Assignment 2:

Repo address: <https://github.com/krystenng/SMU-IS459-assignment-.git>

How to navigate:

Python code for questions 1 and 2:

assignment\_2\_q1q2.py

Python code for question 3:

assignment\_2\_q3.py

Data scraped for question3:

Folder -- assignment2\

result.json

scrapy.cfg

assignment2.parquet

mongodb\_to\_parquet.py

assignment2\

spiders\

\_\_init\_\_.py

spider.py

items.py

middlewares.py

pipelines.py

settings.py

\_\_init\_\_.py

This document consists of:

* Answers for questions 1, 2 and 3 for Assignment 2
* Explanation for questions 1, 2 and 3 for Assignment 2
* Codes that are not found in the repo but used to extract data as evidence
* Screenshots of the data frame extracted
* Screenshots of results from the codes

1. **How large are the communities (connected components)? What are they key words of the community (frequent words)?**

Out of the 4659 authors, 4534 authors are connected as their component ids are the same, 0. Those that have component\_id == author\_id, are only connected to themselves. Sorting by descending order can allow me to view the unconnected components.

A picture containing text

Description automatically generated A picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidence

1. **What are they key words of the community (frequent words)?**

Another one I have tested out is the most used words found in the **contents** and the results I have gotten after executing my code shows that:

Graphical user interface, text

Description automatically generated

I have extracted out the top 200 words used to obtain a more accurate result.

The top 200 words are as follows:

--------------------------------------------------------------------------------------------------------------------------------------

the, to, i, and, is, a, for, of, in, you, can, it, on, my, with, that, but, not, will, if, u, so, play, be, this, have, game, or, just, like, are, got, all, at, me, no, from, get, as, one, now, also, your, more, then, 1, still, only, they, 2, time, its, need, do, go, up, when, dont, think, see, out, what, he, how, was, playing, liao, good, we, there, too, new, some, use, 3, really, lol, even, very, want, by, any, know, after, other, buy, last, server, players, im, which, first, back, than, much, about, them, try, those, team, should, his, using, most, has, who, quite, their, before, many, already, cant, better, an, here, level, why, guild, people, am, games, same, him, 4, make, since, into, build, keep, well, going, win, 5, start, because, cannot, few, leh, sure, long, jin, always, right, maybe, ur, anyone, where, come, say, us, add, find, take, join, next, never, ah, way, free, end, again, guys, season, damage, drop, skill, played, player, run, around, best, our, 10, moi, would, fun, damn, down, been, ppl, hard, another, give, la, deck, wait, feel, turn, actually, dun, set, must, hit, league, over, bad, until, lvl, getting, wanna

--------------------------------------------------------------------------------------------------------------------------------------

From these key words, people are talking about the games and their different seasons. They also discuss about the damage, skill, and things in the game. League should be a quite popular game to talk about since it is being extracted from the top 200 words. I have highlighted what are think that it will be significant to find out what people are talking about in the forum.

I have also tested out for the key words found in **topics** and results I have gotten after executing my code shows that:

A picture containing text

Description automatically generated

The top 20 words are:

thread, part, of, no, official, or, items, 2, sales, fantasy, the, exile, selling, ingame, currency, edmw, dota, premier, league, 2hearthstone.

--------------------------------------------------------------------------------------------------------------------------------------

Out of these words, most people are talking about games like edmw, dota, league, 2hearthstone

Code to extract top 20 words are:

import pyspark

from pyspark.sql import SparkSession

from pyspark.sql.functions import monotonically\_increasing\_id

from graphframes import \*

from nltk import tokenize

from operator import itemgetter

import math

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import string

from nltk.stem import WordNetLemmatizer

spark = SparkSession.builder.appName('sg.edu.smu.is459.assignment2').getOrCreate()

# Load data

posts\_df = spark.read.load('/user/krystenng/parquet-input/hardwarezone.parquet')

# Clean the dataframe by removing rows with any null value

posts\_df = posts\_df.na.drop()

postCollect = posts\_df.collect()

topic\_str = ''

# Remove stop words from the topics and put them into a string

stop\_words = set(stopwords.words('english'))

for post in postCollect:

    if post['topic'] not in stop\_words:

        topic\_str += post['topic'].lower() + ''

# Remove unnecessary punctuations from the string

data = str.maketrans(dict.fromkeys(string.punctuation))

topics\_without\_punctuation = topic\_str.translate(data)

# Lemmatize the words so that the words will be in the base form and it will be easier to identify key words

lemmatizer = WordNetLemmatizer()

topic\_clean = ''

for w in topics\_without\_punctuation:

    topic\_clean += lemmatizer.lemmatize(w)

# Split the words into array

words\_array = topic\_clean.split()

# To put the words and their count into a dictionary: key is the word and the value is the number of times the word appear

df = {}

for word in words\_array:

    if word not in df:

        df[word] = 1

    else:

        df[word] += 1

tf\_score = df

# Calculate the score of the words by using the df\_idf method, as there are no more than one file, only df score is calculated

total\_word\_length = len(words\_array)

tf\_score.update((x, y/int(total\_word\_length)) for x, y in tf\_score.items())

# Function to get the top n words from the dict

def get\_top\_n(dict\_elem, n):

    result = dict(sorted(dict\_elem.items(), key = itemgetter(1), reverse = True)[:n])

    return result

#Get the top 20 words

top20\_words = get\_top\_n(tf\_score, 20)

top20 = ''

for word in top20\_words:

    top20 += word + ', '

top20 = top20[:-2]

print("The most frequent words are the following :" + '\n' + top20)

1. **How cohesive are the communities (Average # of triangles over every user in a community)?**

The average number of triangles are approximately 56663. I took the total number of triangles for all the authors divide by the total number of authors.

Code to do:

Text

Description automatically generated

To get the total number of triangles:

num\_of\_triangles = results.agg({'num\_of\_triangles':'sum'}

A picture containing graphical user interface

Description automatically generated

total\_num\_triangles = num\_of\_triangles.collect()[0][0]

count = results.count()



1. **Is there any strange community?**

Crawl the joined date of all the authors in the hardwarezone forums.

Note: Code of crawling data can be found in “assignment2” folder in the github repo.

After that, I changed the result.json that I have gotten to a parquet file and load the data as shown:

users\_df= spark.read.load('/user/krystenng/parquet-input/assignment2.parquet')

users\_df.show()

A picture containing text, plaque, scoreboard

Description automatically generated

I have tried to check the number of authors who joined in different year since the administrator started the hardwarezone forum in year 2000.

Year 2000: 185, Year 2001: 144, Year 2002: 111, Year 2003: 121, Year 2004: 236, Year 2005: 242,

Year 2006: 233, Year 2007: 356, Year 2008: 380, Year 2009: 335, Year 2010: 371, Year 2011: 330,

Year 2012: 364, Year 2013: 235, Year 2014: 196, Year 2015: 249, Year 2016: 180, Year 2017: 140,

Year 2018: 119, Year 2019: 94, Year 2020: 58, Year 2021: 22

Example of code:

Text

Description automatically generated

The number of authors that joined the forum decreased over the years. However, can see there is a sharp rise in the number of authors who joined the forum in 2007 and 2008.

I have also tried to find out the number of posts each author post on the forum, which is shown as follows:

author\_rdd = users\_df.rdd

author\_rdd.first()

author\_date\_rdd =author\_rdd.map(lambda x: (x[1],(x[2], 1)))

count\_rdd = author\_date\_rdd.reduceByKey(lambda x, y: (x[0], x[1] + y[1]))

test\_rdd = count\_rdd.map(lambda x: (x[0],x[1][0],x[1][1]))

df = test\_rdd.toDF()

df1 = df.withColumnRenamed('\_1', 'author') \

        .withColumnRenamed('\_2', 'joinedDate') \

        .withColumnRenamed('\_3','num\_of\_posts')

A picture containing text

Description automatically generated

I sorted them by descending order to find out whether those who joined the hardwarezone forum in the early years will post more than those who joined in the later date which is shown:

from pyspark.sql import functions as F

A picture containing text

Description automatically generated

It can be seen that the author, Vulpix who joined the hardwarezone forum when it is first established by the administrator on Jan 1, 2000 has the most number of posts on the forum. This is followed by the author, Pyre who also joined in Jan 1, 2000, having the second highest number of posts on the hardwarezone forum.

The top 20 authors with the highest number of posts joined the forum before 2012 which means that those who joined the forum earlier are most likely to use the forum more frequently.

Using the same method as previous, I tried to find the more common topics discussed by the authors with the most number of posts on the hardwarezone forum.

topic\_rdd = users\_df.rdd

topic\_rdd.first()

topic\_author\_date\_rdd =topic\_rdd.map(lambda x: (x[1],(x[2], x[0], 1)))

countpost\_rdd = topic\_author\_date\_rdd.reduceByKey(lambda x, y: (x[0], x[1] + y[1], x[2] + y[2]))

df\_topic\_rdd = countpost\_rdd.map(lambda x: (x[0],x[1][0],x[1][1], x[1][2]))

df\_topic = df\_topic\_rdd.toDF()

df1\_topic = df\_topic.withColumnRenamed('\_1', 'author') \

        .withColumnRenamed('\_2', 'joinedDate') \

        .withColumnRenamed('\_3', 'topic') \

        .withColumnRenamed('\_4','num\_of\_posts')

df1\_topic.sort(F.desc('num\_of\_posts')).show()

Graphical user interface, text

Description automatically generated with medium confidence

The results show that games like Path of Exile, Hearthstone, League of Legends, EDMW are the more common topics discussed by the more frequent users of the hardwarezone forum.

This then supports the key words found in question 1 where the key words found in topics are league, edmw, exile, etc.

Next, I tested what are some of the key words from the topic for the authors who have joined in year 2008. Year 2008 has the most number of authors who joined the forum.

Continuing from the previous [code](#df1_code):

from pyspark.sql import functions as F

from pyspark.ml.feature import Tokenizer

from pyspark.ml.feature import StopWordsRemover

import pyspark.sql.functions as f

tokenizer = Tokenizer(inputCol="topic", outputCol="topic\_token")

tokenized = tokenizer.transform(df1\_topic).select("author", "joinedDate", "topic\_token", "num\_of\_posts")

remover = StopWordsRemover(inputCol='topic\_token', outputCol='topic\_clean')

data\_clean = remover.transform(tokenized).select("author", "joinedDate","topic\_clean", "num\_of\_posts")

data\_clean.show()

filtered\_df = data\_clean.filter(data\_clean.joinedDate.like('%2008'))

results = filtered\_df.withColumn('word', f.explode(f.col('topic\_clean'))) \

    .groupBy('word') \

    .count().sort('count', ascending=False)

word\_count = results.collect()[:40]

for word in word\_count:

    print(word[0] + ',' + str(word[1]) + '\t')

Results for the top 40 words and their count:

Text

Description automatically generated

These games might have gotten popular in year 2008, hence these games are the more popular ones that were discussed by the authors in hardwarezone forum. This is similar to the [results](#results) found previously.