## Front End Engineering-II /Artificial

## Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

Instagram Captioning Model

A red and white sign

Description automatically generated with low confidence

**Supervised By: Submitted By:**

Faculty Name Ishika Gulati, 2210990423

**Department of Computer Science and Engineering**

## Chitkara University Institute of Engineering & Technology,

## Chitkara University, Punjab

**1. Introduction**

**Background**

Generation Z, or Gen Z, is the cohort of individuals born roughly between the mid-1990s and early 2010s, characterized by their digital nativity and unique cultural expressions. This generation's communication style, often dubbed "Gen Z language" or "Gen Z speak," is marked by its informality, creativity, and use of internet-inspired slang, memes, and references. Understanding and emulating this linguistic style is crucial for engaging with Gen Z audiences in various online platforms, including social media.

**Objectives and Scope**

The primary objective of this project is to develop an innovative model capable of generating captions in the distinctive Gen Z style for user-uploaded images. By harnessing state-of-the-art techniques in natural language processing (NLP) and deep learning, our model aims to capture the essence of Gen Z language and cultural references, providing users with personalized and engaging captions for their images. The scope of the project encompasses data collection, preprocessing, model architecture design, training, and deployment of the captioning model within a user-friendly interface.

**Overview of the Gen Z Style Captioning Model**

The proposed Gen Z style captioning model employs a combination of convolutional neural networks (CNNs) for image feature extraction and long short-term memory (LSTM) networks for text generation. CNNs are utilized to extract high-level visual features from input images, while LSTM networks serve as the language generation component, capturing the sequential dependencies and semantic context of the captions. Through the iterative generation process facilitated by LSTM, the model learns to produce captions that embody the linguistic nuances and cultural references inherent to Gen Z communication.

**Significance**

The significance of this project lies in its potential to revolutionize user engagement and content creation strategies, particularly in the realm of social media marketing and communication. By developing a model capable of generating captions in the distinctive Gen Z style, we address a critical gap in current automated captioning systems, which often fail to capture the informal and culturally rich nature of Gen Z communication.

Through personalized and engaging captions, our model enables brands, influencers, and content creators to connect more authentically with Gen Z audiences, fostering deeper engagement and loyalty. By leveraging state-of-the-art techniques in natural language processing and deep learning, we not only cater to the preferences of Gen Z users but also contribute to advancements in AI-driven content generation.

Furthermore, the insights gained from this project have broader implications for understanding the evolving dynamics of online communication and cultural expression. By studying and emulating the linguistic nuances of Generation Z, we gain valuable insights into the future trends of digital communication, informing the development of more inclusive and effective communication strategies across diverse demographics.

**2. Problem Definition and Requirements**

**Problem Definition:**

The core problem addressed in this project is the lack of automated captioning systems that effectively cater to the linguistic style and cultural references of Generation Z (Gen Z) on social media platforms, particularly Instagram. Current captioning models often fail to capture the informal, creative, and culturally rich nature of Gen Z communication, leading to disengagement and disconnect with this demographic. Therefore, the project aims to develop an innovative captioning model specifically tailored to generate captions in the distinctive Gen Z style for user-uploaded images on Instagram.

**Requirements:**

Gen Z Linguistic Style Understanding: The model should be capable of understanding and emulating the linguistic style of Generation Z, including the use of internet-inspired slang, memes, emojis, and pop culture references.

* **Personalization and Engagement**: Captions generated by the model should be personalized and engaging, resonating with Gen Z users and fostering deeper connection and interaction on social media.
* **Image-Text Alignment**: The captions generated should be contextually relevant to the content of the image, effectively conveying the visual information presented.
* **Ease of Use**: The captioning model should be integrated into a user-friendly interface, allowing users to easily upload images and receive Gen Z style captions within seconds.
* **Scalability and Performance:** The model should be scalable to accommodate large volumes of image data and perform efficiently in real-time caption generation scenarios.
* **Accuracy and Quality:** Captions generated by the model should exhibit high accuracy and quality, effectively capturing the essence of the image and delivering coherent and meaningful textual descriptions.
* **Ethical Considerations:** The model should adhere to ethical guidelines, ensuring that generated captions are respectful, inclusive, and free from offensive or inappropriate content.
* **Adaptability and Continual Improvement:** The model should be adaptable to evolving linguistic trends and cultural shifts within the Gen Z demographic, allowing for continual improvement and refinement over time.

**3. Data Collection**

**Source of Data:**

* The dataset utilized in this project was obtained from Kaggle, a platform hosting diverse datasets for research and analysis. It comprises two main components: the instagram\_data folder and the instagram\_data2 folder.
* The instagram\_data folder contains a substantial collection of image files, totalling approximately 20.5k images. These images serve as the visual content for which captions are generated.
* Additionally, the instagram\_data folder includes a CSV file named captions\_csv.csv, which contains corresponding captions for the images in the dataset. Each caption in the CSV file is associated with a specific image file, allowing for the pairing of textual descriptions with visual content.
* The instagram\_data2 folder complements the dataset with an additional set of image files under the img2 subfolder, totalling 14.4k images. The corresponding captions for these images are provided in the captions\_csv2.csv file within the same folder.

For access to the dataset, please visit the following link: <https://www.kaggle.com/datasets/prithvijaunjale/instagram-images-with-captions/data>

**4. Data Cleaning**

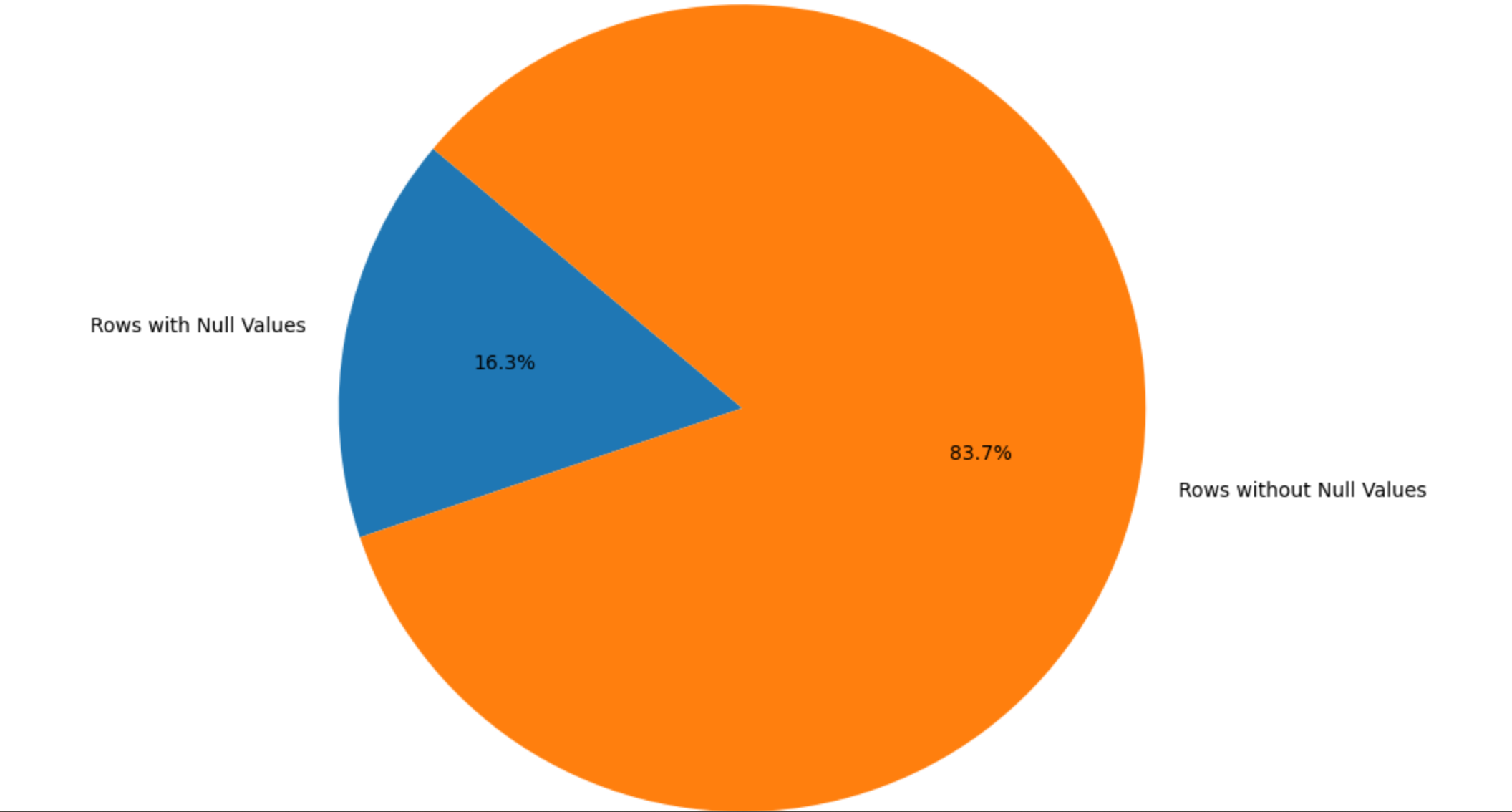
Data cleaning is a crucial preprocessing step aimed at enhancing the quality and consistency of the dataset for subsequent analysis and model training. The following steps outline the data cleaning process implemented in this project:

**Steps in Data Cleaning:**

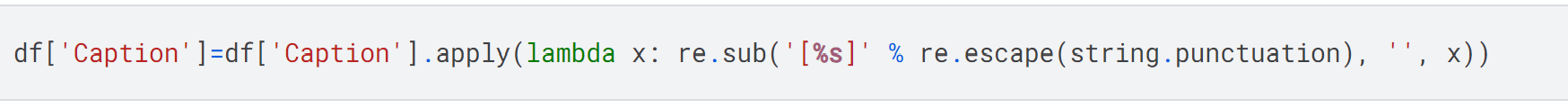
* **Lowercasing:** All textual data, including captions, are converted to lowercase to standardize the text and avoid redundancy caused by case variations.



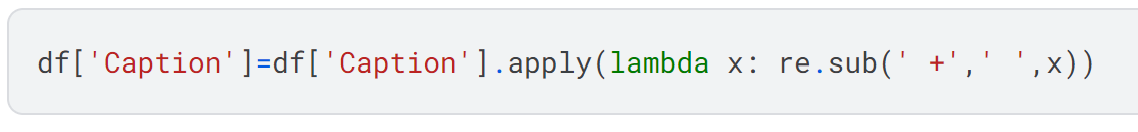
* **Handling Null Values**: Rows with missing or null values are removed from the dataset to maintain data integrity and prevent errors during processing.



* **Punctuation Removal:** Punctuation marks are removed from the textual data to focus solely on the content of the captions and eliminate noise from the dataset.



* **Whitespace Removal**: Extraneous whitespace characters are removed to standardize the formatting of the textual data and improve readability.



* **Tokenization**: Textual data is tokenized into individual words or tokens, enabling subsequent processing steps such as substitution and lemmatization.



* **Substitution of Chat Words :**In the data cleaning process, chat words and acronyms commonly used in informal online communication were replaced with their corresponding full forms to enhance the readability and clarity of the captions. This task was accomplished by defining a dictionary named chat\_words, which mapped abbreviated chat expressions to their expanded forms. Subsequently, a function named chat\_conversion was implemented to iterate through each word in the captions, identifying and replacing any occurrences of chat words with their full forms. This function was then applied to the 'Caption' column of the dataset using the apply() method. By incorporating this step into the data cleaning pipeline, the dataset was effectively standardized, ensuring consistency and coherence in the textual data, which is essential for subsequent model training and evaluation.



* **Spell Check:** A spell-checking mechanism is applied to identify and correct any spelling errors in the textual data, ensuring accuracy and coherence in the captions.
* **Handling Emojis:** Emojis, often used in social media captions, are substituted with their corresponding textual representations using the emojis module to maintain consistency in the dataset.



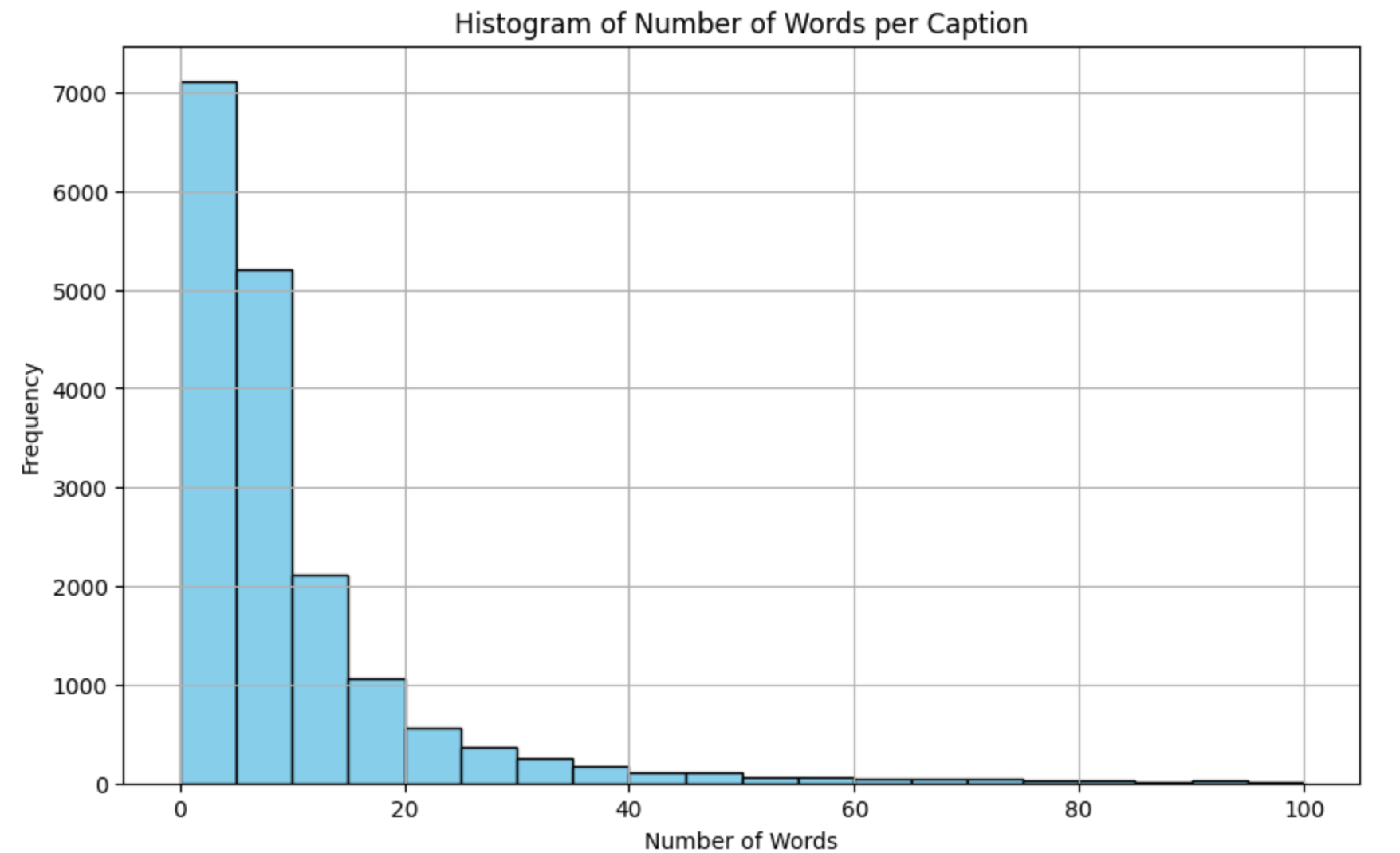
* **Lemmatization using spaCy:** Lemmatization is performed using the spaCy library, which provides efficient and accurate lemmatization capabilities, reducing inflected forms to their base or dictionary form.

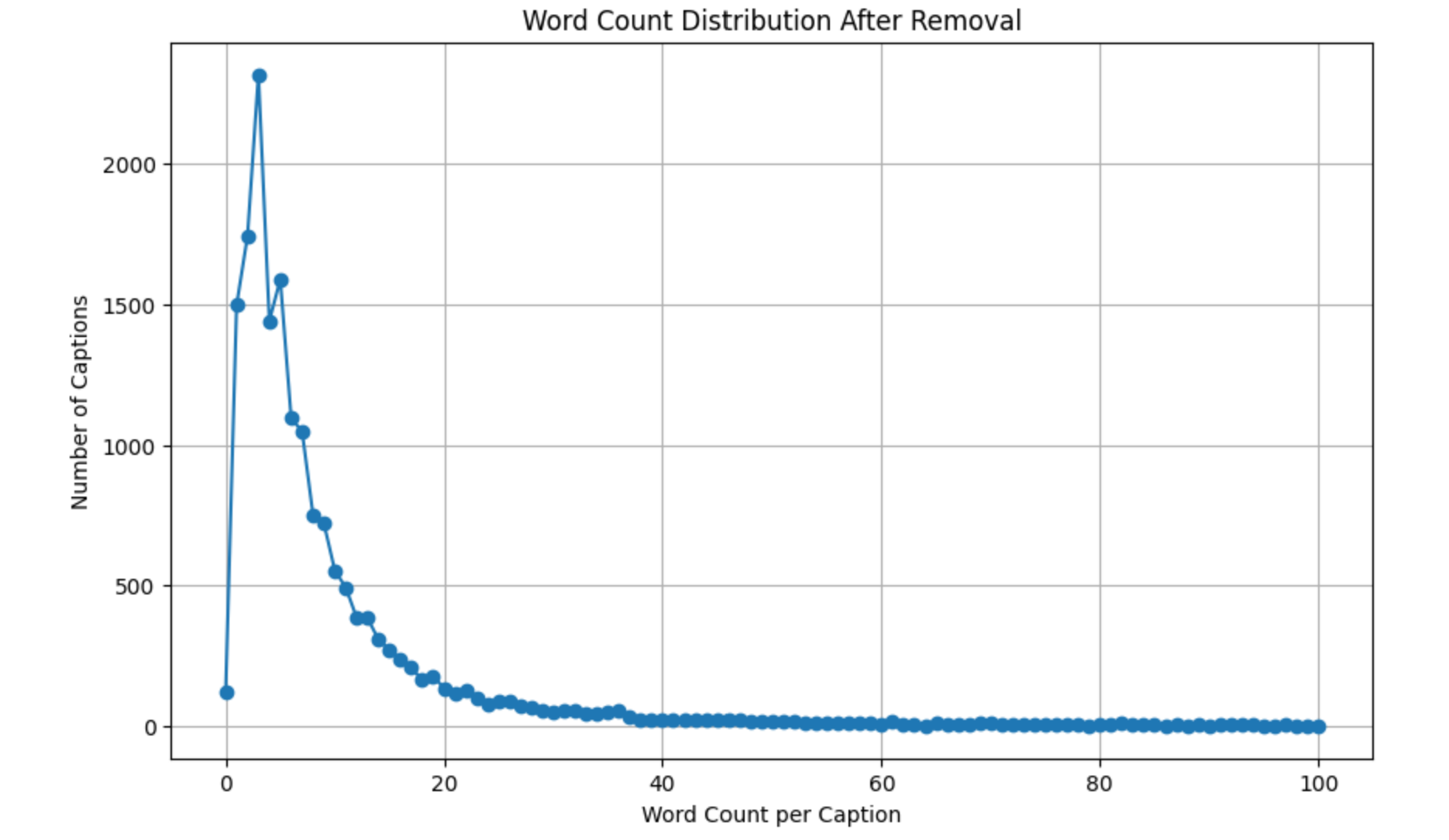
The implementation of these data cleaning steps ensures that the dataset is prepared for further processing, including feature extraction, model training, and evaluation. By standardizing and refining the textual data, the cleaned dataset facilitates more accurate and meaningful insights to be derived from the data, ultimately enhancing the performance and effectiveness of the image captioning model.

**5. Exploratory Data Analysis (EDA) and Data Processing**

**Analysis: Word Count Distribution**

To gain insights into the distribution of caption lengths within the dataset, the word count per caption was calculated using the len() function applied to each caption in the 'Caption' column. The resulting word counts were stored in a new column named 'Word\_Count' in the DataFrame. Subsequently, captions with a word count exceeding 100 were removed from the dataset to ensure focus on captions of manageable length for analysis and processing. By removing excessively long captions, the dataset was refined to contain captions that are concise and informative, facilitating efficient data processing and model development.





**Analysis: Word Cloud**

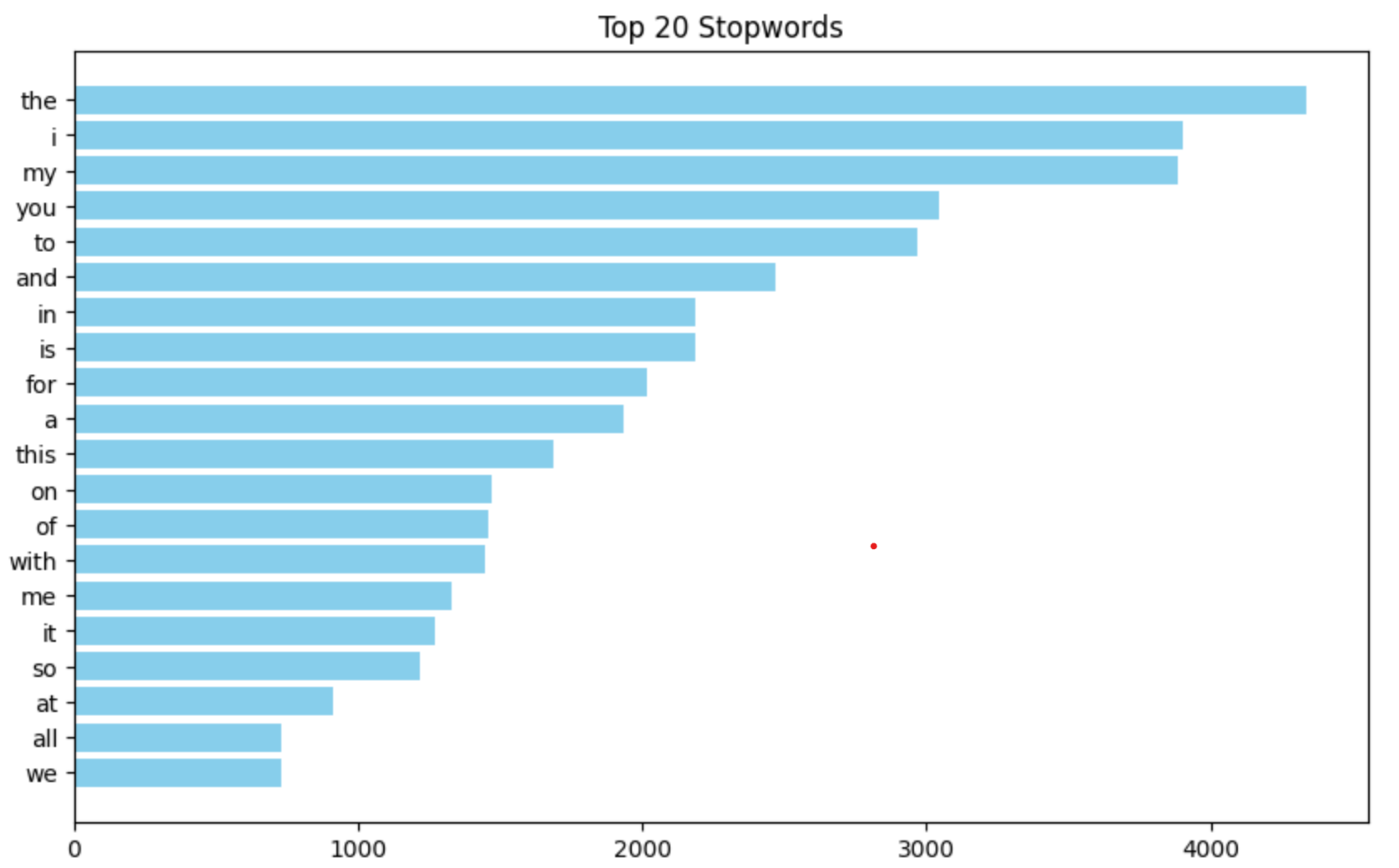
A word cloud was generated to visualize the frequency of words appearing in the captions. This visualization technique provides an intuitive representation of the most common words used in the dataset, with word size indicating relative frequency. By examining the word cloud, we gain insights into the prevalent themes and topics captured in the captions, guiding further analysis and interpretation of the textual data.



**Analysis: Stopwords Removal**

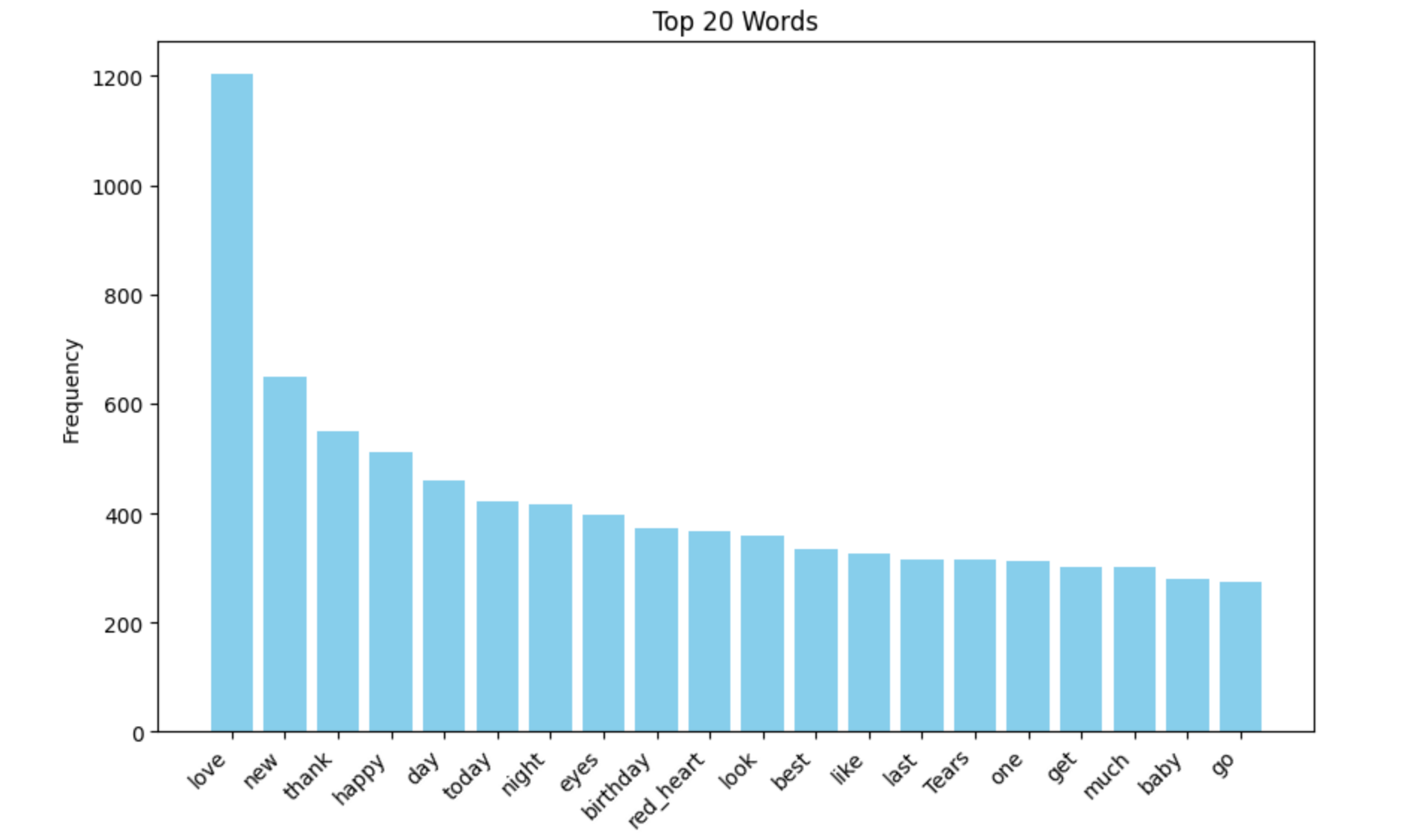
Stopwords are commonly used words in natural language that are considered to have little semantic significance, such as "the", "is", "and", etc. These words often appear frequently in text but contribute little to the overall meaning. In natural language processing, stopwords are typically removed from text data to focus on the more meaningful content.In the provided code snippet, the NLTK library is utilized to download a set of English stopwords. These stopwords are then used to filter out irrelevant words from the captions in the dataset. The function filter\_stopwords() takes a list of tokens (words) as input and returns a filtered list containing only tokens that are not stopwords. The frequency of stopwords in the dataset is then calculated using a Counter object, which tallies the occurrences of each unique stopword across all captions.

By removing stopwords from the captions, we aim to reduce noise in the textual data and focus on the more informative content, thereby improving the quality of subsequent analysis and model training. The frequency of stopwords provides insights into the distribution and prevalence of these common words within the dataset, aiding in the understanding of textual patterns and characteristics.



**Analysis : Top Words in Captions Based on Frequency**

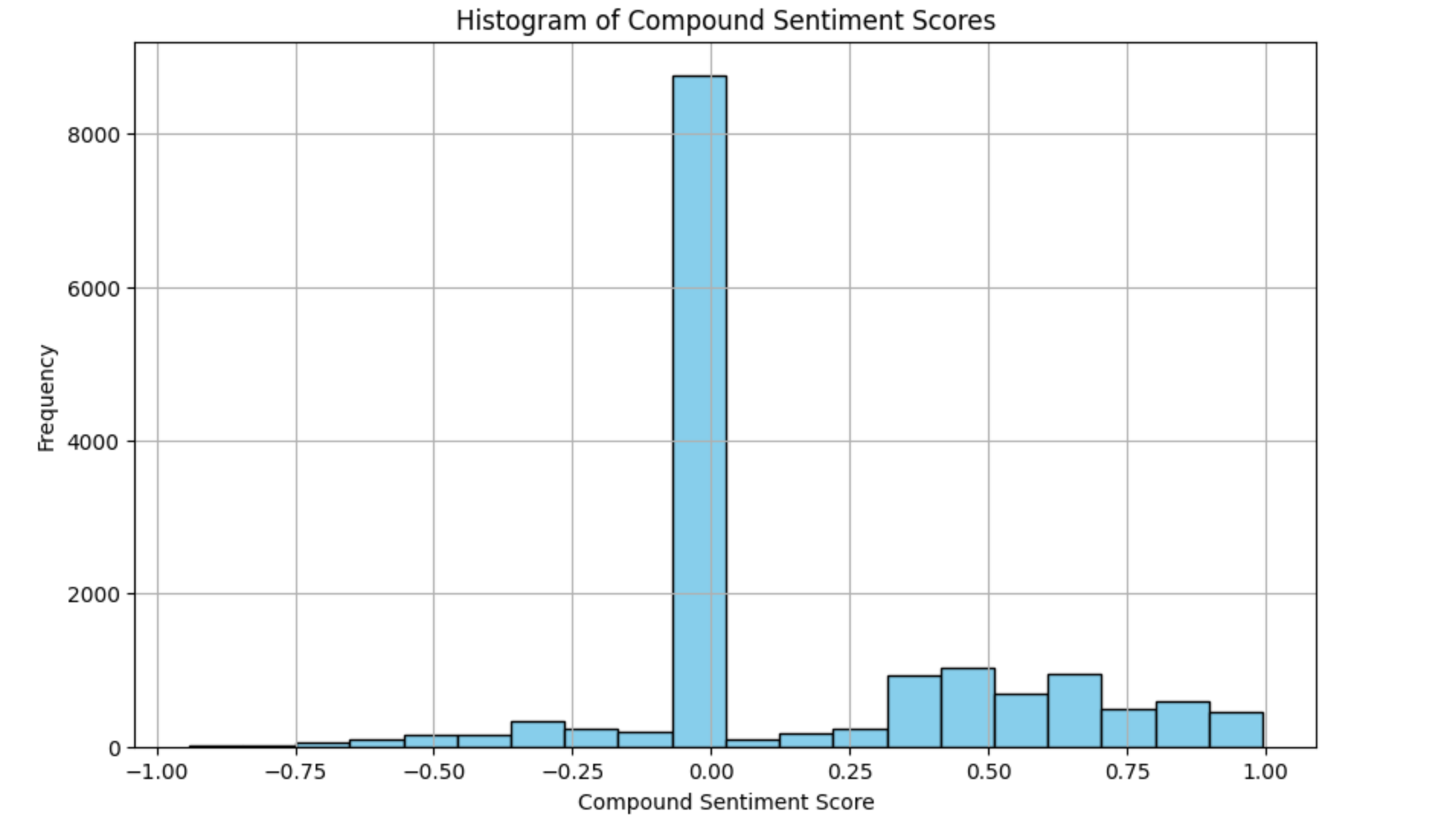
An analysis of the frequency of words in the captions data revealed the top 20 most commonly occurring words. These words provide insights into the prevalent themes and topics captured in the dataset, shedding light on the language patterns and characteristics of the captions.



**Vader Sentiment Analysis**

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool specifically designed for analyzing sentiments expressed in text. It scores text passages based on the positivity, negativity, neutrality, and compound sentiment. In the provided code, sentiment analysis using VADER is applied to the captions in the dataset. The output consists of sentiment scores for each caption, including scores for positivity, negativity, neutrality, and an overall compound score. By examining these scores, we can understand the emotional tone and sentiment expressed in the captions, aiding in the interpretation of user-generated content.

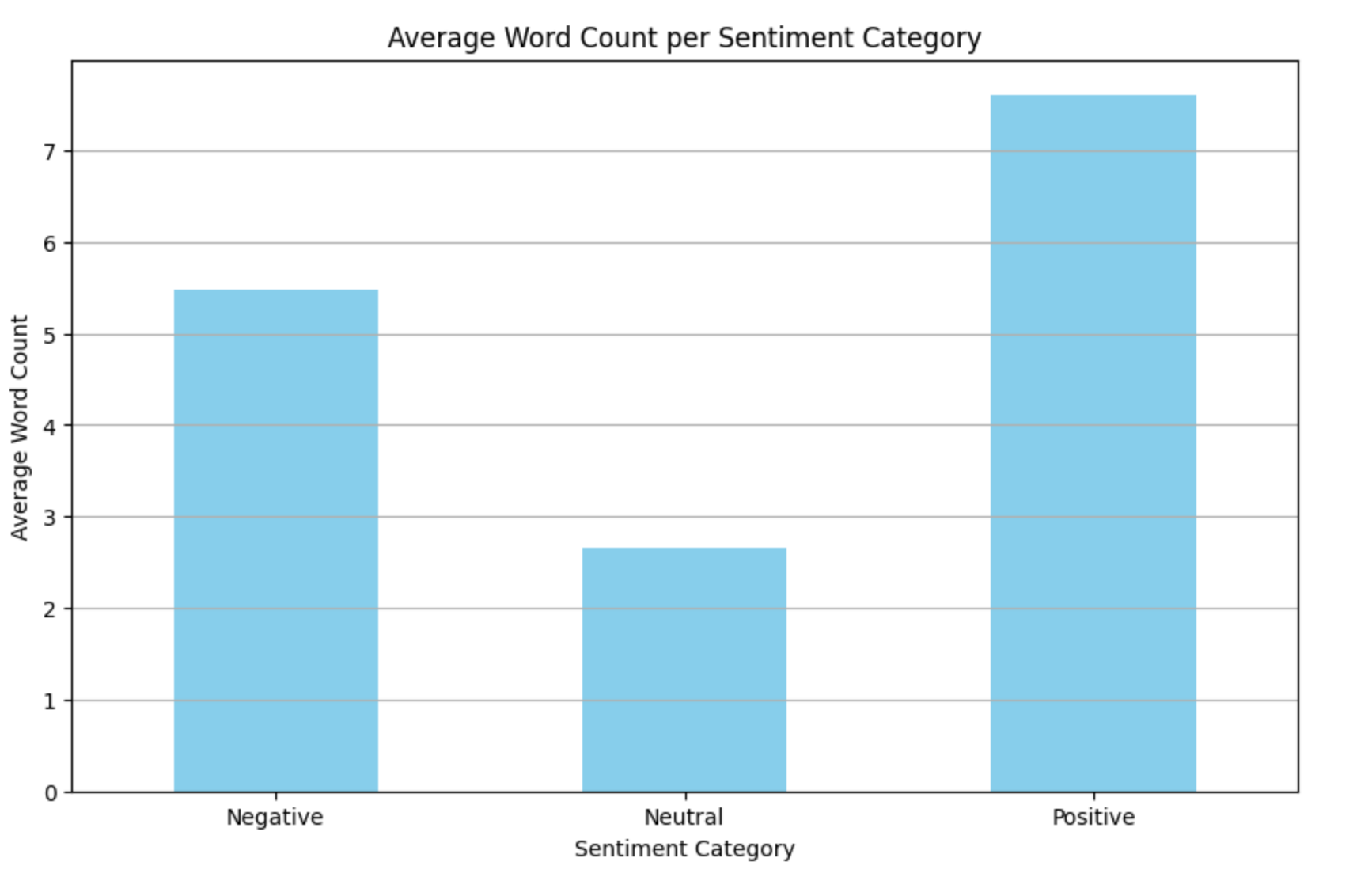




**Correlation Analysis: Sentiment and Number of Words per Caption**

To explore the relationship between sentiment and the number of words per caption, a correlation analysis was conducted. Sentiment scores, including positivity, negativity, neutrality, and compound sentiment, were calculated using VADER sentiment analysis for each caption in the dataset. Additionally, the number of words per caption was computed to quantify the length of textual content.

By examining the correlation between sentiment scores and the number of words per caption, insights into how sentiment may vary with caption length can be gained. A positive correlation suggests that longer captions tend to exhibit more extreme sentiment, while a negative correlation indicates the opposite. Understanding this relationship can provide valuable insights into how users express emotions in their captions based on the length of the accompanying text.



**Flesch Reading Ease Analysis**

The Flesch Reading Ease score is a measure of how easy or difficult it is to understand a passage of text. It is calculated based on the average sentence length and average syllables per word in the text. Higher scores indicate easier readability, while lower scores indicate more complex or difficult-to-read text.

In this analysis, the Flesch Reading Ease score was computed for each caption in the dataset using the textstat.flesch\_reading\_ease function from the textstat library. The distribution of Flesch Reading Ease scores across captions was visualized using a histogram, providing insights into the readability levels of the captions. Additionally, captions with very low Flesch Reading Ease scores (indicating high complexity) were identified and filtered out from the dataset.

**Distribution of Flesch Reading Ease Scores**

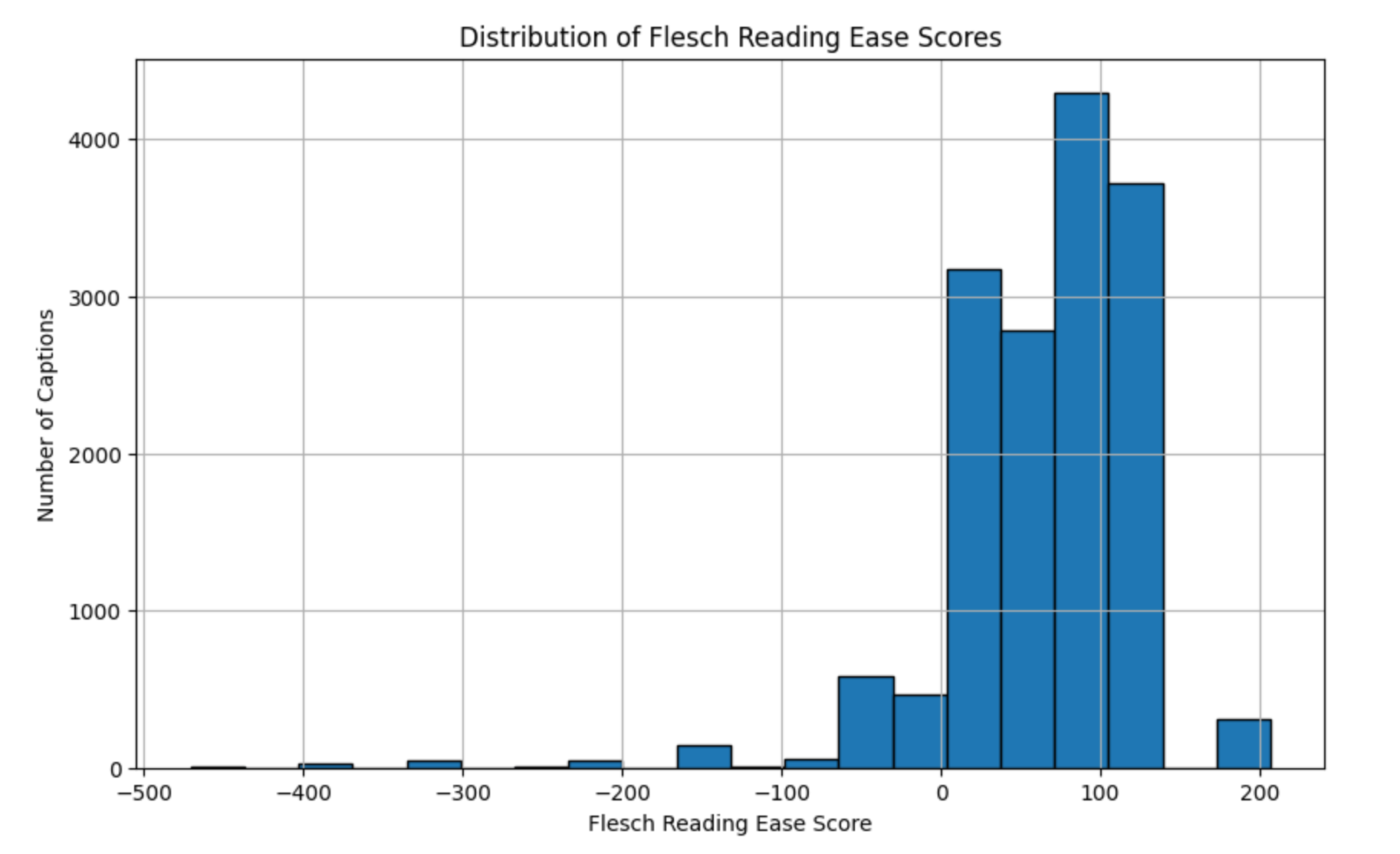
The histogram illustrates the distribution of Flesch Reading Ease scores among the captions. It shows the number of captions falling within different readability ranges, helping to identify trends and patterns in the readability levels of the textual content.

**Identification of Very Confusing Captions**

Captions with Flesch Reading Ease scores between 0 and 29 were classified as "very confusing." The count of captions falling into this category provides insights into the proportion of captions that may be challenging for readers to comprehend.

**Data Filtering**

Captions with very low Flesch Reading Ease scores (below 29) were filtered out from the dataset to ensure that only captions with acceptable readability levels are retained for further analysis and model training. This filtering process helps improve the quality and usability of the dataset by removing overly complex or difficult-to-read captions.



**Analysis : Part-of-Speech (POS) Tagging**

Part-of-speech (POS) tagging is a process in natural language processing (NLP) that involves categorizing each word in a text passage into a specific grammatical category, such as noun, verb, adjective, etc. This analysis aims to understand the distribution of different parts of speech in the captions dataset.

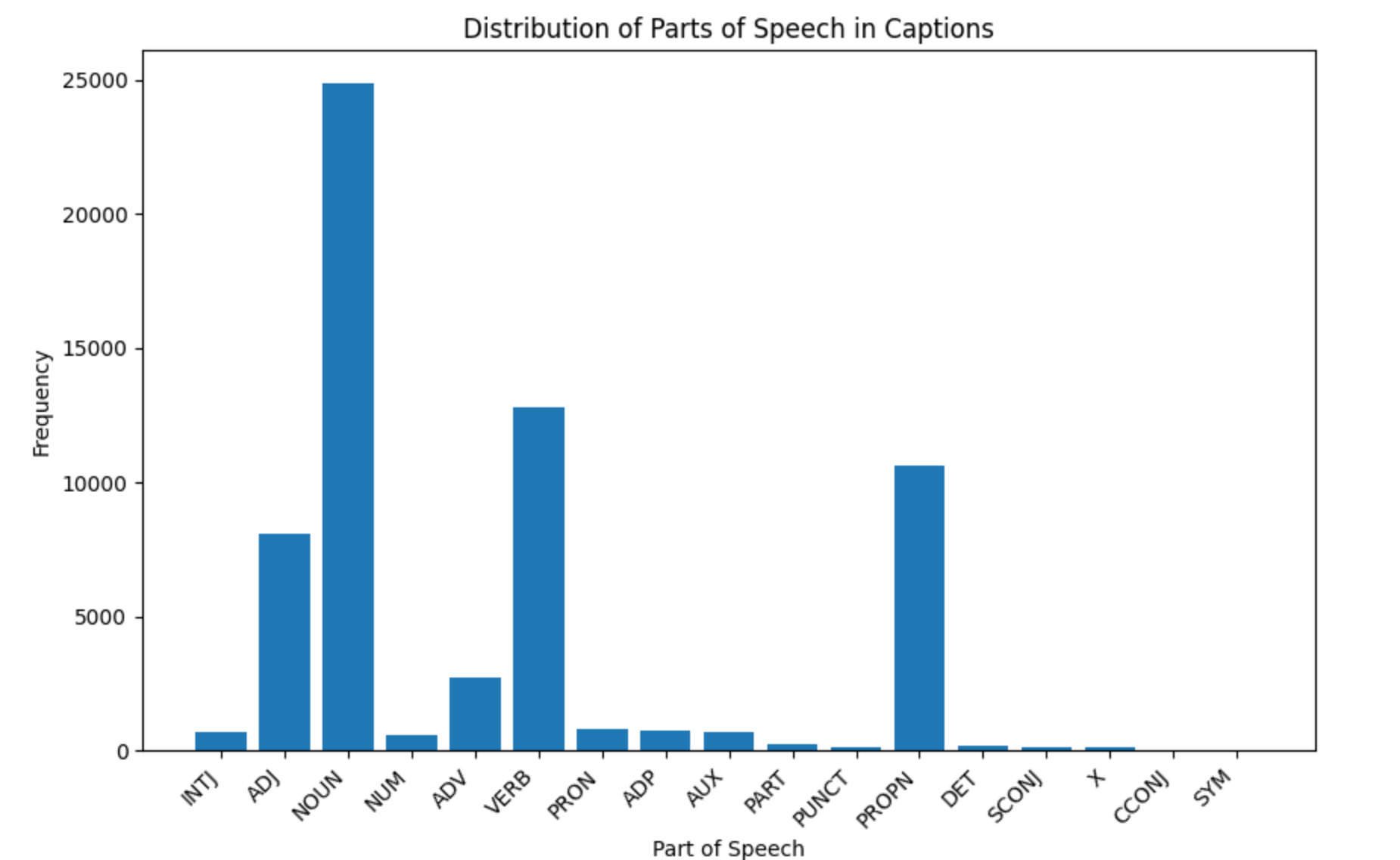
In this analysis:

**POS Tagging:** POS tagging was performed using the spaCy library (en\_core\_web\_sm model). Each caption was tokenized, and each token was assigned a POS tag indicating its grammatical category (e.g., noun, verb, adjective).

**POS Tag Distribution**: The distribution of POS tags across captions was visualized using a bar chart. The frequency of each POS tag was calculated, providing insights into the prevalence of different grammatical categories in the captions dataset.

**POS Tagging Process**

The POS tagging process involved tokenizing each caption and assigning a POS tag to each token using the spaCy NLP library. This process enables the classification of words based on their grammatical roles, facilitating deeper linguistic analysis of the textual data.



**Analysis : N-grams**

N-grams are contiguous sequences of n items (words or characters) within a text passage. Analyzing n-grams provides insights into the patterns and co-occurrences of words or characters in the textual data. This analysis aims to explore the distribution of n-grams within the captions dataset.

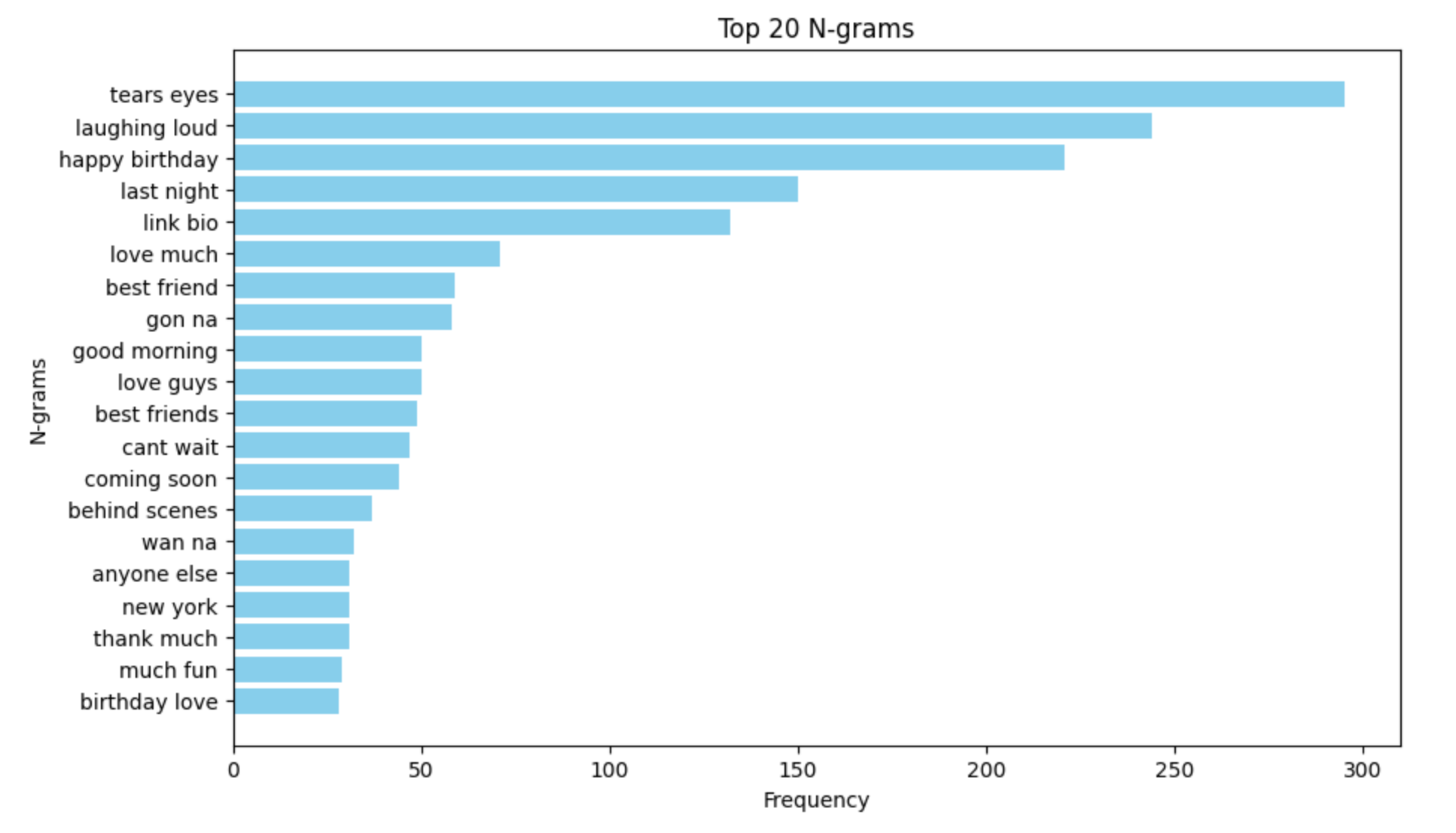
In this analysis:

**N-gram Generation:** N-grams were generated for each caption in the dataset. The dataset was tokenized into individual words, and n-grams of varying lengths (uni-grams, bi-grams, tri-grams, etc.) were created to capture sequential patterns of words within the captions.

**N-gram Frequency**: The frequency of each n-gram was calculated to identify the most common sequences of words occurring in the captions dataset. This provides insights into the prevalent phrases and linguistic patterns present in the textual data.

**N-gram Generation Process**

N-grams were generated by tokenizing each caption into individual words and then creating contiguous sequences of words of varying lengths. This process enables the capture of sequential patterns and co-occurrences of words within the captions.



**6. Model Architecture Overview**

The proposed model architecture consists of three main components: feature extraction, encoder model, and decoder model. Each component plays a crucial role in the process of generating captions for images, enabling the model to effectively learn the relationship between visual features and textual descriptions.

**1. Feature Extraction:**

* The feature extraction component utilizes the InceptionV3 model pre-trained on the ImageNet dataset to extract high-level visual features from input images.
* The last fully connected layer of the InceptionV3 model is removed, leaving the extracted features in a high-dimensional space.

**2. Encoder Model:**

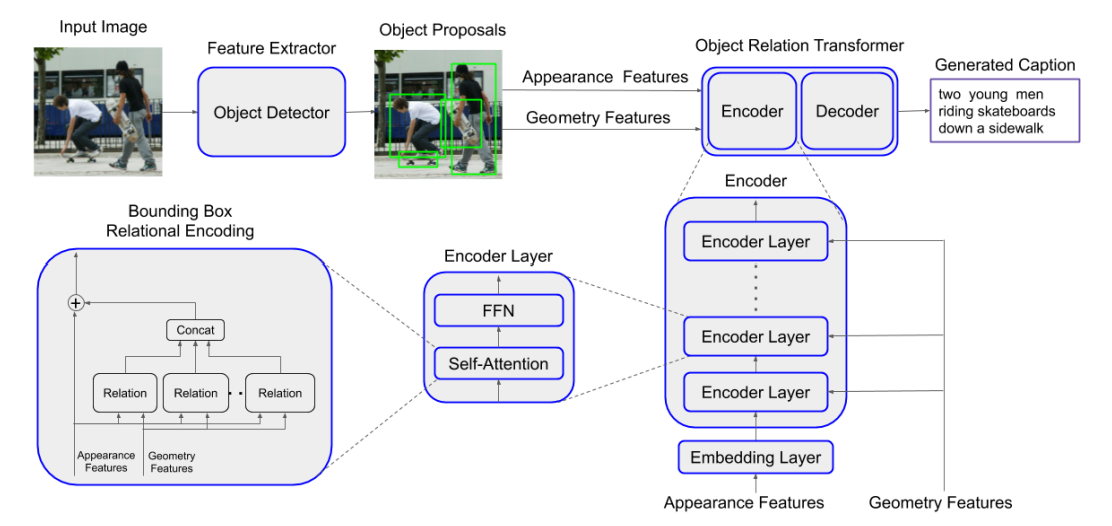
* The encoder model processes the extracted image features and the corresponding textual captions to create a joint representation that captures the semantic relationship between the visual and textual modalities.
* The image features are passed through a dense layer to reduce their dimensionality and enhance their representational power.
* The textual captions are processed using an embedding layer followed by an LSTM (Long Short-Term Memory) layer, which captures the sequential dependencies and semantic context of the captions.
* The outputs of the image and caption processing streams are then merged to create a joint feature representation.

**3. Decoder Model:**

* The decoder model takes the joint feature representation generated by the encoder as input and generates the corresponding textual captions.
* The joint feature representation is passed through additional dense and LSTM layers to further refine the features and facilitate caption generation.
* The final output layer consists of a softmax activation function, which predicts the probability distribution over the vocabulary of words, enabling the model to generate captions word by word.

**Key Points:**

* Integration of Visual and Textual Modalities: The model seamlessly integrates visual features extracted from images and textual information from captions to generate coherent and contextually relevant captions.
* Hierarchical Representation Learning: The encoder-decoder architecture enables hierarchical representation learning, where high-level semantic features are learned from both visual and textual inputs to facilitate accurate caption generation.
* End-to-End Training: The entire model is trained in an end-to-end fashion, allowing it to jointly optimize the feature extraction, encoding, and decoding processes to maximize caption generation performance.
* This comprehensive model architecture enables the generation of descriptive and contextually relevant captions for images, leveraging the complementary information provided by both visual and textual modalities.



**7. Implementation Overview**

**Feature Extraction:**

* The feature extraction process leverages the InceptionV3 convolutional neural network pre-trained on the ImageNet dataset. The model is loaded using TensorFlow's Keras API.
* The final fully connected layer of the InceptionV3 model is removed, leaving the extracted image features in a high-dimensional space.
* Images are preprocessed using the load\_img and img\_to\_array functions from Keras, followed by normalization using the preprocess\_input function.

def extract\_features(df, target\_size=(299, 299), batch\_size=32):

*'''*

*Extract features from photos in the dataset using InceptionV3 model.*

*:param df: DataFrame containing image file names in column 'Image File'.*

*:param target\_size: Tuple specifying the target size for images.*

*:param batch\_size: Batch size for processing images.*

*:return: Dictionary of image names and their extracted features.*

*'''*

*# Load InceptionV3 model pre-trained on ImageNet and remove the top fully connected layer*

base\_model = tf.keras.applications.InceptionV3(include\_top=False, weights='imagenet')

model = Model(inputs=base\_model.inputs, outputs=base\_model.layers[-1].output)

*# Dictionary to store the extracted features*

features = OrderedDict()

*# Base path for image files*

img\_path = '/kaggle/working/combined\_images' *# Path where your images are saved*

*# Process images in batches*

print('Starting feature extraction...')

for i **in** tqdm(range(0, len(df), batch\_size)):

batch\_image\_names = df['Image File'].iloc[i:i+batch\_size].tolist()

batch\_images = []

for image\_name **in** batch\_image\_names:

filename = os.path.join(img\_path, image\_name+'.jpg')

image = load\_img(filename, target\_size=target\_size)

image = img\_to\_array(image)

image = tf.keras.applications.inception\_v3.preprocess\_input(image)

batch\_images.append(image)

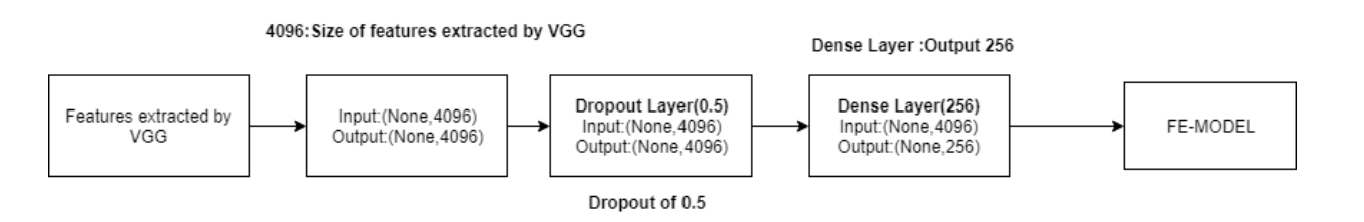
batch\_images = np.array(batch\_images)

features\_batch = model.predict(batch\_images)

for j, image\_name **in** enumerate(batch\_image\_names):

features[image\_name] = features\_batch[j].flatten()

return features

****

InceptionV3

**Dense Layer Application:**

* After feature extraction, a dense layer is applied to the extracted image features to reduce their dimensionality and enhance their representational power.
* The dimensionality reduction process is essential for improving computational efficiency and facilitating effective integration with textual features.

def apply\_dense\_layer(features, output\_dim=100):

*'''*

*Apply a dense layer for dimensionality reduction to the extracted features.*

*:param features: Dictionary containing image names and their extracted features.*

*:param output\_dim: Dimensionality of the output feature vector.*

*:return: Dictionary of image names and their extracted features after the dense layer.*

*'''*

dense\_features = {}

for image\_name, feature\_vector **in** features.items():

dense\_model = tf.keras.Sequential([

Dense(output\_dim, activation='relu', input\_shape=(len(feature\_vector),)),

])

dense\_feature\_vector = dense\_model.predict(feature\_vector.reshape(1, -1))

dense\_features[image\_name] = dense\_feature\_vector.flatten()

return dense\_features

**Preprocessing:**

* Captions are preprocessed to generate sequences suitable for sequence-to-sequence learning. Each caption is split into tokens (words) and padded to a fixed length to ensure uniform input dimensions.
* Captions are represented as one-hot encoded vectors, where each token is represented as a binary vector indicating its presence in the vocabulary.

def preprocessing(dtexts, dimages):

*'''*

*Pre-process captions to generate seq2seq text. A caption is generated up to position 't'*

*and the model learns to predict the caption at the 't+1' position.*

*:param dtexts: List of captions*

*:param dimages: Corresponding list of image features*

*:return: Tuple of (Xtext, Ximage, ytext)*

*Xtext - input text up to point 't'*

*ytext - text to be predicted at 't+1' position*

*Ximage - corresponding image features*

*'''*

assert len(dtexts) == len(dimages), "# of captions and images must match"

Xtext, Ximage, ytext = [], [], []

for text, image **in** zip(dtexts, dimages):

for t **in** range(1, len(text)):

in\_text, out\_text = text[:t], text[t]

in\_text = pad\_sequences([in\_text], maxlen=maxlen, padding='post')[0]

out\_text = to\_categorical([out\_text], num\_classes=vocab\_size)[0]

Xtext.append(in\_text)

Ximage.append(image)

ytext.append(out\_text)

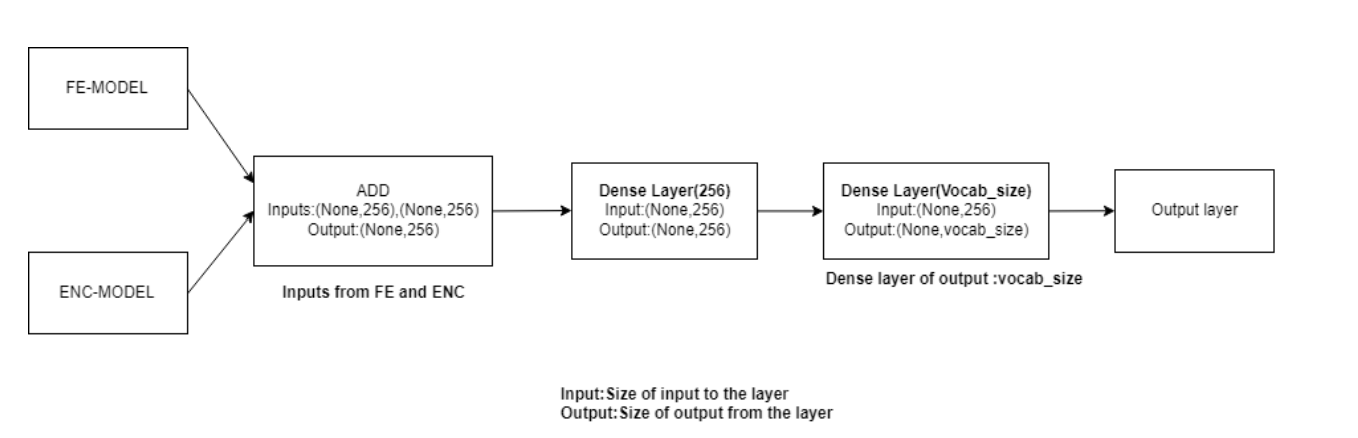
Xtext, Ximage, ytext = np.array(Xtext), np.array(Ximage), np.array(ytext)

print(f"Processed dataset shapes - Xtext: **{**Xtext.shape**}**, Ximage: **{**Ximage.shape**}**, ytext: **{**ytext.shape**}**")

return (Xtext, Ximage, ytext)

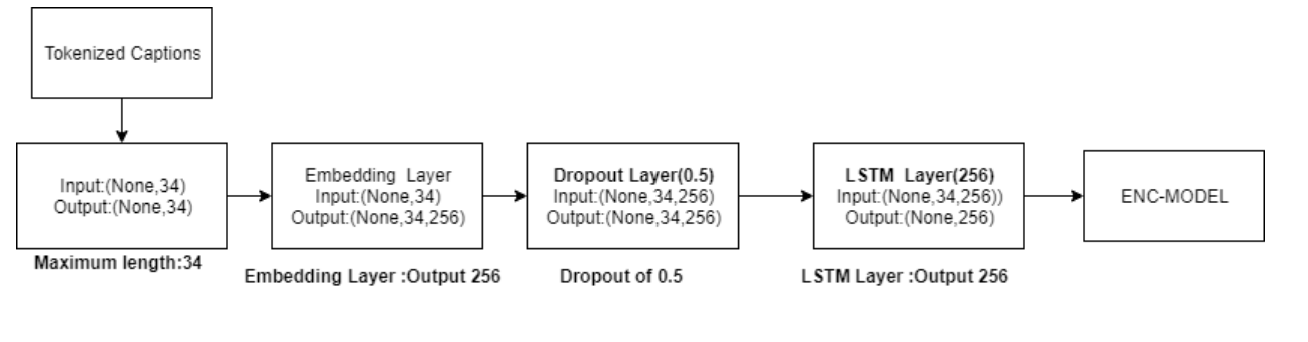
**Encoder Model:**

* The encoder model processes both the extracted image features and the preprocessed textual captions to create a joint representation.
* Image features are passed through a dense layer to enhance their semantic representation.
* Textual captions are processed using an embedding layer followed by an LSTM layer to capture sequential dependencies and semantic context.



**Decoder Model:**

* The decoder model takes the joint representation generated by the encoder as input and generates textual captions word by word.
* The joint representation is passed through additional dense and LSTM layers to refine the features and facilitate caption generation.
* The final output layer consists of a softmax activation function, predicting the probability distribution over the vocabulary of words for each time step.



def make\_model(input\_shape):

*'''*

*Final model that combines extracted image features and their captions*

*:param input\_shape: size of image features*

*:return: model*

*'''*

embedding\_dimension = 64

*# input for image features*

input\_image = Input(shape=(input\_shape,))

image\_dropout = Dropout(0.5)(input\_image)

image\_dense = Dense(256, activation='relu', name="ImageFeature")(image\_dropout)

*# input for captions*

input\_txt = Input(shape=(maxlen,))

text\_embedding = Embedding(vocab\_size, embedding\_dimension, mask\_zero=True)(input\_txt)

text\_lstm = LSTM(256, name="CaptionFeature")(text\_embedding)

*# merging image and caption features*

merged\_features = add([image\_dense, text\_lstm])

merged\_dense = Dense(256, activation='relu')(merged\_features)

output = Dense(vocab\_size, activation='softmax')(merged\_dense)

*# create model*

model = Model(inputs=[input\_image, input\_txt], outputs=output)

model.compile(loss='categorical\_crossentropy', optimizer='adam')

return model

In [ ]:

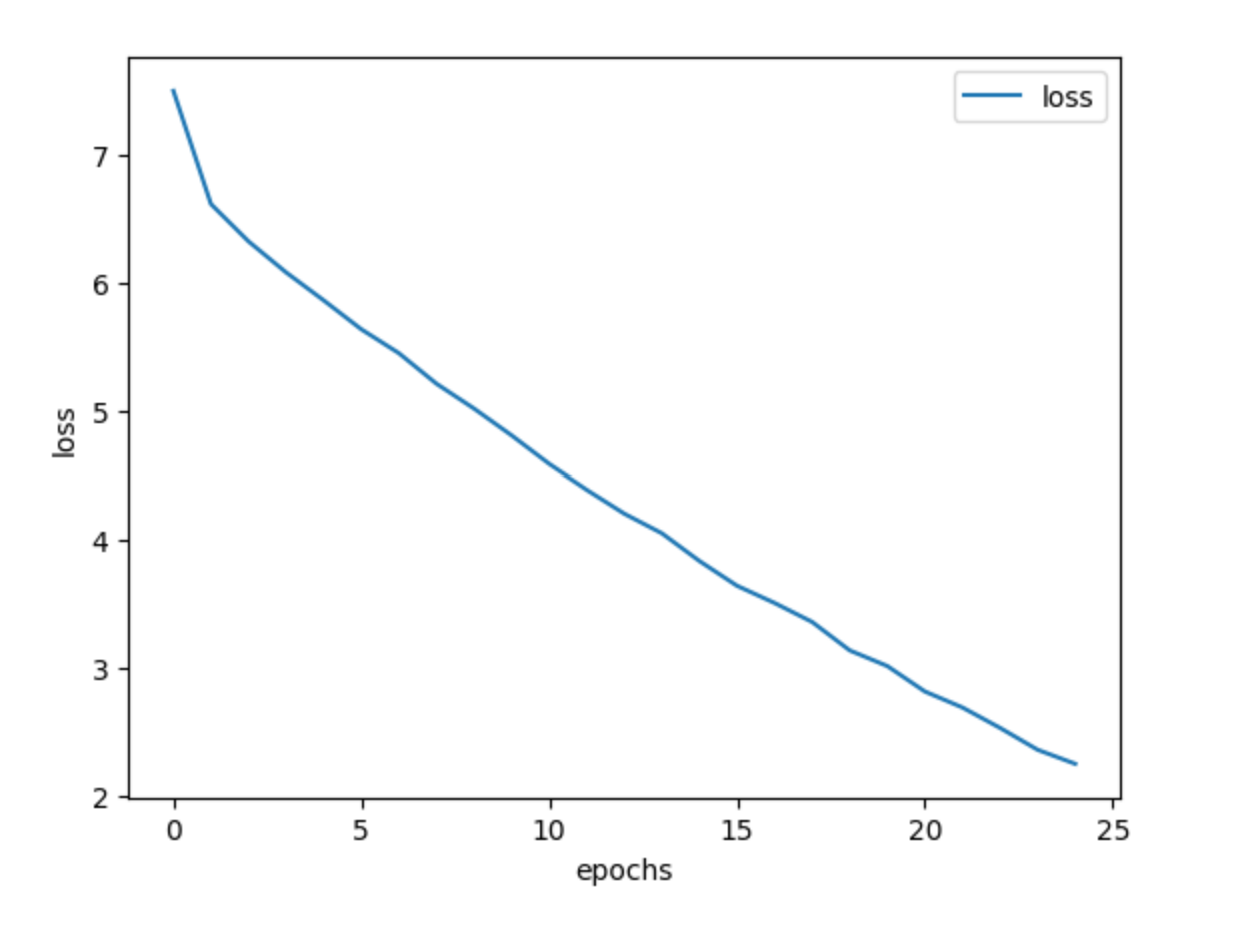
**Training:**

* The entire model is trained end-to-end using backpropagation and gradient descent optimization.
* The loss function used is categorical cross-entropy, measuring the discrepancy between the predicted and actual word distributions.
* Training involves iteratively updating the model parameters to minimize the loss and improve caption generation performance.

**8. Model Evaluation and Visualization**

**Metric Analysis:**

* The performance of the trained model is evaluated using various metrics, including loss. These metrics provide insights into the model's learning progress and effectiveness in generating captions for images.
* The graph displays the loss values across epochs during the training process. Loss values indicate the discrepancy between predicted and actual captions, with lower values indicating better model performance.



**Time of Execution:**

* The time taken for model training and evaluation is recorded to provide insights into the computational resources required for the task. This metric helps assess the efficiency of the model implementation and training process.

import matplotlib.pyplot as plt

def make\_metrics(hist):

*'''*

*Graphs of metrics*

*:param hist: model history*

*:return: None*

*'''*

for label **in** ['loss']:

plt.plot(hist.history[label], label=label)

plt.legend()

plt.xlabel('epochs')

plt.ylabel('loss')

*# Save the plot as an image*

plt.savefig('/kaggle/working/output\_metrics.png') *# Adjust the path for Kaggle environment*

**BLEU Score Calculation:**

* The BLEU score, a metric commonly used in natural language processing tasks, is calculated to evaluate the quality of generated captions.
* BLEU score measures the similarity between predicted and reference captions, with higher scores indicating better caption quality.
* Predicted captions are compared against ground truth captions from the test set to compute the BLEU score.
* Mean BLEU score is calculated to provide a summary measure of the model's performance across the entire test set.
* A higher mean BLEU score indicates better alignment between predicted and reference captions, signifying improved caption generation quality.

from nltk.translate.bleu\_score import sentence\_bleu

def bleu\_score(img\_text\_model, fnm\_test, di\_test, dt\_test, tokenizer, index\_word):

*'''*

*Calculates BLEU score for image captioning predictions.*

*:param img\_text\_model: Neural network model for image captioning*

*:param fnm\_test: List of filenames for images from the test set*

*:param di\_test: List of image feature vectors from the test set, obtained from VGG16*

*:param dt\_test: List of tokenized captions from the test set*

*:param tokenizer: Tokenizer used for encoding text*

*:param index\_word: Dictionary to convert tokens back to words*

*:return: None*

*'''*

*# Initialize containers for tracking results*

nkeep = 5

pred\_good, pred\_bad, bleus = [], [], []

*# Process each image, feature vector, and tokenized text*

for count, (jpgfnm, image\_feature, tokenized\_text) **in** enumerate(zip(fnm\_test, di\_test, dt\_test), 1):

if count % 200 == 0:

print(f"**{**100 \* count / float(len(fnm\_test))**:**.2f**}**% is done..")

*# Construct the true caption from tokens*

caption\_true = [index\_word.get(i, '') for i **in** tokenized\_text if i != 0] *# Handles unknown tokens and removes padding*

caption\_true = caption\_true[1:-1] *# Remove '<START>' and '<END>' tokens*

*# Predict the caption*

caption = predict\_caption(img\_text\_model, image\_feature.reshape(1, -1), tokenizer, index\_word)

caption = caption.split()[1:-1] *# Convert predicted caption to list and remove '<START>' and '<END>'*

*# Calculate BLEU score*

bleu = sentence\_bleu([caption\_true], caption)

bleus.append(bleu)

*# Collect good and bad predictions based on BLEU score*

if bleu > 0.7 **and** len(pred\_good) < nkeep:

pred\_good.append((bleu, jpgfnm, caption\_true, caption))

elif bleu < 0.3 **and** len(pred\_bad) < nkeep:

pred\_bad.append((bleu, jpgfnm, caption\_true, caption))

*# Output mean BLEU score*

print(f'Mean BLEU **{**np.mean(bleus)**:**.3f**}**')

**Sample Outputs:**

* Sample outputs, including both predicted and ground truth captions, are saved for qualitative evaluation. These outputs provide visual representations of the model's performance in generating captions for images.
* Predicted captions are compared against ground truth captions to assess the accuracy and relevance of the generated captions.

**Visual Inspection of Results:**

* Images along with their corresponding predicted and ground truth captions are plotted for visual inspection.
* This visual representation allows for qualitative assessment of the model's ability to generate relevant and descriptive captions for a variety of images.

**9. References**

* <https://www.researchgate.net/publication/329037107_Image_Captioning_Based_on_Deep_Neural_Networks>
* <https://paperswithcode.com/task/image-captioning>
* https://www.researchgate.net/publication/333214768\_Visual\_Image\_Caption\_Generator\_Using\_Deep\_Learning/link/5e08a9bd92851c8364a3d754/download?\_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19