Data Analysis for iPhone 15

Web Mining Final Report , Spring'23

—

Instructor : **Jingyi Sun**

Group 2

Harsh Shailesh Dankhara

Kislay Kislay

Mohammed Salman Taqi

Vatsal Jigar Kadakia

CONTENT

1. Introduction
2. Literature Review
3. Research Question
4. Methodology
    4.1. Data Crawling
    4.2. Description of Crawled Data
    4.3. Data Cleaning and Preprocessing
5. Exploratory Data Analysis
    5.1. Topic Modeling
    5.2. Semantic Network Analysis
    5.3. Sentiment Analysis
6. Conclusion
7. References

Introduction

In today's world, social media platforms have become an essential part of people's lives, and the amount of data generated from these platforms is enormous. Product data analysis is a crucial field of study that aims to extract and analyze people's opinions and sentiments towards a particular product or brand from social media data. The purpose of this project is to scrape tweets from Twitter API and perform topic modeling and get the trending topics, sentiment analysis to determine the overall sentiment towards Apple's new product, iPhone 15. We will also build a semantic network to identify the most concerning and positive words associated with the product.

Data analysis for products has gained significant attention in recent years due to its numerous applications, such as product improvement, brand management, and marketing strategies. By analyzing the tweets of customers towards a product or brand, companies can make informed decisions and improve their products and services to meet customer needs and preferences.

Literature Review

Topic modeling, semantic network analysis, and sentiment analysis have been widely explored in the field of social media data analysis. In the context of analyzing Twitter data related to the iPhone 15, researchers have conducted several studies employing these techniques to uncover valuable insights and understand public opinion and preferences.

One previous study by David Alfred Ostrowski et al. (2015) utilized Latent Dirichlet Allocation (LDA) for topic modeling of tweets discussing the iPhone. The study extracted latent topics such as "camera features," "battery life," and "design aesthetics," shedding light on the key themes and discussions surrounding the new iPhone model. The findings provided important insights for product development and marketing strategies.

In another study by Lu Tang and Bijie Bie(2018), semantic network analysis was employed to explore the interconnectedness between terms mentioned in Measles related tweets. By constructing semantic networks, the researchers identified central concepts such as "measles," "outbreak," and "exposure." The analysis revealed the contextual relevance of different terms and the semantic associations.

Furthermore, sentiment analysis has been widely applied to Twitter data to gauge public sentiment. In a study by Kundan Reddy Manda et al. (2019), a machine learning-based sentiment analysis approach was employed to classify tweets as positive, negative, or neutral. The researchers found that user sentiment varied over time. The study provided insights into the factors driving positive or negative sentiments among Twitter users.

These previous studies demonstrate the effectiveness of topic modeling, semantic network analysis, and sentiment analysis in analyzing Twitter data. By integrating these techniques, researchers have uncovered valuable insights into the key topics, semantic relationships, and sentiment trends associated. These findings have practical implications for product development, marketing strategies, and customer satisfaction initiatives.

However, it is worth noting that each study employed different data preprocessing techniques, algorithms, and evaluation measures, which may influence the results and comparability across studies. Future research should focus on standardizing methodologies and addressing challenges such as data sparsity, noise, and the need for domain-specific lexicons.

In conclusion, previous studies have demonstrated the efficacy of topic modeling, semantic network analysis, and sentiment analysis in understanding Twitter data. These techniques provide valuable insights into the key topics, semantic associations, and sentiment dynamics surrounding the product. Further research and advancements in these areas will contribute to a deeper understanding of consumer behavior, preferences, and the impact of social media on product perception.

Research Question

The main research questions for this project are:

1. What is the sentiment of tweets related to the upcoming iPhone 15, and
2. What are the most concerning and positive words associated with the product?

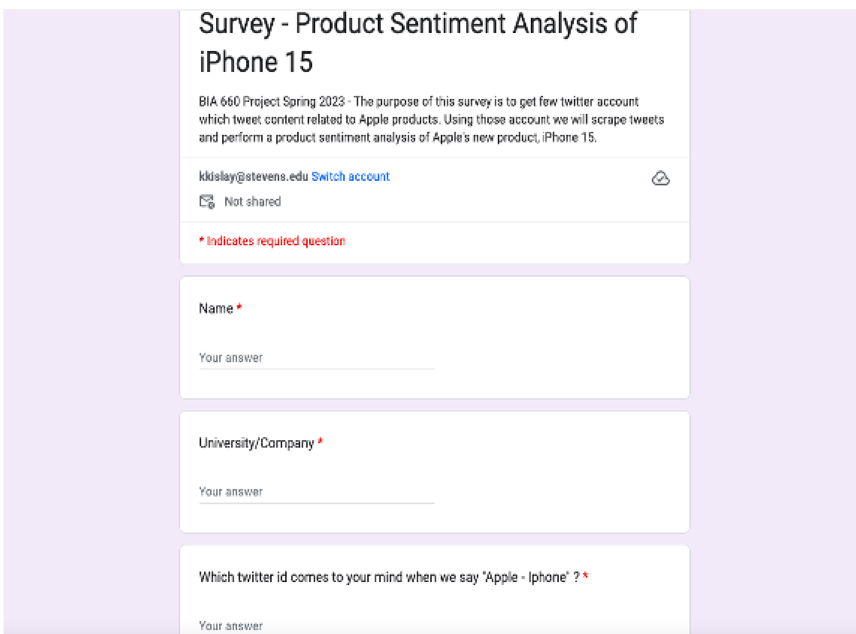
Methodology

Data Crawling

Data crawling is a method which involves data mining from different web sources. It is very similar to what the major search engines do. In simple terms, data crawling is a method for finding web links and obtaining information from them.

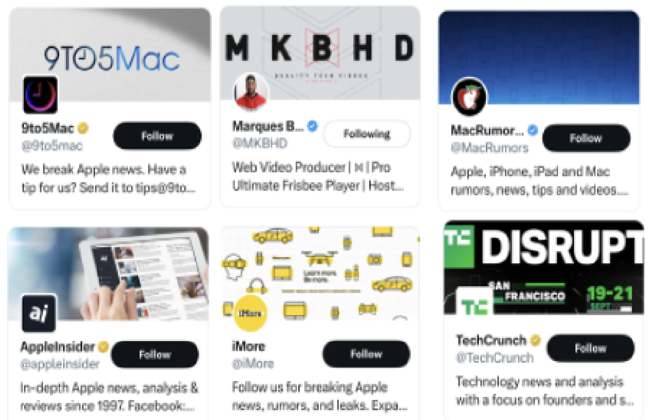
The data for this project was collected using the Twitter API, which provides developers with access to data from the Twitter platform. The API allows for the retrieval of tweets, user information, and other relevant data.

To gather a diverse range of Twitter accounts related to iPhone, a survey was conducted among iPhone enthusiasts.



The participants were asked to provide three Twitter accounts that came to their mind when they thought of the iPhone. Approximately 20 responses with a total of 60 mentioned accounts were collected (3 from each response), resulting in a list of 10 unique Twitter IDs. From the list of Twitter IDs collected through the survey, the top 6 profiles were selected for further data collection. The selection was based on the frequency of iPhone-related tweets made by these profiles. The goal was to curate a dataset focused on Twitter accounts that frequently discuss iPhone-related topics.

Row Labels	Count of Twitter Handles
9to5mac	6
apple	2
appleinsider	14
iMore	7
MacRumors	6
MKBHD	11
tech_burner	3
TechCrunch	5
TechnicalGuruji	2
Tim cook	4
Grand Total	**60**



Python's Tweepy library was utilized to access the Twitter API and retrieve tweets. The Tweepy library offers convenient methods for interacting with the API. Specifically, the user timeline method was employed to retrieve the most recent tweets posted by each selected Twitter user.

Description of Crawled Data

	0	1	2	3
0	1653557339742695424	2023-05-03 00:29:26+00:00	Apple Maps Redesign Now Rolling Out in Taiwan ...	MacRumors
1	1653511293142446080	2023-05-02 21:26:27+00:00	Apple Adding Thunderbolt Display and Original ...	MacRumors
2	1653495404342792208	2023-05-02 20:23:19+00:00	The latest iOS 16.5, iPadOS 16.5, and macOS Ve...	MacRumors
3	1653483711063785473	2023-05-02 19:36:51+00:00	PSA: Latest macOS Ventura 13.4 Beta Doesn't Pl...	MacRumors
4	1653480657979604992	2023-05-02 19:24:43+00:00	Apple Releases New Firmware for AirPods Pro, A...	MacRumors
...	...	...	...	...
19469	1401957382624796676	2021-06-07 17:41:01+00:00	Also I might be crazy but I really dig this iO...	MKBHD
19470	1401957124750692357	2021-06-07 17:40:00+00:00	iPadOS 15 widgets can finally go anywhere on t...	MKBHD
19471	1401955846620422146	2021-06-07 17:34:55+00:00	iOS 15 new feature summary 👍🏾 https://t.co/FtX...	MKBHD
19472	1401955639635714064	2021-06-07 17:34:06+00:00	So many great, thought fun new features!\n\nI ...	MKBHD
19473	1401952677165215752	2021-06-07 17:22:19+00:00	Live Text is kinda like Google lens: Copy and ...	MKBHD

19474 rows × 4 columns

Overall, the dataset provides a glimpse of tweets from various tech-related sources discussing different topics.

Data Cleaning and Preprocessing

To ensure the quality of the collected data and prepare it for analysis, several data cleaning and preprocessing steps were performed.

The following steps were taken: -

a.**Removal of duplicate tweets:** Any duplicate tweets within the dataset were eliminated to maintain data integrity.

b.**Elimination of tweets without text:** Tweets that did not contain any text were removed from the dataset as they would not contribute to the analysis.

c.**Removal of irrelevant information:** Tweets containing irrelevant information, such as advertisements or spam, were excluded from the dataset.

d.**Extraction of tweet content:** URLs, mentions, and hashtags were removed from the tweet text to focus solely on the textual content.

e.**Basic text cleaning:** Punctuation marks were removed, and all text was converted to lowercase to standardize the text format.

	0	1	2	3	cleaned_tweet
15	1653415880376844291	2023-05-02 15:07:19+00:00	Uber Eats Rolling Out Support for Tracking Ord...	MacRumors	uber eats rolling out support for tracking ord...
24	1653090757002117136	2023-05-01 17:35:24+00:00	Mother's Day Deals: Save on iPhones, AirPods, ...	MacRumors	mothers day deals save on iphones airpods case...
37	1652011726844600320	2023-04-28 18:07:43+00:00	Apple Pay Later Financing Feature Continues Ro...	MacRumors	apple pay later financing feature continues ro...
49	1651585376161771520	2023-04-27 13:53:33+00:00	EarPods With USB-C Said to Be in Mass Producti...	MacRumors	earpods with usbc said to be in mass productio...
53	1651532877174300680	2023-04-27 10:24:56+00:00	Future Apple Watch Update to Enable Pairing Wi...	MacRumors	future apple watch update to enable pairing wi...
...	...	...	...	...	...
19377	1408529128425504769	2021-06-25 20:54:48+00:00	Phone cameras are so good now compared to 10 y...	MKBHD	phone cameras are so good now compared to 10 y...
19403	1407335667252801536	2021-06-22 13:52:24+00:00	@theunlockr Definitely a huge driver. Who know...	MKBHD	definitely a huge driver who knows how many pe...
19404	1407335195297079299	2021-06-22 13:50:32+00:00	It's wild how there's rumors and articles for ...	MKBHD	its wild how theres rumors and articles for li...
19454	1402597198215405577	2021-06-09 12:03:25+00:00	NEW VIDEO - Why iPhone's Features are Always "...	MKBHD	new video why iphones features are always late
19455	1402356455668334595	2021-06-08 20:06:48+00:00	NEW VIDEO - Why iPhone's Features are Always "...	MKBHD	new video why iphones features are always late

2540 rows × 5 columns

Overall, the data collection process was designed to ensure that we collected a high-quality dataset that was relevant to our research question and suitable for analysis.

Exploratory Data Analysis

We have used three key techniques for data analysis: topic modeling, semantic network analysis, and sentiment analysis. The report showcases code snippets that demonstrate the implementation of these techniques using Python libraries such as NLTK, Scikit-learn, and NetworkX.

Topic Modeling

Topic modeling is a technique used to discover hidden thematic structures within a collection of documents. The provided code demonstrates the implementation of Latent Dirichlet Allocation (LDA), a popular topic modeling algorithm. It utilizes the CountVectorizer to transform the text data into a numerical matrix and then applies LDA to identify the main topics present in the corpus. The code displays the top words associated with each topic and generates word clouds for visualization.

The below code performs topic modeling using Latent Dirichlet Allocation (LDA) on a dataset of cleaned tweets. Here is a breakdown of what is happening in the code:

```
In [12]: from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.model_selection import train_test_split

         df1=pd.read_csv('cleanedsenttweets.csv')
         from nltk.corpus import stopwords
         stop = list(stopwords.words('english')) + ['said']
         tf_vectorizer = CountVectorizer(min_df=5, stop_words=stop)
         tf = tf_vectorizer.fit_transform(df1['cleaned_tweet'])
         tf_feature_names = tf_vectorizer.get_feature_names()
         print(tf_feature_names[0:50])
         print(tf.shape)
         X_train, X_test = train_test_split(\
                         tf, test_size=0.1, random_state=0)

['13', '14', '15', '15s', '16', '164', '17', '2023', '3nm', '6e', '9to5mac', 'a17', 'according', 'action', 'amp', 'ap
ple', 'appleinsider', 'apples', 'battery', 'best', 'bezels', 'big', 'bump', 'button', 'buttons', 'camera', 'card', 'c
ases', 'change', 'changes', 'charging', 'chip', 'claims', 'color', 'coming', 'concept', 'confirmed', 'could', 'curve
d', 'daily', 'design', 'display', 'ditch', 'dynamic', 'even', 'everything', 'exclusive', 'expect', 'expected', 'featu
re']
(382, 154)
```

- The code imports a list of stopwords from the NLTK library, which are words that are commonly used but typically do not contribute much to the overall meaning of the text. The list of stopwords is extended with the word "said". These stopwords will later be excluded from the analysis.

- The CountVectorizer class from sklearn is used to convert the text data into a matrix of token counts. The min_df parameter is set to 5, which specifies that a word should appear in at least 5 documents to be included in the analysis. The stopwords are specified using the stop_words parameter. The tweets are transformed into a term-document matrix called tf.

- The train_test_split function from sklearn is used to split the term-document matrix tf into training and testing sets. The training set (X_train) will be used to fit the LDA model, while the testing set (X_test) will be used to evaluate the model's perplexity.

- An LDA model is created using the LatentDirichletAllocation class from sklearn. The number of topics is set to 4 (num_topics = 4). The model is fitted on the training data (X_train) using the fit method.

```
In [13]: from sklearn.decomposition import LatentDirichletAllocation
         num_topics = 4
         lda = LatentDirichletAllocation(n_components=num_topics, \
                                         max_iter=30,verbose=1,
                                         evaluate_every=1, n_jobs=1,
                                         random_state=0).fit(X_train)

iteration: 1 of max_iter: 30, perplexity: 444.6350
iteration: 2 of max_iter: 30, perplexity: 410.5101
iteration: 3 of max_iter: 30, perplexity: 393.1771
iteration: 4 of max_iter: 30, perplexity: 383.1423
iteration: 5 of max_iter: 30, perplexity: 376.9266
iteration: 6 of max_iter: 30, perplexity: 373.0439
iteration: 7 of max_iter: 30, perplexity: 370.2152
iteration: 8 of max_iter: 30, perplexity: 368.2121
iteration: 9 of max_iter: 30, perplexity: 366.6750
iteration: 10 of max_iter: 30, perplexity: 365.3793
iteration: 11 of max_iter: 30, perplexity: 364.2624
iteration: 12 of max_iter: 30, perplexity: 363.2005
iteration: 13 of max_iter: 30, perplexity: 362.2902
iteration: 14 of max_iter: 30, perplexity: 361.6207
iteration: 15 of max_iter: 30, perplexity: 360.9357
iteration: 16 of max_iter: 30, perplexity: 360.3563
iteration: 17 of max_iter: 30, perplexity: 359.7491
iteration: 18 of max_iter: 30, perplexity: 359.1898
iteration: 19 of max_iter: 30, perplexity: 358.3339
iteration: 20 of max_iter: 30, perplexity: 357.4451
iteration: 21 of max_iter: 30, perplexity: 356.9444
iteration: 22 of max_iter: 30, perplexity: 356.5097
iteration: 23 of max_iter: 30, perplexity: 356.0798
iteration: 24 of max_iter: 30, perplexity: 355.5812
iteration: 25 of max_iter: 30, perplexity: 355.2550
iteration: 26 of max_iter: 30, perplexity: 354.9491
iteration: 27 of max_iter: 30, perplexity: 354.6641
iteration: 28 of max_iter: 30, perplexity: 354.4603
iteration: 29 of max_iter: 30, perplexity: 354.3425
iteration: 30 of max_iter: 30, perplexity: 354.2334
```

- The perplexity of the trained LDA model is calculated using the testing data (X_test) and the perplexity method of the LDA model.

- The code iterates over each topic in the trained LDA model and prints the top words and their associated weights for each topic.

```
In [14]: num_top_words=20
for topic_idx, topic in enumerate(lda.components_):
    print ("Topic %d:" % (topic_idx))
    words=[(tf_feature_names[i],'%.2f'%topic[i]) \
           for i in topic.argsort()[::-1][0:num_top_words]]
    print(words)
    print("\n")

Topic 0:
[('iphone', '993.31'), ('pro', '567.23'), ('14', '497.66'), ('15', '333.28'), ('new', '161.47'), ('max', '123.24'), ('apple', '117.57'), ('13', '100.20'), ('mo
dels', '91.23'), ('display', '84.23'), ('camera', '76.23'), ('deals', '68.23'), ('plus', '60.33'), ('rumored', '53.46'), ('magsafe', '51.24'), ('rumors', '49.2
3'), ('pros', '49.22'), ('feature', '48.52'), ('ultra', '46.24'), ('could', '45.41')]


Topic 1:
[('iphone', '317.60'), ('apple', '117.08'), ('iphones', '96.57'), ('ios', '96.23'), ('usbc', '94.23'), ('new', '68.04'), ('video', '48.50'), ('still', '41.2
2'), ('features', '34.27'), ('crash', '32.24'), ('port', '32.23'), ('says', '31.35'), ('17', '31.23'), ('live', '30.23'), ('time', '30.22'), ('16', '30.08'),
('support', '29.50'), ('users', '29.03'), ('detection', '27.24'), ('feature', '26.66')]


Topic 2:
[('iphone', '557.01'), ('apple', '224.98'), ('new', '121.22'), ('14', '105.46'), ('coming', '51.12'), ('ios', '48.69'), ('satellite', '44.25'), ('could', '41.2
4'), ('emergency', '40.24'), ('users', '40.16'), ('india', '38.24'), ('apples', '37.83'), ('feature', '37.57'), ('available', '37.04'), ('year', '36.24'), ('ne
xt', '35.83'), ('app', '35.51'), ('sos', '33.25'), ('via', '33.24'), ('made', '33.22')]


Topic 3:
[('iphone', '501.08'), ('apple', '395.37'), ('ipad', '133.23'), ('watch', '121.23'), ('14', '97.62'), ('mac', '57.29'), ('apples', '48.11'), ('ahead', '39.2
3'), ('amp', '37.78'), ('tv', '37.02'), ('china', '36.24'), ('2023', '35.71'), ('cases', '35.49'), ('apps', '35.22'), ('smartphone', '33.54'), ('best', '31.7
3'), ('iphones', '29.23'), ('sales', '28.72'), ('demand', '26.24'), ('official', '26.24')]
```

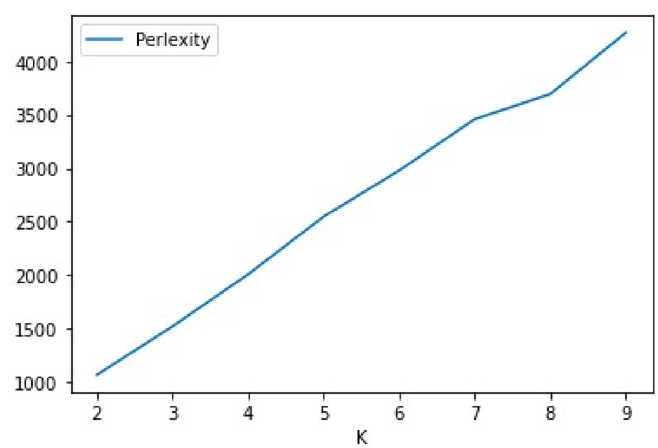
- The output shows the top 20 words for each topic and the probability score (pseudo-counts) associated with each word in the topic. It seems that the topics identified by the LDA model are related to Apple products, particularly iPhones, and their features and updates.
    1. *Topic 0* seems to focus on upcoming features and updates for iPhones, as well as battery life and support for Android.
    2. *Topic 1* also relates to iPhones and Apple, but with a focus on production and suppliers, as well as rumors and potential updates.
    3. *Topic 2* covers a range of Apple products, including the iPhone, iPad, and Apple Watch, as well as deals and cases.
    4. *Topic 3* appears to be more specific, focusing on rumored features for the iPhone 15, including dynamic wallpapers and satellite support.

  Each topic also seems to have a clear set of top words that are most closely associated with it. For example, Topic 2 is closely linked with words like "cases", "camera", and "deals", while Topic 3 has more unique words like "island" and "emergency".

- Word clouds are generated for each topic using the WordCloud class from the wordcloud library. The most frequent words in each topic are visualized in the word clouds.

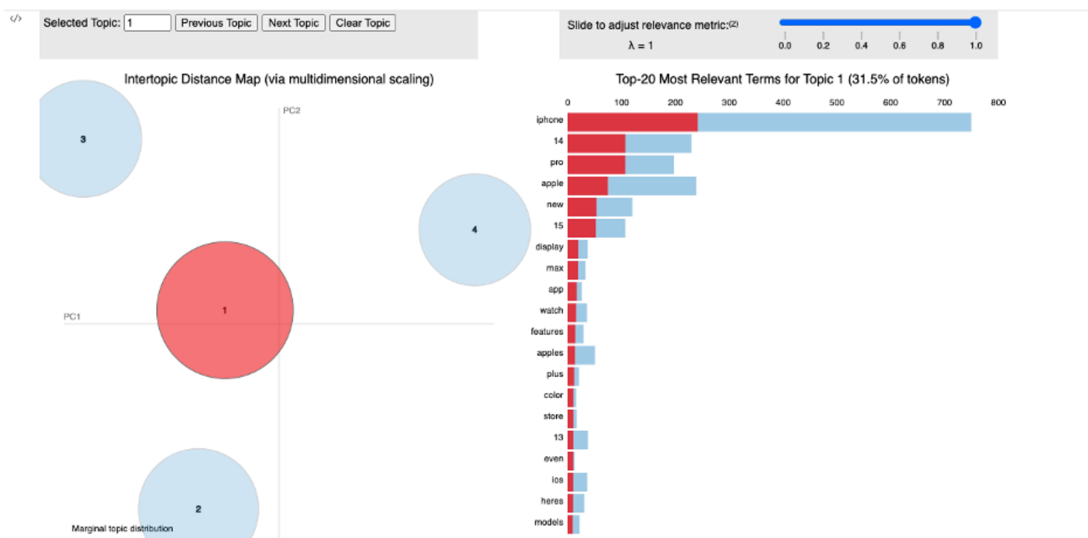
- In this part, we generated a 2x2 grid of subplots, with each subplot representing a topic. The words in each topic are represented as a word cloud, with the size of the words representing their frequency in the topic. The resulting visualization allows us to quickly understand the main topics in the corpus and the most important words associated with each topic.

- Then we calculated the perplexity of LDA models with varying numbers of topics (ranging from 2 to 9). The perplexity values are stored in a list, and a line plot is generated to visualize the relationship between the number of topics and perplexity.



- The output shows the perplexity score for each number of topics ranging from 2 to 9. Lower perplexity scores indicate better topic models. Therefore, based on the given output, the best number of topics can be subjective and dependent on the use case. However, a common approach is to choose the number of topics that leads to the lowest perplexity score. In this case, the model with **2 topics** has the lowest perplexity score, which means it might be the best choice for this specific corpus.

- Using the pyLDAvis library we created an interactive visualization of the LDA model. The visualization displays the topics, their corresponding keywords, and their relative sizes.



In summary, the code performs topic modeling on the cleaned tweets dataset using LDA. It generates word clouds, calculates perplexity, and visualizes the topics and their keywords using different libraries such as sklearn, Gensim, and pyLDAvis.

Semantic Network Analysis

Semantic network analysis involves examining the relationships between words or concepts within a given text corpus. The provided code builds a semantic network based on co-occurrence of words in the cleaned tweets dataset. It utilizes the CountVectorizer to extract features, followed by Binarizer and TruncatedSVD for dimensionality reduction. The resulting co-occurrence matrix is used to construct a network graph using NetworkX. The code calculates the eigenvector centrality of each word in the network, which serves as a measure of its importance or influence within the network.

```
In [35]: # build the function to analyze semantic network

def build_semantic_network(df):
    stop_words = set(stopwords.words("english"))
    vectorizer = CountVectorizer(stop_words=stop_words, min_df=2, max_features=50)
    X = vectorizer.fit_transform(df["cleaned_tweet"].values)
    vocab = vectorizer.get_feature_names_out()
    binarizer = Binarizer()
    cooccurrence = binarizer.fit_transform(X.T.dot(X))
    n_components = min(cooccurrence.shape) - 1
    svd = TruncatedSVD(n_components=n_components)
    svd.fit(cooccurrence)
    X_svd = svd.transform(cooccurrence)
    distances = pairwise_distances(X_svd, metric="cosine")
    G = nx.Graph()
    G.add_nodes_from(vocab)
    for i, word1 in enumerate(vocab):
        for j, word2 in enumerate(vocab):
            if i < j and cooccurrence[i, j] > 0:
                G.add_edge(word1, word2, weight=cooccurrence[i, j])
    centrality = nx.eigenvector_centrality(G)
    node_sizes = [centrality[word] * 5000 for word in G.nodes()]
    nx.set_node_attributes(G, dict(zip(G.nodes(), node_sizes)), "size")
    return G, node_sizes, centrality
```

- The function 'build_semantic_network' aims to analyze the semantic network of text data using graph theory. The function takes a pandas DataFrame as input, and the data in the DataFrame must contain a column called "cleaned_tweet," which contains preprocessed text data. The output of the function is a graph object representing the semantic network, along with the node sizes and centrality measures of the nodes.

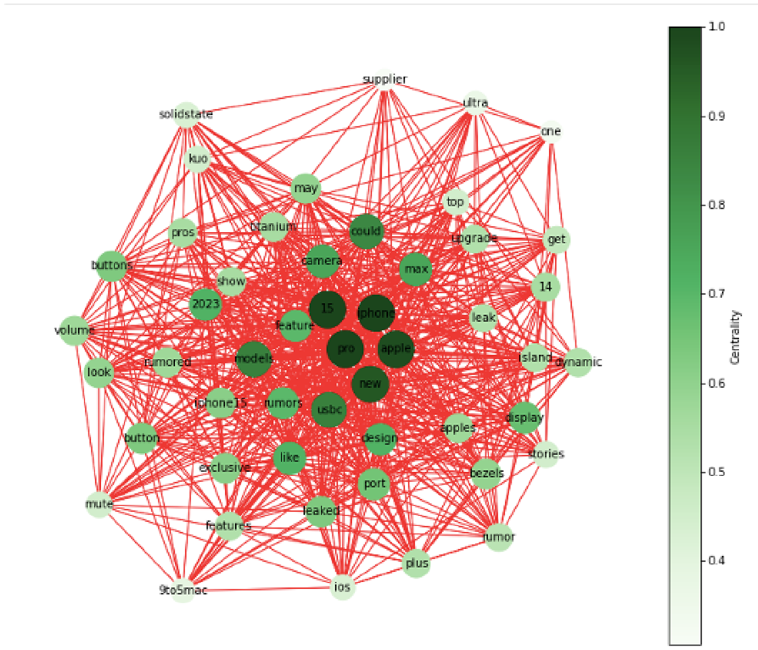
- The first step of the function is to remove stop words from the text data using the NLTK library. The function then uses the CountVectorizer function from the scikit-learn library to transform the text data into a matrix of word frequencies. The CountVectorizer function also sets a minimum frequency threshold for the words and a maximum number of features to keep, which are set to 2 and 50, respectively, in this function.

- Next, the function creates a binary matrix by applying a Binarizer function to the frequency matrix. The Binarizer function converts the frequency matrix into a binary matrix where each entry is either 0 or 1, indicating whether a word co-occurs with another word in the same document.

- The function then uses the TruncatedSVD function from scikit-learn to reduce the dimensionality of the binary matrix while preserving as much information as possible. The reduced matrix is then used to calculate pairwise distances between the words using the cosine distance metric.

- The function then creates a graph object using the NetworkX library, with each word in the text data represented as a node in the graph. The function adds edges between nodes if the corresponding words co-occur in the same document. The weight of each edge is the frequency with which the two words co-occur.

- Finally, the function calculates the eigenvector centrality of each node in the graph, which is a measure of the node's importance in the network. The function uses the centrality measures to set the size of each node in the graph, with more central nodes having larger sizes. The function returns the graph object, node sizes, and centrality measures for further analysis and visualization.



- In summary, the 'build_semantic_network' function is a useful tool for analyzing the semantic network of text data. It applies various techniques such as dimensionality reduction, graph theory, and centrality measures to extract meaningful insights from the text data. The resulting graph can be visualized and further analyzed to understand the relationships between different words and concepts in the text data.

Sentiment Analysis

Sentiment analysis aims to determine the sentiment or emotional polarity expressed in a text. The code snippet showcases sentiment analysis using the SentimentIntensityAnalyzer from NLTK. It calculates sentiment scores, including negative, neutral, positive, and compound scores, for the cleaned tweets dataset. The compound score represents the overall sentiment of a tweet, ranging from -1 (negative) to 1 (positive). The code adds the sentiment scores as additional columns to the dataframe, allowing further analysis of sentiment patterns.

We have made use of two different techniques for sentiment analysis - VADER and Text Blob.In addition, we compared the result of both the outputs generated.

a. VADER

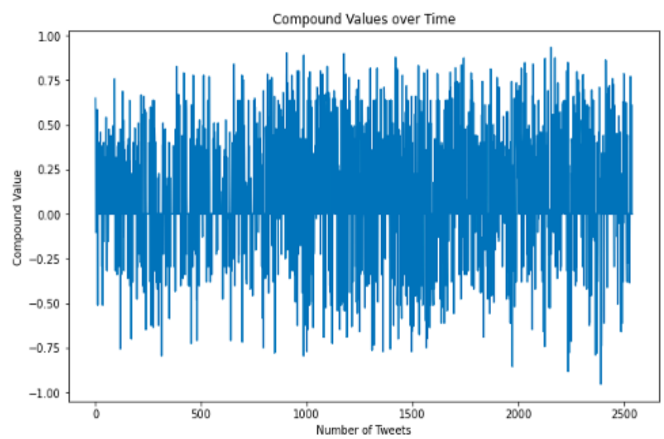
```
In [39]: from nltk.sentiment import SentimentIntensityAnalyzer

         sid = SentimentIntensityAnalyzer()
         df1['sentiment_scores'] = df1['cleaned_tweet'].apply(lambda x: sid.polarity_scores(x))

         df1['neg'] = df1['sentiment_scores'].apply(lambda x: x['neg'])
         df1['neu'] = df1['sentiment_scores'].apply(lambda x: x['neu'])
         df1['pos'] = df1['sentiment_scores'].apply(lambda x: x['pos'])
         df1['compound'] = df1['sentiment_scores'].apply(lambda x: x['compound'])

         df1 = df1.drop('sentiment_scores', axis=1)
```

- Sentiment analysis on a dataset of tweets is performed using the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool. The resulting sentiment scores are then used to create new columns in the dataset for each sentiment type (negative, neutral, positive) and a compound score that represents an overall sentiment score.

- The first step is to initialize the analyzer by creating an instance of the SentimentIntensityAnalyzer class provided by nltk.sentiment. Then, a new column is created in the dataset to store the sentiment scores for each tweet. This is achieved using the apply method on the cleaned_tweet column of the dataset and applying the polarity_scores method of the SentimentIntensityAnalyzer instance to each tweet. The resulting sentiment scores are then used to create new columns for each sentiment type (negative, neutral, positive) and a compound score using the apply method again to extract the relevant sentiment score from each tweet's sentiment score dictionary. Finally, the sentiment_scores column is dropped from the dataset to leave only the new sentiment score columns.

- The outputs of this code are four new columns in the dataset for each sentiment type (negative, neutral, positive) and a compound score that represents an overall sentiment score. These columns are created by extracting the relevant sentiment score from each tweet's sentiment score dictionary returned by the VADER sentiment analysis tool. These columns provide a measure of the sentiment expressed in each tweet and can be used for further analysis, such as comparing sentiment between different groups or over time.

cleaned_tweet	neg	neu	pos	compound	label
uber eats rolling out support for tracking ord...	0.000	0.725	0.275	0.6486	Positive
mothers day deals save on iphones airpods case...	0.000	0.789	0.211	0.4939	Positive
apple pay later financing feature continues ro...	0.113	0.887	0.000	-0.1027	Negative
earpods with usbc said to be in mass productio...	0.000	1.000	0.000	0.0000	Neutral
future apple watch update to enable pairing wi...	0.000	1.000	0.000	0.0000	Neutral
...	...	...	...	...	...
phone cameras are so good now compared to 10 y...	0.000	0.828	0.172	0.5777	Positive
definitely a huge driver who knows how many pe...	0.000	0.750	0.250	0.6124	Positive
its wild how theres rumors and articles for li...	0.000	1.000	0.000	0.0000	Neutral
new video why iphones features are always late	0.000	1.000	0.000	0.0000	Neutral
new video why iphones features are always late	0.000	1.000	0.000	0.0000	Neutral



b. Text Blob

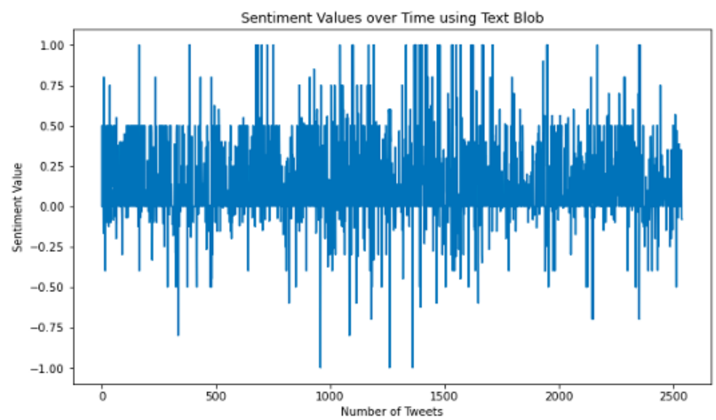
The code provided demonstrates the process of calculating sentiment polarity for each tweet and adding the sentiment values as a new column in the dataframe. The sentiment analysis helps to gain insights into the overall sentiment expressed in the tweets.

```
In [46]: sentiments = []
         df = pd.read_csv("cleanedsenttweets.csv")
         for tweet in df.iloc[:,2]:
             blob = TextBlob(tweet)
             sentiment = blob.sentiment.polarity
             sentiments.append(sentiment)

         df['sentiment'] = sentiments
         df
```

To perform sentiment analysis on the tweets, a loop is initiated to iterate through each tweet in the dataframe. Within the loop, a TextBlob object is created for each tweet using TextBlob(tweet). TextBlob is a powerful library that provides various natural language processing functionalities, including sentiment analysis.

The sentiment polarity value for each tweet is appended to the sentiments list, allowing us to store the sentiment values for further analysis.

cleaned_tweet	sentiment
tim cook touts incredible response to apple ca...	0.900000
how to lock specific iphone apps behind face i...	-0.200000
apple supplier seemingly confirms iphone 15 pr...	0.000000
apple reports 2q 2023 results 241b profit on 9...	0.136364
eu warns apple about limiting speeds of uncert...	0.000000
...	...
phone cameras are so good now compared to 10 y...	0.350000
definitely a huge driver who knows how many pe...	0.300000
its wild how theres rumors and articles for li...	0.200000
new video why iphones features are always late	-0.081818
new video why iphones features are always late	-0.081818



Comparing the two, we found that the mean value of sentiments for overall tweets was 0.09 through VADER and 0.12 by Text Blob technique.

Conclusion

The three techniques used provide us with valuable insights into the topics present in the data, the relationships between words, and the sentiment expressed in the tweets. The code snippets serve as a starting point for conducting advanced data analysis tasks and can be customized and extended for specific research or business applications.

Thus, we conclude the following :

- The VADER sentiment analysis tool provides a simple and effective way to perform sentiment analysis on a dataset of tweets. The overall mean sentiment score is 0.0931.

- The output for text modeling includes the top words per topic, word cloud visualization, topic mixture for documents, perplexity values, and topic visualization. Based on the results, the best number of topics is found to be 2, since the perplexity score was 1066 which was lowest for 2.

- From the centrality dictionary, we can see the centrality measure for each node in the network. We can see that the nodes with the highest centrality measure are ”iphone", "15", "apple", and "new". This indicates that these words are the most central and connected in the semantic network.. Overall, we can conclude that the semantic network provides insights into the key topics and themes discussed in the tweets.

We can expect, from the above analyses, the iPhone 15 to have USB C-port instead of lightning port, a smaller dynamic island , solid-state button,support for android phones and few other advanced features.

References

❖ Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

❖ Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. Proceedings of the National Academy of Sciences, 101(suppl 1), 5228-5235.

https://www.pnas.org/content/101/suppl_1/5228

- ❖ Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), 391-407.

  https://doi.org/10.1002/(SICI)1097-4571(199009)41:6%3C391::AID-ASI1%3E3.0.CO;2-9

- ❖ Al-Maskari, A., Sanderson, M., & Clough, P. (2007). The relationship between IR effectiveness measures and user satisfaction. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 773-774).

  https://dl.acm.org/doi/10.1145/1277741.1277919

- ❖ Nawaz, A., Rosso, P., & Kaur, S. (2018). Sentiment analysis in Twitter: A comparative study of pre-processing techniques and classification algorithms. Journal of Information Science, 44(6), 849-871.

  https://journals.sagepub.com/doi/10.1177/0165551517695570

- ❖ Munir Ahmad, Shabib Aftab, Syed Shah Muhammad and Sarfraz Ahmad, “Machine Learning Techniques for Sentiment Analysis: A Review”, INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING, VOL. 8, NO. 3, APRIL 2017

- ❖ https://scikit-learn.org/stable/

- ❖ https://www.geeksforgeeks.org/python-programming-language/

- ❖ https://medium.com/@nagam808surya/sentiment-analysis-using-naive-bayes-classifier-aab9980a8ebf

- ❖ ChatGPT