CSI4107

INFORMATION RETRIEVAL

AND THE INTERNET

ASSIGNMENT 1

REPORT

Group member:

Chen Chen (Student No. 5774520)

Zhiheng Yi (Student No. 7475788)

INDEX

TASK DIVISION

PROGRAM DESIGN

Zhiheng Yi with Chen Chen

PROGRAMING

Zhiheng Yi

README FILE DESIGN

Chen Chen and Zhiheng Yi

README FILE

Chen Chen

EVALUATION

Zhiheng Yi

FUNCTIONALITY DESCRIPTION

According to the instructions in the assignment sheet, our program can achieve the following functions.

First of all, the program reads the file which records the tweet documents, from the file divides tweet texts and tweet ID's, and stores them separately. The tweet texts are tokenized and removed stop-words from.

Secondly, the program extracts the query titles from the Query document, and stores them in a set of lists as the vocabulary.

Thirdly, each query will be compared with all the documents by repetition to calculate and collect the weight of each document and each query. And then the program calculates the value of cosine with the document's and the query's weight as the relevance.

Fourthly, the program ranks the relevance decreasingly in the required format.

Finally, the results will be evaluated, comparing them with the relevance feedback file using trec_eval script.

OPERATION INSTRUCTION

Since the program is developed in Java, it requires the user's device to have a Java platform such as JDK or JRE. Also, having eclipse installed will make running the program much easier. Otherwise, the CMD.exe will suffice. The running instruction will be introduced separately for both eclipse and DOS users as follows.

The program is in a project folder named "A1Program" in which the contents are listed as:

A sub-folder: .settings

A .chasspath file

A .project file

Two text file topics_MB1-49.txt and Trec_micoblog.txt: used to load the query and document into the program to proceed

A .class file: Query.class

A .java file: Query.java

For the users with eclipse, import the project folder "A1Program" into workspace, in "Package Explorer" find "default package" under " A1Program", and open Query.java file under "default package" with double click. Afterwards, simply click "Run As" button in "Quick Access" bar. The results will be displayed in a text file named as "results.txt" in the project folder.

For the users using CMD.exe to run the program, go to the directory where the project folder "A1Program" is stored, then type "java Query" and hit "Enter". The results will be displayed in a text file named as "results.txt" in the project folder.

PROGRAM DETAIL

ALGORITHMS

STRUCTURE FLOW CHART

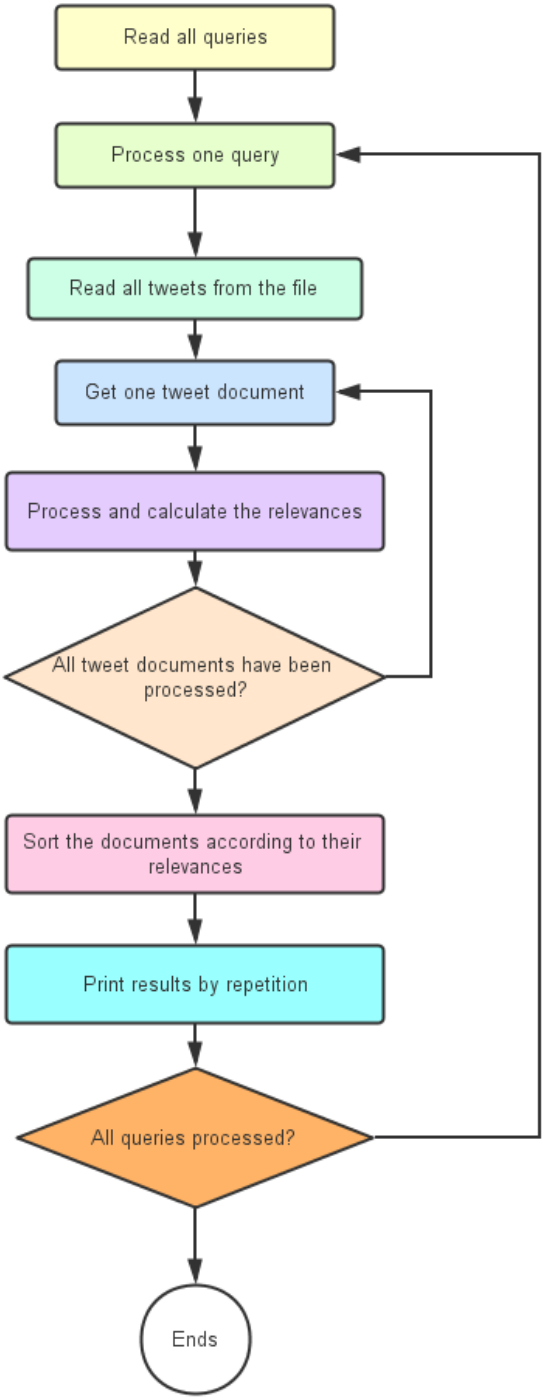

Figure 1 Structure Flow Chart

CHART EXPLANATION

1. Read all queries

Read queries into memory through BufferReader.

2. Process one query

The first step of this procession is to recognize the "<title>" line. Then extract the keywords between "<title>" and "</title>" with interface "split". Afterwards, the keywords are stored in the arraylist "query".

3. Read all tweets from the file

Read queries into memory through BufferReader.

4. Get one tweet document

In a "while" loop, get a tweet each time and use interface "split" to separate tweet text from tweet ID. Then with the help of regular expression and the stop-words list given in the assignment sheet, tokenize the tweet text with "pattern.matcher()" and "list.remove()". Afterwards, use an arraylist to store the tweet text and a string class the tweet ID.

5. Process and calculate the relevance

The weight of queries and tweet texts represents the vectors in a multi-dimension coordinate system to calculate cosine as relevance. In a "for" loop, we collect the weight of a query in an integer array. In another "for" loop, the weight of a tweet text, the same size and type as the one of a query, is done by comparing it with each query. The values in each weight array will be set to "1" if the corresponding words in query are found in the tweet text. After collecting the weights on both sides, we calculate the cosine values by using the following equation.

$$\mathrm{CosSim}(\vec{d}_j, \vec{q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} (w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

In the program, int[] wQuery and wDoc are wij and wiq, whose values change in each calculation.

6. Sort the documents according to the relevance

To sort the documents according to the relevance in a decreasing order, we first store the cosine values gotten from the previous step in a two-dimension array's

second row. The first row is for the corresponding tweet ID. Then we use the exchange sort method to sort the tweets.

7. Print results by repetition

Print the results, in a specified format, in a text file.

DATA STRUCTURE

1. Query

The keywords in queries are stored in an arraylist named "query".

2. Tweet

After being tokenized, the words for calculation in tweet documents are stored in a set of arraylists named "list", each of which stores the words in each tweet document. And the tweet ID's are stored inside of a string class named "tweetId". Both "list" and "tweetId" will change after each calculation of weight and cosine.

3. Weight

There are two kinds of weight, one for each queries and one for tweet documents. Both are stored in the form of integer array, named as "wQuery" and "wDoc". The size of the arrays is the same, the number of the keywords in each query, which will change after each time of cosine calculation.

4. Cosine

We used a two-dimension array, "docCos", to contain the cosine values. The first dimension is the tweet ID, and the second cosine value.

OPTIMIZATIONS

The first optimization we came up with was reducing the size of the vocabulary. Therefore, instead of using the tokenized tweet words as the vocabulary, we used the keywords in the query. As a result, the size of the vocabulary was immensely reduced while the functionality of the program was still maintained.

Secondly, we tried to decrease the times of loops in the program, hence workload of coding could be cut down. In the end, we only used a two-layer circulation. The outer cycle is to extract the keywords in the queries, and the inner one consists of three loops: the first one to tokenize tweets and to calculate weights and the cosine values, the second to sort the cosine values, and the third to print the results.

VOCABULARY

SIZE OF VOCABULARY

148 words

SAMPLES: (THE FIRST 100 WORDS)

2012, 2022, airport, amtrak, and, aristide, arrest, assange, athletes, attack,

attacks, auto, bbc, beck, bell, birth, bombing, bottega, bowl, breakup,

british, budget, campaigns, carbon, certificate, cesar, computer, consumption, court, crash, cuomo,

curfew, cuts, date, detroit, diplomat, dog, drug, egyptian, egyptian,

emanuel, energy, envoy, evacuation, farming, fax, fifa, filling, giffords, global,

hacking, haiti, half, healthcare, holland, house, in, iran, job, jobs,

jordan, kate, keith, known, kubica, kucinich, law, awsuit, maddow, media,

mexico, millan, monoxide, moscow, msnbc, murder, museum, new, nist, nobel,

nomination, nsa, obama, of, olbermann, olive, olympics, oprah, organic, pakistan,

peace, phone, pit, piven, political, politicians, protesters, protests, rachel, recall

RESULT SAMPLES

1 Q0 30260724248870912 1 0.95 myRun
1 Q0 33823403328671744 2 0.89 myRun
1 Q0 32504175552102400 3 0.89 myRun
1 Q0 32415024995631104 4 0.89 myRun
1 Q0 32158658863304704 5 0.89 myRun
1 Q0 30554037510213632 6 0.89 myRun
1 Q0 30371820393734144 7 0.89 myRun
1 Q0 30303184207478784 8 0.89 myRun
1 Q0 30299217419304960 9 0.89 myRun
1 Q0 34952194402811904 10 0.89 myRun
2 Q0 35048150574039040 1 1 myRun
2 Q0 35042178199851008 2 0.82 myRun

2 Q0 34993687440130048 3 0.82 myRun
2 Q0 34782178369863680 4 0.82 myRun
2 Q0 34757123825090560 5 0.82 myRun
2 Q0 34738795341414400 6 0.82 myRun
2 Q0 34634360019750912 7 0.82 myRun
2 Q0 34618905859203072 8 0.82 myRun
2 Q0 34606391796695040 9 0.82 myRun
2 Q0 34530597766434816 10 0.82 myRun
3 Q0 34694060157435904 1 1 myRun
3 Q0 34692276609351680 2 1 myRun
3 Q0 34689356128059392 3 1 myRun
3 Q0 33711164877701120 4 1 myRun
3 Q0 32469924240695296 5 1 myRun
3 Q0 32443364628500480 6 1 myRun
3 Q0 32204788955357184 7 1 myRun
3 Q0 32203898773053440 8 1 myRun
3 Q0 29615296666931200 9 1 myRun
3 Q0 29278582916251648 10 1 myRun
4 Q0 32851298193768448 1 1 myRun
4 Q0 30470121625485312 2 1 myRun
4 Q0 30306064587030528 3 1 myRun
4 Q0 30027043655655424 4 1 myRun
4 Q0 29903758779490304 5 1 myRun
4 Q0 29887625179435008 6 1 myRun
4 Q0 29878816281206784 7 1 myRun
4 Q0 29684273590042624 8 1 myRun
4 Q0 29400624374222848 9 1 myRun
4 Q0 29756448648994816 10 0.94 myRun
5 Q0 30566466117967872 1 0.82 myRun
5 Q0 33288536677421056 2 0.82 myRun
5 Q0 33217960763985920 3 0.82 myRun
5 Q0 33188770152976384 4 0.82 myRun
5 Q0 33158966397763584 5 0.82 myRun
5 Q0 32995392408780800 6 0.82 myRun
5 Q0 32874671481290752 7 0.82 myRun
5 Q0 32874664267087872 8 0.82 myRun
5 Q0 32874659636584448 9 0.82 myRun

5 Q0 32874651365412864 10 0.82 myRun
6 Q0 35005178885181440 1 1 myRun
6 Q0 34952124211134464 2 1 myRun
6 Q0 34899100029689856 3 1 myRun
6 Q0 34728883752280064 4 1 myRun
6 Q0 34713303489978368 5 1 myRun
6 Q0 34668062389051392 6 1 myRun
6 Q0 34584602995589120 7 1 myRun
6 Q0 34484309310193664 8 1 myRun
6 Q0 34431046347005952 9 1 myRun
6 Q0 34255567715307520 10 1 myRun
7 Q0 34857364532236288 1 0.87 myRun
7 Q0 30790044126027776 2 0.87 myRun
7 Q0 30723400687165440 3 0.87 myRun
7 Q0 30809856231342080 4 0.87 myRun
7 Q0 34872339610996736 5 0.71 myRun
7 Q0 35061722091880448 6 0.71 myRun
7 Q0 34759762457395200 7 0.71 myRun
7 Q0 34706220640112640 8 0.71 myRun
7 Q0 34546625972158464 9 0.71 myRun
7 Q0 33712124932923392 10 0.71 myRun
8 Q0 34467845525995520 1 0.87 myRun
8 Q0 29281084667596800 2 0.82 myRun
8 Q0 34313399491891200 3 0.71 myRun
8 Q0 34219593782267904 4 0.71 myRun
8 Q0 34219555756711936 5 0.71 myRun
8 Q0 34057400382132224 6 0.71 myRun
8 Q0 34029986566373376 7 0.71 myRun
8 Q0 33944012440207360 8 0.71 myRun
8 Q0 33923417598066688 9 0.71 myRun
8 Q0 33531241156317184 10 0.71 myRun
9 Q0 35067946019590144 1 1 myRun
9 Q0 35023707030167552 2 1 myRun
9 Q0 34887054940704768 3 1 myRun
9 Q0 31984587735306240 4 1 myRun
9 Q0 31059921093009408 5 1 myRun
9 Q0 30629752956002304 6 1 myRun

9 Q0 30574265715658752 7 1 myRun
9 Q0 30571851667210240 8 1 myRun
9 Q0 30567532691726336 9 1 myRun
9 Q0 30513117570015232 10 1 myRun
10 Q0 31077260689674240 1 0.87 myRun
10 Q0 31075323059638272 2 0.87 myRun
10 Q0 32275485807353856 3 0.87 myRun
10 Q0 31855298419359744 4 0.87 myRun
10 Q0 31609613321244672 5 0.87 myRun
10 Q0 31245512300568576 6 0.87 myRun
10 Q0 31426798810046464 7 0.71 myRun
10 Q0 31426539723689984 8 0.71 myRun
10 Q0 31426139767443456 9 0.71 myRun
10 Q0 31425731942686720 10 0.71 myRun
11 Q0 34199299428581376 1 1 myRun
11 Q0 34530871604289536 2 0.95 myRun
11 Q0 31907420687044608 3 0.71 myRun
11 Q0 33578604684120064 4 0.71 myRun
11 Q0 32838627180421120 5 0.71 myRun
11 Q0 31894628701573120 6 0.71 myRun
11 Q0 33662164124307456 7 0.71 myRun
11 Q0 29246016238657536 8 0.71 myRun
11 Q0 34264130017824768 9 0.71 myRun
11 Q0 34197137193443328 10 0.71 myRun
12 Q0 32117461922877440 1 0.87 myRun
12 Q0 31940748085563392 2 0.87 myRun
12 Q0 33186842786398208 3 0.71 myRun
12 Q0 33085151869145088 4 0.71 myRun
12 Q0 32947773988929536 5 0.71 myRun
12 Q0 32181966761631744 6 0.71 myRun
12 Q0 32124496466935808 7 0.71 myRun
12 Q0 33241016773382144 8 0.71 myRun
12 Q0 32112594718294016 9 0.71 myRun
12 Q0 33213569818558464 10 0.71 myRun
13 Q0 29561670661570560 1 1 myRun
13 Q0 29564454236594176 2 0.87 myRun
13 Q0 29560222905278464 3 0.87 myRun

13 Q0 29558797357809664 4 0.87 myRun
13 Q0 29579396469760000 5 0.87 myRun
13 Q0 30345378041692160 6 0.82 myRun
13 Q0 29540081047961600 7 0.71 myRun
13 Q0 29560614837813248 8 0.71 myRun
13 Q0 29563563311894528 9 0.71 myRun
13 Q0 29559171300982784 10 0.71 myRun
14 Q0 31885796969545728 1 0.87 myRun
14 Q0 31085435891490816 2 0.87 myRun
14 Q0 31059918010187776 3 0.87 myRun
14 Q0 30610467424571392 4 0.87 myRun
14 Q0 30296957977100288 5 0.87 myRun
14 Q0 29980889064669184 6 0.87 myRun
14 Q0 30107939607937024 7 0.82 myRun
14 Q0 30418286688608256 8 0.82 myRun
14 Q0 29025548743221248 9 0.82 myRun
14 Q0 32478310793482240 10 0.71 myRun
15 Q0 33276913967435776 1 0.77 myRun
15 Q0 33087998794932224 2 0.77 myRun
15 Q0 32760806558928896 3 0.77 myRun
15 Q0 32703076989140992 4 0.77 myRun
15 Q0 32678457083170816 5 0.77 myRun
15 Q0 32674038430040064 6 0.77 myRun
15 Q0 32665910925852672 7 0.77 myRun
15 Q0 33231324298874880 8 0.73 myRun
15 Q0 32641649188274176 9 0.67 myRun
15 Q0 34077200302997504 10 0.63 myRun
16 Q0 29528543469764608 1 0.77 myRun
16 Q0 30756647127228416 2 0.63 myRun
16 Q0 34997633260978176 3 0.63 myRun
16 Q0 34995783627575296 4 0.63 myRun
16 Q0 34399319335378944 5 0.63 myRun
16 Q0 34330268860940288 6 0.63 myRun
16 Q0 30160666098667520 7 0.63 myRun
16 Q0 32612223830458368 8 0.63 myRun
16 Q0 32603111843438592 9 0.63 myRun
16 Q0 32492294682705920 10 0.63 myRun

17 Q0 32876528131899392 1 1 myRun
17 Q0 32871838174416896 2 1 myRun
17 Q0 32867803606290432 3 1 myRun
17 Q0 32878820491001856 4 0.82 myRun
17 Q0 32877729216987140 5 0.82 myRun
17 Q0 32877652738048000 6 0.82 myRun
17 Q0 32877609662554112 7 0.82 myRun
17 Q0 32877575688691712 8 0.82 myRun
17 Q0 32877382029287424 9 0.82 myRun
17 Q0 32877247207718912 10 0.82 myRun
18 Q0 30273593405345792 1 0.85 myRun
18 Q0 31396010894819328 2 0.76 myRun
18 Q0 31014649839226880 3 0.76 myRun
18 Q0 29471305610825728 4 0.65 myRun
18 Q0 29372322523648000 5 0.65 myRun
18 Q0 34186767892484096 6 0.65 myRun
18 Q0 30188073790742528 7 0.65 myRun
18 Q0 30085495253897216 8 0.65 myRun
18 Q0 32003494944706560 9 0.65 myRun
18 Q0 29963346505633792 10 0.62 myRun
19 Q0 34046080261824512 1 1 myRun
19 Q0 32864634478272512 2 1 myRun
19 Q0 32579076275306496 3 1 myRun
19 Q0 32403820008968192 4 1 myRun
19 Q0 32342810979999744 5 1 myRun
19 Q0 32324277558575104 6 1 myRun
19 Q0 32550210244706304 7 0.94 myRun
19 Q0 32550327760723968 8 0.94 myRun
19 Q0 34688504977948672 9 0.82 myRun
19 Q0 34686577133232128 10 0.82 myRun
20 Q0 31082136219947008 1 1 myRun
20 Q0 29906116062220288 2 0.98 myRun
20 Q0 29853985930219520 3 0.94 myRun
20 Q0 32218912527482880 4 0.87 myRun
20 Q0 31948737517453312 5 0.87 myRun
20 Q0 31424967899873280 6 0.87 myRun
20 Q0 31161931205181440 7 0.87 myRun

20 Q0 31129345066012672 8 0.87 myRun
20 Q0 31121369240444928 9 0.87 myRun
20 Q0 31121223412883456 10 0.87 myRun
21 Q0 29985282845581312 1 0.87 myRun
21 Q0 29963031203020800 2 0.87 myRun
21 Q0 29612419269525504 3 0.87 myRun
21 Q0 29610928903299072 4 0.87 myRun
21 Q0 29606637102702592 5 0.87 myRun
21 Q0 30993728348880896 6 0.71 myRun
21 Q0 30927494672551936 7 0.71 myRun
21 Q0 30882540759810048 8 0.71 myRun
21 Q0 30858521247485952 9 0.71 myRun
21 Q0 30850469203025920 10 0.71 myRun
22 Q0 32275892562567168 1 1 myRun
22 Q0 32252894585552896 2 1 myRun
22 Q0 32228931012657152 3 1 myRun
22 Q0 32219487591735296 4 1 myRun
22 Q0 32219349217443840 5 1 myRun
22 Q0 32218747884277760 6 1 myRun
22 Q0 32201676312018944 7 1 myRun
22 Q0 32175018792325120 8 1 myRun
22 Q0 32173003508813824 9 1 myRun
22 Q0 32171528254660608 10 1 myRun
23 Q0 35047250950361088 1 0.82 myRun
23 Q0 34846878495412224 2 0.82 myRun
23 Q0 34768755347169280 3 0.82 myRun
23 Q0 34763798942322688 4 0.82 myRun
23 Q0 34755971859357696 5 0.82 myRun
23 Q0 34744935773118464 6 0.82 myRun
23 Q0 34729493578907648 7 0.82 myRun
23 Q0 34080830133379072 8 0.82 myRun
23 Q0 33710662530113536 9 0.82 myRun
23 Q0 33685170267627520 10 0.82 myRun
24 Q0 35022813232373760 1 1 myRun
24 Q0 34978936819548160 2 1 myRun
24 Q0 34784437526855680 3 1 myRun
24 Q0 34734420598464512 4 1 myRun

24 Q0 34669638520406016 5 1 myRun

24 Q0 34665683837001728 6 1 myRun

24 Q0 34646404349562880 7 1 myRun

24 Q0 34640769553928192 8 1 myRun

24 Q0 34543726252527616 9 1 myRun

24 Q0 34518627260567552 10 1 myRun

25 Q0 32609015158542336 1 1 myRun

25 Q0 31738694356434944 2 1 myRun

25 Q0 31550836899323904 3 1 myRun

25 Q0 31286354960715776 4 1 myRun

25 Q0 33633450548264960 5 0.82 myRun

25 Q0 32560154188713984 6 0.82 myRun

25 Q0 32551173739249664 7 0.82 myRun

25 Q0 32528974961713152 8 0.82 myRun

25 Q0 32108003058524160 9 0.82 myRun

25 Q0 31823291815567360 10 0.82 myRun

FINAL RESULTS DISCUSSION

Our program successfully achieves all the objectives as we were required. In order to optimize the functionality and tidy the code, we used a much simpler vocabulary and the structure of multi-layer circulation.

After detailed discussion we agreed that, overall, the algorithms should be the first and the most significant to improve. Although we did some work to make it effective in terms of code load, the program still needs about 30 minutes to yield the retrieval results, which should be much less if we had paid more attention to reduce the run time with a better plan.

Particularly, there are some aspects for further optimization. At first, in the preprocessing and indexing part, there might be better data structures which can not only satisfy the data requirements, but be indexed in a very short time as well. Also, in order to simplify the data structure and the calculation, we used the term frequency weighting method. The term frequency only puts the frequency of queries' keywords in consideration, but doesn't analyze the keywords' proportion of the entire tweet document. As a result, there were many documents with the same relevance value for a single query. On the other hand, in some complex scenarios, the TF-IDF weighting scheme should work better than frequency weighting scheme.

In addition, we might be able to rank these results in a faster way in the retrieval and ranking part. Moreover, we still need further research to understand the differences of using various evaluation methods and the meaning of trec_eval script's output (see Figure 2 in Appendix) better. Consequentially, there should be a way to improve the program according to the evaluation parameters.

Incidentally, the program doesn't support multi-languages retrieval.

APPENDIX

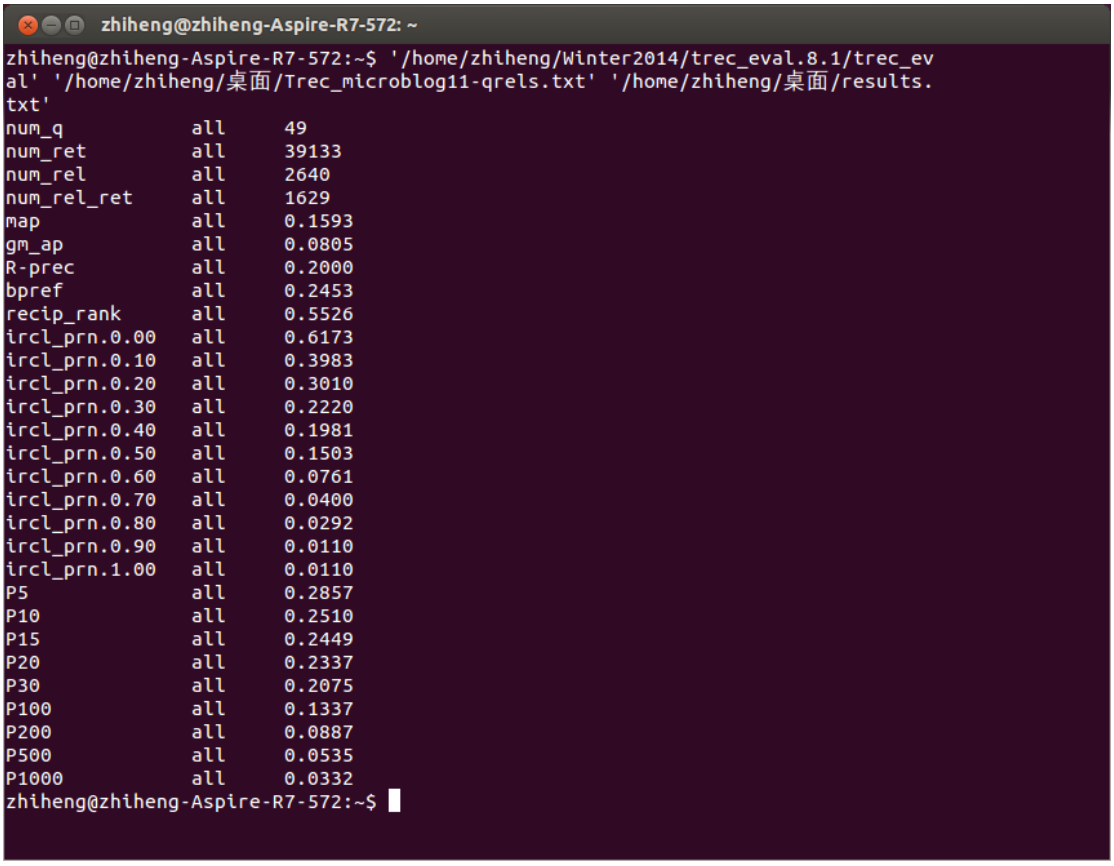

Figure 2 Evaluation Output