

**Python Text Modeling Project Report: Group 4**

Submitted by:

Nitisha Adhikari | Yash Sharma | Yanhui Chen | Ishant Nayer | Nitish Puri

**Brief Introduction to Accessing Twitter Data**

There are different Application Program Interfaces (APIs) that can be used for accessing data on Twitter. The four main objects that are encountered in these APIs are:

* *Tweets*: The basic building block of Twitter, tweets represent the status updates performed by users. They can be embedded, replied to, liked, unliked and deleted[[1]](#footnote-1). There are many fields that this object can contain, such as:
  + *coordinates*: Represents the geographic location of the tweet
  + *created\_at*: UTC time when the Tweet was created
  + *favorite\_count*: Indicates how many times this tweet has been liked
  + *id*: Tweet identifier; a signed 64 bit integer is used for storing this
  + *text:* The actual text of the status update
* *User:* A user can be a person, organization, or some other entity. The users can perform actions such as Tweeting, will have a timeline, and can be followed, searched or mentioned[[2]](#footnote-2). Some of the fields for this object are:
  + *created\_at:* UTC time when the user account was created
  + *favourites\_count:* Number of Tweets liked in the account’s lifetime
  + *followers\_count:* The current number of followers that this account has
  + *friends\_count:* The numbers of users this account is following
  + *id:* User identifier; a signed 64 bit integer is used for storing this
* *Entities:* These provide meta data about the Tweets. Some of the fields used for this object are[[3]](#footnote-3):
  + *hashtags:* This represents the hashtags which are parsed out of the Tweet text.
  + *urls:* Represents URLs included in the text of a Tweet
  + *user\_mentions*: Represents other Twitter users mentioned in the Tweet text
* *Places:* Places represent specific locations that have some geographic coordinates. These locations do not necessarily have to be the locations from which the Tweet was issued, but could also be a place that the Tweet is about. Places are searchable. Some of the fields comprising this object are as follows[[4]](#footnote-4):
  + *bounding\_box:* A bounding box of objects which encloses this place
  + *country:* The country that contains this place
  + *full\_name:* Full name of the place (e.g. Cincinnati, OH)
  + *place\_type:* The type of location (e.g. city)
  + *id:* Place identifier; this is represented as a string and not as an integer

As mentioned earlier, there are different Twitter APIs. A description of each has been given below:

REST APIs

These provide access to read and write Twitter data. These enable us to create Tweets, and read user profiles and follower data[[5]](#footnote-5).

Streaming APIs

These allow developers to quickly access the global stream of Tweet data. These transmit lesser information than the REST APIs, and the client is basically notified whenever a Tweet or some other event has occurred. There are *Public streams*, which are streams of public data flowing through Twitter. These are appropriate for data mining, or following specific users or topics. *User streams* track all of the data related to a single user[[6]](#footnote-6).

Webhook APIs

These provide realtime access to data for an account. After proper configuration of a destination URL, Twitter begins directly posting data to the URL whenever a relevant activity occurs[[7]](#footnote-7).

Twitter Ads API

This allows advertisers to create custom tools and manage their Twitter Ad campaigns.

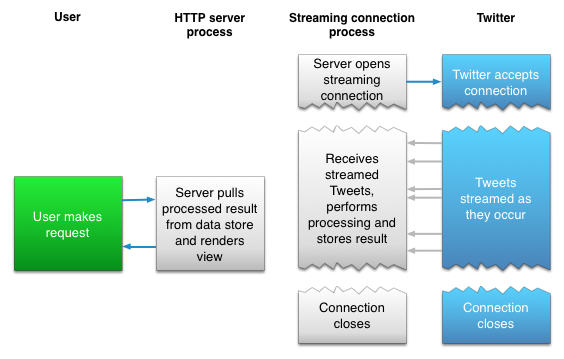
**REST APIs – Data Access Method for this Project**

We access the REST APIs for getting data on Tweets. For performing searches, or reading additional user profile information, these are the appropriate APIs to use. We have used *tweepy* to get the data in Python, which provides access to the REST APIs. *OAuth* is used for authentication of requests. With respect to Rate Limiting, the documentation states that “The rate limit window is divided into 15 minute chunks per endpoint, with most individual calls allowing for 15 requests in each window”[[8]](#footnote-8).

The Search API, which is a part of the REST API, allows us to search for recent or popular tweets. The API allows us to search against a sampling of recent Tweets published in the past 7 days. *Tweepy* allows us to access this and directly fetch Tweets using *API.Search(q,..)*; q represents the search query string. In general, query operators work as demonstrated by some of these examples[[9]](#footnote-9):

|  |  |
| --- | --- |
| Operator | Finds Tweets... |
| watching now | containing both “watching” and “now”. This is the default operator. |
| “happy hour” | containing the exact phrase “happy hour”. |
| love OR hate | containing either “love” or “hate” (or both). |
| beer -root | containing “beer” but not “root”. |
| #haiku | containing the hashtag “haiku”. |
| from:interior | sent from Twitter account “interior”. |
| list:NASA/astronauts-in-space-now | sent from a Twitter account in the NASA list astronauts-in-space-now |
| to:NASA | a Tweet authored in reply to Twitter account “NASA”. |
| @NASA | mentioning Twitter account “NASA”. |

**DATA STREAMING**

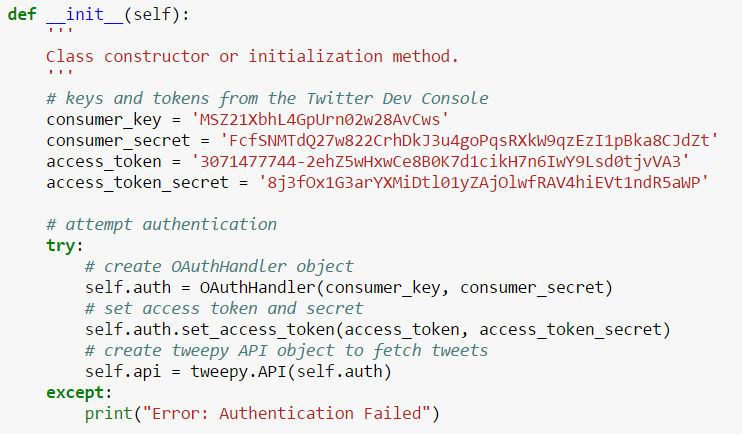


We used the above-mentioned API to extract live tweets from twitter. The above diagram explains the process overview. We use a user key to create a connection between the Twitter servers and the local computer using the twitter API. After establishing the connection, we stream the tweets which are happening live and then process them as and when required. Then the connection is closed.

We have three modules/functions, which are:

* Connection initialization: This function uses the unique user key, access tokens which are associated with a user account and establishes a connection with the twitter server
* Get Tweets: Fetches the tweets related to a keyword/handle
* Get Tween Sentiments: Uses the textblob library available for python 2 and 3, to classify each tweet into a positive, negative or neutral sentiment
* Clean Tweets: Cleans each tweet and remove special characters, links using regular expressions

Below I have explained each of these modules and their workings:



The above codes show that the API needs 4 attributes to connect a local verified machine to the twitter server. The four are:

* Consumer Key
* Consumer secret code
* Access token
* Access token secret code

Each of these 4 attributes are attached to a unique user account and can verify and authenticate the user/local machine to access some restricted data.

We also have an exception handler incase our code failed. We will get an error saying “Error: Authentication Failed” in case the function is unable to authenticate and create a connection.

**Get Tweets**



Following are the two inputs which the user needs to give to extract live tweets:

1. Twitter Handle Name
2. Number of tweets

Once the user gives these two arguments and calls the function, the following processes takes place:

1. The function creates an empty list, which will be used to store the elements of the tweets. This list is called “tweets”. As each attribute of a tweet is collected or fetched, it is appended into this list
2. Then we call the twitter API which fetched the required number of tweets, and store them into a temporary variable called “fetched\_tweets”
3. Then, we parse each tweet one by one. We store two attributes of the tweet, the text and the sentiment. We also make sure that if a tweet has been retweeted then it should not come into our database as a duplicate
4. Then we return the collected tweets
5. All this while, we have an exception handler which would output the error encountered if any

**Yes**

Append to database

Discard Tweet

Create empty tweet list

Fetch tweets from Twitter

Check if repeated tweet

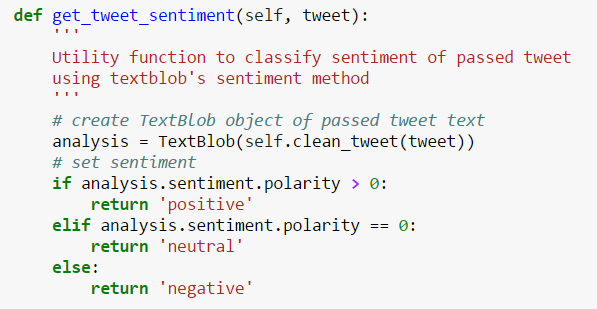
Tweet Database

For each tweet

**No**

**Tweet Sentiment**

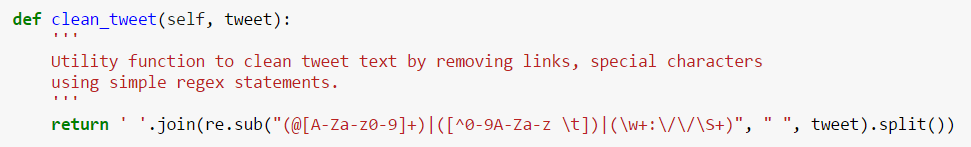
To categorize a tweet as positive/negative/neutral tweet, we again have a function called “get\_tweet\_sentiment” which runs on each tweet parsed by the API. Below is the snapshot of the function and the explanation of its working:



Here we are using the textblob python library which is used for work processing. We find out the polarity of each tweet using the NLP API. The textblob library classifies the text and assigns a polarity number to each of the text/tweets.

* A polarity of more than 0 signifies a positive tweet
* A negative polarity signifies a negative tweet
* While 0 polarity is assumed to be neutral tweet

**Clean Tweets**



The above code shows the snippet where we clean the tweets and return the text without special characters and links.

**Main function**



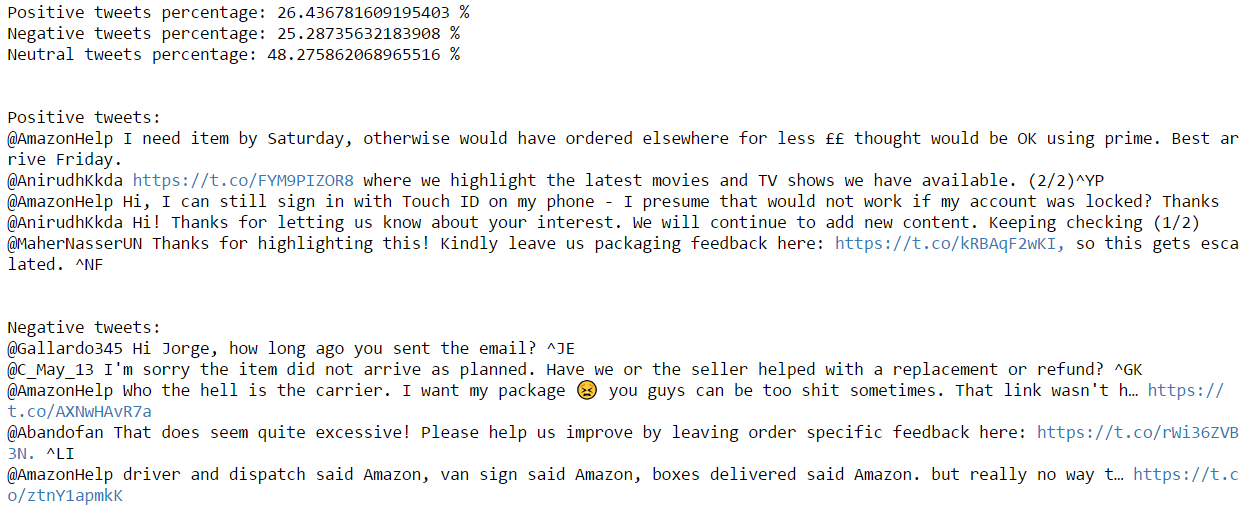
Main function is the one which calls each of the above listed functions and handles the master dataset containing the tweets and their sentiments.

We also display the percentage of positive, negative and neutral tweets. And then we display some examples of the positive and negative tweets which were classified using the textblob library.

**Sentiment Analysis for various Handles**

**Amazon Help**

This is a specific twitter handle which Amazon uses only to cater to customer requests, complaints and communications. We queried 2500 most recent tweets for this handle and these are the results:



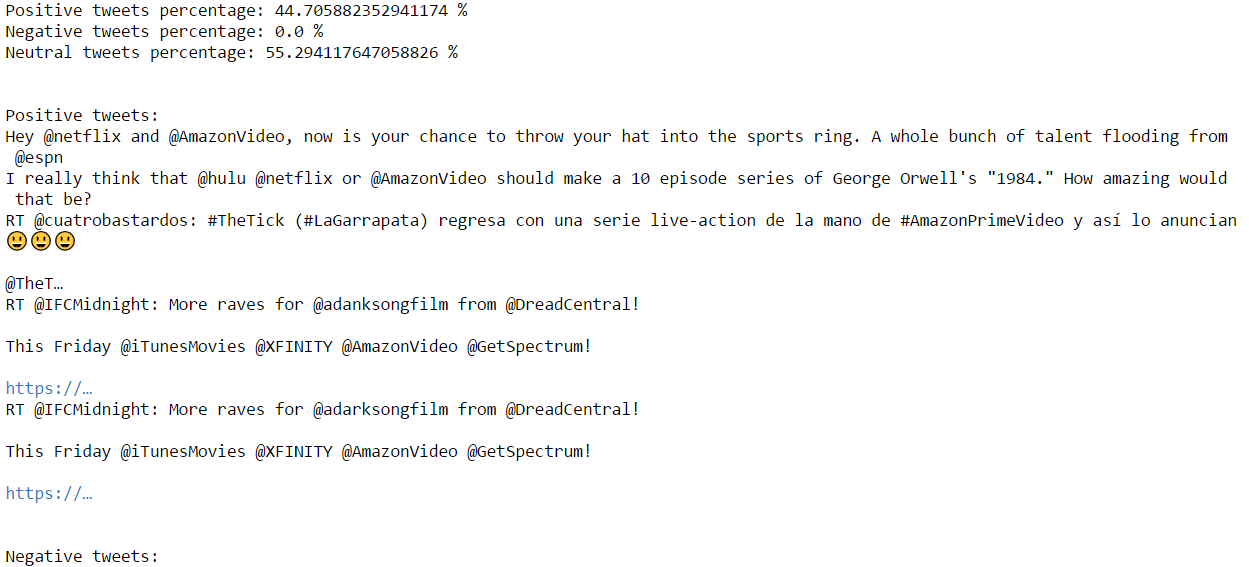
The results above show that approximately 26% of all tweets were classified as positive while 25% of all tweets were classified as negative. Hence this twitter handle has equal number of positive and negative tweets.

**Amazon Music**

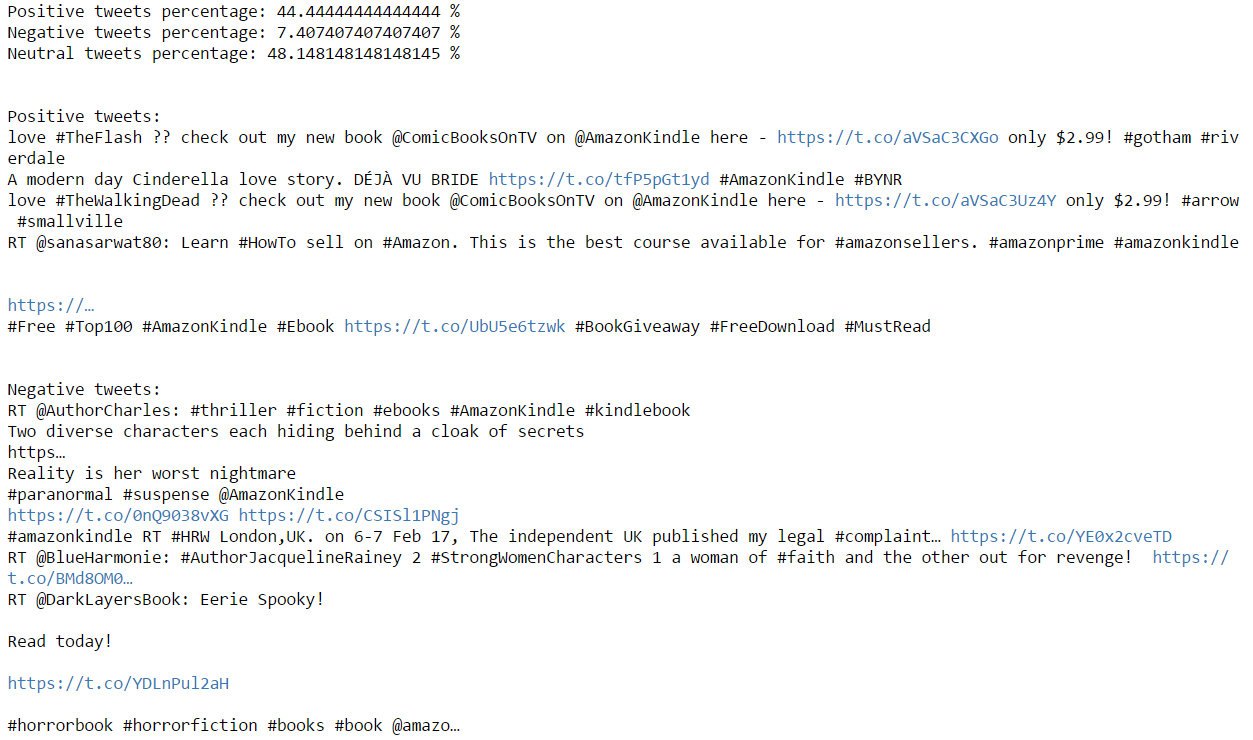


Amazon music has less negative sentiments as compared to Amazon help.

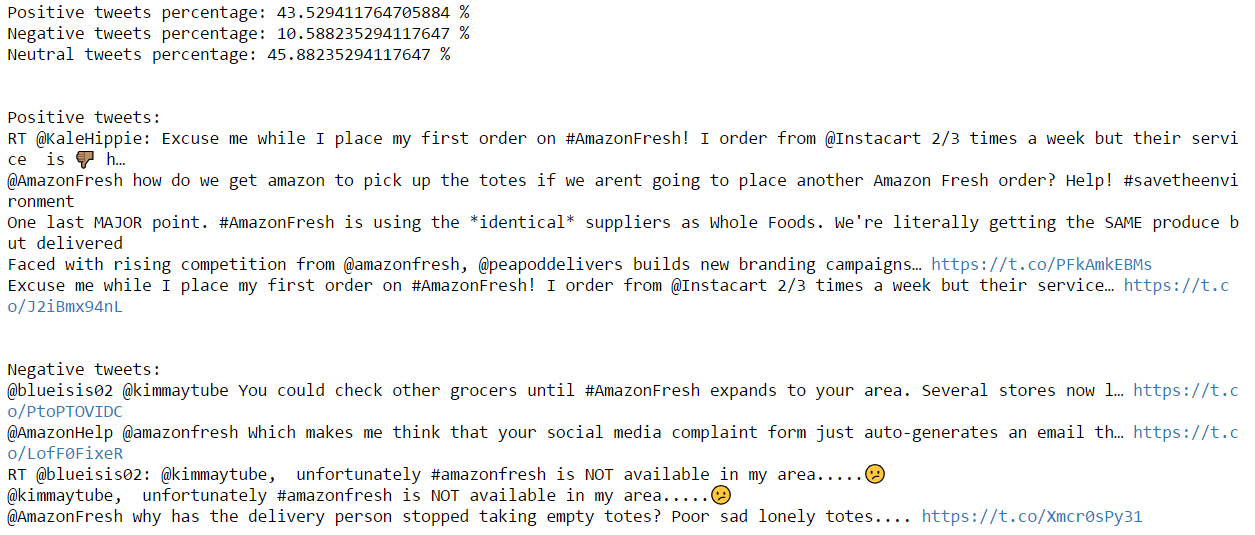
**Amazon Video**



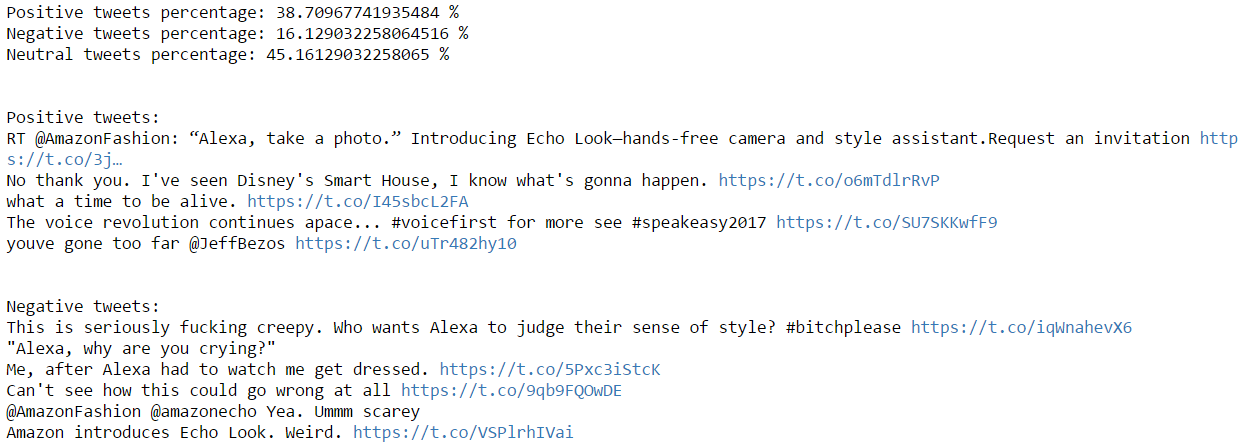
**Amazon Kindle**



**Amazon Fresh**



**Amazon Fashion**



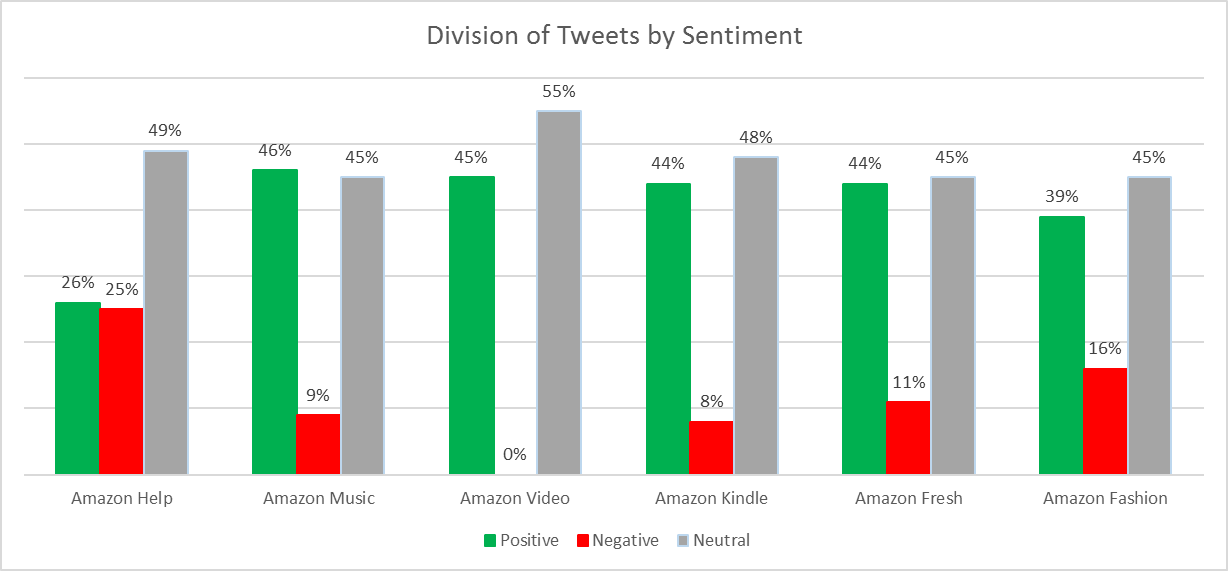
**Summary**

Here is the summary of all the Amazon handles we have analyzed:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Amazon Help | Amazon Music | Amazon Video | Amazon Kindle | Amazon Fresh | Amazon Fashion |
| Positive Tweets | 26% | 46% | 45% | 44% | 44% | 39% |
| Negative Tweets | 25% | 9% | 0% | 8% | 11% | 16% |

We see that Amazon Help has the most percentages of tweets as negative sentiments, since this handle is dedicated to serving customers, enabling a communication channel and catering to customer complaints. But the percentage of negative sentiment tweets are much less when is comes to other handles like Amazon video, music and Kindle.

Amazon Fresh and Fashion also have significantly higher percentages of negative sentiment tweets, given Amazon Fresh is a comparatively new service and does not cater to all the areas, hence lot of users tweets complaining the lack of service in their areas.



1. (Twiiter API Overview - Tweets, n.d.) [↑](#footnote-ref-1)
2. (Twitter API Overview - Users, n.d.) [↑](#footnote-ref-2)
3. (Twitter API Overview - Entities, n.d.) [↑](#footnote-ref-3)
4. (Twitter API Overview - Places, n.d.) [↑](#footnote-ref-4)
5. (Twitter Developer Documentation - REST APIs, n.d.) [↑](#footnote-ref-5)
6. (Twitter Developer Documentation - Streaming APIs, n.d.) [↑](#footnote-ref-6)
7. (Twitter Developer Documentation - Webhook APIs, n.d.) [↑](#footnote-ref-7)
8. (Twitter Developer Documentation - REST APIs, n.d.) [↑](#footnote-ref-8)
9. (Twitter Developer Documentation - Search APIs, n.d.) [↑](#footnote-ref-9)