THE UNIVERSITY OF SYDNEY

COMP5349 Assignment 2 Report

Group: F16B - Fri - SIT117 - 1

Name	Student ID	Unikey	Tutor
Zezheng Zhang	490197930	zzha4270	Shizhe
Jiakun Yu	490346550	jiyu7287	Shizhe
Yihong Wang	450538618	ywan0535	Shizhe
Yue Yang	470510940	yyan9073	Shizhe

Date: 22/05/2019

Introduction

This project performs basic big data analysis on one of the Amazon product review datasets. The category of the data chosen is Video and DVD. The review dataset has the size of 1.4GB and over 5 million reviews where each contains a review id, product id, customer id, comments, ratings and etc. Basic exploratory analysis and sentence processing techniques were performed during the project using Spark on AWS EMR.

Stage One: Overall Statistics

Explanatory analysis of the data is conducted using spark functions such as ‘count()’, ‘distinct()’, ‘groupBy()’, ‘orderBy()’ and ‘approxquantile()’. The results are shown in Table 2.1 below.

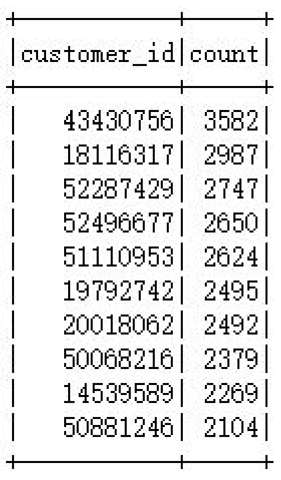
Table 2.1: Overall statistics of the dataset

Total number of reviews	5069140
Number of unique users	2075970
Number of unique products	297919
The largest number of reviews published by a single user	3582
The median number of reviews published by a user	1
The largest number of reviews written for a single product	4969
The median number of reviews published by a user	3

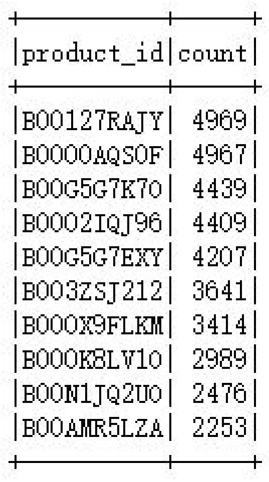
The top 10 users ranked by the number of reviews they publish is acquired using the following command. The dataset is firstly grouped in terms of the customer_id. Then the total number of reviews of each customer is computed and only the top 10 customers are shown. Top 10 products are acquired in a similar fashion. The code and results are shown below.

```
revs.groupBy("customer_id").count().\
                    orderBy("count", ascending=False).show(10)
```

```
revs.groupBy("product_id").count().\
                    orderBy("count", ascending=False).show(10)
```



```
+-----------+-----+
|customer_id|count|
+-----------+-----+
|   43430756| 3582|
|   18116317| 2987|
|   52287429| 2747|
|   52496677| 2650|
|   51110953| 2624|
|   19792742| 2495|
|   20018062| 2492|
|   50068216| 2379|
|   14539589| 2269|
|   50881246| 2104|
+-----------+-----+
```



```
+----------+-----+
|product_id|count|
+----------+-----+
|B00127RAJY| 4969|
|B0000AQS0F| 4967|
|B00G5G7K7O| 4439|
|B0002IQJ96| 4409|
|B00G5G7EXY| 4207|
|B003ZSJ212| 3641|
|B000X9FLKM| 3414|
|B000K8LV1O| 2989|
|B00N1JQ2UO| 2476|
|B00AMR5LZA| 2253|
+----------+-----+
```

Figure 2.1: The top 10 users and products ranked by the number of reviews

Stage Two: Filtering Unwanted Data

Sentence splitting

The following code is implemented to divide user reviews into sentences by ‘function.split()’ which splits the ‘review_body’ into sentences in accordance to ‘.’, ‘?’ and ‘!’. Then user reviews with less than two sentences are omitted.

```
rev_df_filtered = revs.withColumn("sentence", F.split(revs['review_body'], "\. |\? |\!
")).withColumn("sen_no", F.size("sentence")).filter("sen_no>1")
```

Review Filtering

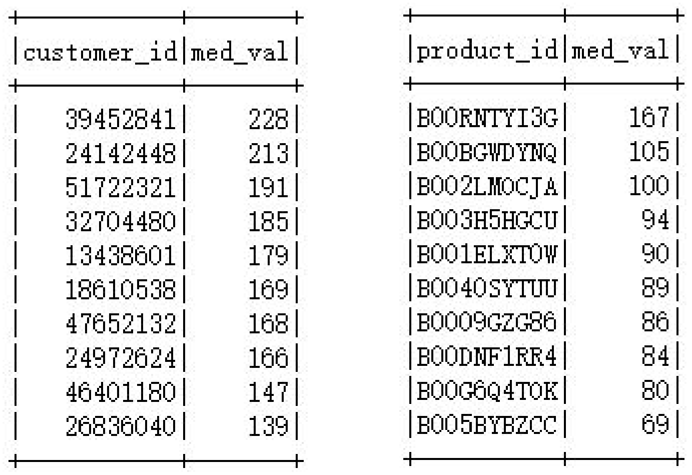
The following reviews are filtered and methods are explained:

- reviews with less than two sentences in the review body.
  Pyspark SQL function ‘size()’ is used to determine the number of sentences in each review and then filtered using SQL function ‘filter()’.

- reviews published by users with less than or equal to the median number of reviews published.
  The ‘window()’ function partitioned by customer_id is used to calculate the median where a filter over the median is applied to filter out the customers with median less than or equal to the overall median value, 1.

- reviews from products with less than or equal to the median number of reviews received.
  Similar to above but using a window partition by product_id and a median value of 3.

The top 10 users ranked by the median number of sentences in the reviews they have published and top 10 products ranked by median is shown as below:



```
+-----------+-------+        +----------+-------+
|customer_id|med_val|        |product_id|med_val|
+-----------+-------+        +----------+-------+
|   39452841|    228|        |B00RNTYI3G|    167|
|   24142448|    213|        |B00BGWDYNQ|    105|
|   51722321|    191|        |B002LMOCJA|    100|
|   32704480|    185|        |B003H5HGCU|     94|
|   13438601|    179|        |B001ELXTOW|     90|
|   18610538|    169|        |B0040SYTUU|     89|
|   47652132|    168|        |B0009GZG86|     86|
|   24972624|    166|        |B00DNF1RR4|     84|
|   46401180|    147|        |B00G6Q4TOK|     80|
|   26836040|    139|        |B005BYBZCC|     69|
+-----------+-------+        +----------+-------+
```

Figure 3.1: The top 10 users and products ranked by the median number of sentences in the reviews

Stage Three: Similarity Analysis with Sentence Embedding

Positive vs. Negative Reviews

The product “B00AMR5LZA” is chosen for further analysis in stage 3 and 4. The processed data in stage 2 is then filtered according to star ratings into positive and negative reviews data frames. The Pyspark ‘explode()’ function is applied to sentences so that each row contains one individual sentence. The individual sentences column is then extracted from the data frame and formatted into RDDs.

Intra-Class Similarity

The following procedures are applied to both positive and negative sentence RDDs respectively.

Table 4.1: Procedures for calculating cosine distance between sentences

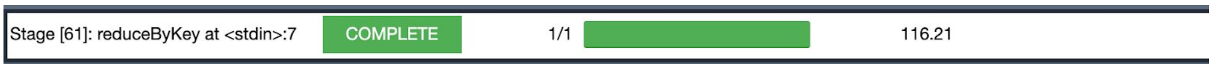
	Procedures	Data Structure
1	MapPartition the review RDD using Google’s Universal Sentence Encoder (cached)	(vector)
2	Zip the 512 dimension vector with Unique ID (cached)	(vector, ID)
3	Create cartesian RDD with itself (cached)	((vector i, vector j), (ID i, ID j))

4	Calculate cosine distance (1- cosine similarity) of each key-value pair in the RDD	(ID i, cosine distance between i and j)
5	Reduce by key to calculate sum of the distance to all other sentences, since cosdist(i,i) = 0 (cached)	(ID i, sum of the distance to all other sentences)

The exact formula of the cosine distance function between x and y is calculated by one minus the dot product of x and y divided by the norm of x and norm of y.

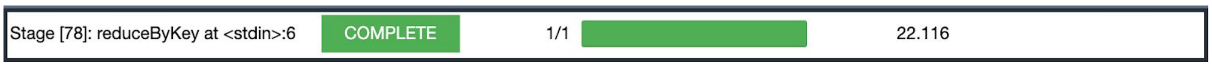
The average distance is then calculated by summing all the distance values in the RDD and divided by the n*(n-1) where n is the total number of sentences in RDD.
The average distance for positive class is: 0.7148459355281314
The average distance for negative class is: 0.7222256875855203

The processing time for computing the intra-class similarity is shown below, which the procedure 5 (i.e. ‘reduceByKey()’) consumes most of the running time**:**



Stage [61]: reduceByKey at <stdin>:7 COMPLETE 1/1 116.21

Figure 4.1: Intra-class similarity computation time of the positive class.



Stage [78]: reduceByKey at <stdin>:6 COMPLETE 1/1 22.116

Figure 4.2: Intra-class similarity computation time of the negative class.

Class Center Sentences

The centre sentence is found by finding the minimum value in the reduced RDD which is in the form of (ID i, sum of distance to all other sentences), as the output of the procedure five in the last section. The ID for the minimum sum value is the centre sentence.

The distance from the centre sentence to all other sentences are then found by filtering the mapped cartesian RDD of the form (ID i, cosine distance between i and j) in the procedure four according to the centre sentence ID. Then the list of the distance to the other sentences is ordered and only the top 10 closest sentences are collected. Results are shown as below where the first row is the centred sentence and the following 10 rows are its closest neighbours. The smaller the number, the closer to the centred sentence.

Table 4.2: Customer_ids and sentences of the *positive* centre sentence and the top ten closest sentences by Google Pre-trained universal sentence encoder:

No.	Customer_id	Sentences
0	R3G3KHQ43Y860A	This is the best bible movie ever made
1	RPHGQC7A7NZHG	One of the best made movie versions of the bible yet

2	RRX99ACI7D1QI	Of all the Bible based films, this was my favorite Jesus
3	R1P0X6G9JNUSHN	Truly a great movie about the BIBLE
4	R2A85C5IBC5LVH	One of the most incredible movies out there about the Bible
5	RFQSUAJDSY6SW	This is one of the best movies that depict the stories of the bible
6	R1VV03CKEK08I0	This mini series is one if the best bible movies made
7	R1NRPFNQIG56FX	It is one of the Best Bible stories ever depicted on Movie setting if not The Best
8	R3NFXPKKG26EYT	Of all of the Biblical movies I've seen, this by far is the best one yet
9	RC0FZ9SNACXIG	Fantastic Movie for Bible lovers
10	R3VAO8UAOKW8JK	Best movie I have seen that tells about the Bible

Stage Four: Similarity analysis with Spark Supported Feature Extractors

Positive vs. Negative Reviews

The exact same RDDs are used as in stage three for positive and negative reviews.

TF-IDF Embedding Mechanism

Intra-Class Similarity
The procedures are nearly identical to stage three where the Spark library function TFIDF is used instead of Google's Universal Sentence Encoder in sentence encoding. The length of the TF-IDF vector is 10000 without PCA. Due to time constraints, PCA is not applied and it will improve the performance by reducing a huge amount of time.

Cosine distance function for this stage is slightly different as the output vector from TFIDF is SparseVector whereas Google's used Numpy. The calculation is the same but with a slightly different format.

The average distances of intra-class similarity are shown below:
The average distance for TFIDF positive reviews is: 0.7569218811530731
The average distance for TFIDF negative reviews is: 0.7600351790298421

The processing time for computing the intra-class similarity by TF-IDF is shown below. Similarly, the procedure 5 (i.e. 'reduceByKey()') consumes most of the running time**:**

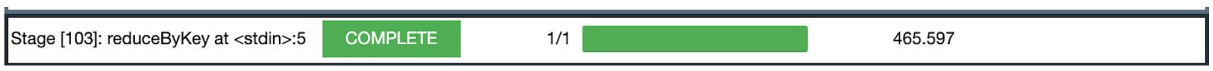

```
Stage [103]: reduceByKey at <stdin>:5    COMPLETE    1/1    465.597
```

Figure 5.1: Intra-class similarity computation time of the positive class by TF-IDF.

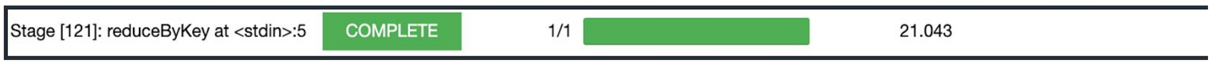

```
Stage [121]: reduceByKey at <stdin>:5   COMPLETE   1/1   21.043
```

Figure 5.2: Intra-class similarity computation time of the negative class by TF-IDF.

Class Center Sentences
The exact same method in stage three is used and results are shown below:

Table 5.1: Customer_ids and sentences of the *positive* centre sentence and the top ten closest sentences by TFIDF:

No.	Customer_id	Sentences
0	RLRHV7WF6NX0T	It has brought the Bible to live for me and …... very compelling
1	R3LFBWNMKXRA7G	I don't approve of their ideas and I don't like …... everything else
2	R2WMLDC0RI9D9X	I found it to be very thought …... our family members.
3	R2CWQLPV2MZL53	I like most of background landscape in the …... ancient Bible time
4	R34K769M3MFI82	I think it would have been better if…... more emotional acting.
5	RVXBCW48RZAXN	I am still not a believer, but this film did raise my …... film to the book
6	R350LYVU4BZJWV	I highly recommend it to both believers …... of the scriptures.
7	R3LFBWNMKXRA7G	It can be difficult to step outside of our own …... better friend
8	R2UO03ITZPNRST	I don't think it is completely accurate as …... communicated well.
9	R1CA4PVZXB01QP	perhaps someone should be looking at a full …... than ever
10	RB11W39PW9689	I am not a religious person, but in the past i …... movie is about

Word2Vec Embedding Mechanism

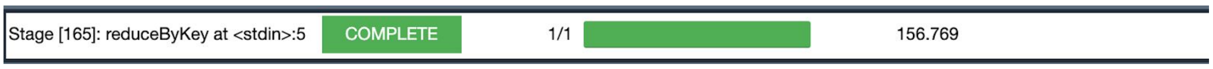
In stage four, the word2vec embedding mechanism is also implemented to compare the performance. The model maps each sentence to a unique fixed-size vector. Typical word2Vec dimensions are 50,100, 200 and 300. In our case, the vector length is 300, which theoretically speed up the calculation process because stage three needs to handle 512 length vector and TF-IDF needs to handle 10000 length vector. The ‘Word2Vec’ is imported from ‘pyspark.ml.feature’ library that makes Spark data frame or RDD suitable for this model.

The input of word2vec is Array[String] Type data frame so the previous calculated sentence RDD needed to transform itself as required. After word2vec transformation, those vectors could compute the pairwise distances by RDD cartesian as the same procedure as in stage three.

From the class centre and its 10 closest neighbours outcomes generated by word2vec embedding, it illustrates that most of the positive review sentences are quite short and length of negative review sentences are normal. Intuitively, they look very similar among the same class, which satisfies the requirement.

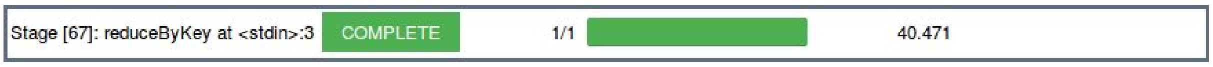
The average distances of intra-class similarity are shown below:
The average distance for Word2Vec positive reviews is: 1.0000091232411779
The average distance for Word2Vec negative reviews is: 1.0000091232411779

The processing time for computing the intra-class similarity by Word2Vec is shown below. Similarly, the procedure 5 (i.e. 'reduceByKey()') consumes most of the running time**:**



```
Stage [165]: reduceByKey at <stdin>:5   COMPLETE   1/1   156.769
```

Figure 5.3: Intra-class similarity computation time of the positive class by Word2Vec.



```
Stage [67]: reduceByKey at <stdin>:3   COMPLETE   1/1   40.471
```

Figure 5.4: Intra-class similarity computation time of the negative class by Word2Vec.

Class Center Sentences
The exact same method in stage three is used and results are shown below:

Table 5.2: Customer_ids and sentences of the *positive* centre sentence and the top ten closest sentences by TFIDF:

No.	Customer_id	Sentences
0	R1KFSV90H1M9I5	this movie is excellent.
1	R1HJUXE8B7WWUK	awesome series if you have not boughten this …… what are you waiting for?
2	R2N3XIWU4JID2K	this covers all things of historical fact.
3	R34GF5Z93B8ELU	it's a great summary that capture your attention.
4	R1MBWS4LYK4GY6	very good.
5	R35AWA5YDSETP	very good.
6	RECGRWZQKIN56	very good.
7	R16S08JK4654YK	very good.
8	R2HVMVVIA2Z2PS	in a lot of areas it is not always accurate with scripture, but overall, pretty good.
9	R1821CPKEEDAMU	a basic for everyone to get their feet wet to be inspired to read the entire bible.
10	R19LT7UG961QVY	humble and compassionate but authoritative.

Performance Comparison of Stage Three and Stage Four

Table 6.1: Computation time comparison between stage three and stage four

	Time Taken of Positive class (n=2661)	Time Taken of Negative class (n=557)	Average Closeness of Positive class	Average Closeness of Negative class
Stage three	116s	22s	0.7148	0.7222
Stage four word2vec (vectorSize=300)	157s	10s	1.0000	1.0000
Stage four TFIDF without PCA	466s	21s	0.7569	0.7600

For both stage three and four, only one step consumes a significant amount of time which is the ‘reduceByKey()’. Other steps such as ‘collect()’ take a minimum amount of time which can be neglected. Hence we are only comparing the time used for ‘reduceByKey()’ which includes a couple of steps as well including zip with index, cartesian, maps and distance calculation.

Stage three consumes the least amount of time to compute the average distance and centre classes overall. The word2vec mechanism took longer than Google’s method with a smaller dimension of vectors. However, the negative class used less time than Google. This is probably due to the overhead on Google’s embedding technique which adds additional time on all computations where small amount of data would be most affected. We also implemented TFIDF without PCA where the vector dimension is over 10000. The dimension is much higher than word2vec mechanism. The time consumption on the TFIDF is approximately triple the amount of word2vec where the dimension of vectors is 300. The additional time is most likely due to more complex distance calculations and sentence transformation.

The average closeness by Word2Vec mechanism is approximately 1 which means the sentence vectors mostly has a dot product equal to 0, hence leading the cosine distance to be 1. This is probably because the sentences are treated as vectors and the exact same sentences appear rarely resulting dot product to become 0.

Another observation is that TF-IDF centre sentences tend to be very long and the other two methods tend to find shorter sentences. The reason for this observation requires further investigation.

Appendix

Dataset location:

rev_data = "s3://amazon-reviews-pds/tsv/amazon_reviews_us_Video_DVD_v1_00.tsv.gz"

Google Pre-trained universal sentence encoder negative class:

0 is the centre sentence
0 R1TQT3UD1NYPEF The makers of it had a great opportunity to tell the story of the Bible
1 R2BOHBJN6W1FFO I have heard that it was created to introduce people into the bible
2 R2TYF4KR0I67D5 Good bible story books are really inspired by the bible
3 R2TYF4KR0I67D5 They should have done the screenplay ….. to add)with the bible
4 R3TPP75ZVB83FE The Bible
5 RFHUIBNG40GHX This is the Bible
6 R1S1GCV6NJM37Q The writers should have READ THE BIBLE before they started …... could improve upon
7 R2NWEHJH3SBRYO The Real Bible
8 R3021YB26T5Q0N This was called THE BIBLE, not Most of The Bible, …... in the Bible
9 RM3GHVBLVWL5 This was the worst recreation of the bible known to man
10 R149PWP0UVY4KY A simple reading of the Bible would show that just the …... interpretation of the Bible

TFIDF negative class:

0 is the centre sentence
0 R1LUM57F7DZUGH Considering this is the history channel, …….even writing about
1 RDT5RGRIELAZI It is gruesome cinematic violence ….. and everybody
2 R31UK8DHK4URJ2 It would have been just as easy to ….. nothing to back it up historically
3 R1LUM57F7DZUGH It doesn't work as a historic …...how the events are even connected
4 R3MMPM137A6F11 It may give the impression of negative …... without intending to do so
5 R2A1N48RYXSW59 Yes, show some historically …... the ultimate goal here, isn't it
6 RGNUHQHATUIBY I literally threw it in the trash …... erroneous positive reviews
7 R117JQ6YWB9FDL If you have not read the …..., at the glaring commissions and omissions
8 R329ZHBZ5WWJWP I've only watched disk one …... reason wound up buying this movie
9 R7XUYN5QXA5CD The camels have perfect white…... clothes in the middle of the desert
10 R19JAWVGYUG0VY I read in an interview with …... to the younger generation

Word2Vec negative class:

0 is the centre sentence
0 RQZT7L2KEGVJN: when i saw, the \\"leaking\\" ark, i knew this did not bode well.
1 RB52RK9IVGUH6: many details of the individual stories were altered or embellished for, i suppose, dramatic effect, thus the need for a disclaimer at the beginning of each episode, always a warning that the content will be altered beyond the approval of many.
2 RPNUUWLSPFJTU: horrible!
3 R2IWBY39860T88: no, now it's reality shows and blood-and-guts vikings and romans and swamp people and heaven knows what's next!
4 R1OXSBNJ4VDZF3: the writers have not only taken so many liberties with the bible to make this movie &#34;agreeable&#34; to a new audience, but it also changes and/or leaves out so much important aspects of the bible that it shouldn't even be sold in christian book stores.
5 RRHOQWC1P180Y: in this series, jesus heals a bunch of people and draws a lot of attention to himself.
6 R20FT0D9JP1CJD: he had hair like lambs wool, skin like brass burnt in an oven(anyone who has burnt brass in an oven knows it is black, not brown, or white or &#34;light-skinned&#34;).
7 RDIHIC8TAYH8K: i've seen some of their past documentaries pertaing to scripture, but never was really impressed.
8 R1J4FT2BUVEL0G: um, maybe not.
9 R3PY3UV1CJPD5T: read the bible, don't rely on some movie to reveal the truth to you.

10 RZ2GQ5E0F4C1L: you changed critical facts in many of the stories i saw...and for what reason....isnt the truth good enough...its the truth that sets us free...we do not need to add hollywood details to gods story...so sorry this went so greatly anticipated...a huge disappointment