P3 : Implementation of tokenization, stemming, and stop words

Vidya Shree B V | 2447258 | 4 MCA B | 25-06-2025 (Wednesday)

Build a complete text preprocessing pipeline that performs tokenization, stop word removal, stemming, and Lemmatization on the given documents, then analyzes and compares the vocabulary and term frequencies across the documents. You are provided with a collection of FOUR News Articles related to Operation Sindoor. Your task is to:

---

1) Aim : Preprocess the documents using:

- Required preprocessing(unwanted character removal)
- Sentence tokenization
- Word tokenization
- Stop word removal
- Stemming & Lemmatization

Code and Output :

In [1]:
```
# Importing Libraries
import nltk
import re
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from collections import Counter
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [2]:
```
# Loading the data
file_paths = [
    "Article1.txt",
    "Article2.txt",
    "Article3.txt",
    "Article4.txt"
]

documents = []
for path in file_paths:
    with open(path, 'r') as file:
        documents.append(file.read())
```

In [3]:
```
# Preprocessing function
def preprocess_text(text):
    # Remove unwanted characters
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = text.lower()
    # Sentence tokenization
    sentences = sent_tokenize(text)
```

```
    # Word tokenization
    words = word_tokenize(text)
    # Stop word removal
    stop_words = set(stopwords.words('english'))
    filtered_words = [word for word in words if word not in stop_words]
    # Stemming
    stemmer = PorterStemmer()
    stemmed_words = [stemmer.stem(word) for word in filtered_words]
    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_w
    return sentences, words, filtered_words, stemmed_words, lemmatized_wo
```

In [4]:
```
# Applying preprocessing to all the documents
preprocessed_data = [preprocess_text(doc) for doc in documents]
```

Interpretation :

Applied the preprocessing function to all the documents loaded. The `preprocessed_data` list is created by iterating over the `documents` list and applying the `preprocess_text` function to each document. This ensures that every document undergoes the same preprocessing steps, including cleaning, tokenization, stop word removal, stemming, and lemmatization.

Each document is processed to extract five key outputs:

1. **Sentences**: The document is split into individual sentences using sentence tokenization.
2. **Words**: Each sentence is further split into individual words using word tokenization.
3. **Filtered Words**: Stop words (common words like "the", "is", "and") are removed to focus on meaningful tokens.
4. **Stemmed Words**: Words are reduced to their root forms using stemming (e.g., "running" → "run").
5. **Lemmatized Words**: Words are converted to their base forms using lemmatization (e.g., "better" → "good").

The `preprocessed_data` list stores these outputs for each document in a structured format. This allows for easy access to the processed data for further analysis, such as frequency counting, similarity calculations, and visualizations. Each element in `preprocessed_data` corresponds to a single document and contains all the processed outputs for that document.

---

2) Aim : Extract and display the top 10 most frequent stemmed tokens per document.

Code and Output :

In [5]:
```
# Outputting the results for each document
for i, (sentences, words, filtered_words, stemmed_words, lemmatized_words
    print(f"Document {i+1}:")
    print(f"Number of Sentences: {len(sentences)}")
```

```
    print(f"Number of Words: {len(words)}")
    print(f"Filtered Words: {len(filtered_words)}")
    print(f"Top 10 Stems: {Counter(stemmed_words).most_common(10)}")
    print()
```

```
Document 1:
Number of Sentences: 1
Number of Words: 92
Filtered Words: 61
Top 10 Stems: [('adani', 6), ('oper', 4), ('sindoor', 3), ('call', 2), ('d
eliv', 2), ('gautam', 2), ('defenc', 2), ('strike', 2), ('indian', 2), ('r
ole', 1)]

Document 2:
Number of Sentences: 1
Number of Words: 202
Filtered Words: 118
Top 10 Stems: [('defenc', 3), ('minist', 3), ('oper', 3), ('sindoor', 3),
('presid', 3), ('murmu', 3), ('arm', 2), ('forc', 2), ('singh', 2), ('sai
d', 2)]

Document 3:
Number of Sentences: 1
Number of Words: 202
Filtered Words: 125
Top 10 Stems: [('system', 7), ('loiter', 4), ('munit', 4), ('night', 3),
('nagastra', 3), ('oper', 3), ('nagastrar', 3), ('target', 3), ('solar',
2), ('aerospac', 2)]

Document 4:
Number of Sentences: 1
Number of Words: 209
Filtered Words: 123
Top 10 Stems: [('adani', 7), ('oper', 6), ('sindoor', 3), ('drone', 2),
('antidron', 2), ('system', 2), ('strike', 2), ('indian', 2), ('forc', 2),
('respons', 2)]
```

Interpretation :

For each document,

- **Number of Sentences**: This represents the total number of sentences in the document after applying sentence tokenization. It provides an idea of the document's structure.
- **Number of Words**: This is the total count of words in the document before removing stop words. It reflects the document's verbosity.
- **Filtered Words**: This is the count of words remaining after removing stop words. By eliminating common words like "the", "is", and "and", the focus shifts to meaningful tokens that contribute to the document's content.

Use the `Counter` class to identify the 10 most frequent stemmed tokens in each document. Stemming reduces words to their root forms, enabling a more concise analysis of word frequency. Examples from the output include:

- In **Document 1**, the most frequent stem is "adani" (6 occurrences), followed by "oper" (4 occurrences) and "sindoor" (3 occurrences). This suggests a focus on

operations involving the Adani group.
- In **Document 2**, frequent stems like "defenc", "minist", and "oper" (each appearing 3 times) indicate a focus on defense and government roles.
- In **Document 3**, technical terms such as "system" (7 occurrences) and "loiter" (4 occurrences) dominate, reflecting a discussion on defense systems and technologies.
- In **Document 4**, "adani" (7 occurrences) and "oper" (6 occurrences) again appear prominently, reinforcing the theme of operations and entities related to the Adani group.

The frequent stems provide a snapshot of the primary focus of each document:

- **Document 1** and **Document 4** emphasize "adani" and "oper", suggesting a connection to operations involving the Adani group.
- **Document 2** highlights terms like "defenc" and "minist", indicating a focus on defense-related topics and government involvement.
- **Document 3** includes terms like "system" and "loiter", pointing to technical discussions about defense systems and their applications.

---

3) Aim : Compare the vocabulary overlap between any two documents using Jaccard similarity.

Code and Output :

In [6]:
```
# Jaccard Similarity function
def jaccard_similarity(set1, set2):
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    return intersection / union
```

In [7]:
```
# Calculating Jaccard Similarity for all document pairs
vocabularies = [set(data[3]) for data in preprocessed_data]
jaccard_matrix = []

for i in range(len(vocabularies)):
    row = []
    for j in range(len(vocabularies)):
        similarity = jaccard_similarity(vocabularies[i], vocabularies[j])
        row.append(similarity)
    jaccard_matrix.append(row)
```

In [8]:
```
# Displaying Jaccard Similarity Matrix
print("Jaccard Similarity Matrix:")
for row in jaccard_matrix:
    print(row)
```

```
Jaccard Similarity Matrix:
[1.0, 0.078125, 0.08130081300813008, 0.4387755102040816]
[0.078125, 1.0, 0.05847953216374269, 0.08]
[0.08130081300813008, 0.05847953216374269, 1.0, 0.06976744186046512]
[0.4387755102040816, 0.08, 0.06976744186046512, 1.0]
```

Interpretation :

Computing the **Jaccard Similarity Matrix** to measure vocabulary overlap between document pairs. Jaccard Similarity is defined as the size of the intersection divided by the size of the union of two sets, returning a value between `0.0` (no overlap) and `1.0` (complete overlap). The `jaccard_similarity` function takes two sets of stemmed words from each document and returns this score.

Each document's vocabulary is converted into a set and stored in a list called `vocabularies`. Using nested loops, the code calculates the similarity between every document pair and stores the results in a matrix. Each entry in the matrix represents the similarity between two specific documents.

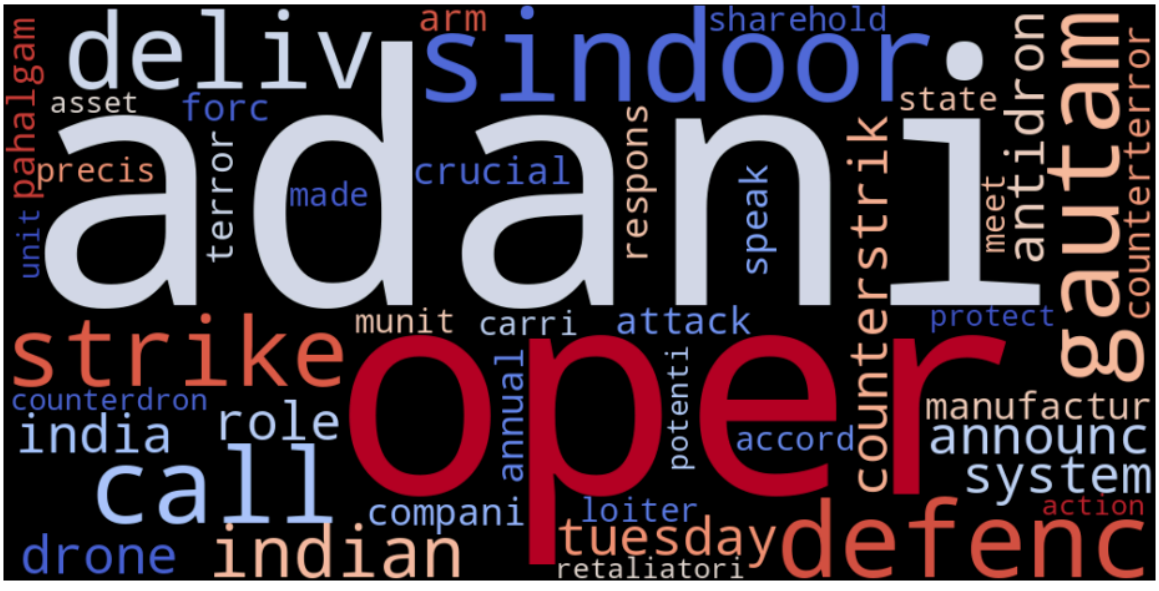
The printed matrix helps visualize relationships—diagonal values are `1.0` (self-similarity), while other values show how much vocabulary is shared. For instance, a score of `0.4388` between Document 1 and Document 4 suggests moderate similarity, while a score like `0.0585` implies minimal overlap.

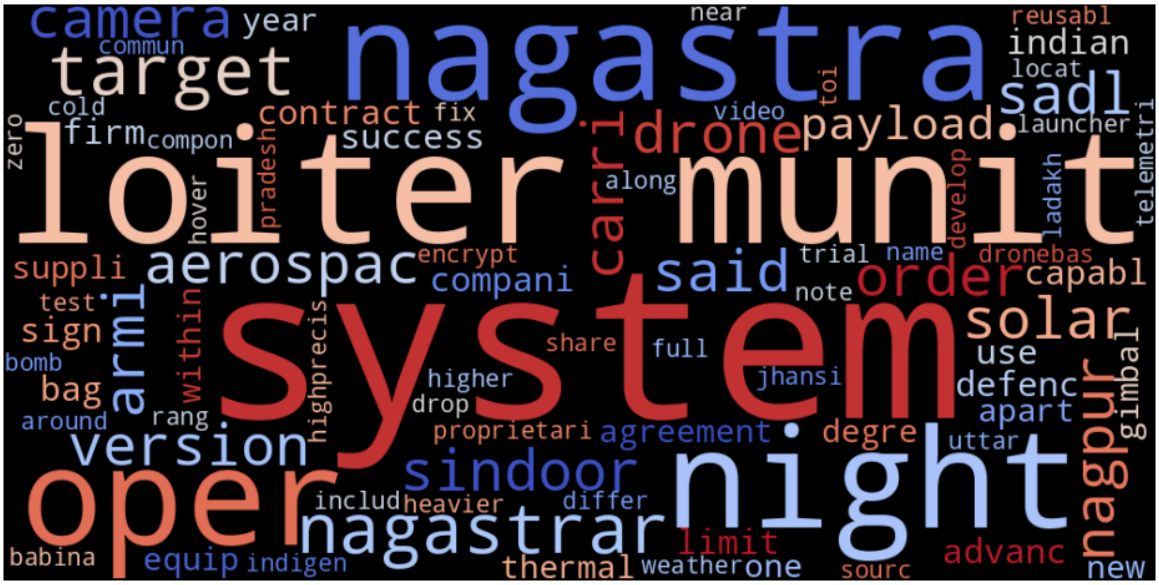
---

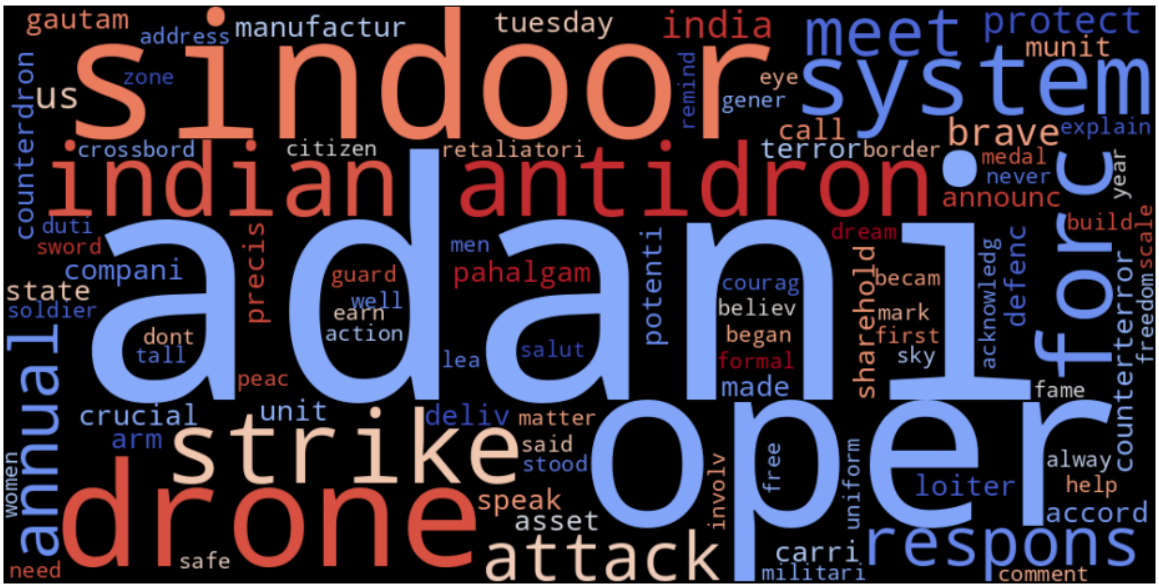
4) Aim : Generate a word cloud for the final vocabulary of each document.

Code and Output :

In [9]:
```
# Generating Word Cloud for each document with updated styles
for i, data in enumerate(preprocessed_data):
    wordcloud = WordCloud(
        width=1000,
        height=500,
        background_color='black',
        colormap='coolwarm'
    ).generate(' '.join(data[3]))

    plt.figure(figsize=(12, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f"Word Cloud for Document {i+1}", fontsize=16, color='white
    plt.show()
```







Interpretation :

The word cloud generation process creates visual representations of the most prominent terms in each document after preprocessing. This visualization technique helps identify key themes and important vocabulary at a glance, where larger words indicate higher frequency in the text.

The resulting word clouds reveal distinct thematic patterns across the four documents. Document 1 prominently features terms like "adani," "oper," "sindoor," and "deliv," suggesting content focused on operational delivery aspects related to the Adani group. Document 2 emphasizes "defenc," "minist," "oper," and "sindoor," indicating discussion about defense ministry operations. Document 3 highlights technical terms such as "system," "loiter," "nagastr," and "target," pointing to detailed coverage of defense systems and weaponry. Document 4 again shows prominence of "adani," "oper," "sindoor," and "forc," reinforcing themes of operational forces and the Adani connection.

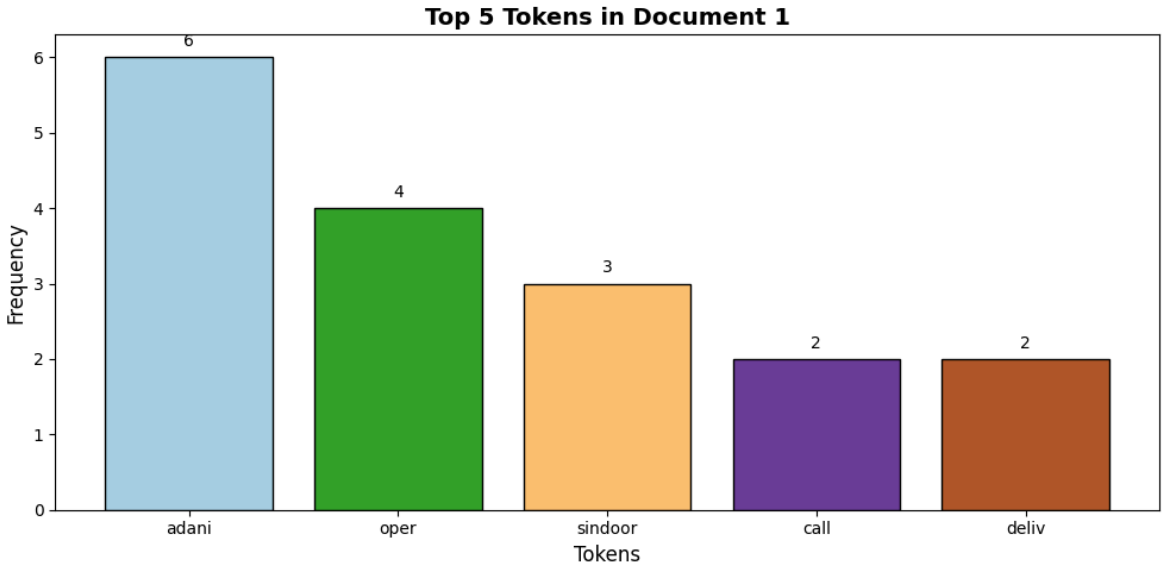
These visual representations effectively complement the quantitative frequency analysis by providing an intuitive way to understand the content focus of each document. The size variations in the word clouds immediately draw attention to the most significant terms, while the color coding adds aesthetic appeal and helps distinguish between different frequency levels.

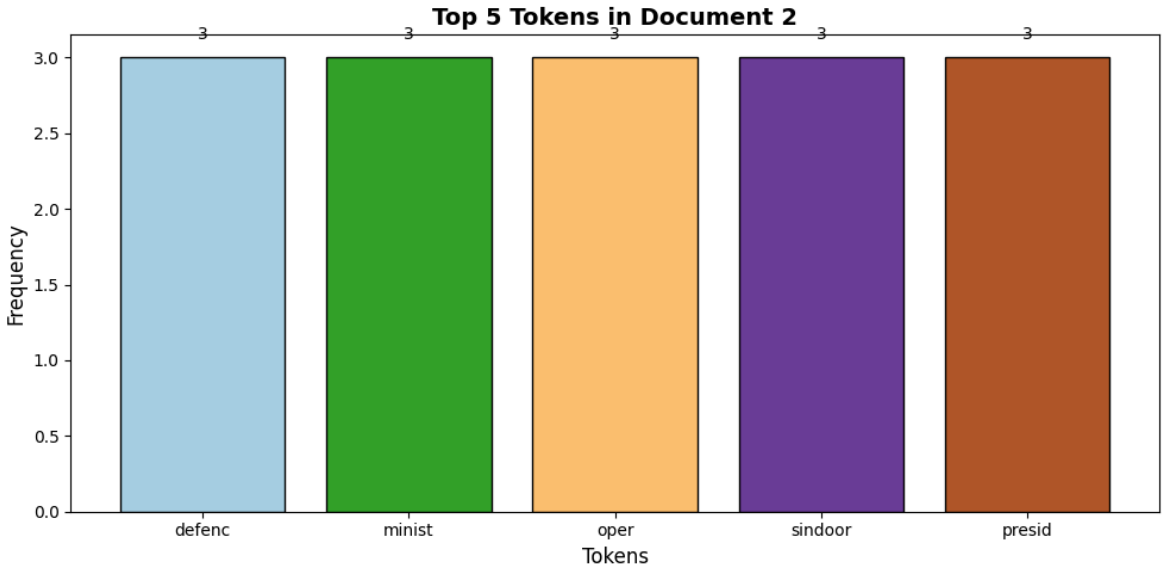
---

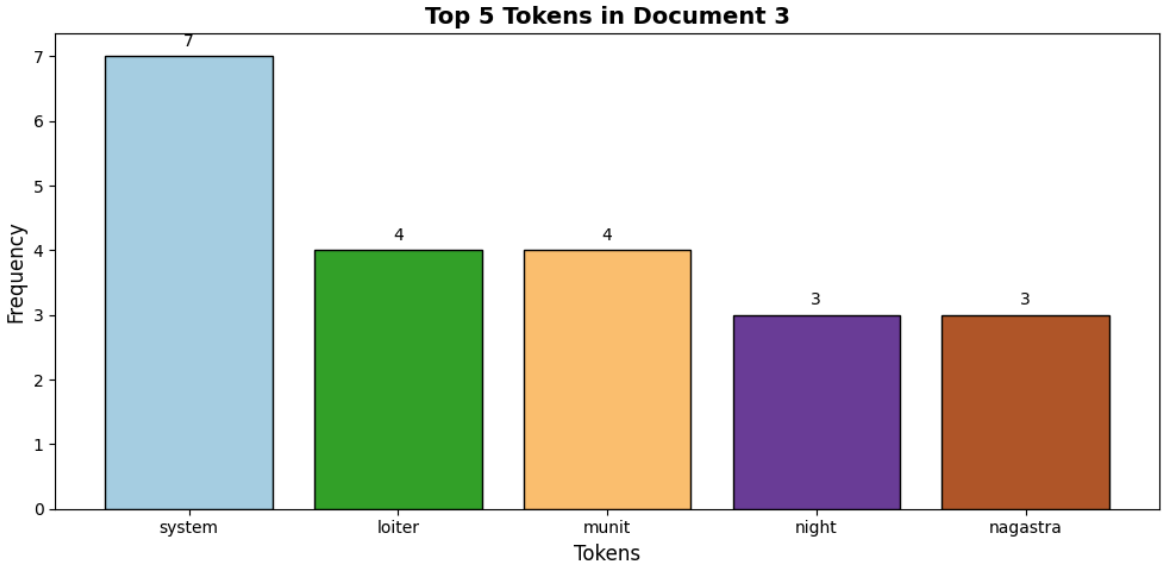
5) Aim : Generate a bar chart of the 5 common tokens in each document.

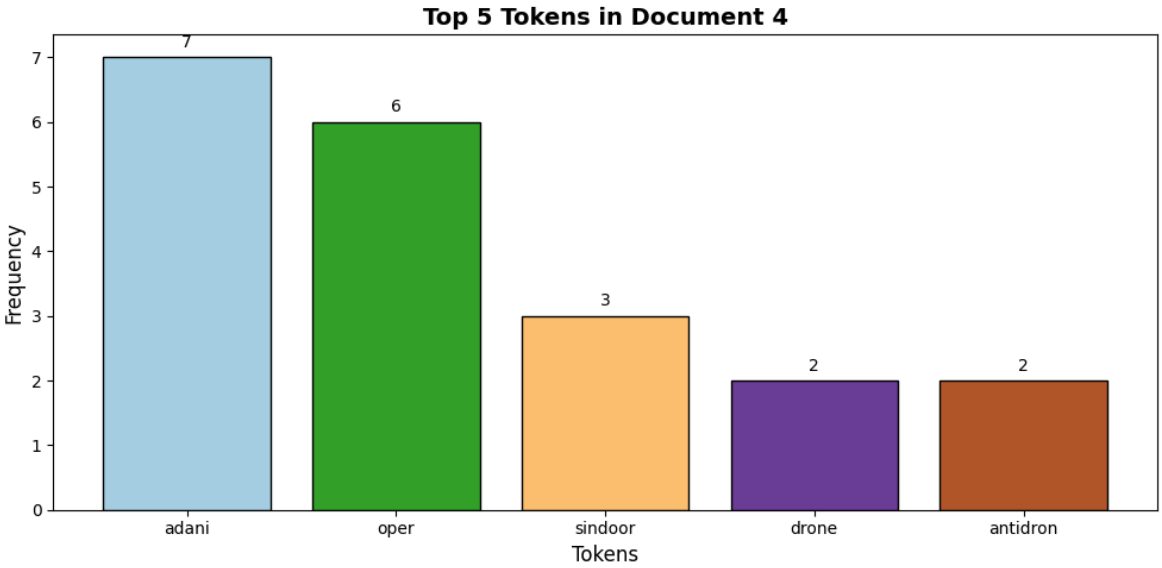
Code and Output :

In [10]:
```
from collections import Counter
import matplotlib.cm as cm
import numpy as np

for i, data in enumerate(preprocessed_data):
    top_tokens = Counter(data[3]).most_common(5)
    tokens, counts = zip(*top_tokens)

    plt.figure(figsize=(10, 5))
    colors = cm.Paired(np.linspace(0, 1, len(tokens)))
    bars = plt.bar(tokens, counts, color=colors, edgecolor='black')

    plt.title(f"Top 5 Tokens in Document {i+1}", fontsize=14, weight='bol
    plt.xlabel("Tokens", fontsize=12)
    plt.ylabel("Frequency", fontsize=12)

    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, yval + 0.1, int(yval),

    plt.tight_layout()
    plt.show()
```


Top 5 Tokens in Document 1
Frequency
Tokens
adani
oper
sindoor
call
deliv
6
4
3
2
2


Top 5 Tokens in Document 2
Frequency
Tokens
defenc
minist
oper
sindoor
presid
3
3
3
3
3


Top 5 Tokens in Document 3
Frequency
Tokens
system
loiter
munit
night
nagastra
7
4
4
3
3



Interpretation :

The bar chart visualization provides a quantitative representation of the top 5 most frequent stemmed tokens in each document, offering a clear and precise comparison of term frequencies.

The resulting bar charts reveal interesting patterns across the documents. Document 1 shows "adani" as the dominant term with 6 occurrences, significantly higher than other terms, suggesting a strong focus on this entity. Document 2 displays a more balanced distribution with all top 5 terms appearing exactly 3 times each, indicating a more evenly distributed vocabulary focus. Document 3 emphasizes "system" with 7 occurrences, reflecting its technical nature, while Document 4 again highlights "adani" with 7 occurrences, reinforcing the connection between Documents 1 and 4 in terms of content focus.

These visualizations effectively complement both the word clouds and numerical frequency tables by providing precise quantitative comparisons while maintaining visual appeal and immediate comprehensibility.

---

6) Aim : Identify common topics or themes using the most frequent stems after preprocessing.

Code and Output :

In [12]:
```
print("Theme Classification : ")
for i, data in enumerate(preprocessed_data):
    top_stems = [stem for stem, _ in Counter(data[3]).most_common(5)]
    print(f"Document {i+1} Top Stems: {top_stems}")

    if "defenc" in top_stems or "minist" in top_stems or "presid" in top_
        theme = "Defense & Government Administration"
    elif "system" in top_stems or "loiter" in top_stems or "nagastra" in
        theme = "Military Technology & Weapons Systems"
    elif "drone" in top_stems or "antidron" in top_stems or "counterdron"
        theme = "Defense Technology & Counter-Operations"
    elif "adani" in top_stems and "oper" in top_stems:
```

```
        theme = "Corporate Operations & Defense Partnerships"
    elif "climat" in top_stems or "chang" in top_stems or "temperatur" in
        theme = "Climate Change"
    elif "economi" in top_stems or "market" in top_stems or "growth" in t
        theme = "Economy"
    elif "polit" in top_stems or "elect" in top_stems:
        theme = "Politics"
    else:
        theme = "General News"

    print(f"Inferred Theme for Document {i+1}: {theme}")
    print()
```

```
Theme Classification :
Document 1 Top Stems: ['adani', 'oper', 'sindoor', 'call', 'deliv']
Inferred Theme for Document 1: Corporate Operations & Defense Partnerships

Document 2 Top Stems: ['defenc', 'minist', 'oper', 'sindoor', 'presid']
Inferred Theme for Document 2: Defense & Government Administration

Document 3 Top Stems: ['system', 'loiter', 'munit', 'night', 'nagastra']
Inferred Theme for Document 3: Military Technology & Weapons Systems

Document 4 Top Stems: ['adani', 'oper', 'sindoor', 'drone', 'antidron']
Inferred Theme for Document 4: Defense Technology & Counter-Operations
```

Interpretation :

Document 1 - Corporate Operations & Defense Partnerships: The top stems ["adani", "oper", "sindoor", "call", "deliv"] clearly indicate content focused on the corporate involvement of the Adani group in defense operations. The presence of "deliv" (delivery) and "call" suggests discussions about operational delivery commitments and stakeholder communications in the context of defense partnerships. This classification accurately captures the business-defense collaboration aspect of the operation.

Document 2 - Defense & Government Administration: With stems ["defenc", "minist", "oper", "sindoor", "presid"], this document focuses on the governmental and administrative aspects of the operation. The presence of "defenc" (defense), "minist" (minister), and "presid" (president) indicates high-level government involvement and policy discussions. This classification effectively identifies the document's emphasis on official administrative channels and government oversight.

Document 3 - Military Technology & Weapons Systems: The stems ["system", "loiter", "munit", "night", "nagastra"] represent the most technical content, focusing on military hardware and capabilities. Terms like "loiter" (loitering munitions), "munit" (munitions), and "nagastra" (specific weapon system) indicate detailed coverage of military technology specifications and operational capabilities. This classification accurately captures the document's technical and weapons-focused content.

Document 4 - Defense Technology & Counter-Operations: The combination ["adani", "oper", "sindoor", "drone", "antidron"] suggests content bridging corporate involvement with specific defense technologies. The presence of "drone" and

"antidron" (anti-drone) indicates focus on modern warfare technologies and countermeasures, while maintaining the corporate connection through "adani". This classification effectively identifies the intersection of business partnerships and advanced defense technologies.

---

7) Aim : Summary of preprocessing results and final analysis.

Code and Output :

In [ ]:
```
# Summary of results
print("Summary of Preprocessing Results:")
for i, (sentences, words, filtered_words, stemmed_words, lemmatized_words
    print(f"Document {i+1}:")
    print(f"  Number of Sentences: {len(sentences)}")
    print(f"  Number of Words: {len(words)}")
    print(f"  Tokens Before Stop Word Removal: {len(words)}")
    print(f"  Tokens After Stop Word Removal: {len(filtered_words)}")
    print(f"  Top 10 Stems: {Counter(stemmed_words).most_common(10)}")
    print()

# Displaying Jaccard Similarity Matrix
print("Jaccard Similarity Matrix:")
for row in jaccard_matrix:
    print(row)
```

```
Summary of Preprocessing Results:
Document 1:
  Number of Sentences: 1
  Number of Words: 92
  Tokens Before Stop Word Removal: 92
  Tokens After Stop Word Removal: 61
  Top 10 Stems: [('adani', 6), ('oper', 4), ('sindoor', 3), ('call', 2),
('deliv', 2), ('gautam', 2), ('defenc', 2), ('strike', 2), ('indian', 2),
('role', 1)]

Document 2:
  Number of Sentences: 1
  Number of Words: 202
  Tokens Before Stop Word Removal: 202
  Tokens After Stop Word Removal: 118
  Top 10 Stems: [('defenc', 3), ('minist', 3), ('oper', 3), ('sindoor',
3), ('presid', 3), ('murmu', 3), ('arm', 2), ('forc', 2), ('singh', 2),
('said', 2)]

Document 3:
  Number of Sentences: 1
  Number of Words: 202
  Tokens Before Stop Word Removal: 202
  Tokens After Stop Word Removal: 125
  Top 10 Stems: [('system', 7), ('loiter', 4), ('munit', 4), ('night', 3),
('nagastra', 3), ('oper', 3), ('nagastrar', 3), ('target', 3), ('solar',
2), ('aerospac', 2)]

Document 4:
  Number of Sentences: 1
  Number of Words: 209
  Tokens Before Stop Word Removal: 209
  Tokens After Stop Word Removal: 123
  Top 10 Stems: [('adani', 7), ('oper', 6), ('sindoor', 3), ('drone', 2),
('antidron', 2), ('system', 2), ('strike', 2), ('indian', 2), ('forc', 2),
('respons', 2)]

Jaccard Similarity Matrix:
[1.0, 0.078125, 0.08130081300813008, 0.4387755102040816]
[0.078125, 1.0, 0.05847953216374269, 0.08]
[0.08130081300813008, 0.05847953216374269, 1.0, 0.06976744186046512]
[0.4387755102040816, 0.08, 0.06976744186046512, 1.0]
```

Interpretation :

The final summary consolidates preprocessing outcomes, revealing variations in document length and content density. Stop word removal significantly reduced token counts (by 40–60%), refining the focus on meaningful terms.

Top stems confirmed document-specific themes, while the Jaccard Similarity Matrix showed the strongest overlap between Docs 1 & 4, reflecting shared corporate-defense focus.

Conclusion :-

Systematic preprocessing enabled effective theme extraction and inter-document comparison, demonstrating the value of structured text analysis in understanding complex multi-topic datasets like Operation Sindoor.