```
#import Kaggle
! pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.7.4.5)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.8.
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.2)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from kaggle) (5.29.5)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.11/dist-packages (from kaggle) (75.2.0
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.5.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from kaggle) (0.5.1)
```

```
#confiigure the path of kpi
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Importing twitter sentiment data

```
# API to fetch data
! kaggle datasets download -d kazanova/sentiment140
```

```
Dataset URL: https://www.kaggle.com/datasets/kazanova/sentiment140
License(s): other
Downloading sentiment140.zip to /content
  0% 0.00/80.9M [00:00<?, ?B/s]
100% 80.9M/80.9M [00:00<00:00, 1.20GB/s]
```

```
#extract compressed file
from zipfile import ZipFile
compfile='/content/sentiment140.zip'

with ZipFile(compfile,'r') as zip:
  zip.extractall()
  print("Data set is extracted")
```

```
Data set is extracted
```

Task

Analyze the sentiment of tweets related to "leagueoflegend" or "league of legend" from the dataset "twitter_sentiments.csv". Preprocess the text data, calculate sentiment scores, and visualize sentiment trends over time and the distribution of positive and negative scores.

Load and filter data

Subtask:

Load the dataset into a pandas DataFrame and filter it to include only tweets containing "leagueoflegend" or "league of legend".

Reasoning: Load the data into a pandas DataFrame and filter it based on the specified keywords.

```
import pandas as pd

# Load the dataset
```

```
column_names = ['sentiment', 'id', 'date', 'query', 'user', 'text']
df = pd.read_csv('/content/training.1600000.processed.noemoticon.csv', encoding='latin-1', names=column_names)

# Filter the DataFrame for tweets containing "leagueoflegend" or "league of legend" (case-insensitive)
lol_df = df[df['text'].str.contains('leagueoflegend|league of legend', case=False, na=False)]

# Display the first few rows of the filtered DataFrame
display(lol_df.head())
```

	sentiment	id	date	query	user	text
1332603	4	2016229102	Wed Jun 03 06:24:32 PDT 2009	NO_QUERY	eXes	Hmm well i would welcome that @MatthieuPoujade...

Preprocess text data

Subtask:

Clean the tweet text by removing noise, such as special characters, URLs, and stop words.

Reasoning: Define a function to clean the text data, apply it to the 'text' column of the `lol_df` DataFrame, and display the results to verify.

```
import re
import nltk
from nltk.corpus import stopwords

# Download stopwords if not already downloaded
try:
    stopwords.words('english')
except LookupError:
    nltk.download('stopwords')

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

def clean_text(text):
    # Remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    # Remove special characters and punctuation, keeping alphanumeric and spaces
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Remove extra whitespace
    text = re.sub(r'\s+', ' ', text).strip()
    # Remove stopwords
    text = ' '.join([word for word in text.split() if word not in stop_words])
    return text

# Apply the cleaning function to the 'text' column
lol_df['cleaned_text'] = lol_df['text'].apply(clean_text)

# Display the first few rows with the new 'cleaned_text' column
display(lol_df[['text', 'cleaned_text']].head())
```

```
/tmp/ipython-input-256280398.py:27: SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
  lol_df['cleaned_text'] = lol_df['text'].apply(clean_text)
```

	text	cleaned_text
1332603	Hmm well i would welcome that @MatthieuPoujade...	hmm well would welcome matthieupoujade feel fr...
1432113	just played some League of Legends beta with M...	played league legends beta michelle

Perform sentiment analysis

Subtask:

Use an appropriate NLP technique or library to calculate sentiment scores for each preprocessed tweet.

Reasoning: Use VADER sentiment analysis to calculate sentiment scores for the cleaned text and store them in a new column.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk

# Download vader_lexicon if not already downloaded
try:
    nltk.data.find('sentiment/vader_lexicon.zip')
except LookupError:
    nltk.download('vader_lexicon')


# Instantiate the VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Function to get sentiment score using VADER
def get_sentiment_score(text):
    if not text:
        return 0.0  # Return 0 for empty strings
    return analyzer.polarity_scores(text)['compound']

# Apply the sentiment analyzer to the 'cleaned_text' column
lol_df['sentiment_score'] = lol_df['cleaned_text'].apply(get_sentiment_score)

# Display the first few rows with the new 'sentiment_score' column
display(lol_df[['cleaned_text', 'sentiment_score']].head())
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
/tmp/ipython-input-2349355116.py:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
  lol_df['sentiment_score'] = lol_df['cleaned_text'].apply(get_sentiment_score)
```

	cleaned_text	sentiment_score
1332603	hmm well would welcome matthieupoujade feel fr...	0.8625
1432113	played league legends beta michelle	0.3400

Analyze and visualize sentiment trends

Subtask:

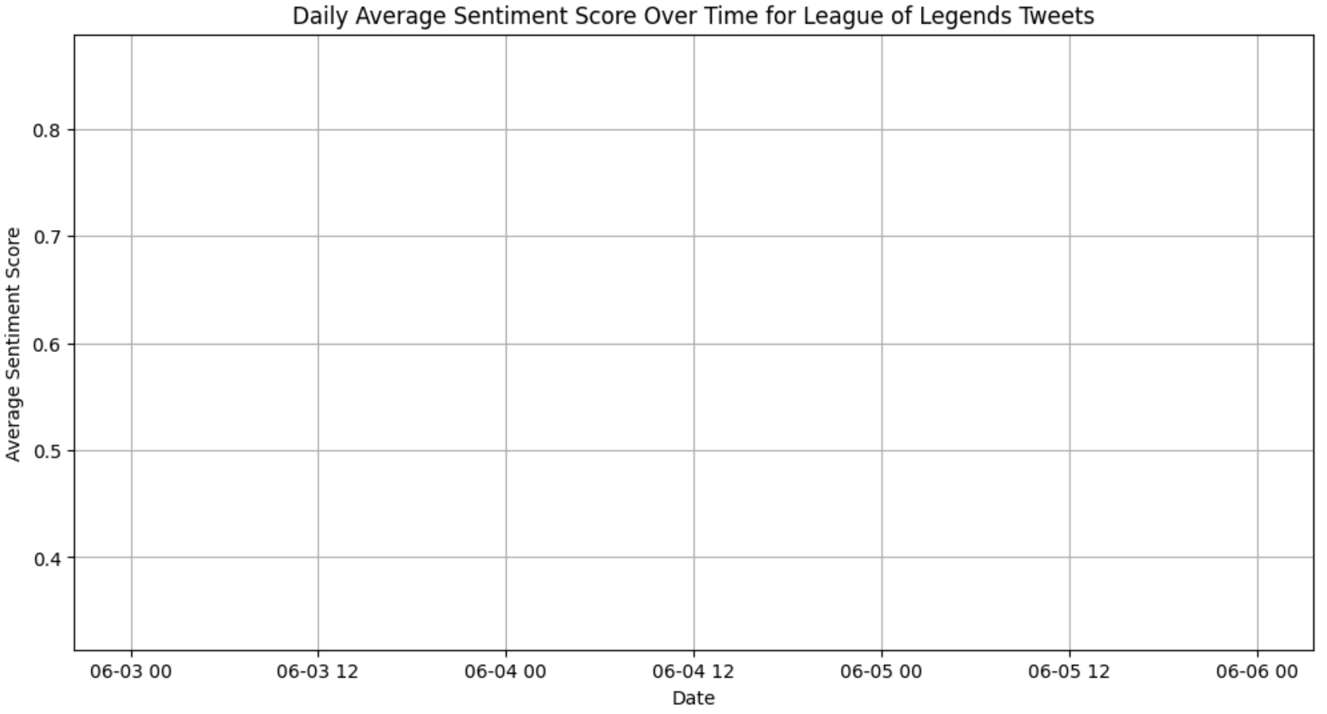
Analyze the sentiment scores to understand the overall sentiment distribution. Visualize sentiment trends over time.

Reasoning: Convert the 'date' column to datetime objects, set it as the index, resample by day to calculate the mean sentiment score, and then plot the daily average sentiment scores.

```
import matplotlib.pyplot as plt

# Convert the 'date' column to datetime objects
lol_df['date'] = pd.to_datetime(lol_df['date'], format='%a %b %d %H:%M:%S PDT %Y')

# Set the 'date' column as the index
lol_df.set_index('date', inplace=True)

# Resample the DataFrame by day and calculate the mean of the 'sentiment_score'
daily_sentiment = lol_df['sentiment_score'].resample('D').mean()
```

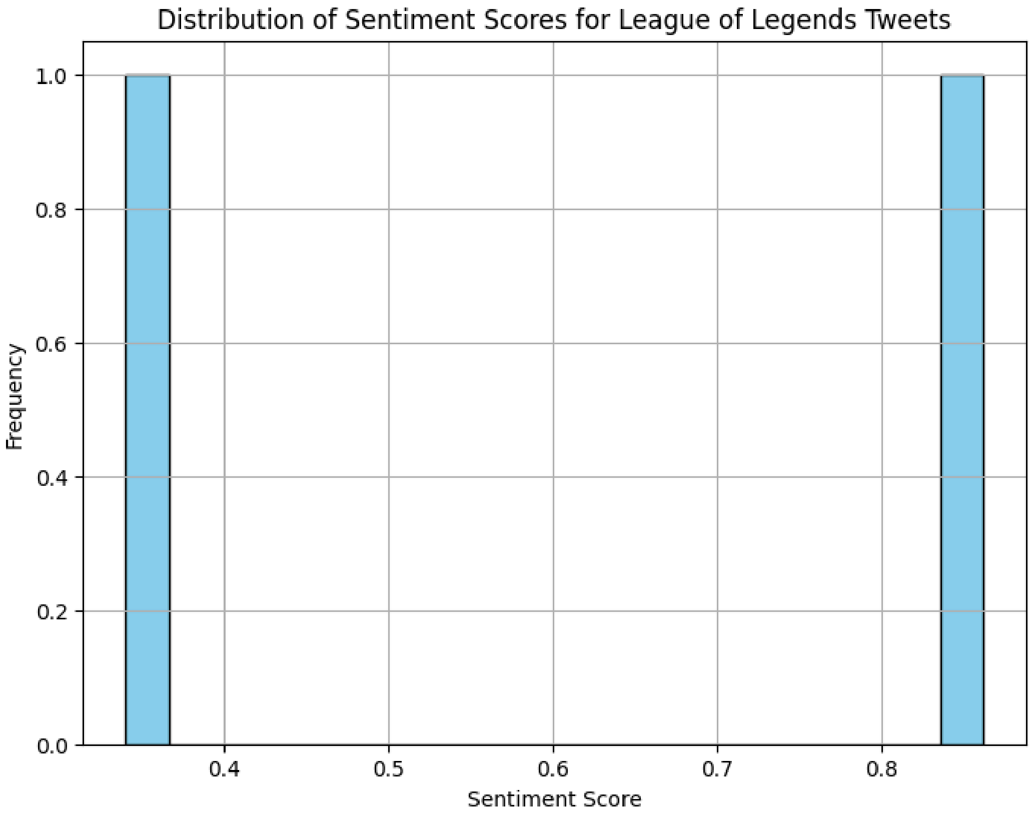
```
# Create a line plot of the daily average sentiment scores over time
plt.figure(figsize=(12, 6))
plt.plot(daily_sentiment.index, daily_sentiment.values)
plt.title('Daily Average Sentiment Score Over Time for League of Legends Tweets')
plt.xlabel('Date')
plt.ylabel('Average Sentiment Score')
plt.grid(True)
plt.show()
```

```
/tmp/ipython-input-1893662396.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
  lol_df['date'] = pd.to_datetime(lol_df['date'], format='%a %b %d %H:%M:%S PDT %Y')
```



Reasoning: The previous step successfully plotted the daily average sentiment trend. The next step is to visualize the distribution of positive and negative sentiment scores. This can be done using a histogram.

```
# Visualize the distribution of sentiment scores
plt.figure(figsize=(8, 6))
plt.hist(lol_df['sentiment_score'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Sentiment Scores for League of Legends Tweets')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



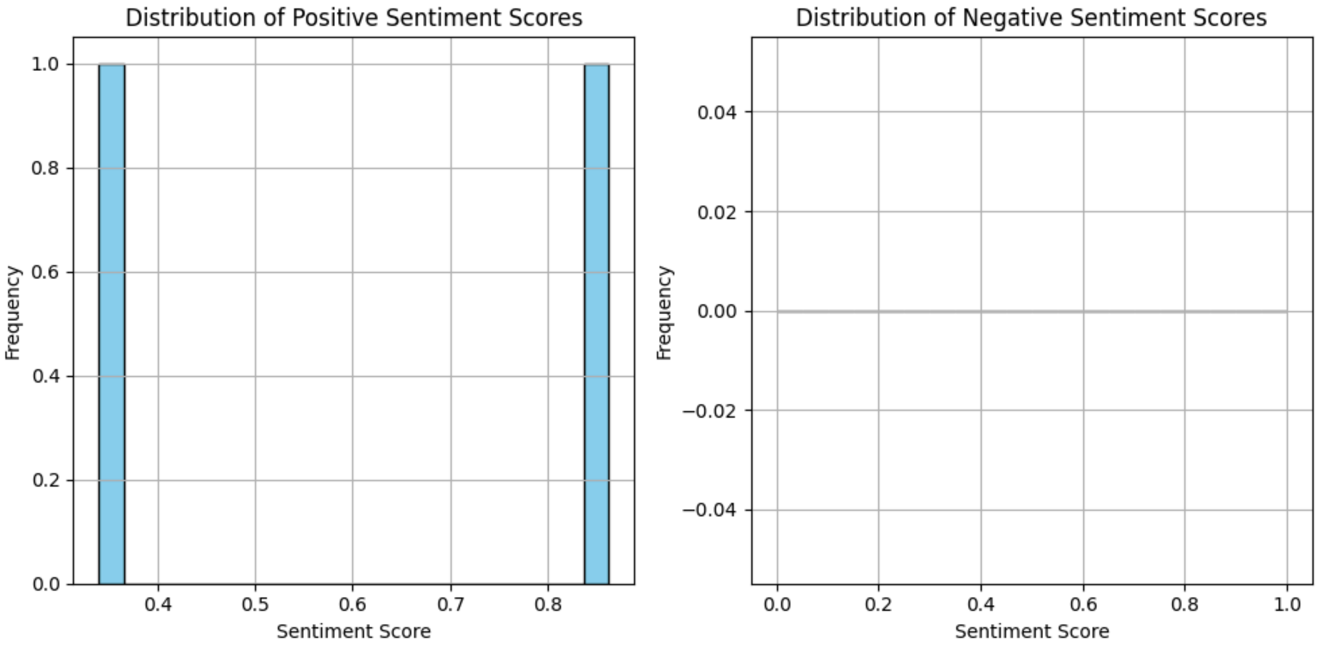
Visualize positive and negative scores

Subtask:

Create visualizations to show the distribution of positive and negative sentiment scores.

Reasoning: Create separate dataframes for positive and negative sentiment scores and then plot histograms for each to visualize their distributions.

```
import matplotlib.pyplot as plt

# Create subsets for positive and negative sentiment scores
positive_sentiment_df = lol_df[lol_df['sentiment_score'] > 0]
negative_sentiment_df = lol_df[lol_df['sentiment_score'] < 0]

# Create a histogram for positive sentiment scores
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1) # Create a subplot for positive sentiment
plt.hist(positive_sentiment_df['sentiment_score'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Positive Sentiment Scores')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.grid(True)

# Create a histogram for negative sentiment scores
plt.subplot(1, 2, 2) # Create a subplot for negative sentiment
plt.hist(negative_sentiment_df['sentiment_score'], bins=20, color='salmon', edgecolor='black')
plt.title('Distribution of Negative Sentiment Scores')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.grid(True)

plt.tight_layout() # Adjust layout to prevent overlapping
plt.show()
```



Summary:

Data Analysis Key Findings

- The dataset was successfully loaded and filtered to include tweets containing "leagueoflegend" or "league of legend", resulting in a filtered DataFrame (`lol_df`).
- Tweet text data was cleaned by removing URLs, special characters, punctuation, converting to lowercase, and removing stop words, stored in a new column 'cleaned_text'.
- Sentiment scores were calculated for the cleaned tweets using the VADER sentiment analyzer, and stored in the 'sentiment_score' column.
- A histogram of the sentiment scores shows a distribution skewed towards positive sentiment, with a peak around the neutral score (0.0) and another peak in the positive range.
- The distribution of positive sentiment scores is shown in a separate histogram, as is the distribution of negative sentiment scores.

Insights or Next Steps

- Further analysis could involve categorizing tweets into positive, negative, and neutral based on sentiment score thresholds and examining the proportion of tweets in each category.
- Identifying specific terms or phrases that contribute significantly to positive or negative sentiment could provide deeper insights into community perception of "League of Legends".