NLP Preprocessing Steps

Natural Language Processing (NLP) preprocessing steps are essential for preparing textual data for machine learning models such as LSTM and other deep learning architectures.

Text Cleaning

Text cleaning involves removing unwanted elements from the text such as punctuation, numbers, special characters, and stopwords. It also includes converting text to lowercase to ensure uniformity.

Example code:

```
import re

def clean_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove non-alphabetic characters
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text


text = "I loved the movie! It was amazing. Visit https://example.com
for more."
cleaned_text = clean_text(text)
print(cleaned_text)
```

Output:

```
i loved the movie it was amazing
```

Tokenization

Tokenization is the process of splitting the text into individual words (tokens). This is a crucial step because the model needs numerical input, and tokenization prepares the text for further numerical encoding.

Example code:

```
from nltk.tokenize import word_tokenize

import nltk

nltk.download('punkt')


# Example text

text = "I loved the movie, it was fantastic!"

# Tokenizing the text

tokens = word_tokenize(text)

print(tokens)
```

Output:

['I', 'loved', 'the', 'movie', ',', 'it', 'was', 'fantastic', '!']

Stopword Removal

Stopwords are common words (e.g., "the", "is", "in") that may not add significant meaning to the text. Removing them helps in reducing the noise in the data.

Example code:

```
from nltk.corpus import stopwords

nltk.download('stopwords')

# Define stopwords

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

```
# Remove stopwords from tokens
filtered_tokens = [word for word in tokens if word not in
stop_words]
print(filtered_tokens)
```

Output:

```
['I', 'loved', 'movie', ',', 'fantastic', '!']
```

Lemmatization

Lemmatization reduces words to their base or root form (lemma) while ensuring that the reduced form is a valid word. Unlike stemming, lemmatization accounts for the context and meaning of the word.

Example code:

```
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
# Example words
words = ["running", "happier", "studies"]
lemmatized_words = [lemmatizer.lemmatize(word, pos='v') for word in
words]
print(lemmatized_words)
```

Output:

['run', 'happy', 'study']

Padding and Truncating Sequences

To ensure that all input sequences are of the same length, padding adds zeros to shorter sequences, and truncation cuts off longer sequences.

Example code:

```
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Example of tokenized sequences of varying lengths

sequences = [

    [1, 2, 3],

    [4, 5, 6, 7, 8],

    [9, 10]

]

# Pad the sequences to a fixed length (e.g., 5)

padded_sequences = pad_sequences(sequences, maxlen=5,

padding='post')

print(padded_sequences)
```

Output:

```
[[ 1  2  3  0  0]

 [ 4  5  6  7  8]

 [ 9 10  0  0  0]]
```

Word Embedding

Word embedding is a technique to convert words into dense vectors that capture semantic relationships between them. The `Embedding` layer in Keras is commonly used to convert words into vectors for deep learning models.

Example code:

```
from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense
```

```
# Define the model
model = Sequential()
model.add(Embedding(input_dim=10000, output_dim=100,
input_length=50))  # Embedding layer
model.add(LSTM(64))  # LSTM layer
model.add(Dense(1, activation='sigmoid'))  # Output layer for binary
classification
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
model.summary()
```

Visualizing Word Frequency

You can visualize the most frequent words in your dataset using bar plots or word clouds.

Example code for Word Frequency Bar Plot:

```
from collections import Counter
import matplotlib.pyplot as plt
import seaborn as sns

# Example word list
word_list = ['movie', 'great', 'loved', 'movie', 'boring', 'slow',
'great']
word_freq = Counter(word_list)

# Extract words and counts
```

```
words = [word[0] for word in word_freq.most_common(5)]
counts = [word[1] for word in word_freq.most_common(5)]

# Plot the most common words
plt.figure(figsize=(10, 6))
sns.barplot(x=counts, y=words)
plt.title('Most Common Words')
plt.show()
```