PERFORMING BAG OF WORDS :

Step 1: Create a file which contains the data.

Step 2: Load the file into the colab environment for performing operations.

```
# Step 2: Read the content of nlp.text
file_path = '/content/nlp.txt' # Make sure 'nlp.text' is uploaded to your Colab environment
try:
    with open(file_path, 'r', encoding='utf-8') as file:
        text = file.read()
    print("File content loaded successfully!")
    print("\nOriginal Text (first 500 characters):\n", text[:500])
except FileNotFoundError:
    print(f"Error: The file '{file_path}' was not found.\nPlease ensure it's uploaded to your Colab env
    text = "This is a sample text for demonstration purposes. It contains punctuation, different casing
    print("\nUsing sample text for demonstration:\n", text)
```

```
File content loaded successfully!

Original Text (first 500 characters):
 In Natural Language Processing (NLP) text data needs to be converted into numbers so that machine learn
One common method to do this is Bag of Words (BoW) model.
It turns text like sentence, paragraph or document into a collection of words and counts how often each
It does not consider the order of the words or their grammar but focuses on counting how often each word
This makes it useful for
```

Step 3: Perform Text Normalization

We will use the `nltk` library for various normalization techniques. First, let's install it and download necessary data.

```
# Install NLTK if you haven't already
!pip install nltk

# Download necessary NLTK data
import nltk
try:
    nltk.data.find('tokenizers/punkt')
except LookupError: # Changed from nltk.downloader.DownloadError
    nltk.download('punkt')
try:
    nltk.data.find('corpora/stopwords')
except LookupError: # Changed from nltk.downloader.DownloadError
    nltk.download('stopwords')
try:
    nltk.data.find('corpora/wordnet')
except LookupError: # Changed from nltk.downloader.DownloadError
    nltk.download('wordnet')
try:
    nltk.data.find('tokenizers/punkt_tab') # Added download for punkt_tab as suggested by the error
except LookupError:
    nltk.download('punkt_tab')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.3)
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.tokenize import word_tokenize

# 1. Convert to Lowercase
normalized_text = text.lower()
print("\n1. Lowercased Text (first 500 characters):\n", normalized_text[:500])

# 2. Remove Punctuation and Numbers
# This regex removes anything that is not a letter or whitespace
normalized_text = re.sub(r'[^a-z\s]', '', normalized_text)
print("\n2. Text after removing punctuation and numbers (first 500 characters):\n", normalized_text[:500

# 3. Tokenization
tokens = word_tokenize(normalized_text)
print("\n3. Tokens (first 50 tokens):\n", tokens[:50])

# 4. Remove Stopwords
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word not in stop_words]
print("\n4. Tokens after removing stopwords (first 50 tokens):\n", filtered_tokens[:50])

# 5. Stemming (using Porter Stemmer)
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]
print("\n5. Stemmed Tokens (first 50 tokens):\n", stemmed_tokens[:50])

# 6. Lemmatization (using WordNet Lemmatizer)
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]
print("\n6. Lemmatized Tokens (first 50 tokens):\n", lemmatized_tokens[:50])

# You can choose to convert the tokens back to a string if needed
final_normalized_text = ' '.join(lemmatized_tokens)
print("\nFinal Normalized Text (first 500 characters, after lemmatization):\n", final_normalized_text[:5
```

```
1. Lowercased Text (first 500 characters):
 in natural language processing (nlp) text data needs to be converted into numbers so that machine learn
one common method to do this is bag of words (bow) model.
it turns text like sentence, paragraph or document into a collection of words and counts how often each
it does not consider the order of the words or their grammar but focuses on counting how often each word
this makes it useful for

2. Text after removing punctuation and numbers (first 500 characters):
 in natural language processing nlp text data needs to be converted into numbers so that machine learnin
one common method to do this is bag of words bow model
it turns text like sentence paragraph or document into a collection of words and counts how often each w
it does not consider the order of the words or their grammar but focuses on counting how often each word
this makes it useful for tasks li

3. Tokens (first 50 tokens):
 ['in', 'natural', 'language', 'processing', 'nlp', 'text', 'data', 'needs', 'to', 'be', 'converted', 'i

4. Tokens after removing stopwords (first 50 tokens):
 ['natural', 'language', 'processing', 'nlp', 'text', 'data', 'needs', 'converted', 'numbers', 'machine'
```

```
5. Stemmed Tokens (first 50 tokens):
 ['natur', 'languag', 'process', 'nlp', 'text', 'data', 'need', 'convert', 'number', 'machin', 'learn',

6. Lemmatized Tokens (first 50 tokens):
 ['natural', 'language', 'processing', 'nlp', 'text', 'data', 'need', 'converted', 'number', 'machine',

Final Normalized Text (first 500 characters, after lemmatization):
 natural language processing nlp text data need converted number machine learning algorithm understand o
```

Step 4: Save the Normalized Text to a File

```
# Define the output file path
output_file_path = '/content/normalized_nlp.txt'

# Prepare the content to be written to the file
output_content = f"""
Original Text (first 500 characters):\n{text[:500]}\n\n1. Lowercased Text (first 500 characters):\n{nor
"""

# Write the structured content to the file
with open(output_file_path, 'w', encoding='utf-8') as outfile:
    outfile.write(output_content)

print(f"Normalized text (with step-by-step breakdown) successfully saved to '{output_file_path}'")
```

```
Normalized text (with step-by-step breakdown) successfully saved to '/content/normalized_nlp.txt'
```

Step 5: Perform Bag of Words (BoW) on the Final Normalized Text

```
from sklearn.feature_extraction.text import CountVectorizer
import pandas as pd

# Initialize CountVectorizer
# We convert the single string into a list containing that string
vectorizer = CountVectorizer()

# Fit and transform the final normalized text
bow_matrix = vectorizer.fit_transform([final_normalized_text])

# Get feature names (words) and their counts
feature_names = vectorizer.get_feature_names_out()
word_counts = bow_matrix.toarray()[0]

# Create a dictionary of word counts
word_frequency = dict(zip(feature_names, word_counts))

# Sort the words by frequency in descending order
sorted_word_frequency = sorted(word_frequency.items(), key=lambda x: x[1], reverse=True)

print("\nBag of Words (Top 20 most frequent words):\n")
# Convert the sorted list of tuples to a string for display and file saving
bow_output = "\n".join([f"{word}: {count}" for word, count in sorted_word_frequency[:20]])
print(bow_output)
```

```
Bag of Words (Top 20 most frequent words):

word: 10
text: 8
document: 4
```

```
vocabulary: 4
bagofwords: 3
counting: 3
like: 3
often: 3
order: 3
vector: 3
algorithm: 2
analysis: 2
appears: 2
bag: 2
bow: 2
classification: 2
collection: 2
data: 2
frequency: 2
grammar: 2
```

Step 6: Update the Output File with Bag of Words Results

```
# Define the output file path
output_file_path = '/content/normalized_nlp.txt'

# Prepare the content to be written to the file
output_content = f"""
Original Text (first 500 characters):\n{text[:500]}\n\n1. Lowercased Text (first 500 characters):\n{re.
"""

# Write the structured content to the file
with open(output_file_path, 'w', encoding='utf-8') as outfile:
    outfile.write(output_content)

print(f"Normalized text and Bag of Words results successfully saved to '{output_file_path}'")
```

```
Normalized text and Bag of Words results successfully saved to '/content/normalized_nlp.txt'
```

INPUT DATA FILE NAMED: nlp.txt

In Natural Language Processing (NLP) text data needs to be converted into numbers so that machine learning algorithms can understand it. One common method to do this is Bag of Words (BoW) model. It turns text like sentence, paragraph or document into a collection of words and counts how often each word appears but ignoring the order of the words. It does not consider the order of the words or their grammar but focuses on counting how often each word appears in the text. This makes it useful for tasks like text classification, sentiment analysis and clustering. The Bag-of-Words (BoW) model is a foundational Natural Language Processing (NLP) technique that represents text data as an unordered collection of words, focusing solely on frequency rather than grammar, order, or context. By tokenizing text and counting token occurrences, it creates numerical vectors, enabling machine learning algorithms to analyze text for tasks like sentiment analysis, classification, and information retrieval. Key Concepts of Bag-of-Words: The Bag-of-Words approach operates by simplifying text into a "bag" of its constituent words, which is inspired by word-frequency counting methods. Vocabulary Building: The first step involves creating a unique vocabulary from the entire dataset, where each unique word corresponds to a feature or dimension. Vectorization: Each document is transformed into a fixed-length numerical vector, where each entry represents the frequency of a vocabulary word appearing in that document. Handling Sparsity: Because most documents only contain a small fraction of the total vocabulary, the resulting vectors are often "sparse" (filled with zeros).

The bag of words performed on the first 500 words of the input text.

FINAL OUTPUT FILE AFTER NORMALIZATION AND PERFORMING BAG OF WORDS :

Original Text (first 500 characters): In Natural Language Processing (NLP) text data needs to be converted into numbers so that machine learning algorithms can understand it. One common method to do this is Bag of Words (BoW) model. It turns text like sentence, paragraph or document into a collection of words and counts how often each word appears but ignoring the order of the words. It does not consider the order of the words or their grammar but focuses on counting how often each word appears in the text.

1. Lowercased Text (first 500 characters):
innaturallanguageprocessingnlptextdataneedstobeconvertedintonumberssothatmachinelearningalgorithmscanund erstanditonecommonmethodtodothisisbagofwordsbowmodelitturnstextlikesentenceparagraphordocumentintoacol lectionofwordsandcountshowofteneachwordappearsbutignoringtheorderofthewordsitdoesnotconsidertheorderoft hewordsortheirgrammarbutfocusesoncountinghowofteneachwordappearsinthetextthismakesitusefulfortaskslike tex tclassificationsentimentanalysisandclusteringthebagofwordsbowmodelisafoundationalnatur

2. Text after removing punctuation and numbers (first 500 characters):
innaturallanguageprocessingnlptextdataneedstobeconvertedintonumberssothatmachinelearningalgorithmscanund erstanditonecommonmethodtodothisisbagofwordsbowmodelitturnstextlikesentenceparagraphordocumentintoacol lectionofwordsandcountshowofteneachwordappearsbutignoringtheorderofthewordsitdoesnotconsidertheorderoft hewordsortheirgrammarbutfocusesoncountinghowofteneachwordappearsinthetextthismakesitusefulfortaskslike tex tclassificationsentimentanalysisandclusteringthebagofwordsbowmodelisafoundationalnatur

3. Tokens after removing stopwords (first 50 tokens): natural language processing nlp text data needs converted numbers machine learning algorithms understand one common method bag words bow model turns text like sentence paragraph document collection words counts often word appears ignoring order words consider order words grammar focuses counting often word appears text makes useful tasks like text

4. Stemmed Tokens (first 50 tokens): natur languag process nlp text data need convert number machin learn algorithm understand one common method bag word bow model turn text like sentenc paragraph document collect word count often word appear ignor order word consid order word grammar focus count often word appear text make use task like text

5. Lemmatized Tokens (first 50 tokens): natural language processing nlp text data need converted number machine learning algorithm understand one common method bag word bow model turn text like sentence paragraph document collection word count often word appears ignoring order word consider order word grammar focus counting often word appears text make useful task like text

Full Final Normalized Text (after lemmatization): natural language processing nlp text data need converted number machine learning algorithm understand one common method bag word bow model turn text like sentence paragraph document collection word count often word appears ignoring order word consider order word grammar focus counting often word appears text make useful task like text classification sentiment analysis clustering bagofwords bow model foundational natural language processing nlp technique represents text data unordered collection word focusing solely frequency rather grammar order context tokenizing text counting token occurrence creates numerical vector enabling machine learning algorithm analyze text task like sentiment analysis classification information retrieval key concept bagofwords bagofwords approach operates simplifying text bag constituent word inspired wordfrequency counting method vocabulary building first step involves creating unique

```
vocabulary entire dataset unique word corresponds feature dimension vectorization document transformed fixedlength numerical vector entry represents frequency vocabulary word appearing document handling sparsity document contain small fraction total vocabulary resulting vector often sparse filled zero

--- Bag of Words Results (Top 20 Most Frequent Words) --- word: 10
text: 8
document: 4
vocabulary: 4
bagofwords: 3
counting: 3
like: 3
often: 3
order: 3
vector: 3
algorithm: 2
analysis: 2
appears: 2
bag: 2
bow: 2
classification: 2
collection: 2
data: 2
frequency: 2
grammar: 2
```