CS5344 Big Data Analytics Technology

Lab 2 (AY2019/2020 Semester 2)

I. Task A

1. Requirement

Write a Spark program that finds that top-N relevant documents given a query comprising of set of keywords

2. Environment

Ubuntu-16.04 pre-configuration which is downloaded from https://nusu-my.sharepoint.com/:u:/g/personal/e0267909_u_nus_edu/EZa_73QIfdlCrivb1NyqiBQBcikJWnlz71giJeIESDZLXQ

Python version 2.7.12

Python libraries: math, os, re, sys

Spark version 2.2.1

3. Command

```
spark-submit Task_A.py in/query.txt in/stopwords.txt in/lab2input out/Task_A
```

4. Algorithm

The Algorithm finds top-N relevant documents given a query comprising of set of keywords:

4.1 Load files

SparkContext.TextFile loads query.txt , split into words and create list of query words

SparkContext.TextFile load stopwords.txt and create a list of stop words

SparkContext.wholeTextFiles loads all documents from input folder and create a RDD with element is a key-value pair, whereby key is file name and value is file content

number_of_documents = RDD.count

4.2 Find number of important words in a document

Apply sequential transform functions for the RDD of documents in step 4.1:

- Flatmap splits all file content into words, and generate into a pair with format RDD ((word, doc), 1)
- Filter removes item which has the word is in listed in stop word list
- Map removes some special symbols such as : !,',-, [,] …
- Filter removes item which has the word is empty string or listed in stop word list.
- ReduceByKey computes number of words in a document, then RDD ((word, doc), 1) is transformed to RDD ((word, doc), numofwords).

4.3 Find number of documents contain a word

Apply sequential transform functions:

- Map transform RDD((word,doc), numofwords)) into RDD(word, doc) (1)
- GroupByKey creates document list for a word RDD (word, [doc1,doc2...])
- Map counts number of documents for a word and transform into RDD (word, numofdocs) (2)

4.4 Calculate TF-IDF of word in document

Apply sequential transform functions:

- From (1) and (2), join RDD (word, doc) and RDD (word, numofdocs) to create RDD (word, (doc, numofdocs))
- Map transform RDD (word, (doc, numofdocs)) to RDD ((word, doc), numofdocs)
- Join RDD ((word, doc), numofwords) and RDD ((word, doc), numofdocs) to create RDD ((word,doc), (numofwords, numofdocs))
- Map calculate TF-IDF for words of documents in RDD ((word, doc), (numofwords, numofdocs)) with the number_of_documents in step 4.1. Finally, we have RDD (doc, (word ,tf-idf))

4.5 Normalize TD-IDF of word in document

Apply sequential transform functions:

- Map transforms (doc, (word ,tf-idf)) to (doc, tf-idf * tf-idf)

- ReduceByKey sums all square of tf-idf for a document, then we get RDD (doc, sum_of_square_of_tf-idf)
- Map gets square root of sum_of_square_of_tf-idf which is also a length of doc vector , then RDD (doc, ||doc||) is created
- Join (doc, (word ,tf-idf)) and (doc, ||doc||) to create (doc, ((word ,tf-idf), ||doc||))
- Map normalizes TF-IDF and transform to (doc, (word, norm_tf-idf))

4.6 Find normalized query vector

From RDD (word, numofdocs) , applying filter to select words available in query string. So the length of query vector ||V|| is a square root of number of words which are available in query string

From RDD (word, numofdocs) again, applying map to normalize and transform RDD (word, numofdocs) into RDD (word , b/||V||) in which b is 1 if the word is available in query string, otherwise b is 0. Finally, we obtain a normalized query vector RDD (word , norm_query_item)

4.7 Calculate Cosine Similarity or relevant score

Apply sequential transform functions:

- Join RDD (word, (doc, norm_tf-idf)) and RDD (word , norm_query_item) to create RDD (word, ((doc, norm_tf-idf,), norm_query_item))
- Map multiplies norm_tf-idf and norm_query_item and transforms to RDD (doc , norm_tf-idf * norm_query_item)
- ReduceByKey sums up the multiplication results for each document and create RDD (doc, relevant_score). Because all doc vectors and query vector are normalized, the sum is also a result of Cosine Similarity or relevant score between a document and query

5. Output top 10 document relevant to query

Sort RDD (doc, relevant_score) by relevant_score and display top 10 documents have highest relevant score

II. Task B

1. Requirement

Write a Spark program that clusters the given set of documents into varying number of clusters using k-means.

2. Environment

Ubuntu-16.04 pre-configuration which is downloaded from https://nusu-my.sharepoint.com/:u:/g/personal/e0267909_u_nus_edu/EZa_73QIfdlCrivb1NyqiBQBcikJWnlz71giJeIESDZLXQ

Python version 2.7.12

Python libraries: math, os, re, sys

Spark version 2.2.1

3. Command

```
spark-submit Task_B.py in/stopwords.txt in/lab2input out/Task_B
```

4. Algorithm

4.1 Normalize TD-IDF for all documents that we do same step from 3.1 to 3.5 in Task A

4.2 Implement K-mean

Datapoint list is derived from getting RDD (doc, (word, norm_tf-idf)) which is grouped by doc to obtain array of normalized TD-IDF of words per document. To represent documents as a vector, structure of datapoint list is a dictionary in which key is document name, value is another dictionary with key is the word belong to document and value is normalized TD-IDF of the word

Centeroid list is initialized by assigning the k values from datapoint list. Whereby the k values are selected following interval index which is calculated by number of documents / k

Find closet centroid for the point by using two metrics Cosine Distance and Euclidean distance:

- Cosine Distance = 1 – Cosine Similarity
- Euclidean Distance = $\text{sqrt}((p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2)$

When the closet centroid is found, the point is assigned to a group whose index is as same as index of the closet centroid in centroid list

Each centroid in list is updated for every iteration by getting average value of all points in same cluster group

Kmean program will stop when there are no points moving among cluster groups

5. Output

The Kmean program results in files for both Euclidean Distance and Cosine Distance with k-mean number starts from 2 to 8