Final Report Paper

Ivan Kho, Alejandro Zuniga, Kinmen Yang, Arshia Singh

December 6, 2014

1 Introduction

The objective of this project is to predict empirical CTR through statistical models using features of data recorded from a search enging over a ten-day period.

The data we are provided with is the Track 2 data of the KDD cup 2012, which includes 12 categorical features in the main data file, and five additional data files: query_tokensid.txt, purchasekeywordid_tokensid.txt, titleid_tokensid.txt, description_tokensid.txt, and userid_profile.txt. The main data was divided into three sections: training, testing, and validation.

The features we chose to employ were gender, age, and the token similarity for QueryId, TitleId, Keyword, and Description. The statistical model we used was the Naive-Bayes classifier.

2 Complete Analysis

2.1 Aggregating Data for Gender and Age Features

We aggregated the gender and age data through AWS using two map reduce processes, using "userid_profile.txt" and the training files provided in s3. The mapper file "mapper_age_1.py" extracted userid, clicks, and impressions from the training files and userid and age from the userid file. For gender we followed a similar procedure, replacing age with gender, using "mapper_gender_1.py" and so on.

For any missing values not included in the data being operated on by the mapper, a value of -1 is assigned for that particular variable. For example, the training files lack data on age, so for that line the mapper outputs: 'userid' \t 'click' \t 'impression' \t 'age', where age = -1. Likewise, for userid_profile.txt, this file lacks the 'click' and 'impression' variables, so a value of -1 is assigned to these two.

This output is then taken and inputted into the reducer file "reducer_age_1.py" which takes the output and begins concatenating the data by pairing the seperate data through userid. The output is of the form: current_age \t current_click \t current_impression. Similarly, gender is done with the files "mapper_gender_1.py"

and "reducer_gender_1.py".

The second map-reduce process takes the output of the first map-reduce process and inputs it into the mapper file mapper_age_2.py and splits the lines into the variables 'age', 'click', 'impression'. This output: 'age' \t 'click' \t 'impression', is then inputted into the reducer file reducer_age_2.py. The reducer aggregates the total number of clicks and impressions for each age. The second map-reduce for gender is done using the files mapper_gender_2.py and reducer_gender_2.py in a similar fashion.

The reducer then outputs data in the form: 'feature value' \t 'feature name' \t 'clicks' \t 'impressions'. We aggregated the data using the shell command: cat part-* >output.txt, allowing us to run the data through the Naive-Bayes Model. An example of the ouput is available in the Appendix (Figure 1).

2.2 Aggregation of Data for Similarity Index

To begin, the files titleid_tokensid.txt, queryid_tokensid.txt, descriptionsid_tokensid.txt, and purchasedkeywordid_tokensid.txt were each run in MapReduce with their corresponding *_append.py file (e.g. title_file_append.py was run with titleid_tokensid.txt file and an identity reducer). These scripts appended an extra column at the start of each line to identify the token file, such as 'title'. This was done because otherwise each of the token files were indistinguishable.

Using the outputs from the above append.py outputs along with the training data, we ran four MapReduce jobs to append the tokens to their subsequent ids. The Map-Reduce files are as follows:

1. mapper_tok_1.py, reducer_tok_1.py
2. mapper_tok_2.py, reducer_tok_2.py
3. mapper_tok_3.py, reducer_tok_3.py
4. mapper_tok_4.py, reducer_tok_4.py

The first map-reduce job compiles the tokens based on the query, then it is fed into the second map-reduce which compiles based on title. The subsequent two mapreduces compile based on keyword and description id. The output is of the format: queryid \t query_tokens \t titleid \t title_tokens \t keyid \t key_tokens \t descrid \t descr_tokens \t click \t impression.

After all MapReduce jobs are completed, a final map reduce is run by using the file token_simi.py as the mapper and an identity reducer. The final map reduce calculates similarity ratios for the number of matching tokens between:

- query to title
- query to description
- query to key
- query to title and description
- query to title and key
- query to description and key

- query to title, key, description

The output of token_simi.py is then fed into the Naive-Bayes Model files. An example of this output can be seen in the Appendix (Figure 2).

2.3 Training of Naive-Bayes Model

After we have the outputs from our previous Map-Reduce processes, we pass it in to "naive_train_model.py", which calculates conditional probabilities P(feature = value |click), P(feature = value |no click), P(click), P(no click), and also unknown values for each feature, P(feature = "UNK" | click), P(feature = "UNK" | no click). For continuous variables, the similarity ratios, it calculates the mean and variance first, which will then be used to calculate the conditional probabilities when predicting. These are all the important values we need for our Naive Bayes Model. This file is run locally and The outputs are put into a text file called naive_probabilities.txt.

After building the dictionary of conditional probabilities, we go back to using MapReduce for prediction. The following files are run to clean up validation data into the format that we want to finally do our predictions. They are run with validation-20, titleid_tokensid.txt, queryid_tokensid.txt, descriptionsid_tokensid.txt, and purchasedkeywordid_tokensid.txt (appended as in Prediction By Similarity Index), and userid_profile.txt:

1. naive_mapper_tok_1.py, naive_reducer_tok_1.py
2. naive_mapper_tok_2.py, naive_reducer_tok_2.py
3. naive_mapper_tok_3.py, naive_reducer_tok_3.py
4. naive_mapper_tok_4.py, naive_reducer_tok_4.py
5. naive_mapper_agegender_5.py, naive_reducer_agegender_5.py
6. naive_token_simi.py, identity reducer

The first 5 mapreduce jobs above follow the file "naive_train_model.py" and are mainly used to clean up the validation data into the format we want for our final mapreduce job for prediction. The map-reduce process of naive_mapper_tok_1.py and naive_reducer_tok_1.py outputs data similar to that of "mapper_tok_1.py" and "reducer_tok_1.py" with the addition of userid, gender, and age features. These jobs are ran sequentially, each output feeding into the next job. An example of the naive_train_model.py" ouput can be seen in the Appendix (Figure 3)

We then run the output of the fifth map-reduce job and input it into the file naive_token_simi.py which is run in conjunction with the identity reducer. This map-reduce process calculates similarity ratios for the number of matching tokens, similar to the process of token_simi.py

This output is then run through the final MapReduce naive_pred_mapper.py, and an identity reducer. This file is run with naive_probabilities.txt as a cache file. For continous variable, the similarity ratio, we assume Normal distribution to predict P(simi = x |click or no click), using the trained mean and variance. The output conditional probabilities are used to calculate P(click |data), such

as P(click |age = 1). For continous variable, the similarity ratio, we assume Normal distribution to predict P(simi = x |click or no click).

To be able to calculate the AUC, we need the probability of click and the actual click. So for the number of impressions for each instance, such as when one given input line is has clicks = 1, impressions = 3, and we predict no click, we expand and output:

P(click | feature), 1
P(click | feature), 0
P(click | feature), 0

2.4 AUC of Validation and Test Sets

In order to calculate the validity of the Naive-Bayes Model, we calculate the AUC (Area under the curve) of the model by creating an ROC graph using a formula of True Positive Rates and False Positive Rates. We did this through R to calculate the AUC. We calculated 3 different AUCs for 3 different sets of features. When we used all of our features, age, gender, token similarity with all possible combination of token comparisons(qtkd(query to title, key, description), qt, qd, qk, qtd, qtk, qdk), we got an AUC of .467. With just age and gender features, we got an AUC of .462. With age, gender, and qtkd similarity, we got an AUC of .466. With just all the similarity features, we got an AUC of .5636. It seems like age and gender aren't good features at all.

The data and the AUC are listed in a table contained in the Appendix (Table 1).

3 Responsibilities of each group member

We met a number of times to collaborate on the project. Initially we had a meeting to go over our project proposal, again to divy up coding responsibilities, and twice more to determinine and implement the AUC and Naive Bayes models, debug, and write up our progress report.

3.1 Ivan Kho

Ivan came to all of our group meetings to help make decisions, work on the code, debug problems that occured, and contributed to this progress report. He wrote part of the codes for the MapReduce jobs and contributed to checking for errors and debugging. In addition, he also wrote the README.md markdown in our repository, explaining the steps to run our model.

3.2 Alejandro Zuniga

Alejandro came to all of our group meetings to help make decisions, work on the code, and debug etc. He wrote the majority of the project summary paper/progress report. He also created a large part of the powerpoint presentation and wrote the majority of the final report.

3.3 Kinmen Yang

Kinmen came to all of our group meetings to help make decisions, work on the code, and debug etc. He wrote the code to calculate the naive bayes probabilities, titled naive_train_model.py. He also wrote all necessary code for our naive mappers and reducers, which handles validation data and prediction. Additionally, he contributed to the progress and final report.

3.4 Arshia Singh

Arshia came to all of our group meetings to help make decisions, work on the code, and debug etc. She developed the age and gender map-reduces and helped write the token similarity map-reduces. She ran some initial code locally to check for errors and debug. She ran all the MapReduce jobs in AWS. She also helped write the progress and final report.

4 Appendix

```
1 -1  age 1533466.000000  59793742.000000
2 1   age 275742.000000   6591266.000000
3 2   age 562991.000000   13077447.000000
4 3   age 1053060.000000  25791578.000000
5 4   age 721183.000000   18092403.000000
6 5   age 543173.000000   12164787.000000
7 6   age 262426.000000   5297832.000000
```

Figure 1: This is the output for Age, formatted as 'Age_Value \t 'age' \t Clicks \t Impressions'.

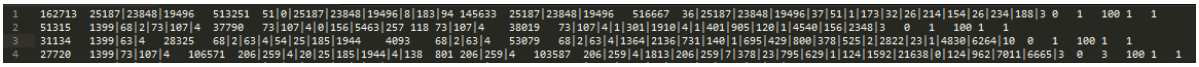

Figure 2: These are a few lines of output, in the format: 'QueryID \t Query_Tokens \t TitleID \t Title_Tokens \t KeyID \t Key_Tokens \t DescriptionID \t Description_Tokens \t Clicks \t Impressions'.

```
qkd_simi    UNK 0   0
qkd_simi    noclick 0.428977967314  0.0611020039713
qkd_simi    click   0.510425441862  0.0683040771182
gender  1   0.180924145224  0.152333840961
gender  0   0.0046557063315 0.0038269283829
gender  UNK 9.7901510493e-08    0.0
gender  2   0.149104098382  0.11726484591
qk_simi UNK 0   0
qk_simi noclick 0.406535127954  0.0612181108333
qk_simi click   0.483721522321  0.0684323738639
qd_simi UNK 0   0
qd_simi noclick 0.287611562264  0.0682657751873
qd_simi click   0.358916497304  0.0832978439451
qtkd_simi   UNK 0   0
qtkd_simi   noclick 0.438785351062  0.0605892449971
qtkd_simi   click   0.520775967806  0.0678484289723
qtk_simi    UNK 0   0
qtk_simi    noclick 0.425270936139  0.0605871769083
qtk_simi    click   0.506969188302  0.0683088607753
qtd_simi    UNK 0   0
qtd_simi    noclick 0.398783234934  0.0647530372482
qtd_simi    click   0.495103660759  0.0715845504159
qt_simi UNK 0   0
qt_simi noclick 0.377727035755  0.0649463080127
qt_simi click   0.474234025997  0.072564448617
age 1   0.0269956482791 0.0222538507193
age 3   0.103096232261  0.0871704741637
age 2   0.0551177510072 0.0440968657255
age 5   0.0531775394421 0.0409507816575
age 4   0.0706049822064 0.0612105183569
age 6   0.0256919921483 0.0177431323111
age UNK 9.79014817389e-08   0.0
Total   Total   0.0347415949022 0.965258405098
```

Figure 3: This is the output from "naive_train_model.py".

Features	Attempt 1	Attempt 2	Attempt 3	Attempt 4c
Age+Gender	X	X	X	
Query:Key	X			X
Query:Title	X			X
Query:Desc	X			X
Query:Title+Key	X			X
Query:Title+Desc	X			X
Query:Key+Desc	X			X
Query:Title+Key+Desc	X		X	X
AUC Score	0.467	0.462	0.466	0.5636

Table 1: AUC Results for Different Feature Combinations.

5 Site Bibliography

KDD Cup 2012, Track 2." KDD Cup 2012. N.p., n.d. Web. 05 Dec. 2014. <https://www.kddcup2012.org/c/kddcup2012-track2>