Abhishek Sethi Homework 5 04/24/2014
CMSC676

Introduction

Information retrieval (IR) systems generally consist of retrieving relevant documents based on a user's keyword input. In order to retrieve these documents from the corpus, generally some sort of weighting scheme and indexing of the terms from the documents is required. This allows for efficient search of documents that are relevant to user inputs. The previous phases of the project were meant to implement a program that will take care of this for us. However, we take a step away from traditional search engine style IR, and take a look at document analysis instead. This phase of the project analyzes the similarities between documents, and then clusters them accordingly.

In order to perform the clustering of documents, some metric must first be established to establish the amount of similarity. The metric chosen in this case is the cosine similarity of the two documents. In order to calculate this measurement, the documents were described as vectors. The vector contents were the frequency of each term in that document, so they were either zero or some positive integer. The cosine metric is calculated using the following formula:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Equation 1

The numerator is the dot product of the two vectors. The denominator is the product of the magnitudes of the two vectors. The similarity is simply the dot product divided by the product of the magnitudes of the vectors [4]. These values were then used to construct a similarity matrix. Assuming n documents, a similarity matrix is an n x n matrix, where each entry is the cosine similarity between the respective documents. The key things to note is that in the similarity matrix (sim), the entries at position x, where sim[x][x] is simply going to be 1 since a document should be similar to itself. Also, the similarity is symmetric so sim[x][y] = sim[y][x]. Applying these properties means that only half of the matrix needs to be filled in (the upper or lower triangle, depending on the implementation).

Once a similarity matrix is constructed, the next step is to begin clustering the documents together. The idea here is to cluster similar documents together so that when searching through the documents, other documents the user could possibly be interested in are also presented without any extra crawling through the corpus. The method for clustering is hierarchical agglomerative clustering using the group average link method. Hierarchical agglomerative clustering is based on the idea of using a hierarchy (generally a tree) to define clusters of documents in a corpus working from the bottom up [1]. The algorithm is simple, yet effective in execution. Initially each document is labeled as its own cluster, so there would be n clusters for n documents. In order to start clustering, a pair of documents from the similarity matrix is selected that has the highest cosine similarity value of all the other entries in the matrix. Basically, this means picking out the maximum value from the matrix each time. The documents are then merged together to create a new cluster with 2 documents now. This process is repeated until 1 cluster remains containing all the documents. The purpose of this is so that a hierarchy can be established, so that if a cut is made at any level in the graph, clusters of similar documents are made based on how much similarity is desired (100%, 50%, etc). An example is shown in figure 1.

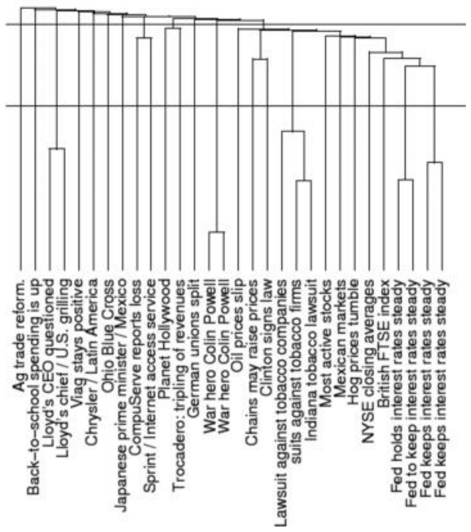

Figure 1: An example of hierarchical agglomerative clustering. As seen, cutting at any level will give a bunch of clusters, some more similar than others [2].

The group average link method comes into play during the merging of 2 clusters. There are multiple ways to choose the new similarity, such as single link and complete link, where you would use the maximum or minimum similarity value, respectively [3]. In group average link, the new similarity for the merged cluster is computed by simply taking the average of the two individual similarity values. This offers a compromise between the single and complete link methods, and is not harder to implement, just a simple computation. This application is a step away from traditional IR searching, but it is a useful application to relevant document retrieval.

Implementation

This project is built upon phase 2 of the programming project. This decision was made because the only data structure required from before is the frequency of the terms in each document. The program can be run by executing:

```
java –jar HW5.jar [input directory] [output directory]
```

Where the input directory contains the corpus for analysis and the output directory is where the cluster list and output strings are written to. The same process for file tokenization from the previous projects is done here to. The next step was to calculate the document magnitudes by simply doing the square root of the sum of the squares of the term frequency in each document. The original data structure of Map<String, Map<String, Integer>> is used, where the first String is the document, the inner map's key is the term, and the value is the frequency of that term in that specific document. The next step was to calculate the similarity matrix. The similarity matrix was established to be a 2-Dimensional array of doubles (double[][]). This allows for efficient access to the data values, but at the higher cost of initialization. Also, the issue here is that the rows/columns cannot be adjusted easily for merging, but that is adjusted for later on. The similarity matrix is created from the frequency map used from the previous phase. The cosine similarity is calculated as described in equation 1. Nothing special was done; just a naïve implementation. The interesting implementation was when performing document clustering.

The algorithm used to perform the clustering is described below:

```
Cluster
    Input: similarity_matrix (double[][]), int files
    Output: Strings containing the documents merged together (printed later)
    //directly manipulates the input similarity matrix
    //previous_max is an object with values, doc1, doc2, similarity value
    previous_max = -1 (no docs);
    While(true)
        //want to stop the clustering process when the maximum value from the matrix
        //is 0.4
        If(previous_max < 0.4 && previous_max > -0.01)
            break;

        //maxValue is an object with values, doc1, doc2, similarity value
        maxValue = find max value from similarity matrix (the documents are saved here to)

        //these values determines which doc is chronologically smaller (ex: document 100 < document 101)
        low = minimum (maxValue.doc1, maxValue.doc2)
        high = maximum (maxValue.doc1, maxValue.doc2)

        //these loops are used to calculate the values for the similarity values of the
        //newly merged clusters
        foreach row in similarity_matrix
            average =(similarity_matrix[row][low] + similarity_matrix[row][high]) / 2
            similarity_matrix[row][low] = average;
        end
        foreach column in similarity_matrix
            average =(similarity_matrix[low][column] + similarity_matrix[high][column]) / 2
            similarity_matrix[low][column] = average;
        end

        //these loops are used to initialize the high end to all 0's.
        Foreach row in similarity_matrix
            Similarity_matrix[row][high] = 0.0
        End
        Foreach column in similarity_matrix
            Similarity_matrix[high][column] = 0.0
        End

        Write output
```

```
                //edit the global clusters array list
                //simply performing a union operation between the two clusters (high and low)
                Clusters.get(low).addCluster(clusters.get(high))
                //an empty cluster is used to identify the higher of the two documents
                Clusters.set(high, Empty_cluster)
                Previous_max = maxValue
                maxValue = -1 (no docs)
        end
end
```

Figure 2: Algorithm used to perform clustering based on the results of the similarity matrix.

Figure 2 describes the algorithm that was used to perform clustering of the matrices. It follows the general idea put forth from hierarchical clustering, but the way merging of clusters is managed is slightly different. Rather than actually shorten the matrix widths and columns, I decided that in the matrix, the "smaller" document name would store the newly merged information, while the "larger" document name would store all 0's. By following this fixed idea, the new similarity values for clusters was calculated relatively easy as shown in the algorithm. A global array list was used to maintain the clusters as well. This simply contained Cluster objects, which is a wrapper containing a list of the documents in the cluster. The ordering between the matrix and array list were exactly the same, so index 0 referred to document 001.html in both the array list and 2-dimensional array for the similarity matrix. Assuming this was consistent, the way the clusters were maintained is shown towards the bottom of the loop in the algorithm. It was also assumed that the intersection between any two clusters would result in an empty set, and my program does enforce this by finding the maximum value each time, and ensuring similarities are maintained appropriately. The high cluster would simply be unioned with the low cluster, and then the high cluster would be set to a singleton class called Empty_cluster, which simply represents a cluster with no documents. This meant that the cluster can now be ignored in further analysis. The output was saved to a second global array list. Both the output and the clusters were written out to a file in the specified output directory. Overall, this program is inefficient, unfortunately. The findMax call inside the loop makes the program run at an O(n^3) time if all the documents to be merged are similar. The program works fine for small corpora of 100 documents, but takes a long time once the full 503 documents were analyzed (data not shown). Overall, the accuracy of the algorithm is more important than the runtime right now, and the algorithm does deliver in terms of accuracy.

Results

Based on the output of my program, the HTML documents that are most similar are documents 101.html and 129.html with a similarity value of 1.0 apparently. The least similar documents are 350.html and 399.html. The document that is closest to the centroid is document 167.html, based on the maximum value from the final merged similarity matrix. Some of these results seemed a bit strange when the documents were visually inspected, but then since the similarities are based on previous implementations of tokenization would explain the variations. The amount of stop words removed may have biased the results towards my implementation. For example, if two documents are similar, then that means the stop word removal may have left only the same tokens between the two. This would in turn affect the frequency counts, thus affecting the results of this project. The full output is shown below in the supplemental data section. Overall, this program does not run very efficiently, but it does get the job done. Small changes can be made, and better algorithmic efficiency can be achieved if the algorithms were re-analyzed.

References

1. “Hierarchical agglomerative clustering”. http://nlp.stanford.edu/IR-book/html/htmledition/hierarchical-agglomerative-clustering-1.html

2. Michael. “Hierarchical cluster analysis”. Stanford. http://www.econ.upf.edu/~michael/stanford/maeb7.pdf

3. Raghavan, P. Manning, C. Mooney, R. Chakrabarti, S. “Hierarchical Clustering”. Lecture.

4. “Tf-idf and Cosine similarity”. http://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/

Supplemental Data

```
Final Clusters after execution:
Cluster = {[1]}
Cluster = {[2]}
Cluster = {[3, 61, 69, 214, 224, 236, 241, 242, 248, 335, 251, 259, 278, 329, 330, 237, 239, 245,
264, 273, 274, 332, 244, 276, 285, 286, 334, 254, 266, 283, 295, 261, 288, 296, 305, 263, 302,
318, 320, 321, 232, 262, 271, 275, 300, 306, 316, 322, 250, 256, 284, 301, 310, 270, 291, 313,
233, 235, 287, 290, 311, 315, 267, 272, 280, 243, 247, 258, 269, 277, 299, 304, 307, 323, 328,
240, 246, 260, 279, 281, 282, 298, 303, 309, 331, 249, 252, 289, 292, 308, 325, 326, 234, 333]}
Cluster = {[4, 12, 225, 21, 83, 149, 180, 190, 6, 14, 23, 59, 201, 231, 85, 134, 209, 471, 485,
497, 499, 503]}
Cluster = {[5]}
Cluster = {[7]}
Cluster = {[8, 219, 229, 114, 404, 406, 409, 411, 415, 416, 425, 429, 462, 484]}
Cluster = {[9, 220, 230]}
Cluster = {[10, 182]}
Cluster = {[11, 37, 101, 86]}
Cluster = {[13, 16, 39, 99, 135, 140, 173, 193, 203, 401, 405, 407, 412, 413, 421, 453, 498, 25,
122, 154, 185, 400, 432, 463, 464, 33, 183, 213]}
Cluster = {[15, 72]}
Cluster = {[17]}
Cluster = {[18, 408, 419]}
Cluster = {[19, 118, 348, 349, 358, 369, 377, 360, 372, 374, 132, 339, 340, 359, 361, 363, 351,
375, 376, 395, 396, 104, 49, 57]}
Cluster = {[20, 93, 129, 155, 158, 194, 204]}
Cluster = {[22, 133]}
Cluster = {[24, 89]}
Cluster = {[26]}
Cluster = {[27]}
Cluster = {[28, 36, 222]}
Cluster = {[29]}
Cluster = {[30, 38]}
Cluster = {[31, 217]}
Cluster = {[32, 56]}
Cluster = {[34, 50]}
Cluster = {[35, 113, 167, 171, 170, 200, 210, 53, 481, 451, 488, 491, 455, 496]}
Cluster = {[40]}
Cluster = {[41, 73, 65, 81]}
Cluster = {[42]}
Cluster = {[43]}
………..
---------------------------------------------------------------------------
Order of clustering that was done:
Most similar documents: SimilarityPair{row=101, column=129, similarity=1.0}
Least similar documents: SimilarityPair{row=350, column=399,
similarity=1.694481118890326E-5}
Merging together clusters: Cluster = {[102]}, Cluster = {[130]}
Merging together clusters: Cluster = {[404]}, Cluster = {[406]}
Merging together clusters: Cluster = {[404, 406]}, Cluster = {[409]}
Merging together clusters: Cluster = {[416]}, Cluster = {[425]}
Merging together clusters: Cluster = {[399]}, Cluster = {[402]}
Merging together clusters: Cluster = {[404, 406, 409]}, Cluster = {[411]}
Merging together clusters: Cluster = {[422]}, Cluster = {[424]}
Merging together clusters: Cluster = {[404, 406, 409, 411]}, Cluster = {[415]}
Merging together clusters: Cluster = {[420]}, Cluster = {[422, 424]}
Merging together clusters: Cluster = {[417]}, Cluster = {[420, 422, 424]}
Merging together clusters: Cluster = {[418]}, Cluster = {[423]}
Merging together clusters: Cluster = {[417, 420, 422, 424]}, Cluster = {[430]}
Merging together clusters: Cluster = {[405]}, Cluster = {[407]}
Merging together clusters: Cluster = {[398]}, Cluster = {[399, 402]}
Merging together clusters: Cluster = {[401]}, Cluster = {[405, 407]}
Merging together clusters: Cluster = {[403]}, Cluster = {[414]}
Merging together clusters: Cluster = {[401, 405, 407]}, Cluster = {[412]}
Merging together clusters: Cluster = {[401, 405, 407, 412]}, Cluster = {[413]}
Merging together clusters: Cluster = {[398, 399, 402]}, Cluster = {[431]}
Merging together clusters: Cluster = {[434]}, Cluster = {[435]}
Merging together clusters: Cluster = {[416, 425]}, Cluster = {[429]}
Merging together clusters: Cluster = {[400]}, Cluster = {[432]}
Merging together clusters: Cluster = {[403, 414]}, Cluster = {[418, 423]}
Merging together clusters: Cluster = {[436]}, Cluster = {[437]}
Merging together clusters: Cluster = {[410]}, Cluster = {[417, 420, 422, 424, 430]}
Merging together clusters: Cluster = {[433]}, Cluster = {[434, 435]}
Merging together clusters: Cluster = {[183]}, Cluster = {[213]}
Merging together clusters: Cluster = {[173]}, Cluster = {[193]}
Merging together clusters: Cluster = {[401, 405, 407, 412, 413]}, Cluster = {[421]}
Merging together clusters: Cluster = {[404, 406, 409, 411, 415]}, Cluster = {[416, 425, 429]}
Merging together clusters: Cluster = {[433, 434, 435]}, Cluster = {[436, 437]}
Merging together clusters: Cluster = {[340]}, Cluster = {[359]}
Merging together clusters: Cluster = {[463]}, Cluster = {[464]}
Merging together clusters: Cluster = {[349]}, Cluster = {[358]}
Merging together clusters: Cluster = {[349, 358]}, Cluster = {[369]}
Merging together clusters: Cluster = {[349, 358, 369]}, Cluster = {[377]}
Merging together clusters: Cluster = {[351]}, Cluster = {[375]}
Merging together clusters: Cluster = {[360]}, Cluster = {[372]}
Merging together clusters: Cluster = {[348]}, Cluster = {[349, 358, 369, 377]}
Merging together clusters: Cluster = {[65]}, Cluster = {[81]}
Merging together clusters: Cluster = {[339]}, Cluster = {[340, 359]}
Merging together clusters: Cluster = {[339, 340, 359]}, Cluster = {[361]}
Merging together clusters: Cluster = {[339, 340, 359, 361]}, Cluster = {[363]}
Merging together clusters: Cluster = {[298]}, Cluster = {[303]}
Merging together clusters: Cluster = {[25]}, Cluster = {[122]}
Merging together clusters: Cluster = {[298, 303]}, Cluster = {[309]}
Merging together clusters: Cluster = {[351, 375]}, Cluster = {[376]}
Merging together clusters: Cluster = {[400, 432]}, Cluster = {[463, 464]}
Merging together clusters: Cluster = {[360, 372]}, Cluster = {[374]}
Merging together clusters: Cluster = {[281]}, Cluster = {[282]}
Merging together clusters: Cluster = {[304]}, Cluster = {[307]}
Merging together clusters: Cluster = {[323]}, Cluster = {[328]}
Merging together clusters: Cluster = {[234]}, Cluster = {[333]}
Merging together clusters: Cluster = {[14]}, Cluster = {[23]}
Merging together clusters: Cluster = {[279]}, Cluster = {[281, 282]}
Merging together clusters: Cluster = {[265]}, Cluster = {[293]}
Merging together clusters: Cluster = {[499]}, Cluster = {[503]}
Merging together clusters: Cluster = {[459]}, Cluster = {[461]}
Merging together clusters: Cluster = {[25, 122]}, Cluster = {[154]}
Merging together clusters: Cluster = {[298, 303, 309]}, Cluster = {[331]}
Merging together clusters: Cluster = {[240]}, Cluster = {[246]}
Merging together clusters: Cluster = {[408]}, Cluster = {[419]}
Merging together clusters: Cluster = {[149]}, Cluster = {[180]}
Merging together clusters: Cluster = {[279, 281, 282]}, Cluster = {[298, 303, 309, 331]}
Merging together clusters: Cluster = {[6]}, Cluster = {[14, 23]}
Merging together clusters: Cluster = {[299]}, Cluster = {[304, 307]}
Merging together clusters: Cluster = {[12]}, Cluster = {[225]}
Merging together clusters: Cluster = {[265, 293]}, Cluster = {[297]}
Merging together clusters: Cluster = {[289]}, Cluster = {[292]}
Merging together clusters: Cluster = {[316]}, Cluster = {[322]}
Merging together clusters: Cluster = {[249]}, Cluster = {[252]}
Merging together clusters: Cluster = {[235]}, Cluster = {[287]}
Merging together clusters: Cluster = {[260]}, Cluster = {[279, 281, 282, 298, 303, 309, 331]}
Merging together clusters: Cluster = {[312]}, Cluster = {[314]}
Merging together clusters: Cluster = {[59]}, Cluster = {[201]}
Merging together clusters: Cluster = {[286]}, Cluster = {[334]}
Merging together clusters: Cluster = {[299, 304, 307]}, Cluster = {[323, 328]}
Merging together clusters: Cluster = {[395]}, Cluster = {[396]}
Merging together clusters: Cluster = {[258]}, Cluster = {[269]}
Merging together clusters: Cluster = {[317]}, Cluster = {[319]}
Merging together clusters: Cluster = {[250]}, Cluster = {[256]}
Merging together clusters: Cluster = {[311]}, Cluster = {[315]}
Merging together clusters: Cluster = {[276]}, Cluster = {[285]}
Merging together clusters: Cluster = {[501]}, Cluster = {[502]}
Merging together clusters: Cluster = {[248]}, Cluster = {[335]}
Merging together clusters: Cluster = {[301]}, Cluster = {[310]}
Merging together clusters: Cluster = {[240, 246]}, Cluster = {[260, 279, 281, 282, 298, 303, 309,
331]}
Merging together clusters: Cluster = {[265, 293, 297]}, Cluster = {[312, 314]}
Merging together clusters: Cluster = {[243]}, Cluster = {[247]}
Merging together clusters: Cluster = {[253]}, Cluster = {[257]}
Merging together clusters: Cluster = {[300]}, Cluster = {[306]}
Merging together clusters: Cluster = {[277]}, Cluster = {[299, 304, 307, 323, 328]}
Merging together clusters: Cluster = {[251]}, Cluster = {[259]}
Merging together clusters: Cluster = {[242]}, Cluster = {[248, 335]}
Merging together clusters: Cluster = {[63]}, Cluster = {[71]}
Merging together clusters: Cluster = {[233]}, Cluster = {[235, 287]}
Merging together clusters: Cluster = {[271]}, Cluster = {[275]}
Merging together clusters: Cluster = {[265, 293, 297, 312, 314]}, Cluster = {[317, 319]}
Merging together clusters: Cluster = {[237]}, Cluster = {[239]}
Merging together clusters: Cluster = {[500]}, Cluster = {[501, 502]}
Merging together clusters: Cluster = {[273]}, Cluster = {[274]}
Merging together clusters: Cluster = {[249, 252]}, Cluster = {[289, 292]}
Merging together clusters: Cluster = {[325]}, Cluster = {[326]}
Merging together clusters: Cluster = {[243, 247]}, Cluster = {[258, 269]}
Merging together clusters: Cluster = {[268]}, Cluster = {[327]}
Merging together clusters: Cluster = {[278]}, Cluster = {[329]}
Merging together clusters: Cluster = {[346]}, Cluster = {[368]}
Merging together clusters: Cluster = {[450]}, Cluster = {[452]}
Merging together clusters: Cluster = {[276, 285]}, Cluster = {[286, 334]}
Merging together clusters: Cluster = {[284]}, Cluster = {[301, 310]}
Merging together clusters: Cluster = {[20]}, Cluster = {[93]}
```