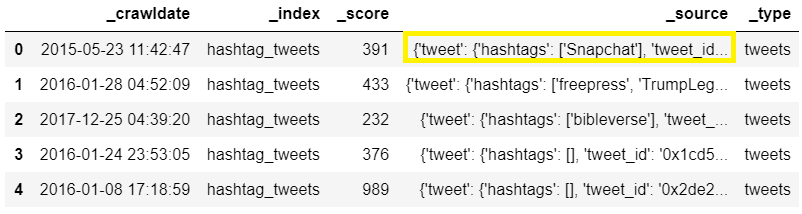
Noa Moshe **Report**

For this assignment we had tweets from tweeter in a json file and we needed to classify them into 7 different emotions.

Data preparation/Preprocessing:

We read the text from the json file and change it to a data frame using pandas. We get the following DF:



In the **\_**source we can see that it is still nested so we un-nest the \_source like so:

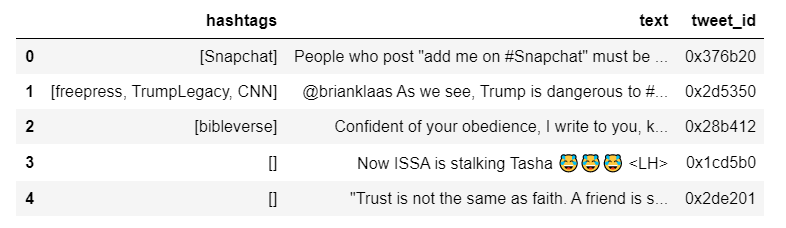
tweets\_df = pd.read\_json('tweets\_DM.json',lines=True)

tweets\_unnested\_df = json\_normalize(tweets\_df.\_source)

tweets\_unnested\_df = tweets\_unnested\_df.rename(index=str, columns={"tweet.hashtags":"hashtags",

"tweet.text":"text",

"tweet.tweet\_id":"tweet\_id"})



We receive the above. The text body and the hashtags are most important features, and, they are usually related. Next, we need split the test and training data by out emotion.csv file which has the result of our training test with the tweet's ID.

After preprocessing, I trained my data using the following 4 methods:

1. Count Vectorizer -NB - 0.44609
2. TF-IDF -NB - 0.37110
3. SLTM - my computer wasn't strong enough for this
4. DNN - 0.41563

The results were surprisingly the best using method number 1. NB does not take into account that Hashtags are dependent on the text, yet it has received the best score. However, getting a better result in Count Vectorizer versus TF-IDF can happen sometimes when there is class imbalance. If you have more instances in one class, the good word features of the frequent class risk having lower IDF, thus their best features will have a lower weight.

In addition, I have also tried using KNN with no success.

Please check out the code for the 4 methods in the file DM19-Lab2-Homework\_Final, under the header "Code for the competition" (this is on the bottom of Lab2, I have left un-working code to show I have at least tried different methods).