In [3]:
```python
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
import numpy as np
import wordcloud
import matplotlib.pyplot as plt
from sklearn.utils import resample
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, Flatten
from wordcloud import WordCloud
from keras.utils import to_categorical
from keras import backend as K
import csv
from keras.models import load_model
```

```
d:\Users\fadia\anaconda3\envs\tf_gpu\lib\site-packages\requests\__init__.py:102: Request
sDependencyWarning: urllib3 (1.26.9) or chardet (5.1.0)/charset_normalizer (2.0.4) does
n't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({})/charset_normalizer ({}) doesn't match a su
pported "
```

In [2]:
```python
df2 = pd.read_csv("train.csv")
df2.head()

# dataset shape to know how many tweets in the datasets
print(f"num of tweets: {df2.shape}")




def overview():
    print(df2.head())
    print(df2.info())

def plot_label_distribution():
    label_counts = df2.iloc[:, 2:].sum()
    plt.bar(label_counts.index, label_counts.values)
    plt.xlabel("Label")
    plt.ylabel("Count")
    plt.title("Distribution of Target Labels")
    plt.show()

def plot_comment_length_distribution():
    df2['comment_length'] = df2['comment_text'].apply(len)
    plt.hist(df2['comment_length'], bins=50)
    plt.xlabel("Comment Length")
    plt.ylabel("Count")
    plt.title("Distribution of text Lengths")
    plt.show()

def word_count_statistics():
    df2['word_count'] = df2['comment_text'].apply(lambda x: len(x.split()))
    print(df2['word_count'].describe())


def plot_top_frequent_words():
    all_words = ' '.join(df2['comment_text']).split()
    word_counts = pd.Series(all_words).value_counts()
    top_20_words = word_counts[:20]

    plt.bar(top_20_words.index, top_20_words.values)
    plt.xlabel("Word")
    plt.ylabel("Count")
    plt.title("Top 20 Most Frequent Words")
    plt.xticks(rotation=45)
    plt.show()


def plot_wordcloud_top_frequent_words():
    all_words = ' '.join(df2['comment_text']).split()
    word_counts = pd.Series(all_words).value_counts()
    top_20_words = word_counts[:20]

    wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from

    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title('Top 20 Most Frequent Words')
    plt.show()



def plot_class_distribution():
    df2['offin'] = ((df2['toxic'] > 0) | (df2['severe_toxic'] > 0)| (df2['obscene'] > 0)
    class_counts = df2['offin'].value_counts()
    plt.hist(df2['offin'], bins=2)
    plt.xlabel("Class")
    plt.ylabel("Count")
    plt.title("Class Distribution")
    plt.xticks([0, 1], ['Non-Hate Speech', 'Hate Speech'])
    plt.show()
```

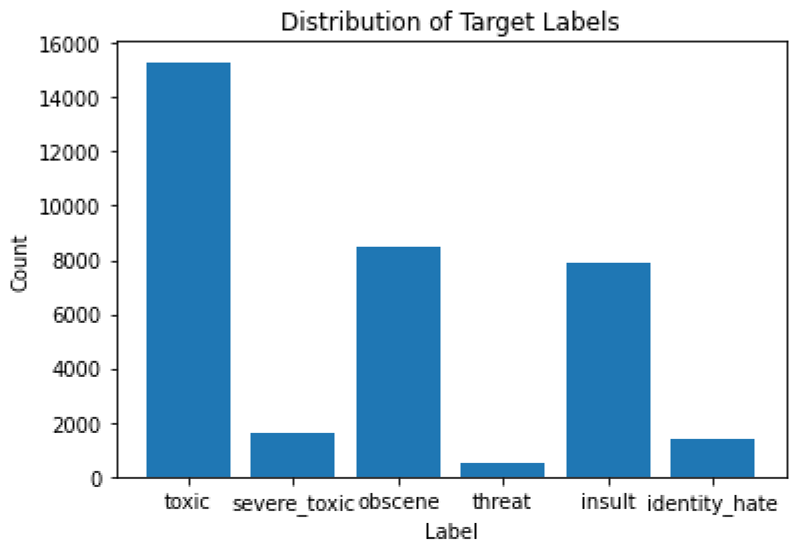
```
num of tweets: (159571, 8)
```

In [3]:
```
overview()
```

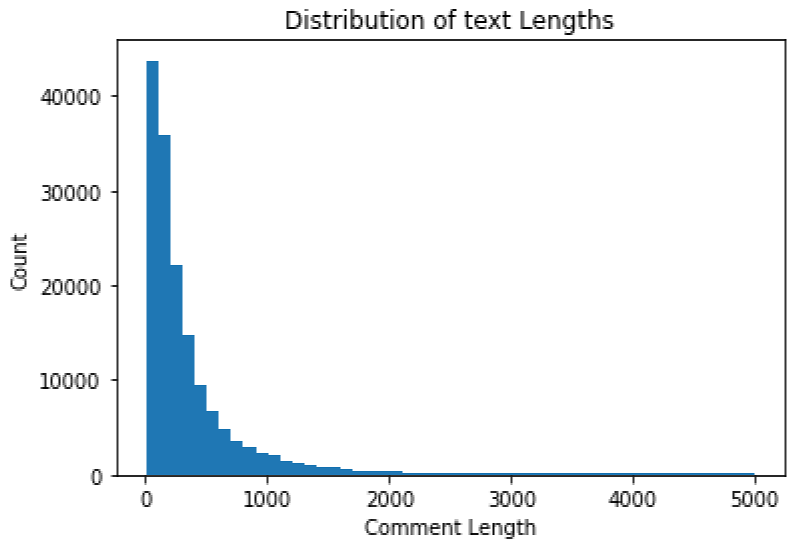
```
                 id                                       comment_text  toxic  \
0  0000997932d777bf  Explanation\nWhy the edits made under my usern...      0
1  000103f0d9cfb60f  D'aww! He matches this background colour I'm s...      0
2  000113f07ec002fd  Hey man, I'm really not trying to edit war. It...      0
3  0001b41b1c6bb37e  "\nMore\nI can't make any real suggestions on ...      0
4  0001d958c54c6e35  You, sir, are my hero. Any chance you remember...      0

   severe_toxic  obscene  threat  insult  identity_hate
0             0        0       0       0              0
1             0        0       0       0              0
2             0        0       0       0              0
3             0        0       0       0              0
4             0        0       0       0              0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159571 entries, 0 to 159570
Data columns (total 8 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   id             159571 non-null  object
 1   comment_text   159571 non-null  object
 2   toxic          159571 non-null  int64
 3   severe_toxic   159571 non-null  int64
 4   obscene        159571 non-null  int64
 5   threat         159571 non-null  int64
 6   insult         159571 non-null  int64
 7   identity_hate  159571 non-null  int64
dtypes: int64(6), object(2)
```

```
memory usage: 9.7+ MB
None
```

In [4]:
```
plot_label_distribution()
```



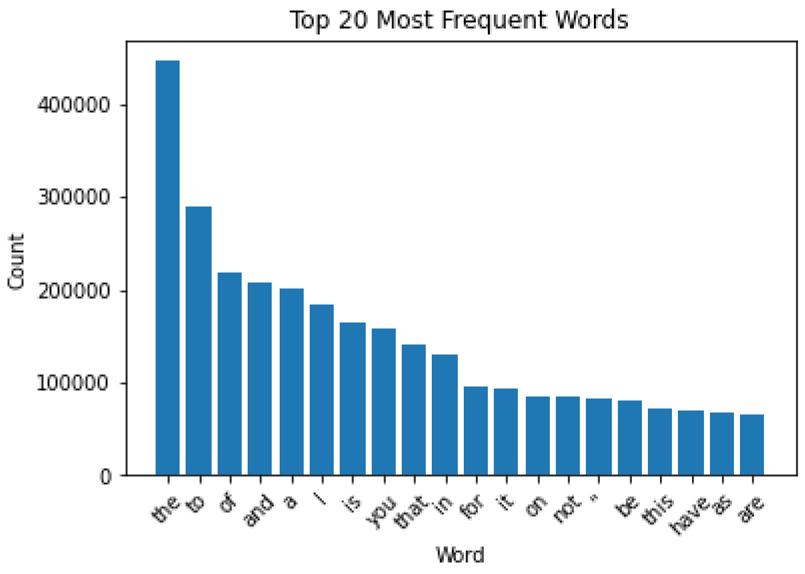
In [5]:
```
plot_comment_length_distribution()
```



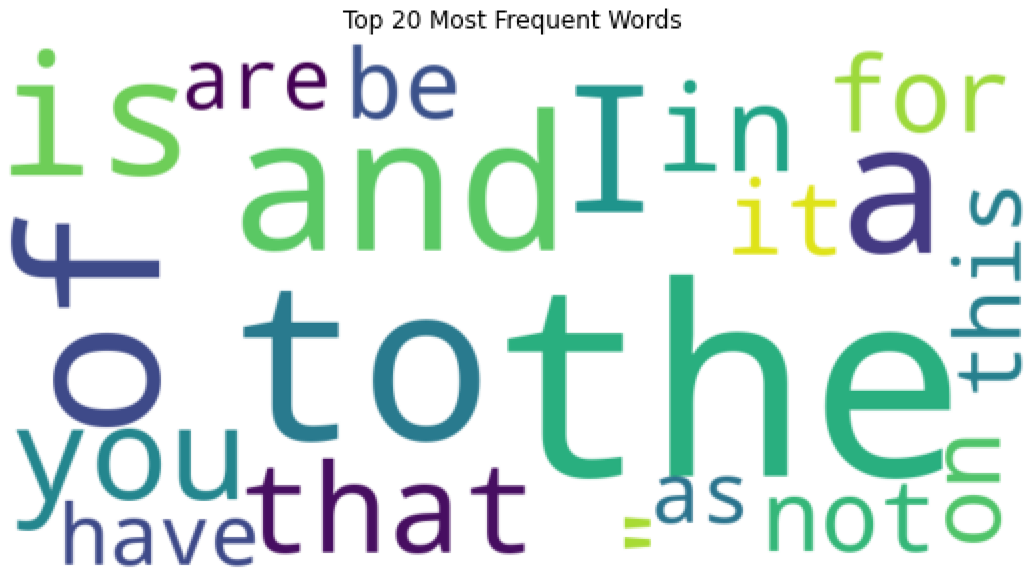
In [6]:
```
word_count_statistics()
```

```
count    159571.000000
mean         67.272518
std          99.231355
min           1.000000
25%          17.000000
50%          36.000000
75%          75.000000
max        1411.000000
Name: word_count, dtype: float64
```

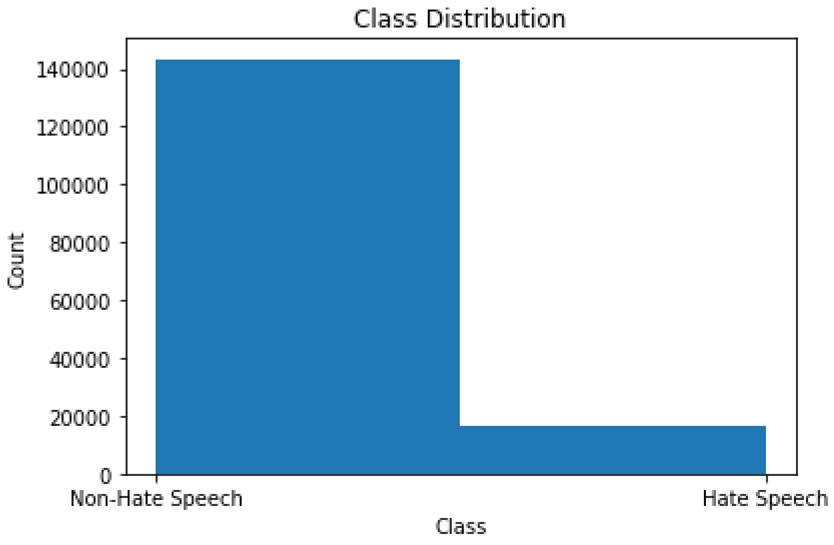
In [7]:
```
plot_top_frequent_words()
```



```
In [8]: plot_wordcloud_top_frequent_words()
```



```
In [9]: plot_class_distribution()
```



Pre-processing and building the model

In [10]:
```
df_offin_0 = df2[df2['offin'] == 0]
df_offin_1 = df2[df2['offin'] == 1]
df_offin_0_downsampled = resample(df_offin_0, replace=False, n_samples=16000, random_sta
df_offin_1_downsampled = resample(df_offin_1, replace=False, n_samples=16000, random_sta
balanced_df = pd.concat([df_offin_0_downsampled, df_offin_1_downsampled])
balanced_df = balanced_df.sample(frac=1, random_state=42)
df2=balanced_df
labels = list(df2['offin'])
stop_words = set(stopwords.words('english'))
stop_words.add("rt")
stop_words.add("pue")
def preprocess2(tweet):
    tweet = re.sub(r'@\w+', '', tweet)

    # Remove hashtags
    tweet = re.sub(r'#\w+', '', tweet)

    # Remove retweet indicators
    tweet = re.sub(r'RT', '', tweet)


    # Remove stop words
    stop_words = set(stopwords.words("english"))
    tweet = ' '.join([word for word in tweet.split() if word not in stop_words])

    # Remove non-letter characters

    cleaned_tweet = re.sub('[^a-zA-Z]', ' ', tweet)
    return cleaned_tweet


def preprocess(datas):
    clean = []

    clean = [preprocess2(str(text)) for text in datas]


    return clean

tweet = list(df2['comment_text'])
labels = list(df2['offin'])
clean_tweet = preprocess(tweet)
```

In [ ]:
```
def write_list_to_csv(data_list, output_file):
    # Ensure the output_file ends with .csv
    if not output_file.endswith('.csv'):
        output_file += '.csv'

    with open(output_file, 'w', newline='', encoding='utf-8') as csv_file:
        csv_writer = csv.writer(csv_file)
        # Write each item in the list as a separate row in the CSV file
        for item in data_list:
            csv_writer.writerow([item])

    # Name of the output CSV file
    output_csv_file = "new_data"

    # Call the function to write the clean_tweets to the CSV file
    write_list_to_csv(clean_tweet, output_csv_file)
```

In [ ]:
```
# Tokenize the text
tokenizer = Tokenizer()
tokenizer.fit_on_texts(clean_tweet)
sequences = tokenizer.texts_to_sequences(clean_tweet)
```

```python
# Pad sequences to have the same length
max_sequence_length = max(len(seq) for seq in sequences)
padded_sequences = pad_sequences(sequences, maxlen=max_sequence_length)

labels = np.array(labels)

# Split the data into training, validation, and testing sets
train_ratio = 0.7
val_ratio = 0.15
test_ratio = 0.15

num_samples = len(padded_sequences)
num_train = int(train_ratio * num_samples)
num_val = int(val_ratio * num_samples)

train_X = padded_sequences[:num_train]
train_y = labels[:num_train]
val_X = padded_sequences[num_train:num_train+num_val]
val_y = labels[num_train:num_train+num_val]
test_X = padded_sequences[num_train+num_val:]
test_y = labels[num_train+num_val:]
```

In [ ]:
```python
def get_model():
    # Define the neural network model
    embedding_dim = 100

    model = Sequential()
    model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=embedding_di
    model.add(LSTM(64, return_sequences=True))  # Add LSTM layer
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    # Evaluate the model on the test set
    test_loss, test_accuracy = model.evaluate(test_X, test_y, verbose=0)
    print("Test Loss:", test_loss)
    print("Test Accuracy:", test_accuracy)

    return model

model=get_model()
# Train the model
model.fit(train_X, train_y, validation_data=(val_X, val_y), epochs=10, batch_size=32)
```

In [ ]:
```python
#model.save('final_text_model.tf')
```

```
WARNING:absl:Found untraced functions such as _update_step_xla, lstm_cell_layer_call_fn,
lstm_cell_layer_call_and_return_conditional_losses while saving (showing 3 of 3). These
functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: text_model.tf\assets
INFO:tensorflow:Assets written to: text_model.tf\assets
```

Classification with saved model

In [4]:
```python
def classification(tweets):
    print(tweets)
    tokenizer = Tokenizer()
    clean_tweet = read_tweets_from_csv(r"new_data.csv")  # Read tweets from CSV file
    tokenizer.fit_on_texts(clean_tweet)
    tweet_sequence = tokenizer.texts_to_sequences([tweets])
    padded_tweet_sequence = pad_sequences(tweet_sequence, maxlen=1403)
    model = load_model(r'final_text_model.tf')
    prediction = model.predict(padded_tweet_sequence)
    return prediction

def read_tweets_from_csv(csv_file):
    tweets = []
    with open(csv_file, 'r', encoding='utf-8') as file:
        csv_reader = csv.reader(file)
        for row in csv_reader:
            if row:
                tweets.append(row[0])
    return tweets
```

In [5]:
```
classification(" does not suppress reading any   does not ban   does not ban   does not
```

```
 does not suppress reading any   does not ban   does not ban   does not ban  and  does n
ot push    ALL of which  does   apparently you agree with
1/1 [==============================] - 1s 888ms/step
```

Out[5]:
```
array([[3.8282815e-05]], dtype=float32)
```