Project report - Phase 5

Shrinivas Kane

April 23, 2014

1 Introduction

This phase aims to understand clustering hierarchical agglomerative clustering using cosine similarity between two documents in corpus. We will use term weights generated from phase 2 as input

2 Requirements

2.1 Input

Term weighted matrix for corpus

2.2 Output

List of documents to be merged as a new cluster in each pass.

2.3 Operations

- Find cosine similarity between tow documents
- Build cosine similarity matrix for corpus
- Find most similar documents in corpus and merge them as cluster

3 Algorithm

We used java IO packages to read and write to files.

3.1 Find cosine similarity between tow documents

We have used cosine similarity measure to measure similarity between two documents. Cosine similarity between two documents is calculated as follows:

$$\text{Cosine similarity} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} * \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

3.2 Build cosine similarity matrix for corpus

We Use cosine similarity measure to calculate similarity matrix for corpus using following pseudo code

Data: Document vector
Result: Similarity matrix
initialization i=0,j=0,n= number of documents;
while *i less than n* **do**
  while *j less than n* **do**
    if *i not equal to j* **then**
      compute Cosine similarity.
      Update similarity matrix.
    end
  end
end

Algorithm 1: create similarity matrix

3.3 Find most similar documents in corpus and merge them as cluster

We have picked up two most similar document from corpus and merged them as one cluster. This newly generated cluster act as new document in corpus and individual merged documents are deleted from similarity matrix. Following pseudo code used to generate clusters.

Data: cosine similarity matrix
Result: Clustered corpus
initialization i=0,j=0,n= number of documents;
while *No Document remain for merge* **do**
  Find two most similar documents in similarity matrix
  Merge these documents to become one cluster
  Remove merged documents from matrix
  Add (newly generated cluster as new document in corpus)
  recompute similarity matrix
end

Algorithm 2: Cluster documents

4 Results

We used memory based approach to store matrix as a list. With overall complexity of O(n*n) to build matrix. List based matrix representation helped us to retrieve top similar documents in constant time which eventually speedups the process.
Our implementation showed following results:
There are multiple documents with same similarity score we used natural descending sorting on docId in case of tie break. This is why document 420,417 merged before 130,102.

Criterion	Doucment Pair	Score
Most similar	102,130 and 420,417	1
Most Disimilar	1,13	0.0
Centroid	182	0.2

Results showen in above table may vary with other implementation due to different tokenization and term weighting schema(s).

5 Challenges and future work

Recomputing similarity matrix for each pass is costly and in memory approach may not scale to large corpus size.Currently we merged one document per pass.Merging multiple documents in one pass can be future extension

6 Conclusion

We have successfully built similarity matrix and clustered the documents in corpus