Zewail city of science and technology
Big Data Analytics CIE 427

Student Name: Ahmed Eid Abdullah

Student Name: Ahmed Kamal

Student ID: 201701264 - 201700295

Doctor Name: ElSayed Hemayed

Teaching assistant Names: Ahmed Sameh – Mohamed ElAref

Course (Code) Name: Big Data Analytics CIE 427

Mini Project 1

Analysis of Reddit website

Zewail city of science and technology
Big Data Analytics CIE 427

Table of content :-

Zewail city of science and technology
Big Data Analytics CIE 427

Our Approach and how we do preprocess on the data

We downloaded the Hadoop on google colab,

1- Firstly, we read the input json file and convert it into to python dictionary {}to access it easily and we do this job by built in library.

2- Then we accessed only the subreddit and body keys and values.

3- In Mapper.py , we did our preprocessing as the following:

- Lower each body
- Removing any non-alphanumeric words from each body, by using simple regex.
- Removing any links such as https://youtube.com , or https://anything.anytld (We did this step , as we noticed that the body contains some links)
- Tokenizing each body (to get its topics)
- Stemming each tokenized word , then we consider each word as topic, we checked if this word is not stop words or not , not numeric , and we put a condition on the word's length (as we noticed some tokenized word may be very long, so we checked if the length is less than 15, words of length less than 15 usually be understandable) , and we take only the noun words (because the topics usually be nouns not verbs for example)

```
for line in sys.stdin:
    #read as dict
    json_obj = json.loads(line)
    # take only subreddit
    subreddit_feature = json_obj["subreddit"]
    # take only body
    body_feature = json_obj["body"]
    #preprocessing starts
    # lower the body
    lowered_body = body_feature.lower()
    non_alphanumeric_body = re.sub(r'[^\w ]+', '', lowered_body) # substitte in thing that is n
    no_links_body = re.sub(r'https?:\/\/.*[\r\n]*', '', non_alphanumeric_body) # i noticed that
    # we will tokenize body to extract topics
    tokenization_of_body = nltk.tokenize.word_tokenize(no_links_body)
```

- Print each subreddit and its (stemmed_word which the topic , 1) separated by tab , so we count in the reducer.

- Print each stemmed_word and its upvotes value (from the json ups key)

```
    for word in tokenization_of_body:
        ##check if the word is not stop word or numberic
        if word not in English_stopwords and len(word) <15 and not word.isnumeric() and word in nouns: # probably not a natural word
            stemmed_word = stemmer.stem(word)
            ## print the subreddit and its topics
            print(f"{subreddit_feature}:topics", (stemmed_word, 1), sep='\t')
            ## print the topics and its upvotes
            print(f"{stemmed_word}:upvotes", (json_obj["ups"],), sep='\t')
```

4- In reducer, we get the most common 10 topics related with each subreddit and count the upvotes of each topic. We take only the upvotes that more than 500 , because we found that many topics count 70 , 20 , 120 , and many more in this range, so to avoid this rare counted topics, we put condition on upvotes more than 500.

```python
#!/usr/bin/python3

from collections import Counter
import sys
topic = None
word_upvotes = None
# here count will be initilzed to zeros we will use Counter object
count = Counter()
##initalize the counter to count the upvotes
count_upvotes = 0
word_counting = 0
for line in sys.stdin:
    key, value = line.strip().split("\t")
    # because strip function will convert it to string
    value = eval(value)
    *key, category = key.split(":") # as the dict contains key and value (category)
    # first question , topics of every subreddit
    if category  =="topics":
      #check first if the topic is none
        if topic is None:
            topic=key
        # if not , then count the most common discussed topics
        if topic != key:
            print(f"{topic}:{category}", count.most_common(10), sep='\t') # top three discussed topics
            topic = key
            count = Counter()
        count[value[0]] += value[1]
    # third question , highest upvotes
    if category  =="upvotes":
      if word_upvotes is None:
        word_upvotes = key
      if word_upvotes != key:
        if word_counting > 500: ## we take 120 , as an average number , because we notice that many numbers and we want the h
          print(f"{word_upvotes}:{category}", count_upvotes, sep='\t')
          word_upvotes = key
          count_upvotes = 0
          word_counting = 0
      count_upvotes += value[0]
      word_counting += 1
# print the top three discussed topics
if category == "topics" and topic:
    print(f"{topic}:{category}", count.most_common(10), sep='\t')
# print upvotes of the topics
if category == "upvotes" and word_upvotes:
  if word_counting > 500:
    print(f"{word_upvotes}:{category}", count_upvotes, sep='\t')


```

5- We open outputs files and convert them into CSV file, to be visualized later.

```python
f = open("part-00000", "r")
Lines = f.readlines()
```

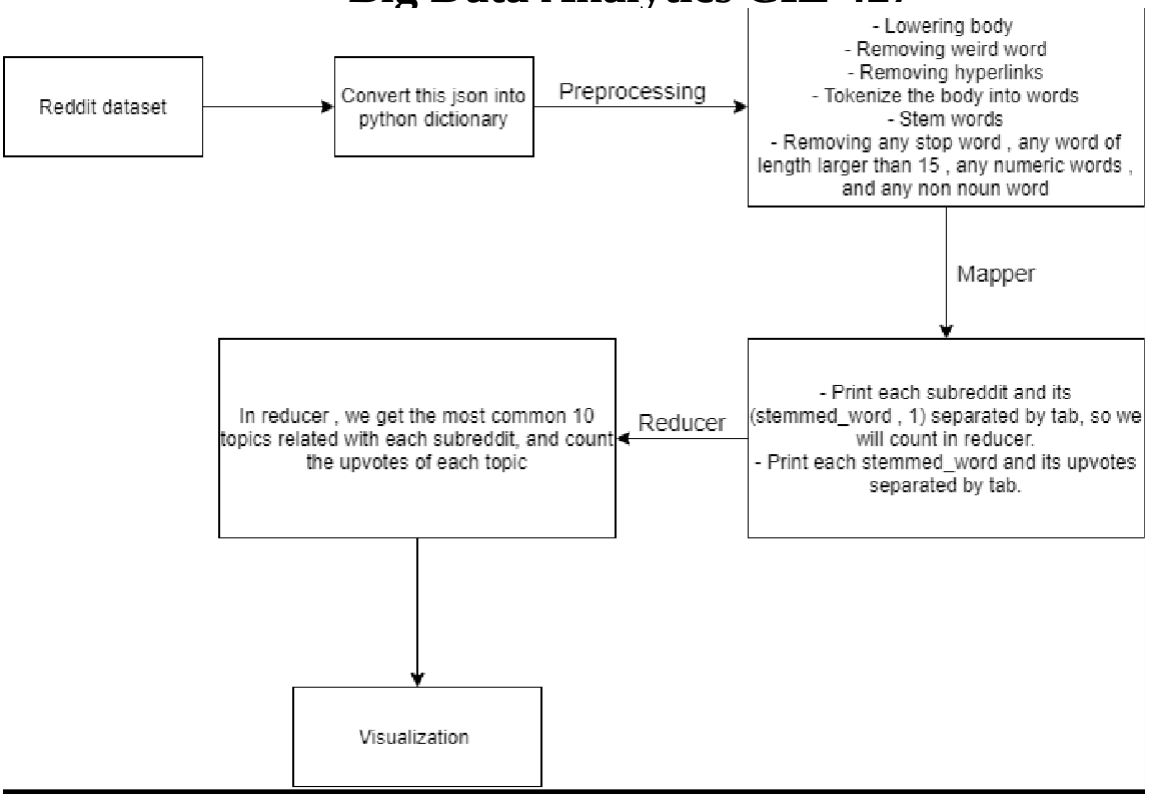
```python
topics = []
upvotes=[]


for line in Lines:
    key, value = line.strip().split("\t")
    value = eval(value)

    *key, case = key.split(":")
    key = ":".join(key) # handle the case where key has a :
    if case == "upvotes":
        if value > 500:
            upvotes.append((key, value))
    elif case == "topics":
        topics.append((key, value))
```

```
[ ] import csv
    header = ['SUBREDDIT NAMES', 'TOP THREE DISCUSSED TOPICS']

    with open('TOP THREE DISCUSSED TOPICS FOR EVERY SUBREDDIT.csv', 'w', encoding='UTF8', newline='') as f:
        writer = csv.writer(f, delimiter=',')
        writer.writerow(header)
        # write multiple rows
        writer.writerows(topics)
```

```
header = ['SUBREDDIT NAMES', 'TOP THREE DISCUSSED TOPICS']

with open('TOP UPVOTES.csv', 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f, delimiter=',')
    writer.writerow(header)
    # write multiple rows
    writer.writerows(upvotes)
```

6- We used two websites to make most of charts which are https://app.datawrapper.de/ **and** https://online.visual-paradigm.com/

Pipeline diagram4



Code Environment

- Programming language: Python
- Where we run the code: Google Colab https://colab.research.google.com/
- Submitted files:

1.(mapper.py)

2.(reducer.py)

3.(Hadoop_Notebook_MapReduce.ipynb)

4.(Convert to csv notebook.ipynb)

- How can you run the code?

1. Open “Hadoop_Notebook_MapReduce” on google colab anf follow instructions in it to download and unzip both hadoop and the dataset

2. Upload mapper and reducer file to google colab then run the codes sequencelly

   To get the output (part-00000) file

3. Open “Convert to csv notebook” on google colab and upload the output file (part-00000) this notebook is created to convert the output to csv file to give options of Excel function that can be applied on csv file and also tons of online websites that enabled us to make elegant charts and graphs .

Challenges we faced and how we solve them?

We faced the following challenges :

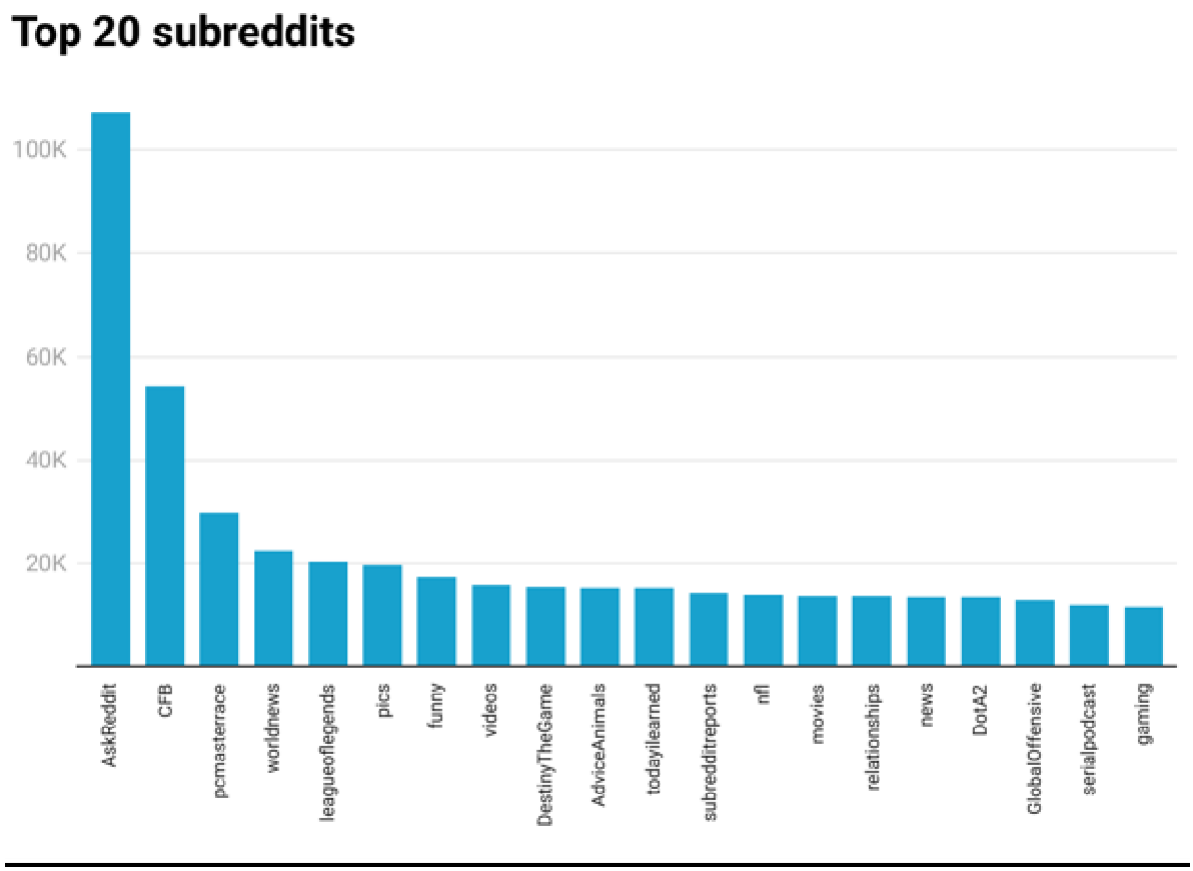
1. We cannot unzip the dataset correctly as it has special format called bz2 , none of normal compressing software such as winRare was capable of unzipping the dataset , so we figured out a way by using a library in python called bz2 is specialized to unzip this kind of compression , but first we mount colab on our drive to be able to read the dataset directly without uploading it

2. We first run the dataset locally on our machine but that was very very slow

to processing on the whole dataset , but Aiad Assad pointed that we can use hadoop on Google colab and he tested it and it was like super power compared to our testing time , then we shifted to google colab to use hadoop online and all credit for this point goes to Aiad , this point enabled us to run on the whole dataset with suitable amount of time ,that make our life easier and was superior guide from Aiad

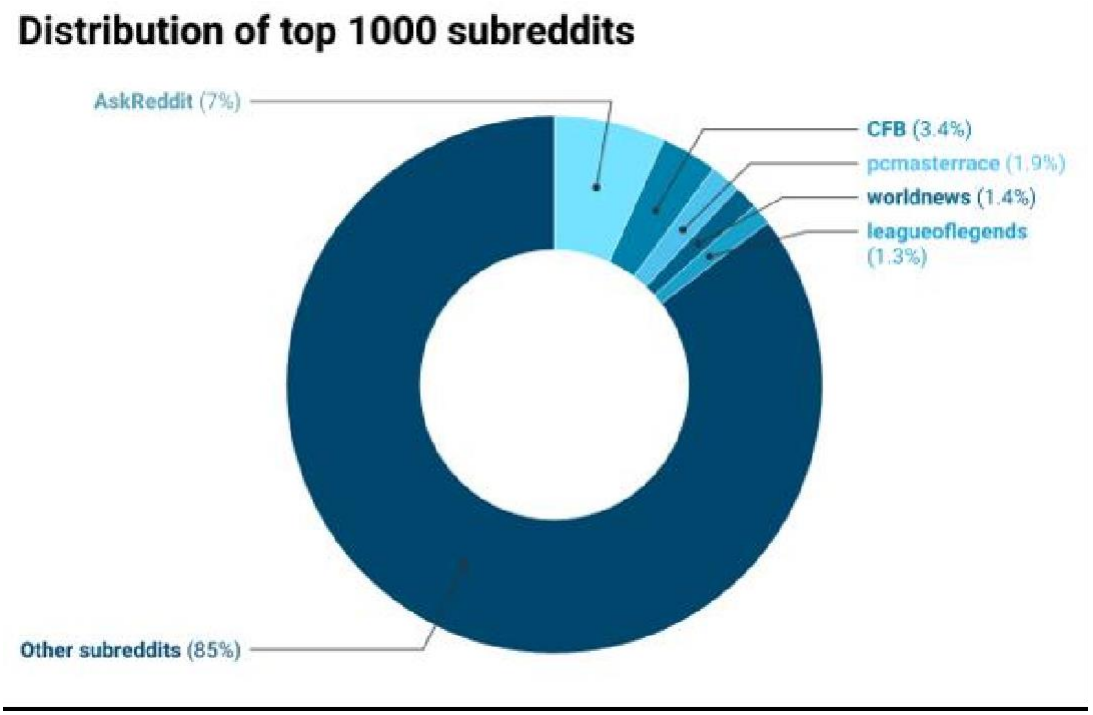
Results and graphs

We found that the most common subreddit is AskReddit. For simplicity we visualize the top 20 subreddits only to be well visualized.



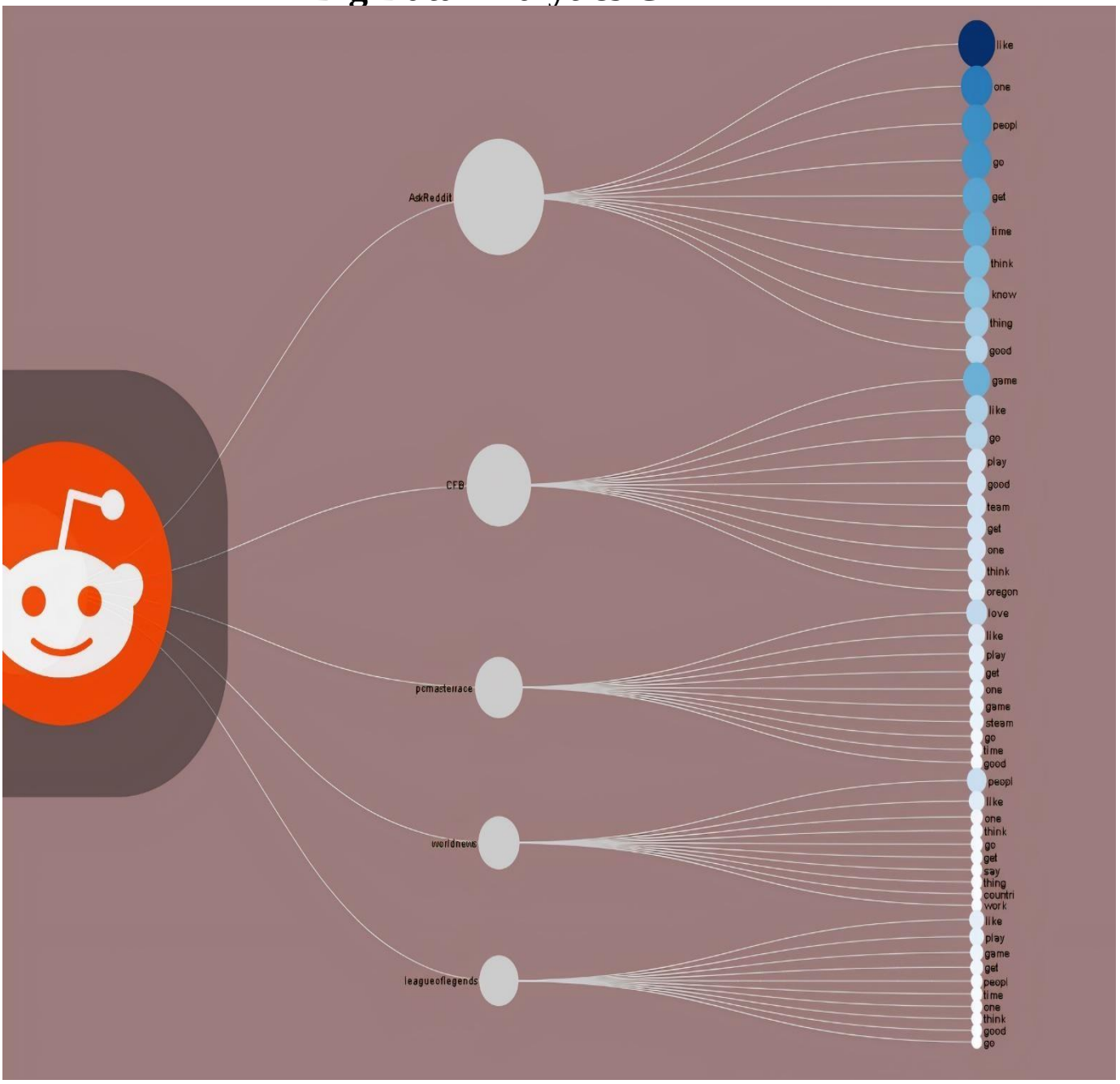
The distribution of top 1000 subreddits

The percentage of AskReddit subreddit is 7% (very high percentage), CFB is 3.4%, pcmasterrace is 1.9% , worldnews is 1.4% , leageoflegends is 1.3% , while the other 995 topics is 85%



Visualization of the Most 10 discussed topics associated with the top 5 subreddit in Reddit

linear Dendrogram Graph



After deletion of commas and squre brackets , we use Excel functions to sort subreddits

Based on their summation on top ten topics counter , then some preprocessing of the data

To be suiyable to be inserted into this site which is can create linear dendogram with our data.

BIG NOTE

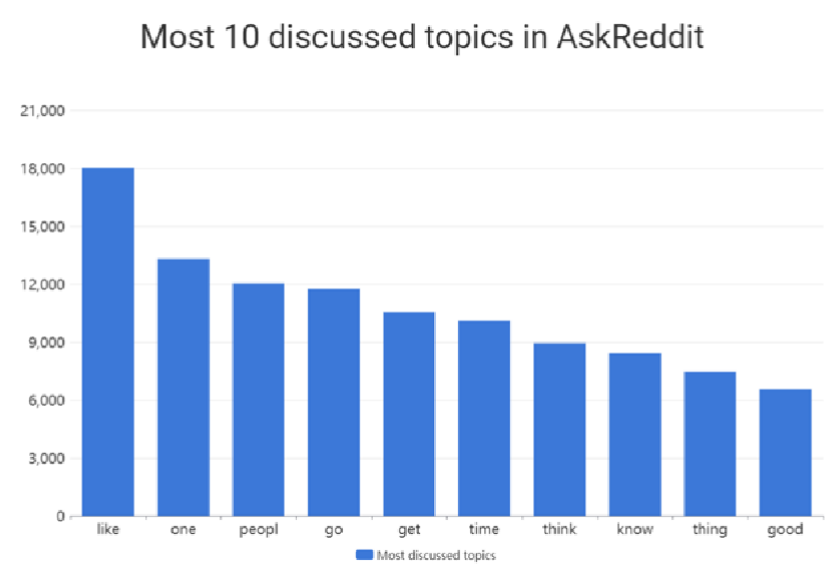
1. Drak blue means highest frequency topic
2. White means lowest frequency topic
3. Size of the word circle means (bigger circle size higher frequency topic)
4. All topics for every subreddit is sorted descending based on their frequency in the subreddit

✓ ***BIG BIG NOTE : Creative Tip***

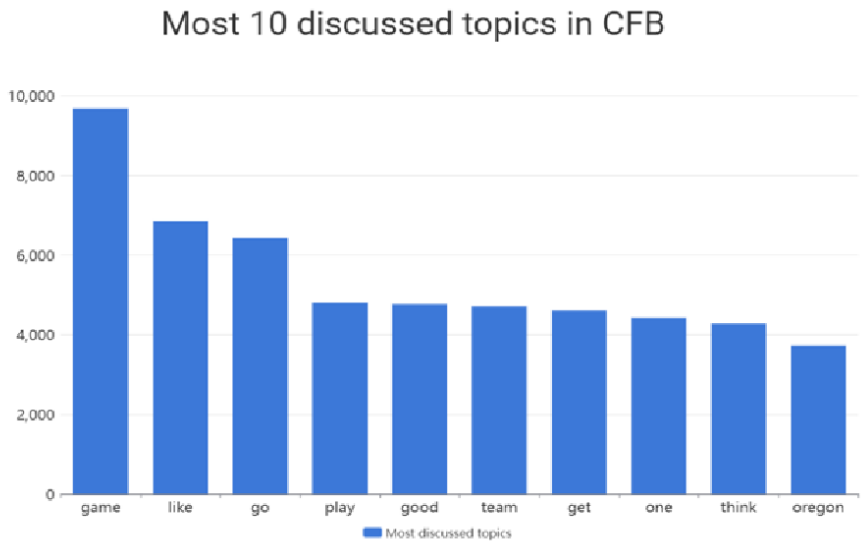
This graph is expandable which means we can this fancy graph for all subreddit in reddit website and for each subreddit we demonstrate every topic in that subredddit with size of circle indicating frequency of that topic (this can be a very powerful and efficient tool to know the trending topic from where it created and when and how big this topic compared to other topics) this is very similar to HASHTAG that can tackle the number of people talk about trending topic and who created this HASHTAG.

- We choose to make linear dendrogram graph on only top five subreddit with their associated top ten topic due to processing power .

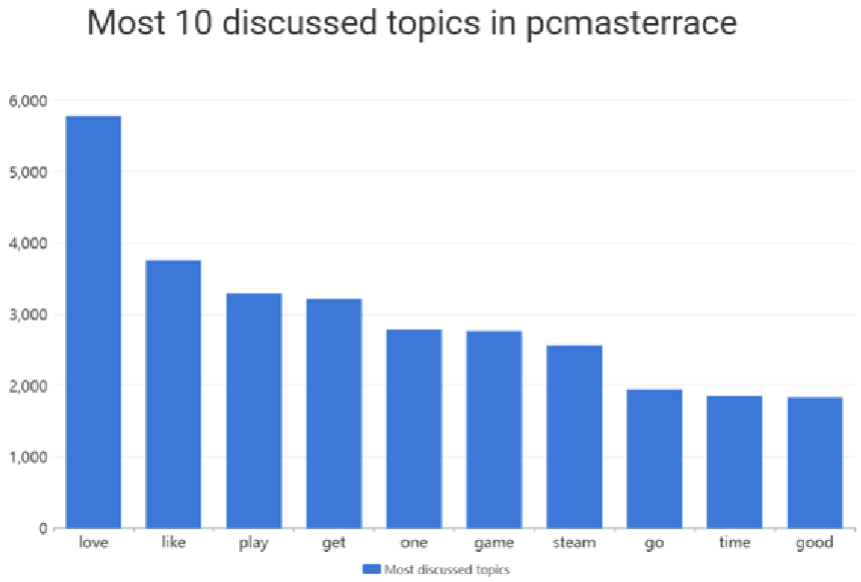
Top 10 most discussed topics in top 5 subreddits



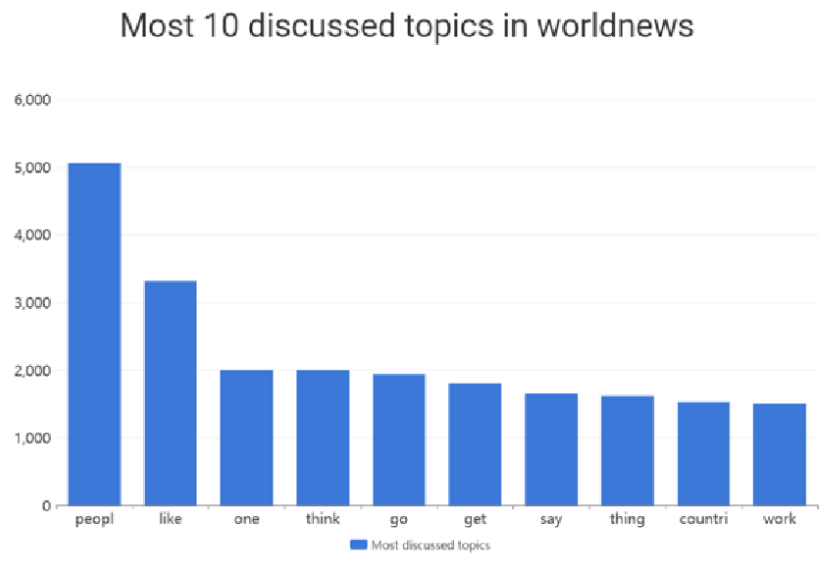
Top 10 most discussed topics in CFB subreddit



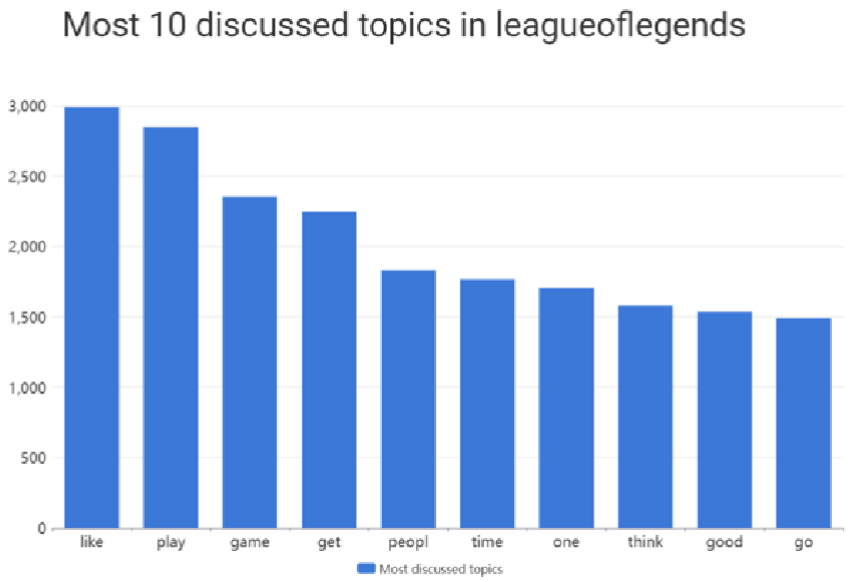
Top 10 most discussed topics in pcmasterrace subreddit



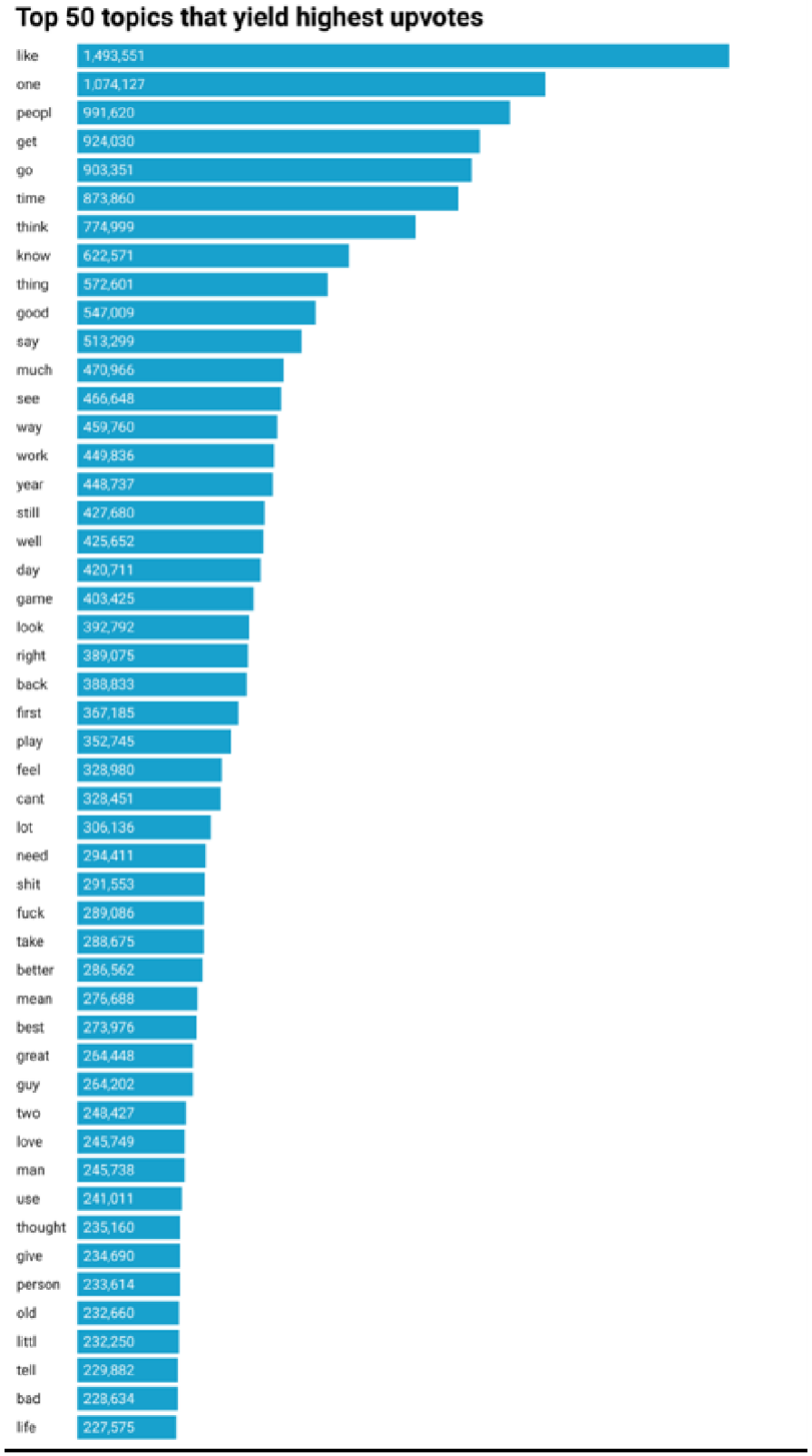
Top 10 most discussed topics in worldnews subreddit



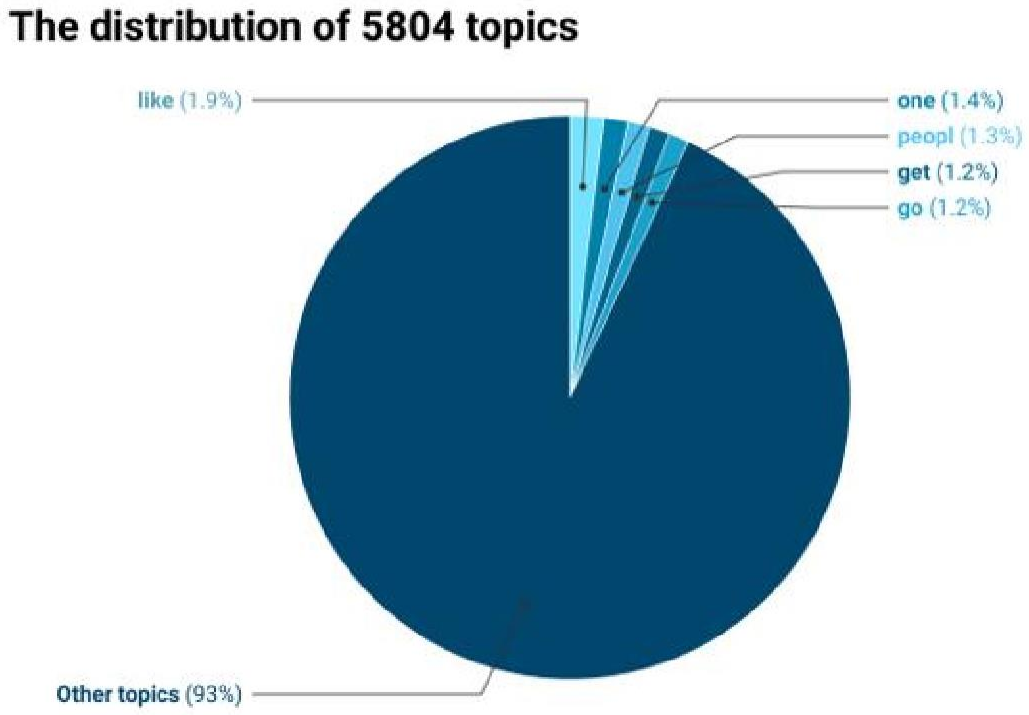
Top 10 most discussed topics in leagueoflegends subreddit



Top 50 topics that yield highest upvotes


Top 50 topics that yield highest upvotes
like 1,493,551
one 1,074,127
peopl 991,620
get 924,030
go 903,351
time 873,860
think 774,999
know 622,571
thing 572,601
good 547,009
say 513,299
much 470,966
see 466,648
way 459,760
work 449,836
year 448,737
still 427,680
well 425,652
day 420,711
game 403,425
look 392,792
right 389,075
back 388,833
first 367,185
play 352,745
feel 328,980
cant 328,451
lot 306,136
need 294,411
shit 291,553
fuck 289,086
take 288,675
better 286,562
mean 276,688
best 273,976
great 264,448
guy 264,202
two 248,427
love 245,749
man 245,738
use 241,011
thought 235,160
give 234,690
person 233,614
old 232,660
littl 232,250
tell 229,882
bad 228,634
life 227,575

Distribution of top 5804 topics in all subreddits



Why our product is very efficient?

1. Our product can run on any dataset of any size, because we use Hadoop system (that deals with any big data)
2. Produce good and detailed visualizations: It can generate multi charts for further analysis of Reddit (in our case). For simplicity, we visualize only the top 20 subreddits and how are they distributed, most 10 topics of the top 5 subreddits, the top 50 topics that yield highest number of upvotes (people like it), and how the top 5804 topics are distributed.

3. Easy to run: We used google colab to run our code, which is very easy to run as all you need is to run each cell consecutively.

4. We used good data preprocessing by using different NLP libraries to remove stop words, remove not understandable characters, remove words that more than 15 characters, remove hyperlinks, we work only on nouns (as topics cannot be verbs for example), we exclude the numeric words, and we did many more preprocessing.

5. We convert all outputs into CSV files and then convert them into charts to be readable and understandable.

Creative/Innovative

- Take only Nouns from body (we download word net) to check in word is considered as noun or not wordNet use synset (presented in NLTK here) and word embedding to know that.
- Recommended creative tip that we thought of but we cannot applied it -> we observe that wordNet is consider some verb as topic also for example like wordNet consider it as noun because it can means admiration, so we think of 2 or 3 N-gram to consider the context of the sentence to be 100 % accurate in determination if this word is noun verb based on the context , this approach will

capture only topic that 100% people talk about such as HASHTAG technology , but note this process is need powerful processing and even more with the increasing of N-gram.

- Put a threshold on up votes (>500) to be included , this threshold we see it suitable to consider topic is highly up voted or not , because if we make it small random number , the algorithm will consider any topic is highly up voted (consider case if topic is get 100 up vote but from friends of user and not relevant to the decision maker if this topic is trending or not) that is why we have high up vote threshold to capture trending topic case or even policy violation detection (higher down votes is will also be considered from the decision maker or from the data analyst)
- Before calculating Controversiality we notice that in first 10 sample its value is zero , then we checked it for the whole dataset we found it always zero , so it impossible to calculate rate of replies because Controversiality is always zero
- Linear dendrogram is very powerful tool that we created to visualize everything for each subreddit

  Dendogram can describe as it is tree has all ancestors (subreddit) with its children (topics) with their frequency (circle size) this can easily expanded to include all reddit website information if processing power is available

Conclusion

We did analysis of Reddit website, we worked on Hadoop, as our dataset is very large 30GB, we convert all outputs in CSV and then visualize all outputs. In the future we can implement GUI and connect this code with it to be a real product.