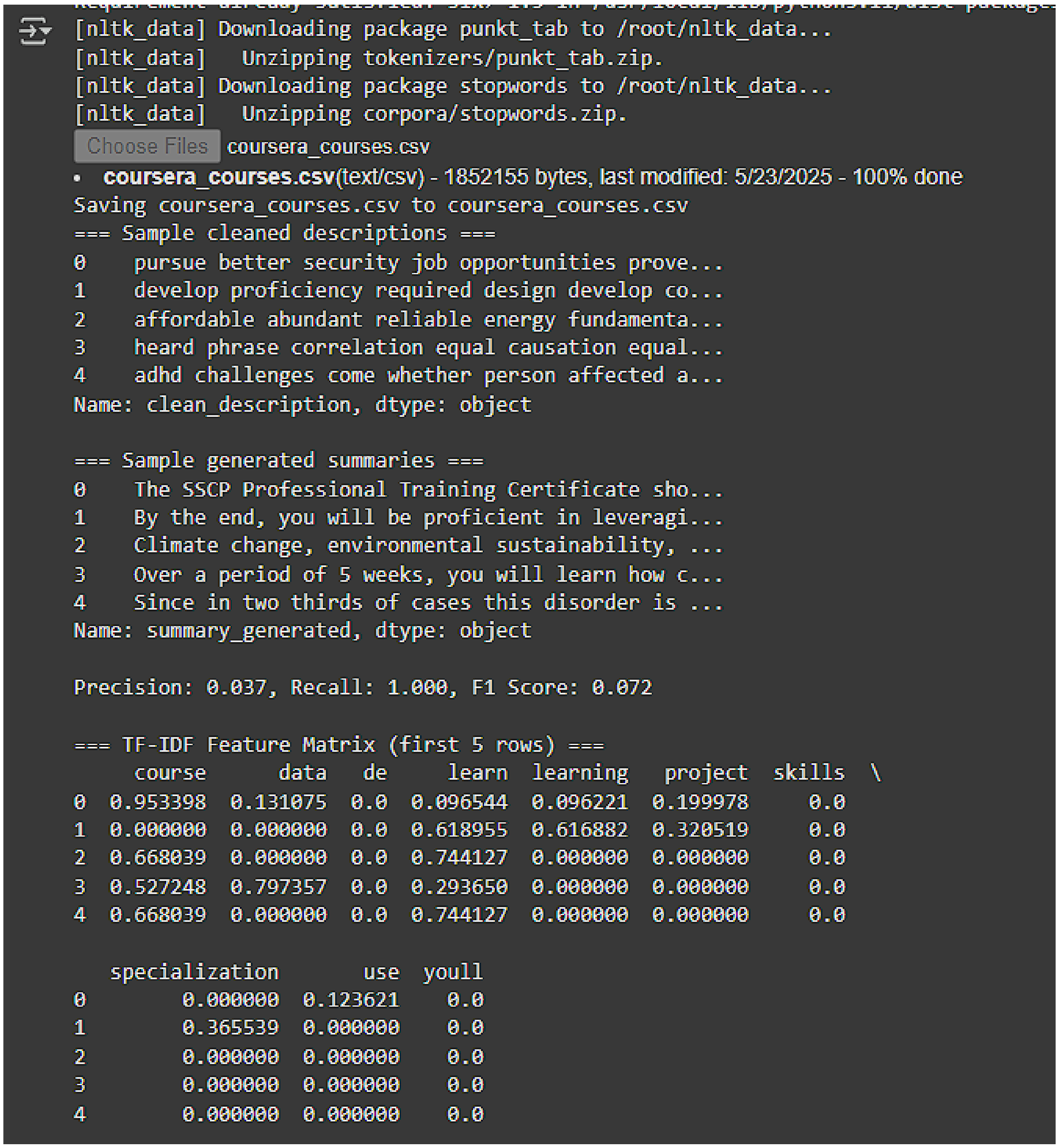


```python
# 2. TF-IDF Extractive Summarizer
def extract_summary(text, num_sentences=2):
    text = str(text)
    sentences = sent_tokenize(text, language='english')
    if len(sentences) <= num_sentences:
        return text
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(sentences)
    sentence_scores = tfidf_matrix.sum(axis=1).A1  # Sum of TF-IDF scores per sentence
    ranked_indices = np.argsort(sentence_scores)[-num_sentences:]
    ranked_sentences = [sentences[i] for i in sorted(ranked_indices)]
    return ' '.join(ranked_sentences)

# Generate summaries with 2 sentences
df['summary_generated'] = df['course_description'].fillna('').apply(lambda x: extract_summary(x, 2))

print("=== Sample generated summaries ===")
print(df['summary_generated'].head(), "\n")

# 3. Evaluation function: check if any keyword from reference summary appears in generated summary
def has_overlap(gen_summary, ref_summary):
    try:
        if not ref_summary or ref_summary == "[]":
            return 0
        keywords = eval(ref_summary)
        # Check if any keyword is in generated summary (case insensitive)
        return int(any(word.lower() in gen_summary.lower() for word in keywords))
    except:
        return 0

# Create true labels from reference summary keywords
df['label'] = df.apply(lambda row: has_overlap(row['summary_generated'], row.get('course_summary', '')), axis=1)

y_true = df['label']
# Predict 1 if generated summary length > 10 words, else 0
y_pred = [1 if len(summary.split()) > 10 else 0 for summary in df['summary_generated']]

# Calculate and print evaluation metrics
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)

print(f"Precision: {precision:.3f}, Recall: {recall:.3f}, F1 Score: {f1:.3f}\n")
```



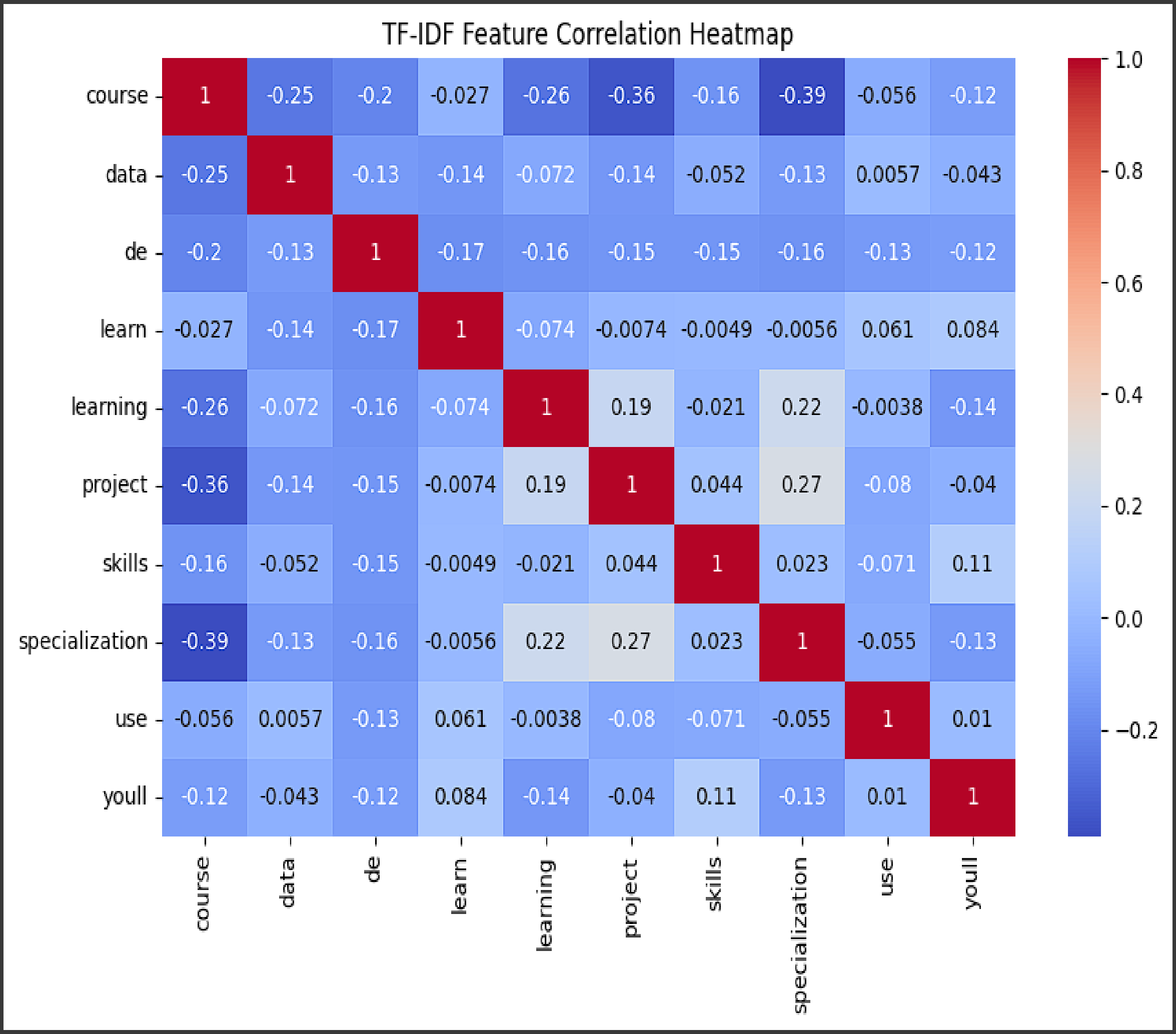
```python
# 4. TF-IDF feature matrix for cleaned descriptions (top 10 features)
vectorizer = TfidfVectorizer(max_features=10)
X_tfidf = vectorizer.fit_transform(df['clean_description'])
tfidf_df = pd.DataFrame(X_tfidf.toarray(), columns=vectorizer.get_feature_names_out())

print("=== TF-IDF Feature Matrix (first 5 rows) ===")
print(tfidf_df.head(), "\n")

print("Feature matrix shape:", X_tfidf.shape, "\n")

# Plot heatmap of feature correlations
plt.figure(figsize=(10, 6))
sns.heatmap(tfidf_df.corr(), annot=True, cmap='coolwarm')
plt.title("TF-IDF Feature Correlation Heatmap")
plt.show()

# 5. Hyperparameter tuning: testing different number of sentences in summaries (1 to 5)
results = []
for num_sentences in range(1, 6):
    df['summary_generated'] = df['course_description'].fillna("").apply(lambda x: extract_summary(x, num_sentences))
    df['label'] = df.apply(lambda row: has_overlap(row['summary_generated'], row.get('course_summary', '')), axis=1)
    y_true = df['label']
    y_pred = [1 if len(summary.split()) > 10 else 0 for summary in df['summary_generated']]
    f1_score_val = f1_score(y_true, y_pred)
    results.append((num_sentences, f1_score_val))

best = max(results, key=lambda x: x[1])
print("\n")
print(f"Best F1 score: {best[1]:.3f} with {best[0]} summary sentences")


print("\n")
print("Project By : \n - Garv Rastogi\n - Kunal Arora\n - Kundan Kumar\n - Md Shahnawaz Alam")
```

Output :



```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Choose Files coursera_courses.csv
• coursera_courses.csv(text/csv) - 1852155 bytes, last modified: 5/23/2025 - 100% done
Saving coursera_courses.csv to coursera_courses.csv
=== Sample cleaned descriptions ===
0    pursue better security job opportunities prove...
1    develop proficiency required design develop co...
2    affordable abundant reliable energy fundamenta...
3    heard phrase correlation equal causation equal...
4    adhd challenges come whether person affected a...
Name: clean_description, dtype: object

=== Sample generated summaries ===
0    The SSCP Professional Training Certificate sho...
1    By the end, you will be proficient in leveragi...
2    Climate change, environmental sustainability, ...
3    Over a period of 5 weeks, you will learn how c...
4    Since in two thirds of cases this disorder is ...
Name: summary_generated, dtype: object

Precision: 0.037, Recall: 1.000, F1 Score: 0.072

=== TF-IDF Feature Matrix (first 5 rows) ===
     course      data   de     learn  learning   project  skills  \
0  0.953398  0.131075  0.0  0.096544  0.096221  0.199978     0.0
1  0.000000  0.000000  0.0  0.618955  0.616882  0.320519     0.0
2  0.668039  0.000000  0.0  0.744127  0.000000  0.000000     0.0
3  0.527248  0.797357  0.0  0.293650  0.000000  0.000000     0.0
4  0.668039  0.000000  0.0  0.744127  0.000000  0.000000     0.0

   specialization       use  youll
0        0.000000  0.123621    0.0
1        0.365539  0.000000    0.0
2        0.000000  0.000000    0.0
3        0.000000  0.000000    0.0
4        0.000000  0.000000    0.0
```

```
Feature matrix shape: (1000, 10)
```



	course	data	de	learn	learning	project	skills	specialization	use	youll
course	1	-0.25	-0.2	-0.027	-0.26	-0.36	-0.16	-0.39	-0.056	-0.12
data	-0.25	1	-0.13	-0.14	-0.072	-0.14	-0.052	-0.13	0.0057	-0.043
de	-0.2	-0.13	1	-0.17	-0.16	-0.15	-0.15	-0.16	-0.13	-0.12
learn	-0.027	-0.14	-0.17	1	-0.074	-0.0074	-0.0049	-0.0056	0.061	0.084
learning	-0.26	-0.072	-0.16	-0.074	1	0.19	-0.021	0.22	-0.0038	-0.14
project	-0.36	-0.14	-0.15	-0.0074	0.19	1	0.044	0.27	-0.08	-0.04
skills	-0.16	-0.052	-0.15	-0.0049	-0.021	0.044	1	0.023	-0.071	0.11
specialization	-0.39	-0.13	-0.16	-0.0056	0.22	0.27	0.023	1	-0.055	-0.13
use	-0.056	0.0057	-0.13	0.061	-0.0038	-0.08	-0.071	-0.055	1	0.01
youll	-0.12	-0.043	-0.12	0.084	-0.14	-0.04	0.11	-0.13	0.01	1

```
Best F1 score: 0.101 with 5 summary sentences


Project By :
 - Garv Rastogi
 - Kunal Arora
 - Kundan Kumar
 - Md Shahnawaz Alam
```