gyanankurtfidf

July 13, 2025

1 TF-IDF Assignment Submission

Name: Gyanankur Baruah
Roll Number: 202405005

Topic: Implementation of TF-IDF using 4 Text Documents in Python

[ ]:
```
!pip install -q scikit-learn pandas seaborn matplotlib
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[ ]:
```
# Define four distinct documents
documents = [
    "Data science and machine learning are evolving fast.",
    "Visualization helps communicate data insights clearly.",
    "Classification and clustering are core to analytics.",
    "Dashboards with Python unlock business decisions."
]
```

[ ]:
```
# Apply TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(documents)

# Convert to DataFrame for analysis
df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())
df
```

[ ]:
```
   analytics       and       are  business  classification   clearly  \
0   0.000000  0.300912  0.300912  0.000000        0.000000  0.000000
1   0.000000  0.000000  0.000000  0.000000        0.000000  0.421765
2   0.400218  0.315537  0.315537  0.000000        0.400218  0.000000
3   0.000000  0.000000  0.000000  0.408248        0.000000  0.000000

   clustering  communicate      core  dashboards  …     helps  insights  \
0    0.000000     0.000000  0.000000    0.000000  …  0.000000  0.000000
```

```
1    0.000000     0.421765  0.000000    0.000000  …  0.421765  0.421765
2    0.400218     0.000000  0.400218    0.000000  …  0.000000  0.000000
3    0.000000     0.000000  0.000000    0.408248  …  0.000000  0.000000

   learning   machine    python   science        to    unlock  visualization  \
0  0.381669  0.381669  0.000000  0.381669  0.000000  0.000000       0.000000
1  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000       0.421765
2  0.000000  0.000000  0.000000  0.000000  0.400218  0.000000       0.000000
3  0.000000  0.000000  0.408248  0.000000  0.000000  0.408248       0.000000

       with
0  0.000000
1  0.000000
2  0.000000
3  0.408248

[4 rows x 24 columns]
```

1.1 Explanation of TF-IDF Output

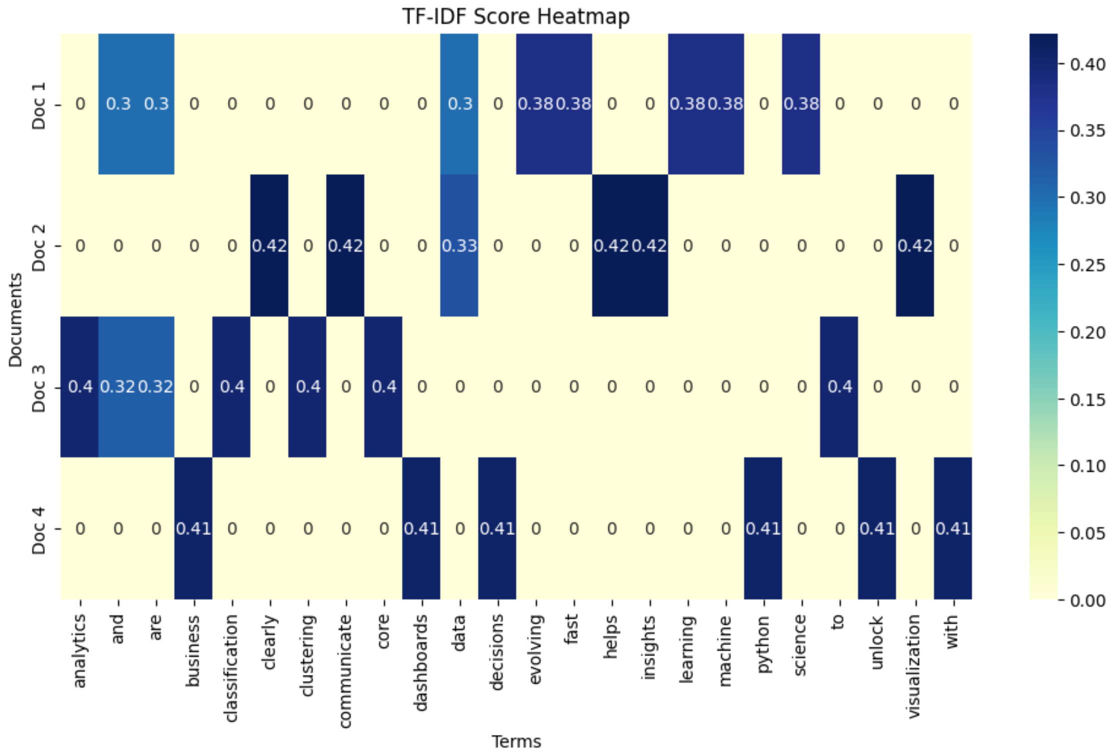
This DataFrame shows how important each term is in its corresponding document.
- High TF-IDF values highlight terms that are unique and meaningful in one document.
- For example, "dashboards" scores highly in Document 4 and not elsewhere—indicating its uniqueness.
- Common terms like "data" or "python" appear in multiple documents with lower scores.

[ ]:
```
def top_keywords_per_doc(tfidf_matrix, feature_names, top_n=5):
    for i, row in enumerate(tfidf_matrix):
        top_indices = np.argsort(row)[::-1][:top_n]
        keywords = [(feature_names[j], row[j]) for j in top_indices]
        print(f"\nDocument {i + 1} Top Keywords:")
        for word, score in keywords:
            print(f"  {word}: {score:.3f}")

top_keywords_per_doc(X.toarray(), vectorizer.get_feature_names_out())
```

```
Document 1 Top Keywords:
  machine: 0.382
  learning: 0.382
  science: 0.382
  evolving: 0.382
  fast: 0.382

Document 2 Top Keywords:
  visualization: 0.422
  clearly: 0.422
  helps: 0.422
```

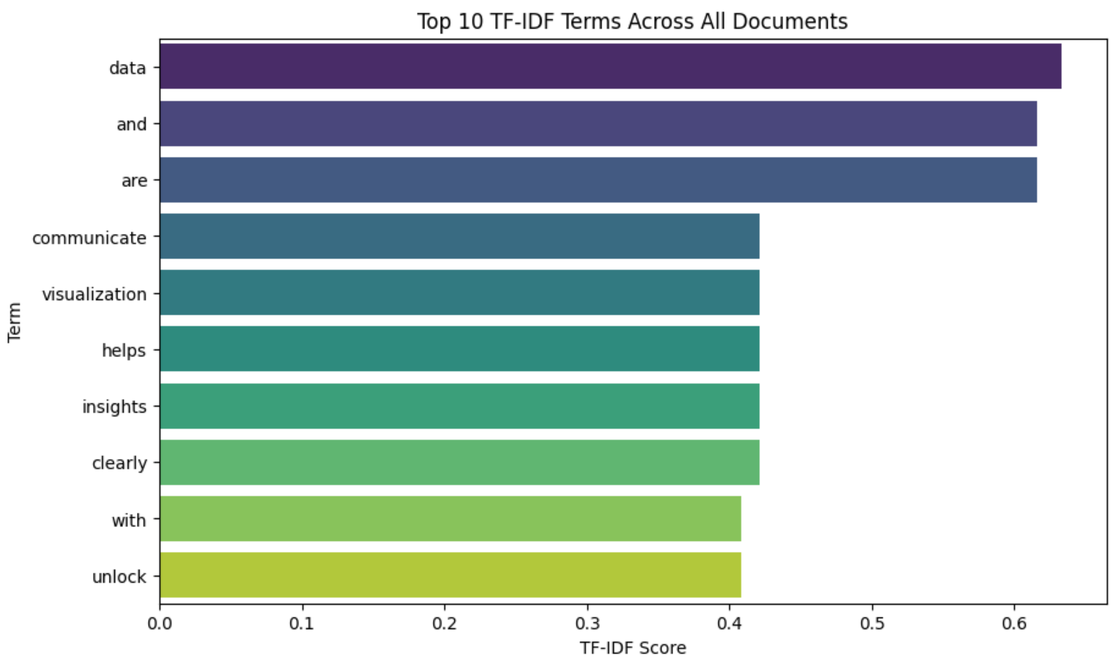
```
  insights: 0.422
  communicate: 0.422

Document 3 Top Keywords:
  to: 0.400
  classification: 0.400
  analytics: 0.400
  core: 0.400
  clustering: 0.400

Document 4 Top Keywords:
  with: 0.408
  unlock: 0.408
  python: 0.408
  dashboards: 0.408
  decisions: 0.408
```

[ ]:
```python
plt.figure(figsize=(12, 6))
sns.heatmap(df, annot=True, cmap="YlGnBu", xticklabels=True, yticklabels=[f'Doc␣
↪{i+1}' for i in range(len(documents))])
plt.title("TF-IDF Score Heatmap")
plt.xlabel("Terms")
plt.ylabel("Documents")
plt.show()
```



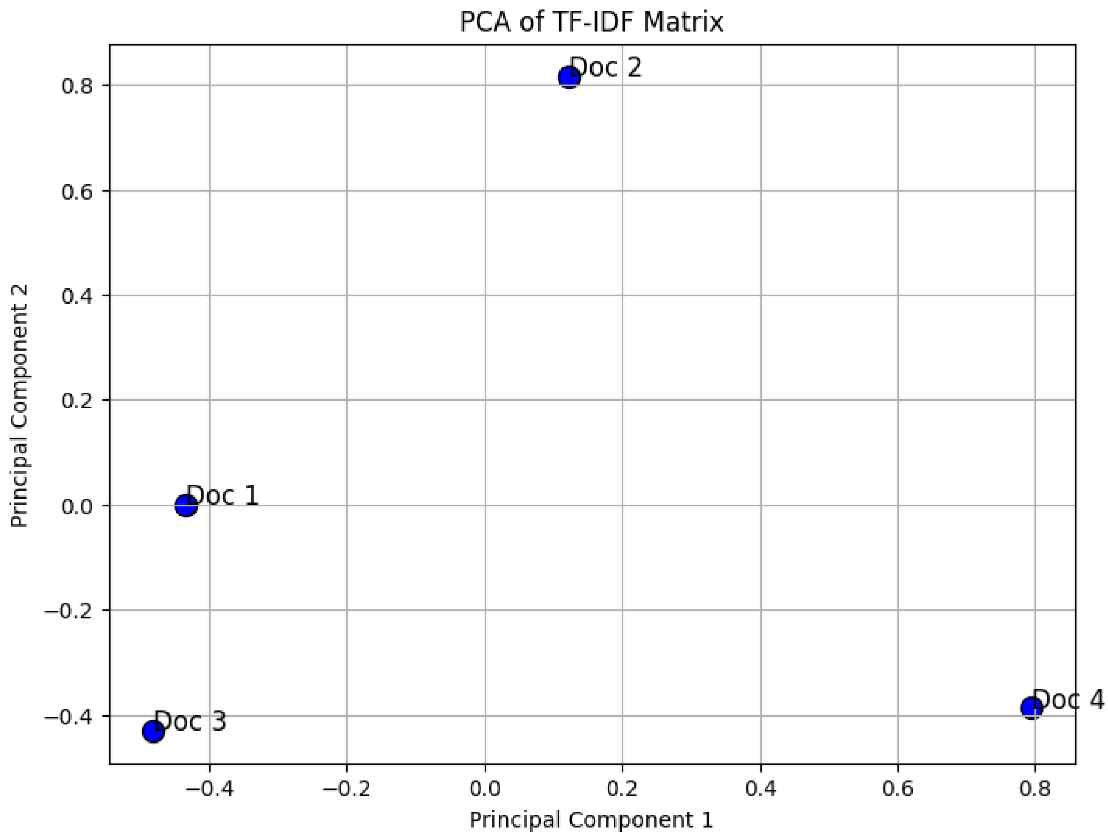
[ ]:
```python
# Sum TF-IDF scores across all documents
term_scores = df.sum(axis=0)

# Get top 10 terms
top_terms = term_scores.sort_values(ascending=False).head(10)

# Plot bar chart
plt.figure(figsize=(10, 6))
sns.barplot(x=top_terms.values, y=top_terms.index, palette="viridis")
plt.title("Top 10 TF-IDF Terms Across All Documents")
plt.xlabel("TF-IDF Score")
plt.ylabel("Term")
plt.show()
```

```
/tmp/ipython-input-10-1603655645.py:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=top_terms.values, y=top_terms.index, palette="viridis")
```



[ ]:
```python
from sklearn.decomposition import PCA
```

```python
# Reduce TF-IDF matrix to 2D
pca = PCA(n_components=2)
pca_result = pca.fit_transform(X.toarray())

# Plot
plt.figure(figsize=(8, 6))
plt.scatter(pca_result[:, 0], pca_result[:, 1], c='blue', s=100, edgecolors='k')
for i, txt in enumerate([f'Doc {i+1}' for i in range(len(documents))]):
    plt.annotate(txt, (pca_result[i, 0], pca_result[i, 1]), fontsize=12)
plt.title("PCA of TF-IDF Matrix")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.grid(True)
plt.show()
```



[ ]:
```python
from sklearn.metrics.pairwise import cosine_similarity

similarity_matrix = cosine_similarity(X)
pd.DataFrame(similarity_matrix, columns=[f'Doc {i+1}' for i in␣
 ↪range(len(documents))],
```

```
            index=[f'Doc {i+1}' for i in range(len(documents))])
```

[ ]:

```
          Doc 1     Doc 2     Doc 3  Doc 4
Doc 1  1.000000  0.100061  0.189898    0.0
Doc 2  0.100061  1.000000  0.000000    0.0
Doc 3  0.189898  0.000000  1.000000    0.0
Doc 4  0.000000  0.000000  0.000000    1.0
```

1.2 Conclusion

In this task, I used a method called TF-IDF to find which words are most important in each of four documents. It helped show which words were special to each document and which were common. I also used tables and charts to better understand the results. This technique is useful for studying text, finding keywords, or helping computers understand written language.