Denmark Technical University

02807 COMPUTATIONAL TOOLS FOR DATA SCIENCE

Final Report

Authors:

Enrique Yegros (s171839)

Emmanuele Yaafi (s161046)

7-December-2018

Table of Content

1. Introduction

Majority of products and services owners rely on review comment section in order to shape their products or mend and improve their services. This means that review comments of products and services have become an important source of information for both consumers and owners. From the information, diverse conclusions can be drawn that can affect products or services either positively or negatively.

It is absolutely central that a product or service owner respond review comment. This means he takes into consideration the concerns shared by many and emphasizes positives common in the review comments.

This report assume ownership of Amazon review comment data. It is going to view the data with data science looking glass with four main objectives to be achieved.

1. Separate the given review to good or bad comments.
2. Identify prevalent good reviews and bad reviews.
3. Design a mechanism to classify a new review comment into a positive or negative one.
4. Improve the speed at which the classification can be done.

In order to achieve these objectives, the report is divided into 6 sections including this one. Section two expatiate the problem space this report is about. Section three discuss the data science tools adopted in pursuit of our objectives. How the selected tools are applied in our project is discussed in section four. We discuss our findings in section six and conclude in the final section.

2. Problem Space: *Identification of the problem in the field of data science and that is technically challenging.*

In this section we will explore in detail problem confronting us and the solutions we devise.

Amazon's Challenge

Amazon is the world's largest e-commerce business and it wants to remain so in the foreseeable future. For this reason it is embarking on consumer oriented drive. The

success of the consumer oriented drive will depend on its ability to incorporate consumers' reviews in its daily operations.

The Task

The above challenge present a task whereby consumer reviews must be mined, cleaned, splitted and categorized and then stored. Comparison of words are done and estimate group of words that appear frequently. Mechanism must be device that will accept new input and re-categorizes the new such into 'good' or 'bad'. In order incorporate the the review comments daily, this mechanism must be as fast as possible.

Existing solution

The existing solution that Amazon is using is by hiring many review agents to manually go through the comments and draw management attention to adverse comments. This mode of practice has led to situations where agents have sometimes deleted or blocked comments. As result, similar adverse comments keep coming. This time, making references to previously written adverse comments that have been deleted. Due to the deletion of bad comment, it cast the company into bad light as uncaring and do not take customers' concern serious.

3. ANALYTICAL FRAMEWORK: ***Analyze the problem and choose relevant tools to solve the problem.***

In the previous section, Amazon's challenge, the task and its existing practice of handling review comments were discussed . In this section, we provide a case for data science tools that can used to accomplish the task and provide what seems to be expensive way and inefficient way of handling review comments. We will then discussed the theoretical underpinnings of the tools.

Considering the challenge and the task to be performed, we need a set of tools that can analyse and synthese customers' review comments. That is to say, we need tools capable of breaking down our data and put together in a way that simply perform the task. In our view, many tools can be utilized but specific tools from data science are more suitable. To be specific here, k-shingle of natural language processing will able to remove undesirable elements the data contains and re-fashion the data into 'good' or 'bad'. The bloom filter will then speed up the process of re-classifying the new review comments.

The NLP is how computers are used to make sense of human language[1]. Different applications can be used. In this project, k-shingles that is part of the tool set of NLP is used. The natural language toolkit (NLTK) package in python that is able to apply k-shingles to tokenize[2], stem and remove specific things from the text.

In the bloom filter we used the mmh3 hash function and a bit array. Bloom filter is a space efficient probabilistic data structure used in order to test if an element is in a set or not.

There are 3 things we need to choose in order to use bloom filter, the number of hash functions, the length of the bit array and the probability of false positives. The best way to do this is to decide the percentage of false positives you are willing to accept and from there calculate the rest, considering that the number of items to be added to the bloom filter is known, thus:

- $m = -\frac{(n\ ln(P))}{(ln2)^2}$, where 'm' is the size of Bit Array, 'n' is the number of elements to be inserted and 'P' is the probability of false positive.
- $k = \frac{m}{n}\ ln(2)$, where 'k' is the optimum number of hash functions, 'm' is the size of the bit array and n is the number of elements to be inserted.

We can also calculate the probability of false positive, but since we already decided to use a fix value we skipped the formula. The caveat here is that we can have a faster search in the bloom filter and be more space efficient, but is all at the expense of having a higher false positive probability.

For the hash function, as we mentioned earlier, we will be using murmur from mmh3 in python. The hash function need to be independent and uniformly distributed, murmur qualifies as a fast, simple non cryptographic hash meeting this criteria.

We also decided to try our problem in a more primitive way, by hard coding a classification where we contain a long (extremely long) list of strings and then get a smaller one and try to see if there are any matches. This as we can guess takes a long time to do, specially if there are many reviews in the queue.

---

[1] https://en.wikipedia.org/wiki/Natural_language_processing accessed 03-11-2018

[2] A **token** is the technical name for a sequence of characters — such as hairy, his, or :) — that we want to treat as a group. See http://www.nltk.org/book_1ed/ch01.html accessed 03-11-2018)

4. IMPLEMENTATION OF TOOLS: The Implementation of relevant tools to solve the identified problem.

In the previous section we briefly expose the technical tools we use in the pursuit of our objectives. This section explains how these tools are applied to tackle specific aspects of our task. There are three main part of this section. Part one basically looks at the data processing and cleaning while part two looks at the application of NLP. In part three, we apply bloom filter to achieve speed during re-classification.

Part I

The amazon review data was downloaded from “kaggle’s” website. The first challenge we encounter was uploading the data into python. This is because the file format required some decoding. There were 2 files, train and test, we decided to only use the test file since the train file was massive and our computer capabilities would not suffice or render a considerably low speed. The test data contained 400000 reviews, 200000 positive and 200000 negative, we consider this number enough for the purpose of this project.

After loading the file the data was then divided multiple times. The first time we divided the data into the train and test reviews, 350000 and 50000 respectively, then it was divided into line strings where each line was a review comment. All comments were then splitted into two, good and bad reviews. Doing this helped us gather and group all words that we deemed positive and negative. All comments with two stars and below are regarded as bad comments and the comments with more than three stars are considered good, we ignored the case with neutral reviews.

Part II

We carried on by tokenizing each review, making all words lowercase and eliminating punctuations and terms that were viewed unnecessary. We then continue by dividing each line into shingles, this is in order to have a better understanding of the context of the review. We decided that using shingles has more value than just one word tokens, this is because a review could be misrepresented, for example someone could write a

review saying something in a sarcastic manner and we will be picking the bad/good words that could lead to misclassification of the review.

The shingles have size 3, meaning each shingle contains 3 words, and there will be a total of k-2 shingles per line or review. This shingles are then passed to our bloom filter function. Knowing the size of the bloom filter we are constructing made the task simpler since we do not have to worry about approximating the number of items.

Part III

In the bloom filter we hash the shingles and save them into a bit array, this saves time and space. There will be two bit-arrays saved, one for the shingles taken from the good reviews and similarly for the bad reviews. When we get a new review we clean the review in the same manner as we did with the train reviews and separate the review into k-2 shingles for k words in the review and pass it through our bloom filter, but this time we only hash without saving it. We then searched through both bit-arrays created earlier to see if the position where the hashed shingle should go is taken or not, thus we are able to classify the shingles. Now, depending on how many shingles are classified as good or bad we are able to say whether a review is a positive or a negative one. The bloom filter only tell us the possibility of existence or if it does not exist therefore there is always the possibility of having a false positive, but it is possible to lower this probability in exchange of efficiency and memory as we discussed in the last section. We decided to use multiple values of p for the probability of false positive.

In order to see which reviews were classify correctly we also divided the test data into good and bad and proceeded in the same fashion as above. By doing this, we saved an array of the size of the comments for each and with zeros, then every time we classify correctly, we change the number to a one. We also saved the index of the misclassified reviews in order to investigate further this misclassification.

5. RESULTS AND DISCUSSIONS: Argue clearly for the choices made when designing and developing the solution.

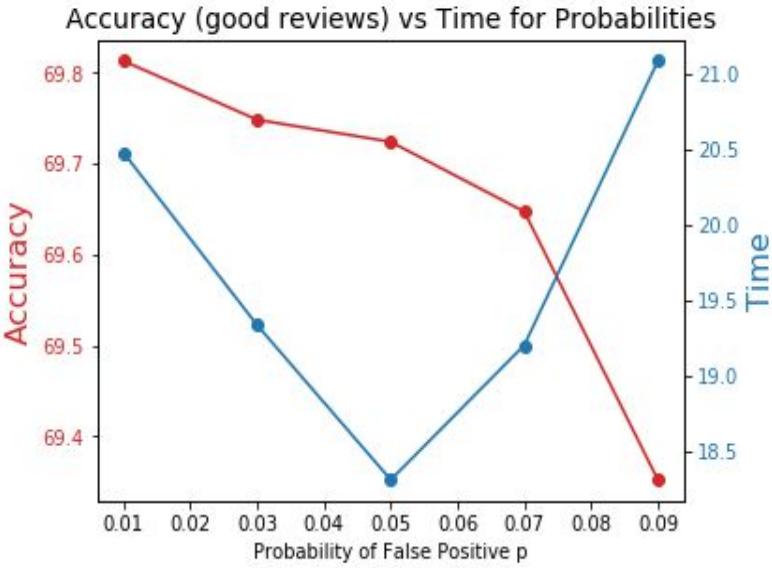
This section summarizes the result of we obtained at the implementation section.

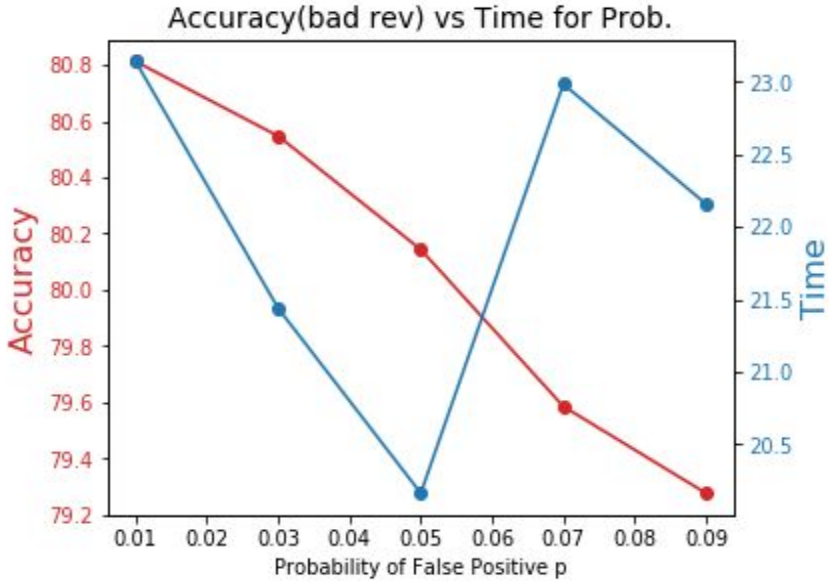
We decided to start the project by using a hard coded implementation by creating lists and checking if an element is in the list. This implementation took a great deal of time since we had to go through millions of shingles. Checking 27.5 million shingles around 50 times per review leads to a lot of computational power and time. Considering this we decided towards a more efficient solution, thus the bloom filter.

The bloom filter allow us to check the probability of the existence of an object in a set or the certainty that the object does not exist in the set. We are not able to check the elements nor eliminate it after it has been added to the set in the bloom filter, but this shortcomings are compensated by the efficiency in terms of speed to check element existance and space saved. With this in mind we thought that bloom filter was a great choice for the problem at hand since we had millions of shingles, approximately 14 million shingles for the bad reviews and 13 million shingles for the good reviews.

In order to see the results and compare the efficiency and space saved we decided to use multiple probabilities of false positives of p. We cleaned and separate the data into shingles once, this task took a few minutes. After this process we used five different p values, [0.01, 0.03, 0.05, 0.07, 0.09] in order to create the bit array and obtain the number of hash functions. We did this twice, one for the negative review shingles and the positive review shingles.

In the next step we checked the existence of the review shingles in this two lists by hashing it and checking if the space where they are supposed to be saved in is empty. After doing this for all comments and looking at the ones that were supposed to be classify as positive and similarly for the ones that were supposed to be classify as negative we were able to find the accuracy.





As we can see in the graphs, the accuracy is around 69.8% when checking the positive reviews and 80.0% when checking the negative reviews. As the probability of false positive p increases the accuracy decreases and the time it takes to check the probability of existence of all review shingles drops until p=0.05, then the time increases again. We are not able to explain why this occurs.

We have looked at the efficiency in terms of speed and accuracy, no we will look at how the space changes along with the changes in p.

p-value	Bit-Array size Good-Shingles	Bit-Array size Bad-Shingles
0.01	127008953	137061551
0.03	96709617	104364062
0.05	82621229	89160595
0.07	73341444	79146327
0.09	66410282	71666572

Hash-functions-Good	Hash-Functions-Bad
8803589767	9500382763
2234466611	2411321844
1145373438	1236028300
726234501	783715048
511467774	551949803

The bit-array size decreases rapidly when increasing the value of p, as we can see in the table above, thus occupying less space. Similarly we can see that the number of hash functions also decreases with the increase in p. The table above show us the results for multiple values of p ranging from 0,01 to 0,09. We consider that using values higher than 0,09 was not going to yield anymore insights therefore we stopped there. In

Section 3 we showed the calculations done in order to obtain these values when the number of elements to be added to the bloom filter and the value of p are known.

Here is an examples of a comment that was misclassified:

"better than the studio versions: on the CD, i prefer the live versions to the studio versions. they have much more energy and excitement. the DVD is great too. it has all of their biggest hits plus some fan-favorite album tracks. i love how they get the audience involved. my only complaint is that there are a few times when mike and chester say some cuss words that get edited out, and there isnt an un-edited version available. for some reason linkin park wont have bad words in their music, or in this case, dvd's. also, the dvd is sorely lacking of any features, such as back-stage interviews or anything. its still a great CD/DVD combo though, and i highly recommend it to any LP fan"

We are able to observed that this comment, that originally was label as positive, is a mixture between a positive and a negative one. This might be the reason why it was misclassified as negative.

The classification gives some promising results considering that the accuracy for both was around 70%, taking into this into consideration and the time it took to classify the comments we could argue that using a p=0,01 is the best choice.

Conclusion.

The results we obtained were somewhat better than expected, but the most interesting part was seeing the difference when changing values. The accuracy did not differed by much when changing the probability value, the time decreases until we reach p=0.05, these seems to be the most efficient in terms of computational time. The bit array size and number of hash functions do have big changes. We can conclude that having a low value for p might not be the most efficient in all cases, but it will yield the best accuracy and in our case is the best choice.

There is always room for improvement and many ideas came to us during the course of the project, ideas for future work. One of these ideas was the use of parallel computation to speed up our process in order to consider bigger data sets. Another idea was the use of bag of words in order to have a feature vector to train a model in order to perform classification. We also consider the normalization of text, stemming the reviews and removing stopped words, we wanted to try this during the course of this project, but

we were not unable to do so due to time. These are a the most important ideas for future work we had.

How to run the code:

The code is easy to run, we did not have enough time to create a more interactive setup. It is possible to run all the code at once if the data file is in the same directory, this will yield a a few variables, but the most important ones are tab_prediction_good, indexes_good_miss, tab_prediction_bad andindexes_bad_miss.
The tab_prediction shows a one if predicted correctly, it need to be specify if you are trying to predict a good review or a bad review in this case (is already set up for the train and test reviews).

If the user wants to predict a new review (singular) there is a few lines of code that allows this to be done, comments were added in line 308 to do so. It is still necessary to run the entire code with the files in the correct place in order to train the model. The new review need to be add as a string (**NEW_REVIEW= str("")**) and then the user need to run the line **"new_tab_prediction, new_indexe, NEW_tt = test_reviews(bloomf_good, bloomf_bad, NEW_REVIEW, "UNKOWN")"** where the tab prediction will give a 1 if positive review or a zero if negative.