BDA MINI PROJECT REPORT

Submitted by

PC28 Ravi Yadav
PC30 Sumanth Vullamparthi
PC34 Aashi Tapadia
PC51 Kaushiv Agarwal

Submitted to

Prof. Nitin Pise

School of Computer Science Engineering

MIT World Peace University Kothrud,

Pune – 411038

Problem statement : To do the sentiment analysis on IMDB movie reviews

Aim: To scrape a good amount of data, preprocess it, analyse it and then perform sentiment classification with a good accuracy.

Methodology:

1. Data scraping :

In the 1st phase we scraped data from IMDB website to get reviews of approximately 500 movies and about 11k rows of data. Our dataset consisted of 3 columns namely Movie name, review and review rating. For data scraping we used the beautiful soup library of python.

The steps followed were:

- Getting review page links of all the movies and store them in a text file called movie_review_links.txt

```
def get_review_links(res):

   movie_tags = res.find_all('a', attrs={'class': None})
   movie_tags = [tag.attrs['href'] for tag in movie_tags
             if tag.attrs['href'].startswith('/title') &
tag.attrs['href'].endswith('/')]
   movie_links = [base_url + tag + 'reviews' for tag in movie_tags]
   return movie_links
```

- Getting links to all the individual reviews in each of the movie review links. There were approximately 25 reviews per movie(We will look more into it in the preprocessing phase)

```
def get_single_review_links(reviews_link):
   review_res = getHTMLcontent(reviews_link)
   review_tags = review_res.find_all('a', attrs={'class': 'title'})
   single_review_links = [tag['href'] for tag in review_tags]
   single_review_links = [base_url + tag for tag in single_review_links]
   return single_review_links
```

- Get data from each review : Movie name, review text, review rating

```
def get_review_data(review_link):
   movie_title = ''
   review_text = ''
   rating= ''
   review_res = getHTMLcontent(review_link)
   try:
       tag = review_res.find('h1')
       movie_title = tag.a.text
   except:
       print('no title found')
   try:
       tag = review_res.find('div', attrs={'class': 'text show-more__control'})
       review_text = tag.getText()
   except:
       print('no text found')
   try:
       rating=int(review_res.find_all('span', attrs={'class':
'rating-other-user-rating'})[0].find_all('span',  attrs={'class': None})[0].getText())

   except:
       print('no rating found')
   review_doc = {'movie_name' : movie_title, 'review' : review_text,'rating': rating}

   return review_doc
```

To run this on large data we used multi threading using the multiThreading pool library of python

```
start = time.time()
result =[]
with ThreadPoolExecutor(max_workers = 100) as exe:
   # Maps the method 'get_review_data' with a list of values.
   result = exe.map(get_review_data,single_review_links)
end = time.time()
print('execution time: ' + str(end - start))
```

- Dumping this data to mongoDB for further analysis by connecting python to mongo compass using the pymongo library

2. Data analysis

- Analysing duplicates and NaN values:
  There were a total of 11000 rows in the raw dataset we scrapped.But after analysing we found out that there were many duplicates and also NaN values in some columns. So we removed all the duplicates and replaced all empty strings with NaN. After this Number of rows reduced to 8723 in number. Then we analysed the individual count of null values for each column and found that there were 938 values in the ratings column .We decided to remove these rows as our prediction was dependent on this column. Final size of the dataset became 7785 rows.

  We created a final dataset of this 7785 row for our final analysis and map reduce technology.

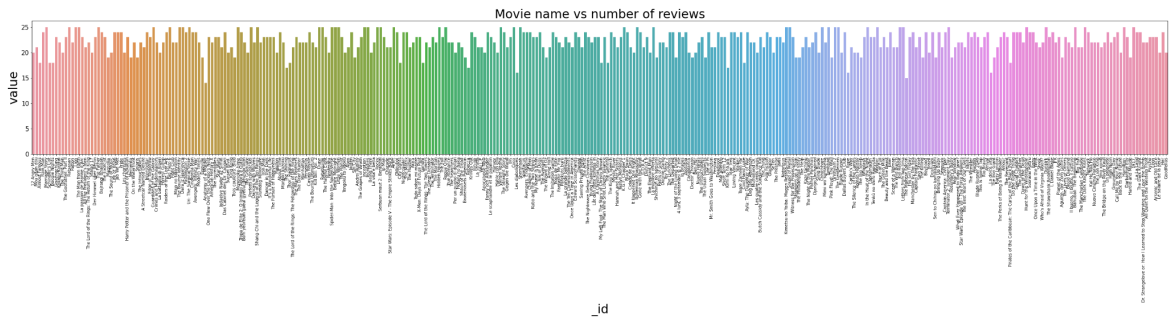
- **1st query using map reduce**

Query for finding reviews per movie - Mapreduce

```
In [215]: from bson.code import Code

In [216]: mapping = Code("function(){emit(this.movie_name, 1)}")

In [217]: reduce = Code("function(key, value){return Array.sum(value)}")

In [218]: result = db.removed_duplicates.map_reduce(mapping, reduce, "movie_review_count")

In [219]: movie_review_count = db.movie_review_count.find()

In [220]: movie_review_count_df = pd.DataFrame(movie_review_count)

In [221]: movie_review_count_df.head()
```

Out[221]:

	_id	value
0	12 Angry Men	20.0
1	Mou gaan dou	21.0
2	Almost Famous	18.0
3	Rain Man	24.0
4	Marriage Story	25.0



After analyzing this plot we concluded that there were 350 unique movies in our dataset and most of the movies were having around 25 reviews. There were only some movies that had a review count less than 15.

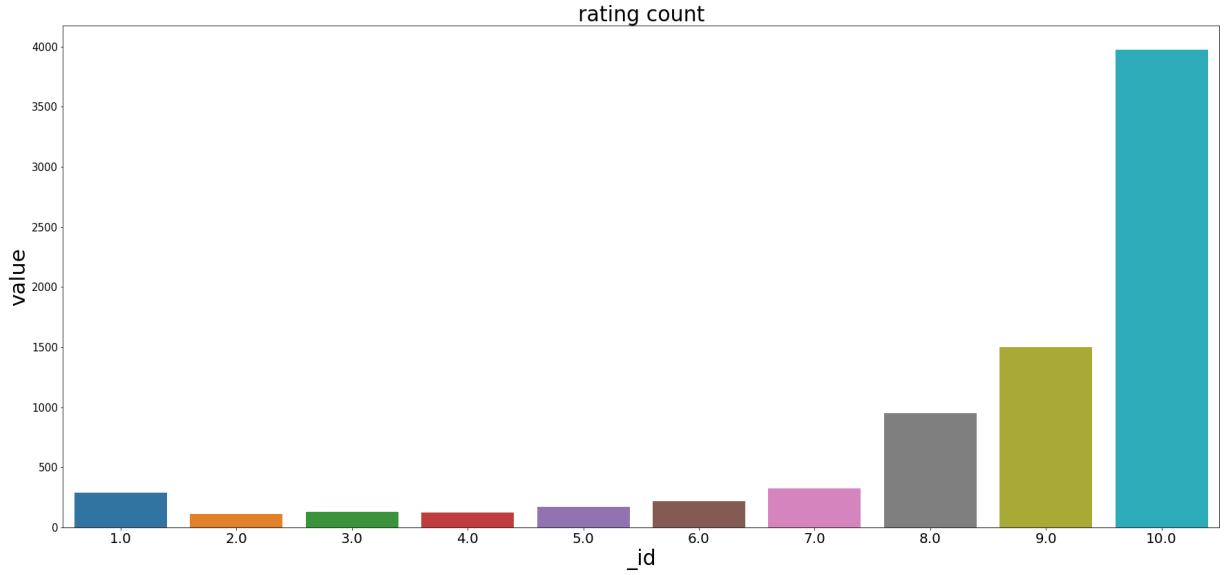
- **2nd query using map reduce**

Query for calculating number of reviews in each rating(1 - 10) - Mapreduce

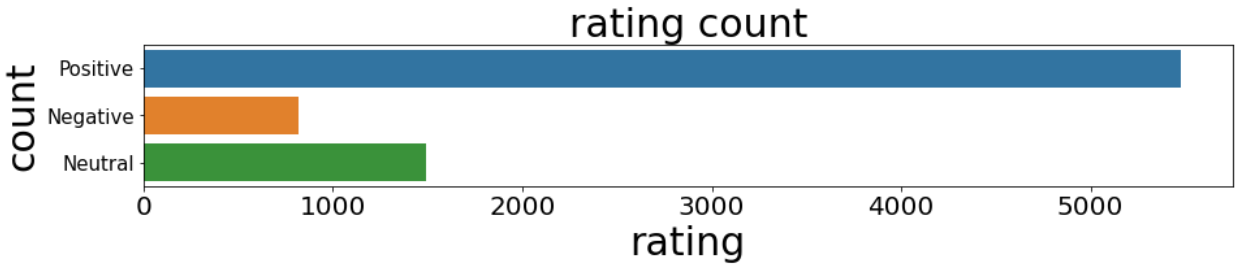
```
In [224]: mapping = Code("function(){emit(this.rating, 1)}")
          reduce = Code("function(key, value){return Array.sum(value)}")
          result = db.removed_duplicates.map_reduce(mapping, reduce, "rating_count")

In [225]: rating_count = db.rating_count.find()

In [226]: rating_count_df = pd.DataFrame(rating_count)
          rating_count_df.head()
```

Out[226]:

	_id	value
0	10.0	3976.0
1	7.0	323.0
2	6.0	216.0
3	9.0	1500.0
4	8.0	952.0



After plotting this graph we found that there were many reviews that had rating of 10 and so we decided to make groups of rating categories for assigning negative,Positive and neutral categories for each review.
After assigning labels and sentiment to reviews we found that still positive reviews were more in number.



- **Data Cleaning**

  We created a function to remove stopwords, punctuations, white spaces from the text. This function also tokenized it and lemmatized it.

In [269]:
```
df
```

Out[269]:

	index	_id	movie_name	review	rating	label	sentiment
0	6089	614f68bfee63d6915969a60c	Short Term 12	[see, movie, realize, academy, work, movie, n,...	10	2	Positive
1	11099	61501e96bb541ff327e496d9	The Lunchbox	[old, say, repeat, ritesh, batra, s, charming,...	9	2	Positive
2	9528	61501e95bb541ff327e490b6	Harry Potter and the Prisoner of Azkaban	[s, anything, movie, prove, difficulty, separa...	8	1	Neutral
3	1807	614f34b8ee63d69159699551	Whiplash	[saw, 24, hour, ago, best, fest, sundance, las...	8	1	Neutral
4	5856	614f68bfee63d6915969a523	Airlift	[story, must, spread, airlift, compelling, tha...	10	2	Positive
...	...	...	...	...	...	...	...
7780	3114	614f34b9ee63d69159699a6c	Kimi no na wa.	[go, see, kimi, na, ha, without, know, anythin...	10	2	Positive
7781	3749	614f34b9ee63d69159699ce7	Memento	[lose, memory, would, go, close, one, worst, e...	10	2	Positive
7782	6474	614f68c0ee63d6915969a78d	Black Swan	[darren, aronofsky, prof, absolute, genius, mo...	9	2	Positive
7783	10598	61501e95bb541ff327e494e4	Bringing Up Baby	[lot, film, 1930s, 1940s, incorrectly, refer, ...	8	1	Neutral
7784	8368	61501e94bb541ff327e48c2e	Strangers on a Train	[let, swap, murders, wife, father, seemingly, ...	9	2	Positive

7785 rows × 7 columns

- **Analysis by word cloud**

  We tried to analyse this review column by creating different word clouds for each positive, negative and neutral sentiment. Below given is one such cloud

3. Training our Model

We split our data in an 80-20 ratio and created a model by using KNN with TFidf. After Training this model gave us 70% accuracy and gave fairly good results Below are the screenshots of results.

```
In [292]: accuracy_score(y_test, y_pred)
Out[292]: 0.7084136159280668

In [301]: text_clf.predict(["ok,I am learning Natural Language Processing in fun fashion!"])
Out[301]: array(['Neutral'], dtype=object)

In [296]: text_clf.predict(["great,thats very interesting"])
Out[296]: array(['Positive'], dtype=object)

In [298]: text_clf.predict(["that's bad"])
Out[298]: array(['Negative'], dtype=object)

In [299]: text_clf.predict(["Modi Goverment"])
Out[299]: array(['Positive'], dtype=object)
```

Conclusion

Hence we were able to scrape movie reviews from IMDB and apply sentiment analysis on the reviews. There were a few problems with the dataset which were responsible for the inconclusive and inaccurate results. The first problem that was noticed was that the amount of positive reviews were more as compared to the negative ones and this increased the count of positive sentiments and the same can be seen in the bar graph. The second problem was the lack of data and its complex nature which made it hard to scrape and analyze. The above problems can be solved in the future by selecting a more neutral source of data and in adequate quantity.