TEAM 34: Sungjae Lee sungjael@andrew.cmu.edu
Chen Wang chenw@andrew.cmu.edu

Part I: Extend NgramCount

Goal:

- Modify the source files inside mapred.nagramcount package so that it takes “n” as a command line option, and outputs counts of corresponding n-grams.
- Can modify any code except for Entry.java.

Method:

- Original UnigramCount Code (v0.1)

v0.1 Mapper: The original code of NgramCount only count the occurance of one word, so all tweets are tokenized in Mapper and each word is regarded as a key. Every time there is a new word, a NullWritable object is added as a value on the word as a key.
v0.1 Reducer: The number of NullWritable objects are counted and reduced to the occurrence of that word.

- Extended NgramCount Code (v1.1)

v1.1 Driver: add a parser for input arguments “-n” with input value “N”; and pass the value of “N” through the Configuration object.
v1.1 Mapper: use input parameter n as the number of grams to extract the n contiguous words from tweets; N contiguous words are regarded as one key and are counted once.
v1.1 Reducer: count the occurrence of N contiguous words (a key) according to the total value associate with the key.

Results

- **ID:**j-2RGN4DFHZ2K2T
    - N = 3
        - Sample Result: "dark center of 1
        - Time for 5 instances on EMR: 276958 ms
        - Results Bucket on EMR: **chenw-output/3gram10m/**
    - N = 4
        - Sample Result:
        - Time for 5 instances on EMR: 283545 ms
        - Results Bucket on EMR: **chenw-output/4gram10m/**
    - N = 5
        - Sample Result: "body of work i will 1
        - Time for 5 instances on EMR: 302660 ms
        - Results Bucket on EMR: **chenw-output/5gram10m/**

TEAM 34: Sungjae Lee sungjael@andrew.cmu.edu
Chen Wang chenw@andrew.cmu.edu

Part 2: Speed up HashtagSim

Inverted Index

How the speedup works :

Differ from original version of mapping hashtags to words, we count all words and reduce each word's count into pairs of hash tags.

We used the same function to get all the feature vectors for each tag, then with the feature_vector we have we created reverse indexed list as below.

This is how it looks like after we created reverse-indexed.

```
a       #job:1;#sna:2;#nodexl:2;#socia:1;
do      #nodexl:1;#sna:2;#socia:!;
..
```

Then we map/reduced the unique pairs as

```
#1      #1           5
#excel #nodexl       7
..
```

What is the expected speed up :

The original version of HashtagSim reads every hashtag and its word feature into the buffer and create a job to compare the loaded hashtag feature with all other hashtag features. Then, if there are N hashtags, the program will compute Loading... times of hashtag feature similarity between two hashtags. The time complexity is Loading....

The speedup version extracts all the words from tweets data and generates inverted index for each word with hashtags that have used these words. Therefore, if two hashtags do not share same words, they will not be compared. The time complexity of the speedup version is linearly increasing with the total number of words, namely Loading..., where M is the total number of words. For each word, the time is taken by counting hashtag pairs and it will vary among words.

As the tweets database increase, the total words in the database will be slowly increasing but the total number of hashtags will increase a lot. Therefore, speedup algorithm will work especially well for large database.

However, we run 10K database on local machine to get the number of hashtags and the number of words. We also approximated the speedup by running the 10k tweets database on a single machine.

Total Number of Hashtags in 10k tweets: Loading...
Total Number of Words in 10k tweets: Loading...
Run time of brute force code on a single micro instance: 1009996 ms
Run time of optimized code on a single micro instance: 67558 ms
Expected Speedup: 15X , which can be approximated as Loading...

What is the observed speedup :

Original verison of 10 million tweets on EMR with 5 instances: > 2 hours (The results got in 2 hours is less than ¼ of the total results. We terminate the run because we do not want to waste too many credits on the brute force code.)
Optimized version of 10 million tweets on EMR with 5 instances: 73 min
Speedup: more than 6X