

Project 1: Hadoop Ecosystem

Overview:

We used Hadoop 3.3.4 on Colab Environment and applied some methods to extract information from the Reddit Dataset, we first subdivided the data into two parts, User related part, and topics related part. Then on each part we merge the required data, distributing on different map reducers and each return the results.

Steps

Setting up Colab environment

We downloaded hadoop 3.3.4 on colab and changed the java version to be java 8 version. After that we started working on a colab.

Packages Used:

NLTK
Pandas
Matplotlib

Challenges

Topic Extraction

It can be challenging to extract a topic from raw text. We followed a naive, yet easy way of extracting the most frequent, “nouns”, that do not stop words as topic.

There were some challenges though:

- “Unicode characters of different languages. For that we converted the comment’s text to
- “ASCII”, ignoring errors. This way, any unicode character will be removed.
- A lot of meaningless tokens appeared, like “http” or gibberish with no alphanumeric
- characters. We removed them using regex
- Words can appear with different endings. For that we needed the word origin, for which we used Lemmatization

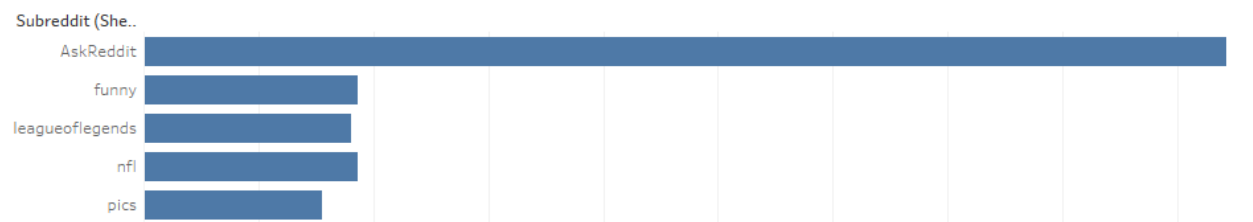
Requirements

Requirement 1:

For this part we started by a mapreduce job to count the frequency of the subreddit. Then with another mapreduce, we have taken the top five subreddit. After that we filter the data and only get the topics related to the top 5 subreddit. Also, we have put the user with them. Finally, we extract the most 5 topics in each top subreddit. For the visualization part. We have used Tableau dashboard.

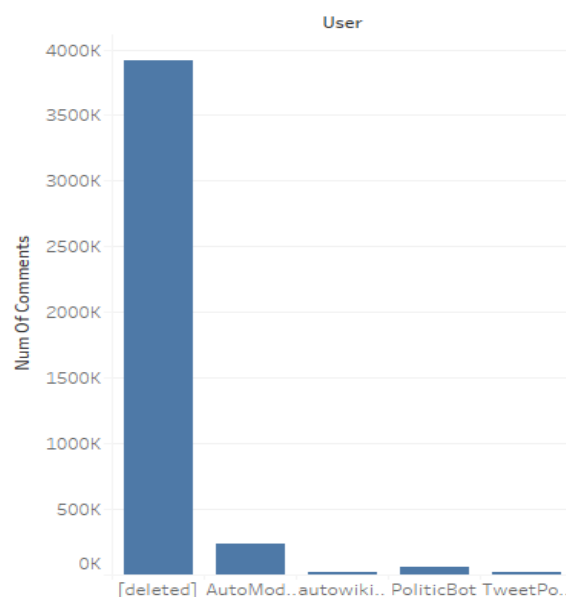
Top Five Subreddit

Top Five Subreddits

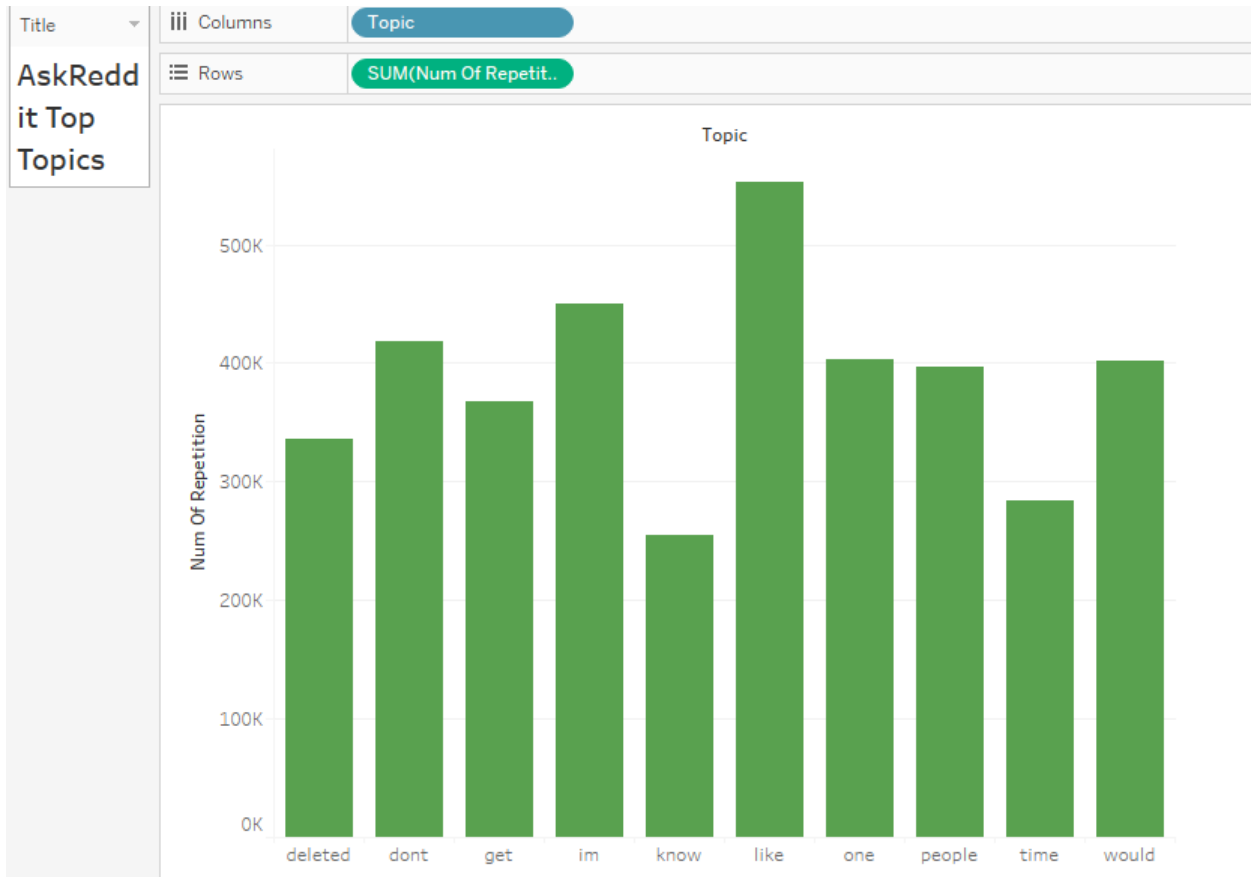


Top five users

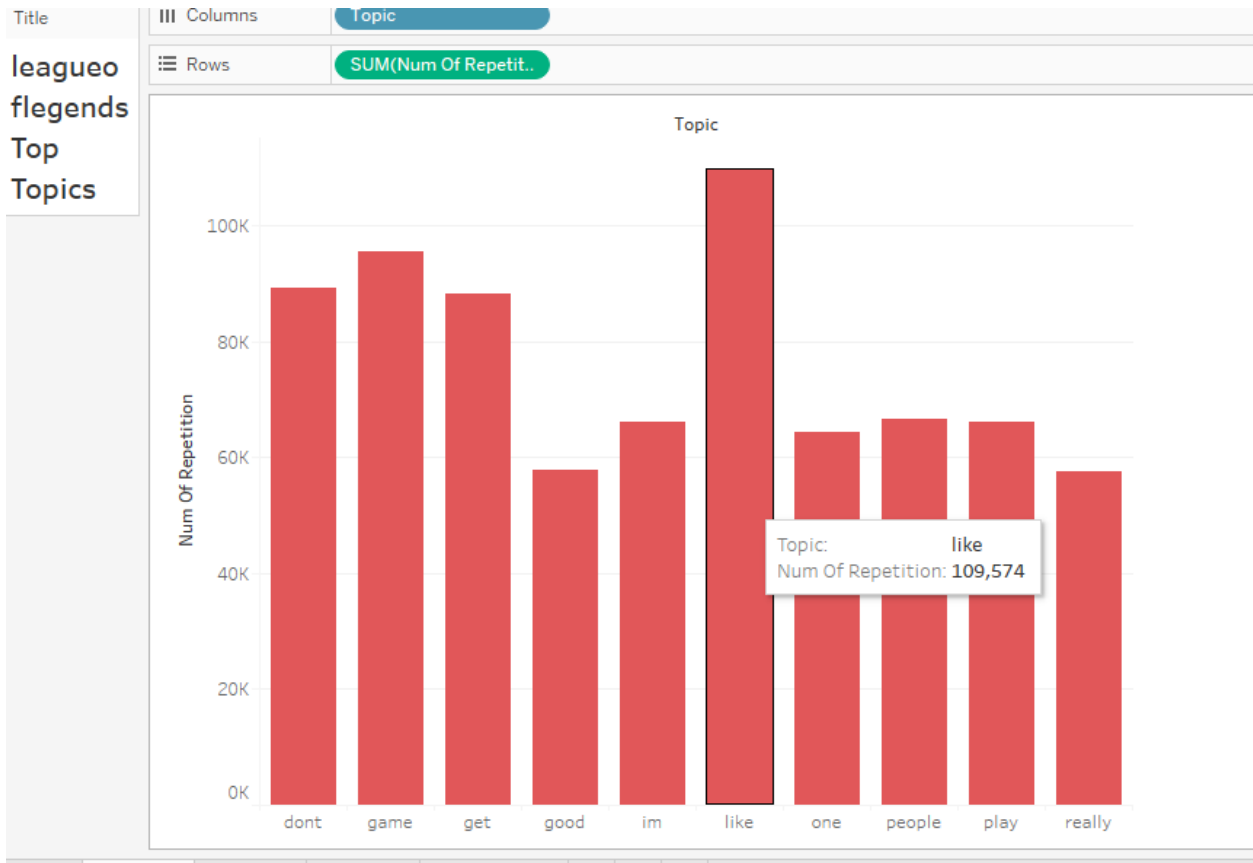
User vs Num Of Comments



Top 5 topics on Askreddit



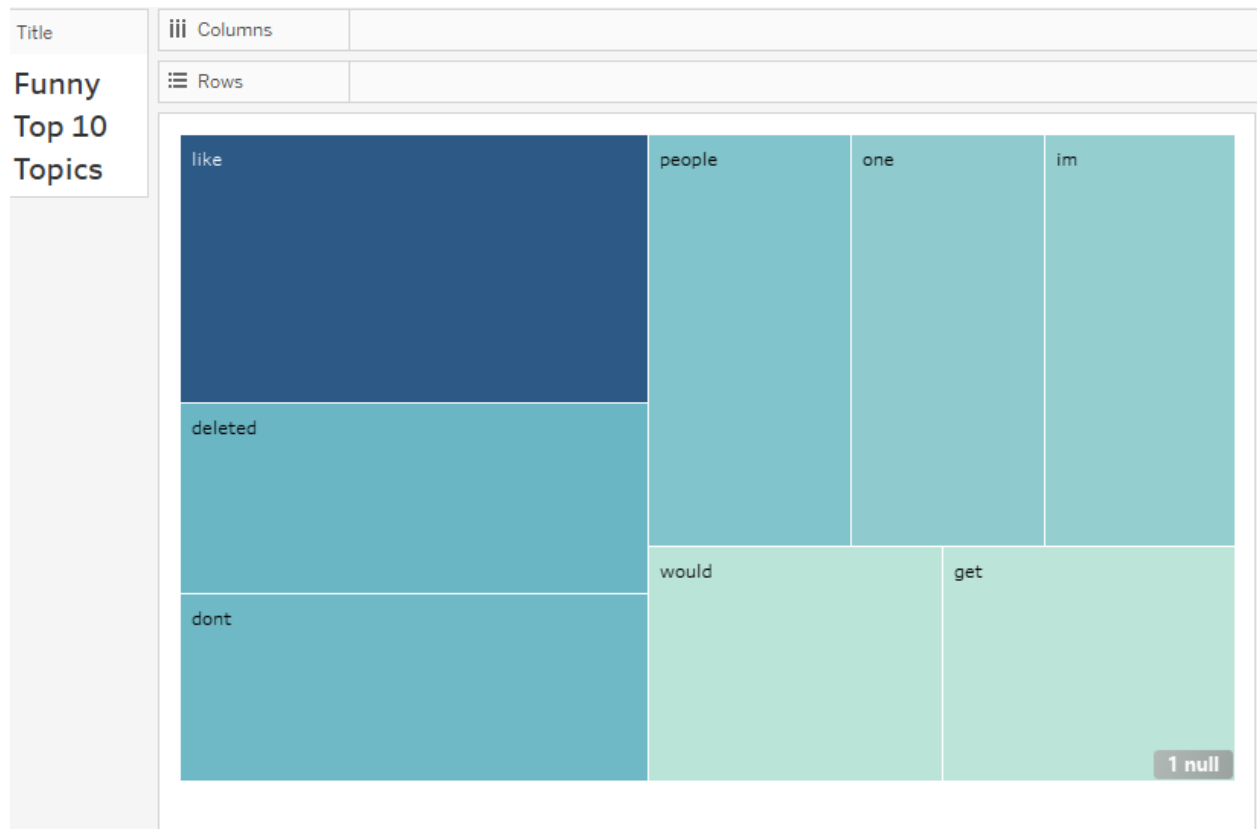
Top 5 topics on LOL



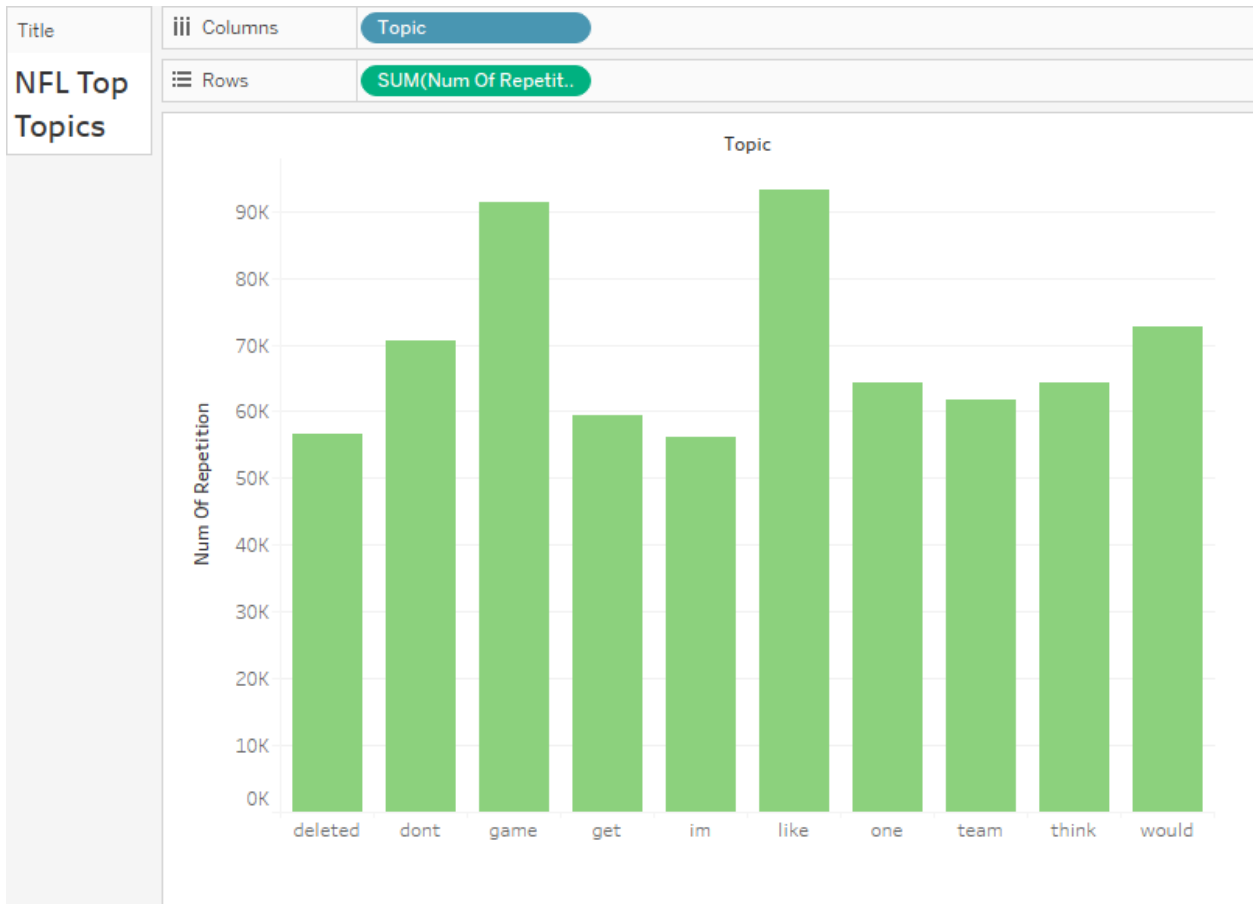
Name: Mazen Ahmed Hassan
Name: Ibrahim Hamada Ibrahim

ID: 201801897
ID: 201800739

Top 5 topics on Funny



Top 5 topics on NFL



Top topics in each subreddit with username.

Topic	User	Subreddit	Freq
people	deleted	funny	1108
thats	deleted	funny	601
get	deleted	funny	718
like	deleted	funny	1322
deleted	deleted	funny	66223

Topic	User	Subreddit	Freq
post	AutoModerator	AskReddit	73488
contact	AutoModerator	AskReddit	48048
question	AutoModerator	AskReddit	105720
please	AutoModerator	AskReddit	109745
deleted	deleted	AskReddit	333841

Topic	User	Subreddit	Freq
AutoModerator	post	pics	1828
deleted	deleted	pics	57007
AutoModerator	shorter	pics	914
people	deleted	pics	1045
dont	deleted	pics	836

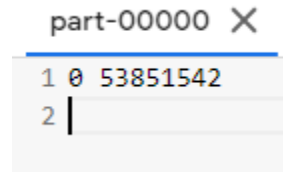
We can clearly see that there are relevant topics in the subreddits, when topics are extracted per subreddit. This is contrary to the case in Requirement 3, when we try to get the top scoring topics overall, they aren't meaningful. So grouping them in subreddits makes it more evident and clear. For example, in the League Of Legends subreddit, a common topic is about game characters, like "Jax", "Zyra", or even the topic "game" itself. In NFL, hockey, and NBA, we can see the topic "game", "penalty", "time", which are all relevant to the subreddits they belong to.

Requirement 2:

Rate of replies compared to controversiality of comment/post

We believe the second requirement is infeasible.

After running a mapreduce job to see the count of the values that the controversiality takes. We found that it is always 0. Therefore, we can not compare to something that is always 0.



part-00000 X		
1	0	53851542
2		

Requirement 3:

Topics that yield the highest number of upvotes and/or lowest of downvotes

Same thing as in controversiality here, Down always takes 0. The output of the mapreduce was exactly as shown in the controversiality. However, the ups and the scores, which always have the same value, have negative values. Therefore after thinking about that, we take the max positive score as the topic that yields the highest number of upvotes and the negative one as the lowest number of downvotes. After running the map the most upvoted 5 topics were as shown below.

Topic	Sum of upvotes
people	513297
dont	520448
im	540325
one	581205
like	779594

For the downvotes it is shown in the below table:

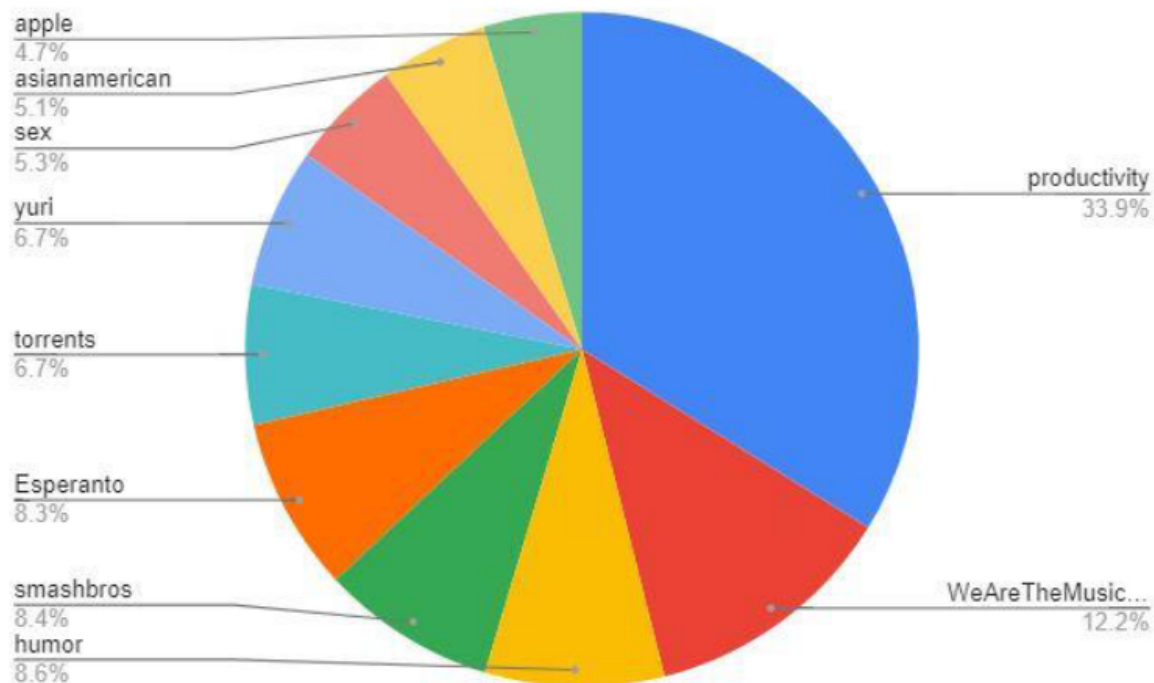
Topic	Sum of upvotes
lols	-334
resolute	-320
mattingly	-319
laurel	-297
whaaaaaa	-288

Creative Requirement:

The usage of curse per subreddit:

we moved to investigate something else, but similar: the dirtiest subreddits and the cleanest subreddits.

The list of curse words we counted upon are 4 words.



Interestingly, people who are talking about being productive use curse words the most! While the cleanest subreddits were a lot containing zero curse words.

A suggested method for better topic extraction

Topic extraction via word frequency is inherently a naive approach, for several reasons:

- Word count doesn't necessarily reflect the topic at hand.
- It doesn't capture the context, or relate the words to each other
- It is done per comment, and not on the whole dataset level

A suggested method to solve these points is using Clustering of word embedding. Clustering can be visualized in the figure below

The suggested sequence is as follows

- If we cluster together the word vectors, using readymade word embeddings (like word2vec). A benefit of word embedding is that the euclidean distance between word vectors is a good measure of word similarity
- Then, since subreddits themselves represent an abstract layer over topics, say a "meta-topic", we can set the cluster count to a multiple of the number of subreddits.
- As a result, now the topics are represented by "cluster centroids" of word vectors.
- A comment's topic is decided by which cluster centroid it's nearest to
- However, this method will require significantly more computations
- Topic extraction of a comment will have to compare the distance of each word to all topic centroids, which can be the order of 100k distance computations per word per comment!
- It will require k-means clustering on hadoop. It's doable by making each iteration as a map-reduce job, and saving the result in a temporary file