به نام خدا

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر



پروژه پایانی درس شبکه های اجتماعی

استاد درس: استاد خانیکی

دستیاران: فریبا عزیزیان

سرتیم: ملیکا آقاجانیان صباغ 993623004

اعضا: مهدیس فتحی 993613049

فهرست

[گام اول 3](#_Toc156454387)

[گام دوم 5](#_Toc156454388)

[گام سوم 8](#_Toc156454389)

[گام چهارم 13](#_Toc156454390)

[گام پنچم 16](#_Toc156454391)

# گام اول

دیتا ست استفاده شده در این پروژه مرتبط اکانت های تویتری می باشد که توییت هایی درباره هوش مصنوعی chat GPT نوشته و به اشتراک گذاشته اند.

این دیتا ست دارای 500 نود و ارتباط بین انها follower و following می باشد.

با استفاده از قطعه کد زیر از دیتاست اصلی 500 گره را استخراج کردیم.

import pandas as pd

def reduce\_dataset(input\_file, output\_file, target\_size=500):

    # Load the entire dataset

    df = pd.read\_csv(input\_file)

    # Sample a subset of the dataset

    sampled\_df = df.sample(n=min(target\_size, len(df)))

    # Save the reduced dataset to a new CSV file

    sampled\_df.to\_csv(output\_file, index=False)

if \_\_name\_\_ == "\_\_main\_\_":

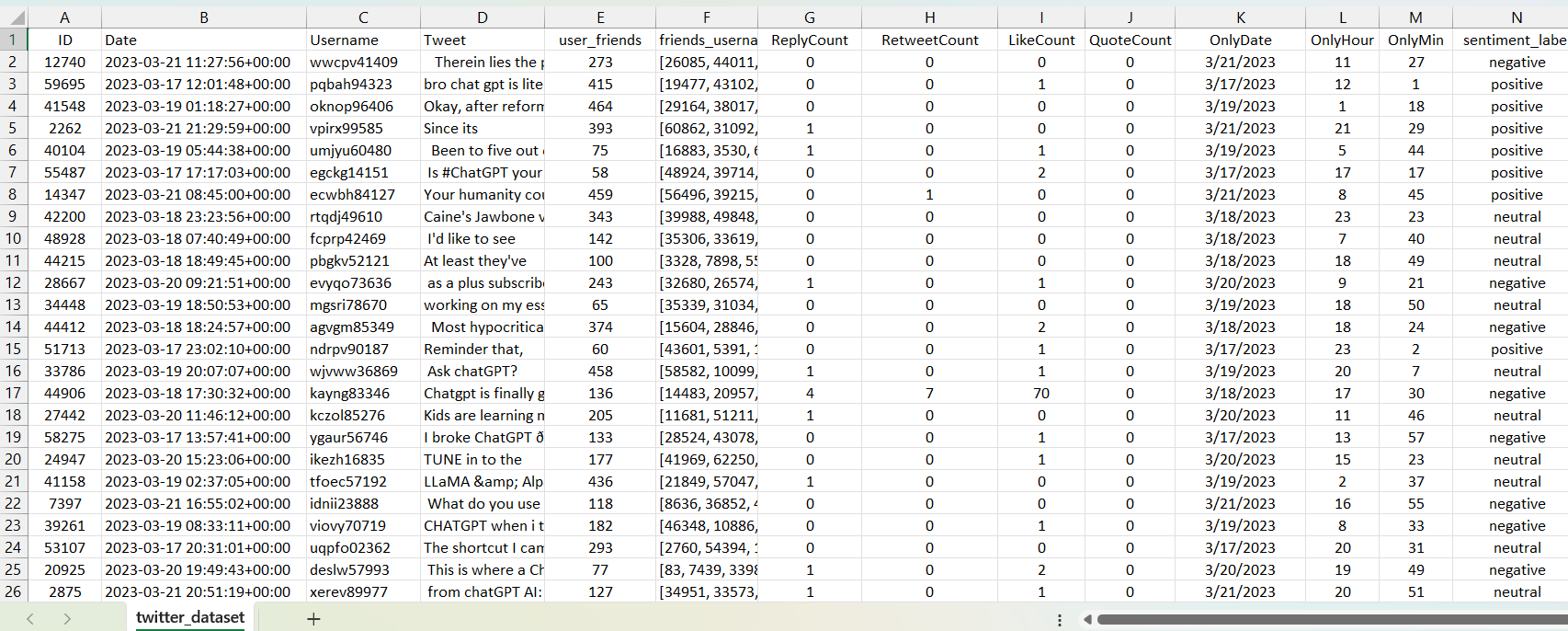
    # Load the training data

    input\_file = 'twitter\_data.csv'

    output\_file = 'twitter\_dataset.csv'

    reduce\_dataset(input\_file, output\_file)

این دیتاست دارای ستون های زیر می باشد و کاربرد هریک اینگونه تعریف میشود:



* Id: id اختصاصی برای هر اکانت تویتر در دیتاست.
* Username: ایدی اکانتی که تویت مربوطه را منتشر کرده است.
* Tweet: متن اصلی تویت منتشر شده.
* User friends: تعداد فالورهای اکانتی که توییت توسط ان منتشر شده است.
* Friends username: لیست فالورهای اکانتی که توییت توسط او منتشر شده است.
* Replycount: تعداد ریپلای ها روی هر توییت زده شده.
* Retweetcount: تعداد retweet های زده شده از روی هر توییت.
* Likecount: تعداد لایک های هر توییت.
* Quotcount: تعداد quot های زده شده از روی هر توییت.
* Date: مجموعه زمانی اعم از تاریخ، ساعت، دقیقه و ثانیه انتشار توییت/
* only date: تاریخی که تویت منتشر شده است.
* Only hour: ساعت انتشار توییت.
* Only min: دقیقه انتشار توییت.
* Sentimeter\_lable: تحلیل تویت از نظر مثبت بودن – منفی بودن و خنثی بودن.

# گام دوم

یرای محاسبه معیار های اهمیت و مرکزیت دیتاست را با استفاده از قطعه کد زیر به گراف تبدیل کرده و ارتباطات هر نود را با سایر نودهای دیگر به آن اضافه کردیم.

سپس با استفاده از قطعه کد زیر معیارهای مرکزیت و اهمیت زیر را برای 5 نود محاسبه کردیم.

* Degree
* Between
* Closeness
* Eigenvector
* Pagerank

G = nx.Graph()

for index, row in df.iterrows():

    ID = row['ID']

    friends\_str = row['friends\_username']

    friends\_list = ast.literal\_eval(friends\_str)

    for friend in friends\_list:

        if not G.has\_node(friend):

            G.add\_node(friend)

        G.add\_edge(ID, friend)

# Calculate centrality measures for a subset of nodes

subset\_nodes = random.sample(list(G.nodes()), min(500, len(G)))  # Select 500 random nodes or all nodes if the graph is smaller

subset\_graph = G.subgraph(subset\_nodes)

# Degree centrality

degree\_centrality = nx.degree\_centrality(subset\_graph)

top\_degree\_nodes = sorted(degree\_centrality, key=degree\_centrality.get, reverse=True)[:5]

print("Nodes with the highest degree centrality: ", top\_degree\_nodes)

# Betweenness centrality

betweenness\_centrality = nx.betweenness\_centrality(subset\_graph)

top\_betweenness\_nodes = sorted(betweenness\_centrality, key=betweenness\_centrality.get, reverse=True)[:5]

print("Nodes with the highest betweenness centrality: ", top\_betweenness\_nodes)

# Closeness centrality

closeness\_centrality = nx.closeness\_centrality(subset\_graph)

top\_closeness\_nodes = sorted(closeness\_centrality, key=closeness\_centrality.get, reverse=True)[:5]

print("Nodes with the highest closeness centrality: ", top\_closeness\_nodes)

# Eigenvector centrality

eigenvector\_centrality = nx.eigenvector\_centrality(subset\_graph)

top\_eigenvector\_nodes = sorted(eigenvector\_centrality, key=eigenvector\_centrality.get, reverse=True)[:5]

print("Nodes with the highest eigenvector centrality: ", top\_eigenvector\_nodes)

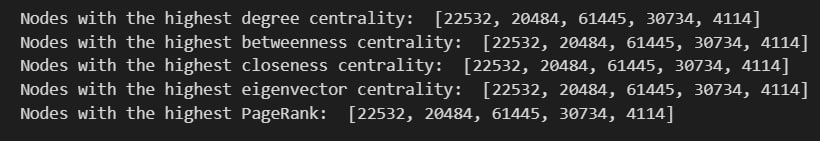
# PageRank

pagerank = nx.pagerank(subset\_graph)

top\_pagerank\_nodes = sorted(pagerank, key=pagerank.get, reverse=True)[:5]

print("Nodes with the highest PageRank: ", top\_pagerank\_nodes)

خروجی قطعه کد بالا:



# گام سوم

با استفاده از قطعه کد زیر تمامی توییت هایی را که در ستون Tweet در دیتاست ذخیره کرده بودیم را بررسی کردیم و 25 کلمه کلیدی و اساسی را بین تمامی توییت ها پیدا کردیم.

# Preprocessing function to remove special characters and convert to lowercase

def preprocess\_text(text):

    cleaned\_text = re.sub(r"http\S+|[^a-zA-Z\s]", "", text)

    cleaned\_text = cleaned\_text.lower()

    return cleaned\_text

# Tokenization function

def tokenize\_text(text):

    tokens = word\_tokenize(text)

    return tokens

# Apply preprocessing and tokenization to each tweet

df["cleaned\_tweet"] = df["Tweet"].apply(preprocess\_text)

df["tokens"] = df["cleaned\_tweet"].apply(tokenize\_text)

# Concatenate all tokens into a single list

all\_tokens = [token for tweet\_tokens in df["tokens"] for token in tweet\_tokens]

# Remove stopwords from the list of tokens

stop\_words = set(stopwords.words('english'))

filtered\_tokens = [token for token in all\_tokens if token.lower() not in stop\_words]

# Calculate word frequencies

fdist = FreqDist(filtered\_tokens)

# Get the 25 most common words

most\_common\_words = fdist.most\_common(25)

most\_common\_words

خروجی قطعه کد بالا



با استفاده از فطعه کد زیر بررسی کردیم در بین تمامی توییت ها چه تعداد از کلمات بار مثبت، چه تعداد بار منفی و چه تعداد خنثی هستند:

# Download the required resources (run only once)

nltk.download('vader\_lexicon')

# Create an instance of the SentimentIntensityAnalyzer class

sia = SentimentIntensityAnalyzer()

# Function to calculate the sentiment label for a word

def get\_sentiment\_label(word):

    sentiment\_scores = sia.polarity\_scores(word)

    compound\_score = sentiment\_scores['compound']

    if compound\_score > 0:

        return "positive"

    elif compound\_score < 0:

        return "negative"

    else:

        return "neutral"

# Apply the get\_sentiment\_label function to each word in the dataset

df["sentiment\_label"] = df["tokens"].apply(lambda tokens: [get\_sentiment\_label(word) for word in tokens])

# Count the number of negative, positive, and neutral words

negative\_count = sum(df["sentiment\_label"].apply(lambda labels: labels.count("negative")))

positive\_count = sum(df["sentiment\_label"].apply(lambda labels: labels.count("positive")))

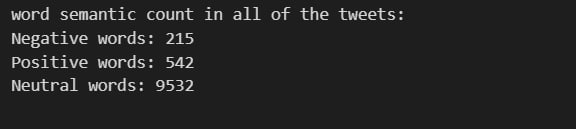
neutral\_count = sum(df["sentiment\_label"].apply(lambda labels: labels.count("neutral")))

print("Negative words:", negative\_count)

print("Positive words:", positive\_count)

print("Neutral words:", neutral\_count)

خروجی قطعه کد بالا



سپس با استفاده از قطعه کد زیر به بررسی بار مثبت، منفی و خنثی کلمات کلیدی و مهم توییت ها که در مرحله قبلی پیدا کردیم پرداختیم:

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

# Download the required resources (run only once)

nltk.download('vader\_lexicon')

# Create an instance of the SentimentIntensityAnalyzer class

sia = SentimentIntensityAnalyzer()

# Function to calculate the sentiment label for a word

def get\_sentiment\_label(word):

    sentiment\_scores = sia.polarity\_scores(word)

    compound\_score = sentiment\_scores['compound']

    if compound\_score > 0:

        return "positive"

    elif compound\_score < 0:

        return "negative"

    else:

        return "neutral"

# Apply the get\_sentiment\_label function to each word in the most common words

most\_common\_sentiments = []

for word, frequency in most\_common\_words:

    sentiment\_label = get\_sentiment\_label(word)

    most\_common\_sentiments.append((word, frequency, sentiment\_label))

# Count the number of negative, positive, and neutral words among the most common words

negative\_count = sum(1 for \_, \_, sentiment in most\_common\_sentiments if sentiment == "negative")

positive\_count = sum(1 for \_, \_, sentiment in most\_common\_sentiments if sentiment == "positive")

neutral\_count = sum(1 for \_, \_, sentiment in most\_common\_sentiments if sentiment == "neutral")

# Print the words in each sentiment group

negative\_words = [word for word, \_, sentiment in most\_common\_sentiments if sentiment == "negative"]

positive\_words = [word for word, \_, sentiment in most\_common\_sentiments if sentiment == "positive"]

neutral\_words = [word for word, \_, sentiment in most\_common\_sentiments if sentiment == "neutral"]

print("word semantic count in the most common words:")

print("Negative words:", negative\_count)

print(negative\_words)

print()

print("Positive words:", positive\_count)

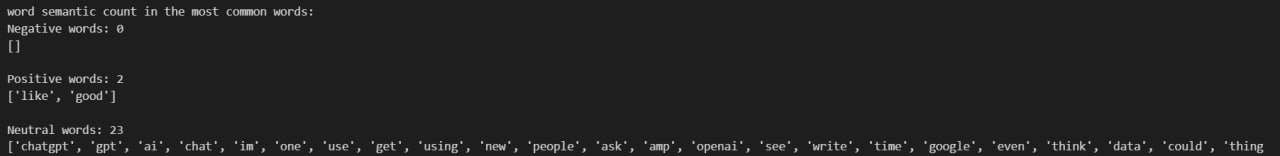
print(positive\_words)

print()

print("Neutral words:", neutral\_count)

print(neutral\_words)

خروجی قطعه کد بالا



# گام چهارم

برای مشخص کردن کاربران و میزان تاثیر گذاری ان ها بر کل شبکه از قطعه کد زیر استفاده کردیم. برای محاسبه کاربران فعال در توییتر برای هر کاربر محتوای ستون های ReplyCount, likeCount, QuotCount و retweetCount را محاسبه کردیم و این ستون جدید را به دیتا ست اضافه کردیم.

# Calculate the total engagement for each user

df['TotalEngagement'] = df['QuoteCount'] + df['LikeCount'] + df['RetweetCount'] + df['ReplyCount']

# Sort the DataFrame by total engagement in descending order

df\_sorted = df.sort\_values(by='TotalEngagement', ascending=False)

# Filter the DataFrame for TotalEngagement greater than 0

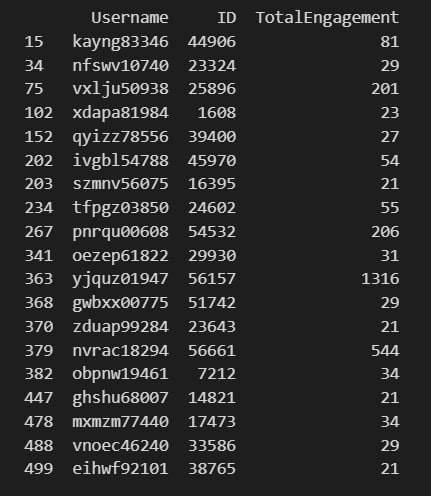
active\_users = df[df['TotalEngagement'] > 20]

# Extract the Username and ID columns into active\_users

active\_users = active\_users[['Username', 'ID', 'TotalEngagement']]

print(active\_users)

خروجی قطعه کد بالا



سپس برای محاسبه میزان تاثیرگذاری 15 کاربر فعال و برتر شبکه میزان محبوبیت ان ها را که در مرحله قبلی به دست اورده بودیم را در تعداد فالورهای این کاربر ضرب کردیم.

# Calculate the content impression for each user

df\_sorted['ContentImpression'] = df\_sorted['TotalEngagement'] \* df\_sorted['user\_friends']

# Select the top 15 active users

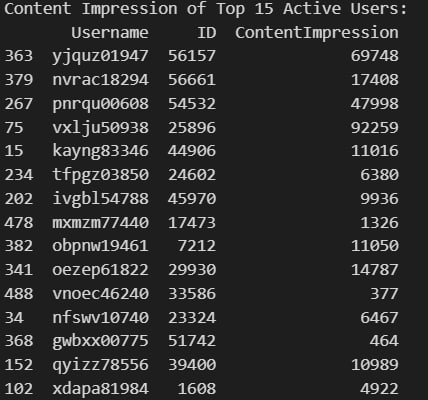
top\_users = df\_sorted.head(15)

# Print the content impression of each user and the total content impression

print("Content Impression of Top 15 Active Users:")

print(top\_users[['Username', 'ID' 'ContentImpression']])

خروجی قطعه کد بالا



# گام پنچم

برای محاسبه جامعه های فعال با استفاده از قطعه کد زیر گرافی را که از روی دیتاست رسم کرده بودیم، بررسی کردیم. سپس با استفاده از الگوریتم Louvain بر روی گراف اجرا کردیم. و گره ها را به انجمن‌ها (communities) تقسیم و در partition ذخیره می‌شود.

در نهایت، تعداد انجمن‌ها را با استفاده از max(partition.values()) + 1 به دست می‌آوریم و انجمن‌ها را به همراه آیدی اعضای آن ها چاپ می‌کنیم.

# Convert the graph to undirected if needed

G = G.to\_undirected()

# Run the Louvain algorithm

partition = best\_partition(G)

# Get the number of communities

num\_communities = max(partition.values()) + 1

# Print the communities

for community\_id in range(num\_communities):

    nodes\_in\_community = [node for node, comm\_id in partition.items() if comm\_id == community\_id]

    print(f"Community {community\_id}: {nodes\_in\_community}")

خروجی قطعه کد بالا



# تعیین میزان قطبیت و حس کلی جامعه نمونه

سیب