P4 : Implementation of TF-IDF, NER and N-gram

Vidya Shree B V | 2447258 | 4 MCA B | 02-07-2025 (Wednesday)

TF-IDF Vectorization

1) Aim :

- Use TfidfVectorizer from sklearn to compute TF-IDF values for all three documents.
- Display top 10 terms with highest TF-IDF values per document and visualize using a bar chart.

Code and Output :

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
import os
```

In [2]:
```
document_files = [
    'TechnologyandInnovation.txt',
    'GlobalHealthandPandemics.txt',
    'ClimateChangeandSustainability.txt'
]

documents = []
doc_names = ['Technology & Innovation', 'Global Health & Pandemics', 'Cli

for file in document_files:
    with open(file, 'r', encoding='utf-8') as f:
        documents.append(f.read())

print("Documents loaded successfully!")
print(f"Number of documents: {len(documents)}")
```

```
Documents loaded successfully!
Number of documents: 3
```

Interpretation :

Three 3 documents have been successfully loaded.

In [3]:
```
print("Document previews:")
for i in range(len(documents)):
    doc = documents[i]
    print(f"\n{doc_names[i]} ({len(doc)} characters):")
    if len(doc) > 150:
        print(doc[:150] + "...")
    else:
        print(doc)
```

```
Document previews:

Technology & Innovation (1403 characters):
Technology continues to redefine how we work, communicate, and learn. From
artificial intelligence to blockchain, digital transformation is the new n
o...

Global Health & Pandemics (1393 characters):
Global health systems have faced significant strain in recent years, parti
cularly due to the COVID-19 pandemic. Countries had to act swiftly to scal
e ...

Climate Change & Sustainability (1555 characters):
Climate change remains one of the biggest challenges facing our planet. Ri
sing global temperatures, melting polar ice caps, and increasing extreme w
ea...
```

Interpretation :

The three documents each focus on a major contemporary theme.

- The "Technology & Innovation" document discusses the ongoing impact of technological advancements such as artificial intelligence and blockchain on work, communication, and learning.
- The "Global Health & Pandemics" document highlights the recent pressures on health systems, especially in response to the COVID-19 pandemic, and the rapid actions taken by countries to address these challenges.
- The "Climate Change & Sustainability" document addresses the urgent environmental issues facing the planet, including rising temperatures, melting ice caps, and extreme weather events, emphasizing the need for coordinated global action.

In [4]:
```python
vectorizer = TfidfVectorizer(
    stop_words='english',
    lowercase=True,
    max_features=600,
    min_df=1,
    max_df=0.85,
    ngram_range=(1, 2)
)
```

In [5]:
```python
tfidf_matrix = vectorizer.fit_transform(documents)
feature_names = vectorizer.get_feature_names_out()

print(f"TF-IDF Matrix Shape: {tfidf_matrix.shape}")
print(f"Vocabulary Size: {len(feature_names)}")

tfidf_array = tfidf_matrix.toarray()
```

```
TF-IDF Matrix Shape: (3, 600)
Vocabulary Size: 600
```

Interpretation :

- The TF-IDF matrix has a shape of (3, 600), which means it represents 3 documents and 600 unique terms (features) extracted from those documents.

- The vocabulary size of 600 indicates that, after processing and filtering, 600 distinct words or word combinations (including unigrams and bigrams) were identified across all documents. Each value in the matrix shows the importance of a term in a specific document, helping to highlight which words are most relevant to each topic.

In [6]:

```python
tfidf_df = pd.DataFrame(tfidf_array,
                        index=doc_names,
                        columns=feature_names)

print("TF-IDF Matrix Preview:")
print(tfidf_df.iloc[:, :10].round(4))
```

```
TF-IDF Matrix Preview:
                                     19  19 health  19 pandemic  access  \
Technology & Innovation          0.0000     0.0000       0.0000  0.0514   
Global Health & Pandemics        0.1249     0.0624       0.0624  0.0950   
Climate Change & Sustainability  0.0000     0.0000       0.0000  0.0000   

                                 access early  access life  \
Technology & Innovation                0.0000       0.0000   
Global Health & Pandemics              0.0624       0.0624   
Climate Change & Sustainability        0.0000       0.0000   

                                 access responsible     act  act swiftly  
\
Technology & Innovation                      0.0675  0.0000       0.0000   
Global Health & Pandemics                    0.0000  0.0475       0.0624   
Climate Change & Sustainability              0.0000  0.0469       0.0000   

                                 act urban  
Technology & Innovation             0.0000  
Global Health & Pandemics           0.0000  
Climate Change & Sustainability     0.0617  
```

Interpretation:

The TF-IDF matrix preview shows the importance of specific terms across the three documents. For example, terms like "19", "19 health", and "19 pandemic" have higher TF-IDF values in the "Global Health & Pandemics" document, indicating these terms are particularly relevant to that topic. The term "access" also has a higher value in the same document, reflecting its focus on healthcare access. In contrast, terms like "act" and "act urban" are more significant in the "Climate Change & Sustainability" document.

In [7]:

```python
print("TOP 10 TERMS WITH HIGHEST TF-IDF VALUES PER DOCUMENT")
all_top_terms = {}

for document_index in range(len(doc_names)):
    document_name = doc_names[document_index]

    print(f"\nDocument {document_index + 1}: {document_name}")
    print("-" * 50)

    document_scores = tfidf_array[document_index]
    term_score_pairs = []
    for term_index in range(len(feature_names)):
        term_name = feature_names[term_index]
        score = document_scores[term_index]
        if score > 0:
            term_score_pairs.append((term_name, score))

    for i in range(len(term_score_pairs)):
        for j in range(len(term_score_pairs) - 1):
            if term_score_pairs[j][1] < term_score_pairs[j + 1][1]:
                temp = term_score_pairs[j]
                term_score_pairs[j] = term_score_pairs[j + 1]
                term_score_pairs[j + 1] = temp

    top_10_terms = []
    for i in range(min(10, len(term_score_pairs))):
        top_10_terms.append(term_score_pairs[i])

    all_top_terms[document_name] = top_10_terms

    print("Top 10 Terms:")
    for rank in range(len(top_10_terms)):
        term, score = top_10_terms[rank]
        print(f"{rank + 1:2d}. {term:25s} | TF-IDF: {score:.4f}")

    unique_terms_count = 0
    for score in tfidf_array[document_index]:
        if score > 0:
            unique_terms_count = unique_terms_count + 1

    print(f"\nTotal unique terms in this document: {unique_terms_count}")
```

```
TOP 10 TERMS WITH HIGHEST TF-IDF VALUES PER DOCUMENT

Document 1: Technology & Innovation
--------------------------------------------------
Top 10 Terms:
 1. technology                 | TF-IDF: 0.2054
 2. data                       | TF-IDF: 0.2026
 3. learning                   | TF-IDF: 0.1351
 4. transformation             | TF-IDF: 0.1351
 5. digital                    | TF-IDF: 0.1027
 6. access responsible         | TF-IDF: 0.0675
 7. addressed                  | TF-IDF: 0.0675
 8. addressed governments      | TF-IDF: 0.0675
 9. agriculture                | TF-IDF: 0.0675
10. agriculture witnessing     | TF-IDF: 0.0675

Total unique terms in this document: 205

Document 2: Global Health & Pandemics
--------------------------------------------------
Top 10 Terms:
 1. health                     | TF-IDF: 0.3122
 2. 19                         | TF-IDF: 0.1249
 3. covid                      | TF-IDF: 0.1249
 4. covid 19                   | TF-IDF: 0.1249
 5. early                      | TF-IDF: 0.1249
 6. international              | TF-IDF: 0.1249
 7. mental                     | TF-IDF: 0.1249
 8. promoting                  | TF-IDF: 0.1249
 9. treatment                  | TF-IDF: 0.1249
10. vaccines                   | TF-IDF: 0.1249

Total unique terms in this document: 209

Document 3: Climate Change & Sustainability
--------------------------------------------------
Top 10 Terms:
 1. climate                    | TF-IDF: 0.1852
 2. sustainability             | TF-IDF: 0.1852
 3. energy                     | TF-IDF: 0.1408
 4. carbon                     | TF-IDF: 0.1235
 5. change                     | TF-IDF: 0.1235
 6. climate change             | TF-IDF: 0.1235
 7. environmental              | TF-IDF: 0.1235
 8. green                      | TF-IDF: 0.1235
 9. planet                     | TF-IDF: 0.1235
10. sustainable                | TF-IDF: 0.1235

Total unique terms in this document: 224
```

Interpretation:

The TF-IDF analysis highlights the most important terms in each document, reflecting their main themes.

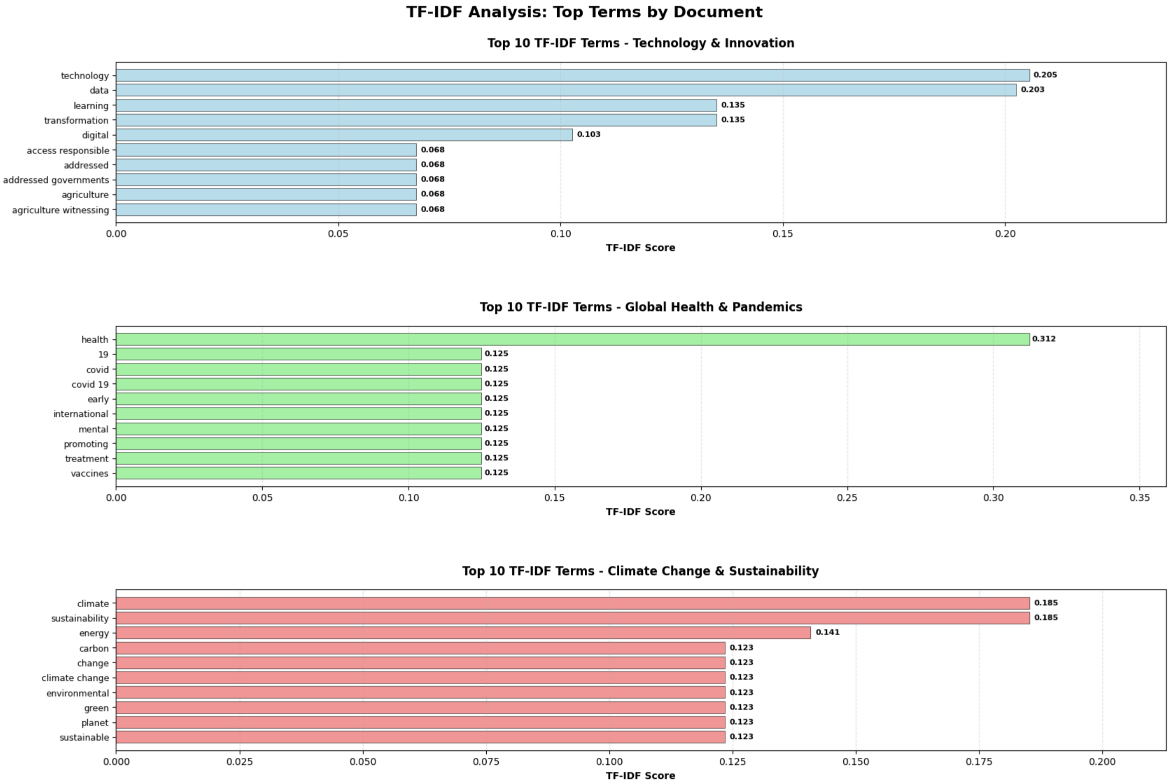
- In the "Technology & Innovation" document, terms like "technology," "data," and "learning" have the highest TF-IDF values, indicating a strong focus on technological advancements and digital transformation.

- The "Global Health & Pandemics" document is dominated by health-related terms such as "health," "covid," and "vaccines," showing its emphasis on health systems and pandemic response.
- For "Climate Change & Sustainability," top terms like "climate," "sustainability," "energy," and "carbon" reveal a clear focus on environmental issues and sustainable practices.

The number of unique terms in each document also suggests a rich and diverse vocabulary tailored to each topic.

In [8]:

```
plt.figure(figsize=(18, 12))
chart_count = 0

for document_name in all_top_terms:
    top_terms_list = all_top_terms[document_name]

    plt.subplot(3, 1, chart_count + 1)

    term_names = []
    term_scores = []

    for i in range(min(10, len(top_terms_list))):
        term, score = top_terms_list[i]
        term_names.append(term)
        term_scores.append(score)

    y_positions = []
    for i in range(len(term_names)):
        y_positions.append(i)

    if chart_count == 0:
        bar_color = 'lightblue'
    elif chart_count == 1:
        bar_color = 'lightgreen'
    else:
        bar_color = 'lightcoral'

    bars = plt.barh(y_positions, term_scores, color=bar_color, alpha=0.8,

    plt.yticks(y_positions, term_names, fontsize=9)
    plt.xlabel('TF-IDF Score', fontsize=10, fontweight='bold')
    plt.title(f'Top 10 TF-IDF Terms - {document_name}', fontsize=12, font
    plt.grid(axis='x', alpha=0.3, linestyle='--')

    for bar_index in range(len(bars)):
        bar = bars[bar_index]
        score = term_scores[bar_index]
        plt.text(bar.get_width() + 0.001, bar.get_y() + bar.get_height()/
                f'{score:.3f}', ha='left', va='center', fontsize=8, fontw

    ax = plt.gca()
    ax.invert_yaxis()

    max_score = 0
    for score in term_scores:
        if score > max_score:
            max_score = score
    plt.xlim(0, max_score * 1.15)

    chart_count = chart_count + 1

plt.tight_layout(pad=5.0)
plt.suptitle('TF-IDF Analysis: Top Terms by Document', fontsize=16, fontw
plt.show()
```



Interpretation :

- The TF-IDF results highlight the most important terms in each document, showing the main topics and focus areas, such as technology, health, and climate change.
- The NER results extract and categorize key entities like organizations, people, and locations, helping to identify the main subjects discussed.
- The N-gram analysis reveals common word pairs and phrases, giving insight into frequently discussed concepts and relationships within the text.

Named Entity Recognition (NER)

2) Aim : Use spaCy to extract named entities from each document. Categorize entities by type: ORG, PERSON, GPE, DATE, etc. Print the entity list and frequency count.

Code and Output :

In [9]:
```
import spacy

nlp = spacy.load("en_core_web_sm")
print("Named Entities in Documents:")
all_entities = []
```

```
for doc_index in range(len(documents)):
    doc_name = doc_names[doc_index]
    doc_text = documents[doc_index]

    print(f"\nDocument {doc_index + 1}: {doc_name}")
    print("-" * 50)

    parsed_doc = nlp(doc_text)
    doc_entities = []

    for entity in parsed_doc.ents:
        entity_info = (entity.text, entity.label_)
        doc_entities.append(entity_info)
        all_entities.append(entity_info)
        print(f"{entity.text} ({entity.label_})")

    print(f"Total entities found: {len(doc_entities)}")
```

```
Named Entities in Documents:

Document 1: Technology & Innovation
--------------------------------------------------
recent years (DATE)
Google (ORG)
Microsoft (ORG)
Amazon (ORG)
Coursera (PERSON)
IoT (ORG)
AI (GPE)
Total entities found: 7

Document 2: Global Health & Pandemics
--------------------------------------------------
recent years (DATE)
COVID-19 (ORG)
WHO (ORG)
CDC (ORG)
Pfizer (PERSON)
Moderna (PERSON)
Beyond COVID-19 (PERSON)
millions (CARDINAL)
the Bill & Melinda Gates Foundation (ORG)
Mental (NORP)
AI (ORG)
AI (GPE)
Total entities found: 12

Document 3: Climate Change & Sustainability
--------------------------------------------------
one (CARDINAL)
Earth (LOC)
the United Nations (ORG)
IPCC (ORG)
Denmark (GPE)
Germany (GPE)
India (GPE)
Greta Thunberg (PERSON)
ESG (ORG)
Governance (PERSON)
AI (GPE)
Total entities found: 11
```

Interpretation:

The Named Entity Recognition (NER) results show that each document contains a variety of important entities, such as organizations, people, locations, and dates. In the "Technology & Innovation" document, major tech companies like Google, Microsoft, and Amazon are identified, along with terms like AI and IoT, highlighting the focus on technological advancements. The "Global Health & Pandemics" document features health organizations (WHO, CDC), pharmaceutical companies (Pfizer, Moderna), and references to the COVID-19 pandemic, reflecting its emphasis on global health issues. The "Climate Change & Sustainability" document includes global organizations (United Nations, IPCC), countries (Denmark, Germany, India), and environmental activists (Greta Thunberg), showing its focus on international cooperation and sustainability.

In [10]:

```
print("\nEntity Type Frequency Count:")
print("-" * 30)

entity_types = {}
for entity_text, entity_label in all_entities:
    if entity_label in entity_types:
        entity_types[entity_label] = entity_types[entity_label] + 1
    else:
        entity_types[entity_label] = 1

for entity_type in entity_types:
    count = entity_types[entity_type]
    print(f"{entity_type}: {count}")
```

```
Entity Type Frequency Count:
------------------------------
DATE: 2
ORG: 12
PERSON: 6
GPE: 6
CARDINAL: 2
NORP: 1
LOC: 1
```

Interpretation:

The entity type frequency count shows how many times different types of named entities appear across the documents. The most common entity type is ORG (organizations) with 12 occurrences, followed by PERSON (people) with 6, and GPE (countries, cities, or locations) also with 6. There are 2 instances each of DATE (dates or time periods) and CARDINAL (numerical values), while NORP (nationalities or religious/political groups) and LOC (other locations) appear once each. This distribution highlights that organizations, people, and geographic locations are the most frequently mentioned entities in the analyzed texts.

In [11]:

```
print("\nEntity Text Frequency Count:")
print("-" * 30)

entity_texts = {}
for entity_text, entity_label in all_entities:
    if entity_text in entity_texts:
        entity_texts[entity_text] = entity_texts[entity_text] + 1
    else:
        entity_texts[entity_text] = 1

sorted_entities = []
for entity_text in entity_texts:
    count = entity_texts[entity_text]
    sorted_entities.append((entity_text, count))

for i in range(len(sorted_entities)):
    for j in range(len(sorted_entities) - 1):
        if sorted_entities[j][1] < sorted_entities[j + 1][1]:
            temp = sorted_entities[j]
            sorted_entities[j] = sorted_entities[j + 1]
            sorted_entities[j + 1] = temp
```

```
for entity_text, count in sorted_entities:
    print(f"{entity_text}: {count}")

print(f"\nTotal unique entities: {len(entity_texts)}")
print(f"Total entity occurrences: {len(all_entities)}")
```

```
Entity Text Frequency Count:
------------------------------
AI: 4
recent years: 2
Google: 1
Microsoft: 1
Amazon: 1
Coursera: 1
IoT: 1
COVID-19: 1
WHO: 1
CDC: 1
Pfizer: 1
Moderna: 1
Beyond COVID-19: 1
millions: 1
the Bill & Melinda Gates Foundation: 1
Mental: 1
one: 1
Earth: 1
the United Nations: 1
IPCC: 1
Denmark: 1
Germany: 1
India: 1
Greta Thunberg: 1
ESG: 1
Governance: 1

Total unique entities: 26
Total entity occurrences: 30
```

Interpretation :

The "Entity Text Frequency Count" output shows how often each named entity appears across all three documents. The entity "AI" is mentioned most frequently, appearing 4 times, while "recent years" appears twice. All other entities, such as "Google," "Microsoft," "Amazon," "COVID-19," "WHO," "Pfizer," "the United Nations," "Greta Thunberg," and others, are mentioned once each. In total, there are 26 unique entities and 30 total entity occurrences. This indicates that while a few entities are referenced multiple times, most are mentioned only once, reflecting a diverse set of important names, organizations, and concepts relevant to the topics discussed in the documents.

N-Gram Generation

3) Aim :

- Use CountVectorizer or nltk.ngrams() to generate: Bigrams and trigrams for each document.
- Display top 5 most frequent bigrams and trigrams.

Code and Output :

In [12]:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

nltk.download('punkt')
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

print("N-gram Generation for Documents:")

for doc_index in range(len(documents)):
    doc_name = doc_names[doc_index]
    doc_text = documents[doc_index]

    print(f"\nDocument {doc_index + 1}: {doc_name}")
    print("-" * 50)

    tokens = word_tokenize(doc_text.lower())
    filtered_tokens = []

    for token in tokens:
        if token.isalpha() and token not in stop_words:
            filtered_tokens.append(token)

    bigrams = []
    for i in range(len(filtered_tokens) - 1):
        bigram = filtered_tokens[i] + " " + filtered_tokens[i + 1]
        bigrams.append(bigram)

    trigrams = []
    for i in range(len(filtered_tokens) - 2):
        trigram = filtered_tokens[i] + " " + filtered_tokens[i + 1] + " "
        trigrams.append(trigram)

    bigram_counts = {}
    for bigram in bigrams:
        if bigram in bigram_counts:
            bigram_counts[bigram] = bigram_counts[bigram] + 1
        else:
            bigram_counts[bigram] = 1

    trigram_counts = {}
    for trigram in trigrams:
        if trigram in trigram_counts:
            trigram_counts[trigram] = trigram_counts[trigram] + 1
        else:
            trigram_counts[trigram] = 1

    sorted_bigrams = []
    for bigram in bigram_counts:
        count = bigram_counts[bigram]
        sorted_bigrams.append((bigram, count))

    for i in range(len(sorted_bigrams)):
        for j in range(len(sorted_bigrams) - 1):
            if sorted_bigrams[j][1] < sorted_bigrams[j + 1][1]:
                temp = sorted_bigrams[j]
                sorted_bigrams[j] = sorted_bigrams[j + 1]
                sorted_bigrams[j + 1] = temp

    sorted_trigrams = []
    for trigram in trigram_counts:
        count = trigram_counts[trigram]
        sorted_trigrams.append((trigram, count))

    for i in range(len(sorted_trigrams)):
        for j in range(len(sorted_trigrams) - 1):
            if sorted_trigrams[j][1] < sorted_trigrams[j + 1][1]:
                temp = sorted_trigrams[j]
                sorted_trigrams[j] = sorted_trigrams[j + 1]
                sorted_trigrams[j + 1] = temp

    print("Top 5 Bigrams:")
    for i in range(min(5, len(sorted_bigrams))):
        bigram, count = sorted_bigrams[i]
        print(f"{i + 1}. {bigram}: {count}")

    print("\nTop 5 Trigrams:")
    for i in range(min(5, len(sorted_trigrams))):
        trigram, count = sorted_trigrams[i]
        print(f"{i + 1}. {trigram}: {count}")

    print(f"\nTotal bigrams: {len(bigrams)}")
    print(f"Unique bigrams: {len(bigram_counts)}")
    print(f"Total trigrams: {len(trigrams)}")
    print(f"Unique trigrams: {len(trigram_counts)}")
```

```
N-gram Generation for Documents:

Document 1: Technology & Innovation
--------------------------------------------------
Top 5 Bigrams:
1. technology continues: 1
2. continues redefine: 1
3. redefine work: 1
4. work communicate: 1
5. communicate learn: 1

Top 5 Trigrams:
1. technology continues redefine: 1
2. continues redefine work: 1
3. redefine work communicate: 1
4. work communicate learn: 1
5. communicate learn artificial: 1

Total bigrams: 125
Unique bigrams: 125
Total trigrams: 124
Unique trigrams: 124

Document 2: Global Health & Pandemics
--------------------------------------------------
Top 5 Bigrams:
1. global health: 1
2. health systems: 1
3. systems faced: 1
4. faced significant: 1
5. significant strain: 1

Top 5 Trigrams:
1. global health systems: 1
2. health systems faced: 1
3. systems faced significant: 1
4. faced significant strain: 1
5. significant strain recent: 1

Total bigrams: 129
Unique bigrams: 129
Total trigrams: 128
Unique trigrams: 128

Document 3: Climate Change & Sustainability
--------------------------------------------------
Top 5 Bigrams:
1. climate change: 2
2. change remains: 1
3. remains one: 1
4. one biggest: 1
5. biggest challenges: 1

Top 5 Trigrams:
1. climate change remains: 1
2. change remains one: 1
3. remains one biggest: 1
4. one biggest challenges: 1
5. biggest challenges facing: 1
```

```
Total bigrams: 145
Unique bigrams: 144
Total trigrams: 144
Unique trigrams: 144
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/swathishreebv/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/swathishreebv/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Interpretation :

Interpretation:

The N-gram analysis identifies the most common word pairs (bigrams) and three-word sequences (trigrams) in each document.

- The Technology & Innovation document shows a connected narrative starting with "technology continues" and flowing through "continues redefine", "redefine work", "work communicate", and "communicate learn". This creates a logical sequence showing how technology continuously reshapes our work and communication methods.
- The Global Health & Pandemics document follows a similar pattern with "global health" connecting to "health systems", then "systems faced", "faced significant", and "significant strain". This sequence tells the story of how global health systems encountered major challenges during recent crises.
- The Climate Change & Sustainability document stands out because "climate change" appears as a bigram twice, making it the most frequent word pair. This repetition emphasizes the central importance of climate change in the document. The trigrams show a clear progression: "climate change remains", "change remains one", "remains one biggest", describing climate change as one of the biggest challenges.

All three documents show high diversity in their word combinations - the Technology document has 125 total bigrams with all being unique, the Health document has 129 bigrams (all unique), and the Climate document has 145 bigrams with 144 being unique. This indicates that each document uses varied vocabulary without much repetition, suggesting rich and diverse content coverage within each topic area.

Wordcloud & Clustering

4) Aim :

- Visualize TF-IDF using word clouds.
- Use TF-IDF features to cluster similar documents using KMeans.

Code and Output :

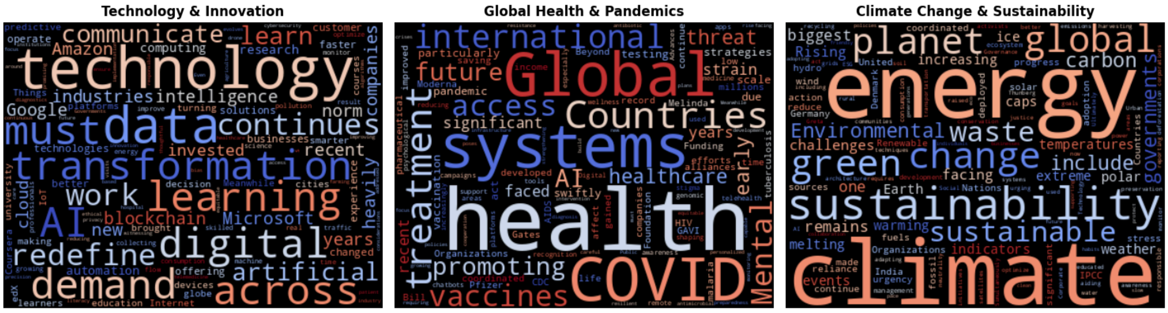
In [13]:

```
from wordcloud import WordCloud
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt

print("Word Cloud Generation:")

plt.figure(figsize=(15, 10))

for doc_index in range(len(documents)):
    doc_name = doc_names[doc_index]
    doc_text = documents[doc_index]

    wordcloud = WordCloud(width=400, height=300, background_color='black'

    plt.subplot(1, 3, doc_index + 1)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'{doc_name}', fontsize=12, fontweight='bold')

plt.tight_layout()
plt.show()

print("\nTF-IDF Word Clouds:")

plt.figure(figsize=(15, 10))

for doc_index in range(len(documents)):
    doc_name = doc_names[doc_index]

    tfidf_scores = tfidf_array[doc_index]
    word_freq = {}

    for term_index in range(len(feature_names)):
        term = feature_names[term_index]
        score = tfidf_scores[term_index]
        if score > 0:
            word_freq[term] = score

    wordcloud = WordCloud(width=400, height=300, background_color='black'

    plt.subplot(1, 3, doc_index + 1)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f'TF-IDF - {doc_name}', fontsize=12, fontweight='bold')

plt.tight_layout()
plt.show()

print("\nDocument Clustering using KMeans:")

kmeans = KMeans(n_clusters=2, random_state=42)
cluster_labels = kmeans.fit_predict(tfidf_array)

print("Cluster assignments:")
for doc_index in range(len(documents)):
    doc_name = doc_names[doc_index]
    cluster = cluster_labels[doc_index]
    print(f"{doc_name}: Cluster {cluster}")

cluster_centers = kmeans.cluster_centers_

print("\nTop terms for each cluster:")
```

```python
for cluster_index in range(len(cluster_centers)):
    print(f"\nCluster {cluster_index}:")
    center = cluster_centers[cluster_index]

    term_scores = []
    for term_index in range(len(feature_names)):
        term = feature_names[term_index]
        score = center[term_index]
        term_scores.append((term, score))

    for i in range(len(term_scores)):
        for j in range(len(term_scores) - 1):
            if term_scores[j][1] < term_scores[j + 1][1]:
                temp = term_scores[j]
                term_scores[j] = term_scores[j + 1]
                term_scores[j + 1] = temp

    for i in range(min(10, len(term_scores))):
        term, score = term_scores[i]
        if score > 0:
            print(f"  {i + 1}. {term}: {score:.4f}")

print("\nCluster Analysis:")
cluster_counts = {}
for cluster in cluster_labels:
    if cluster in cluster_counts:
        cluster_counts[cluster] = cluster_counts[cluster] + 1
    else:
        cluster_counts[cluster] = 1

for cluster in cluster_counts:
    count = cluster_counts[cluster]
    print(f"Cluster {cluster}: {count} documents")
```

```
Word Cloud Generation:
```

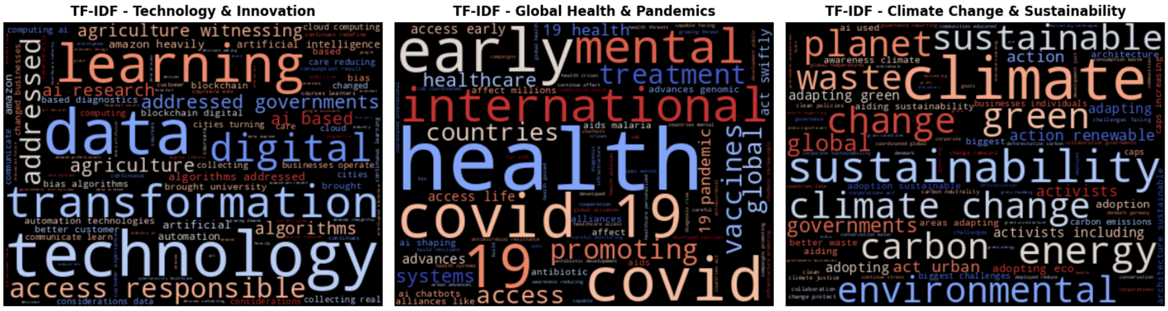


Technology & Innovation

Global Health & Pandemics

Climate Change & Sustainability

```
TF-IDF Word Clouds:
```



TF-IDF - Technology & Innovation

TF-IDF - Global Health & Pandemics

TF-IDF - Climate Change & Sustainability

```
Document Clustering using KMeans:
Cluster assignments:
Technology & Innovation: Cluster 0
Global Health & Pandemics: Cluster 0
Climate Change & Sustainability: Cluster 1

Top terms for each cluster:

Cluster 0:
  1. health: 0.1561
  2. technology: 0.1027
  3. data: 0.1013
  4. digital: 0.0751
  5. access: 0.0732
  6. healthcare: 0.0732
  7. learning: 0.0675
  8. transformation: 0.0675
  9. 19: 0.0624
  10. covid: 0.0624

Cluster 1:
  1. climate: 0.1852
  2. sustainability: 0.1852
  3. energy: 0.1408
  4. carbon: 0.1235
  5. change: 0.1235
  6. climate change: 0.1235
  7. environmental: 0.1235
  8. green: 0.1235
  9. planet: 0.1235
  10. sustainable: 0.1235

Cluster Analysis:
Cluster 0: 2 documents
Cluster 1: 1 documents
```

Interpretation :

The word cloud visualizations provide an immediate visual representation of the most prominent terms in each document. The regular word clouds show the most frequently used words, with larger text indicating higher frequency. The TF-IDF word clouds are more sophisticated, displaying terms sized according to their TF-IDF importance scores, which better highlight document-specific key terms rather than just common words.

The KMeans clustering analysis grouped the three documents into 2 clusters based on their TF-IDF feature similarities. Interestingly, the Technology & Innovation and Global Health & Pandemics documents were placed together in Cluster 0, while the Climate Change & Sustainability document forms its own Cluster 1. This clustering suggests that the technology and health documents share more vocabulary and thematic similarities with each other than either does with the climate document.

- Cluster 0 is characterized by a mix of technology and health terms. The top terms include "health" (0.1561), "technology" (0.1027), "data" (0.1013), "digital" (0.0751), "access" (0.0732), and "healthcare" (0.0732). This combination reflects how both documents discuss access to services, digital transformation, and data-driven approaches - the technology document talks about digital access and data, while the health document discusses healthcare access and health data systems.

- Cluster 1 is purely focused on environmental themes, with "climate" and "sustainability" both scoring 0.1852 as the top terms. Other prominent terms include "energy" (0.1408), "carbon" (0.1235), "change" (0.1235), and various environmental concepts. This cluster represents a distinct thematic area that doesn't overlap significantly with the other two documents.

The cluster distribution shows that 2 documents belong to Cluster 0 and 1 document to Cluster 1, indicating that the climate document is thematically quite different from the technology and health documents, which share some common vocabulary around access, systems, and digital approaches.