Content Based Ranking

*Determining contribution of each document to the dataset

Ritesh Ghodrao
2015A7PS0096G

Soham Kadam
2015A7PS0067G

Abhishek Jain
2015A7PS0025G

Vatsal Bhanderi
2015A7PS0008G

***Abstract*—Nowadays we are confronted with rapidly increasing number of documents. Maintaining an overview seems to be impossible. This report provides an enhanced content based duplicate document detection technique that uses various similarity measures. Documents are represented as vectors and different mathematical models are applied to quantify the similarity. A total of 19 different similarity measures(vector distance, boolean distance, structural similarity) are implemented. These calculations are made for every document pair in the dataset. Further, the cohesion of every document is calculated against set of other documents in the dataset. The cohesion value is ranked in order to identify the contribution of every document in the dataset.**

I. Introduction

The detailed idea, procedure and empirical results are discussed in the following section.Various modules including numpy, gensim, sklearn, stemming, scipy are used. The graphs are plotted using matplot to analyse the results. The approach with psuedo codes is presented in the next section followed by results and findings.

II. Proposed Approach

A. Pre-processing of documents :

Consider a corpus having set of classes (C = c1, c2, ..., cn) of documents (D = d1, d2, ..., dp) . All documents are pre-processed which includes lexical-analysis,stop-word elimination and stemming and then index terms are extracted. The term-document matrix is constructed using the vector space model, where TF-IDF values are used to measure the weight of the terms (t) in their respective document (t_{ij}) (Table 1).

	d_1	d_2	d_3	...	d_p
t_1	t_{11}	t_{12}	t_{13}	...	t_{1p}
t_2	t_{21}	t_{22}	t_{23}	...	t_{2p}
t_3	t_{31}	t_{32}	t_{33}	...	t_{3p}
.	.	.	.	...	.
.	.	.	.	...	.
.	.	.	.	...	.
t_r	t_{r1}	t_{r2}	t_{r3}	...	t_{rp}

B. Similarity Measures

The following mathematical measures are used to compute similarity. Each document is taken as a vector and pairwise similarities are calculated.

Following are the distance functions between two numeric vectors u and v(u and v are considered boolean vectors for 11-18). The values are normalised to give results between 0 to 1, here 0 would mean completely dissimilar and 1 means completely similar(identical) documents.

1) Cosine Similarity:

$$cosine - similarity(u, v) = 1 - \frac{u \cdot v}{|u|_2 |v|_2}$$

2) Bray Curtis:

$$bray - curtis(u, v) = \frac{\sum |u_i - v_i|}{\sum |u_i + v_i|}$$

3) Canberra:

$$canberra(u, v) = \sum_i \frac{|u_i - v_i|}{|u_i| + |v_i|}$$

4) ChebyShev:

$$ChebyShev(u, v) = max_i |u_i - v_i|$$

5) CityBlock:

$$city - block(u, v) = \sum_i |u_i - v_i|$$

6) Correlation:

$$correlation(u, v) = 1 - \frac{(u - \overline{u})(v - \overline{v})}{|(u - \overline{u})|_2 |(v - \overline{v})|_2}$$

$\overline{u}$ and $\overline{v}$ are mean of u and v respectively.

7) Euclidean:

$$euclidean(u, v) = ||u - v||_2$$

8) Minkowski:

$$minkowski(u, v) = ||u - v||_p$$

here p is the order of norm of difference

$$||u - v||$$

9) Sq Euclidean:

$$sqeuclidean(u, v) = (||u - v||_2)^2$$

10) W-Minkowski:

$$W - minkowski(u,v) = (\sum(|w_i(u_i - v_i)^p))^{1/p}$$

here w is weight and p is the order of norm of difference

$$||u - v||$$

$^*c_{ij}$ is the number of occurrences of u[k]=i and v[k]=j for k<n , n being total number of distinct terms in corpus

11) Hamming:

$$hamming(u,v) = \frac{c_{10} + c_{01}}{n}$$

12) Dice:

$$dice(u,v) = \frac{C_{TF} + C_{FT}}{2C_{TT} + C_{FT} + C_{TF}}$$

13) Jaccard:

$$jaccard(u,v) = \frac{C_{TF} + C_{FT}}{C_{TT} + C_{FT} + C_{TF}}$$

14) Russellrao:

$$russellrao(u,v) = \frac{n - C_{TT}}{n}$$

15) Roger-Stanimoto:

$$Roger - Stanimoto(u,v) = \frac{R}{C_{FT} + C_{TF} + n}$$

$$here, R = 2(C_{TF} + C_{FT})$$

16) Sokal Michener:

$$sokal - michener(u,v) = \frac{R}{S + R}$$

$$here, R = 2(C_{TF} + C_{FT}) and S = C_{FF} + C_{TT}$$

17) Sokal Sneath:

$$sokal - sneath(u,v) = \frac{R}{C_{TT} + R}$$

$$here, R = 2(C_{TF} + C_{FT})$$

18) Yule:

$$yule(u,v) = \frac{R}{C_{TT} + C_{FF} + R/2}$$

$$here, R = 2 * C_{TF} * C_{FT}$$

19) Structural Similarity: Structure-based similarity between two documents d_p and d_q is generally measured by computing how many such terms are there that are common to both d_p and d_q and also they should maintain same order in both documents. The formula used to compute this similarity between two documents d_p and d_q is

$$struct(d_p, d_q) = a/b$$

where 'a' represents the number of terms pairs that are common to both d_p and d_q and maintain the same order in both the documents. 'b' represents total combination of common term pairs.

C. Psuedo Codes

1) Computing the Similarities : The following algorithm is used to calculate pairwise similarity for all features(except structural similarity). In case of features (11-18), we use boolean matrix instead of tf-idf .

DATA: tf-idf matrix, boolean matrix
RESULT: similarities between all document pairs
$similarity - list \leftarrow \phi$
for each t_i in tf-idf matrix **do**
  $temporaryList \leftarrow \phi$
  for each array t_j in tf-idf matrix **do**
    $sim \leftarrow similarity\ between\ t_i\ and\ t_j$
    add sim to temporaryList
  end for
  add temporaryList to similarity-list
end for

2) Calculating the Inversion Count: This is used for efficiently calculating Structural Similarity.The algorithm based on Divide and Conquer paradigm. The time complexity of this approach is O(n log(n)).In divide step, we divide problem in two parts which are then solved recursively. The key concept is to count the number of inversion in merge procedure. In merge procedure, we pass two sub-list, the element is sorted and inversion is found by following algorithm.

DATA: 1-D array
RESULT: number of inversions in array
$count \leftarrow 0$
$i \leftarrow left$
$j \leftarrow mid$
C is the Sorted list
Traverse list1 and list2 unti mid or left is encountered
compare list1[i] and list[j]
if $list1[i] \leq list2[j]$ **then**
  c[k++] = list1[i++]
else
  c[k++] = list2[j++]
  count = count + mid -i
end if
add rest elements of list1 and list2 in c
copy sorted list c back in original list
return count

3) Computing the Structural Similarity : The following algorithm is used to compute Structural Similarity. This uses inversion count, the time complexity of this computation is $O(D^2 * N * log(N))$, where D is number of documents and N is number of terms in a document.

DATA: term list of all documents
RESULT: structural based similarities between all document pairs
$global\ variable\ count;$
$structural - similarity \leftarrow \phi$
for i in range of (0, lengthOfTermList) **do**
  $temporaryList \leftarrow \phi$
  $dictionary1 \leftarrow OrderedDictionaryoftermList$
  $index \leftarrow 0$

for each j in keyOfDictionary **do**
$dictionary[j] \leftarrow index$
$count \leftarrow index + 1$
end for
for each j in range of (0,lengthOfTermList) **do**
$dictionary2 \leftarrow emptyOrderedDictionary$
for each k in range of (0,lengthOfTermList[j]) **do**
if termList[j][k] is present in dictionary1 and not present in dictionary2 **then**
add termList[j][k] in dictionary2 with index k
end if
end for
$finalList \leftarrow \phi$
for each k in keys of dictionary2 **do**
add dictionary1[k] in finalList
end for
count number of inversions in finalList
$numofInv \leftarrow count$
$count \leftarrow 0$
if $lengthOffinalList \leq 1$ **then**
add -1 to temporaryList
else
add

$$\frac{2 * noOfInv}{(length(finalList) * length(finalList) - 1)}$$

to temp
end if
end for
add temporaryList to structured-similarity
end for

4) Computing the Cohesion value for every document : Cohesion value for a document is harmonic mean of its average-similarities with all other documents in the corpus.

DATA: Array of float
RESULT: harmonic mean of array
for harmonic mean **do**
$sum \leftarrow 0$
for i in range of (0, lengthOfArray) **do**
if $array[i] == 0$ **then**
$harmonicMean \leftarrow 0$
end if
$sum \leftarrow sum + \frac{1}{array[i]}$
end for
if $sum \neq 0$ **then**
$harmonicMean \leftarrow \frac{1}{sum}$
end if
end for

5) Ranking the documents :

DATA: Similarities of Documents
RESULT: ranked list of documents
$harmonicMean \leftarrow \phi$
$harmonicMean - list \leftarrow \phi$
$index \leftarrow 1$
for each i in similarity-matrix **do**
add harmonic-mean of i in dictionary with value = index
append the previous step value in list
$index \leftarrow index + 1$
end for
sort dictionary
$rankedDocuments \leftarrow valuesOfDictionary$

III. Experimental Analysis

A. Experimental Set up

DUC 2001 dataset is used for testing the results. The final matrix with cohesion value for each document is calculated and normalised. This value is an approximate measure to uniqueness of documents and also quantifies the contribution of the document to corpus. A detailed graphical analysis is made to draw conclusions from results obtained. *we considered 206 documents to analyse the graphs better.

B. Discussion

Fig 1 We calculated the average similarity from 19 similarity features. This value is normalised between 0 to 1. The calculations are calculated for all n*n document pairs. Factually, each document is identical to itself, thus giving similarity value of 1, this can be verified from dark diagonal line in heat map. The other darker regions are mostly concentrated around this diagonal line only, which indicate that documents which belong to same set(or folder) are similar. Moreover, the color is uniform over the region, asserting that all documents have some similar documents present in the corpus.

Fig. 1. Heat Map with average similarity between documents

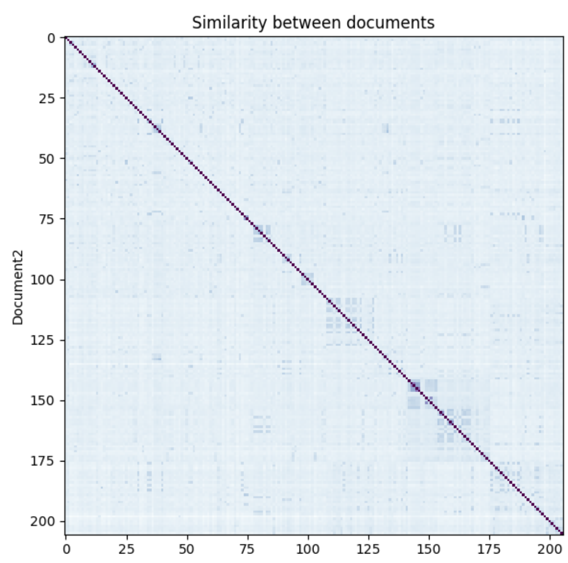


Fig 2 We analysed Structural Similarity for the documents independently. The results support our findings in Fig 1. The result can again be verified with darker diagonal

line. The darker region is again concentrated around the diagonal line. This provides evidence for similarity for documents that belong to same set have more similarity. This approach also signifies that such documents are near-duplicates.The similarity is uniform and significantly less for other documents pairs.

Fig. 2. Structural Similarity between documents

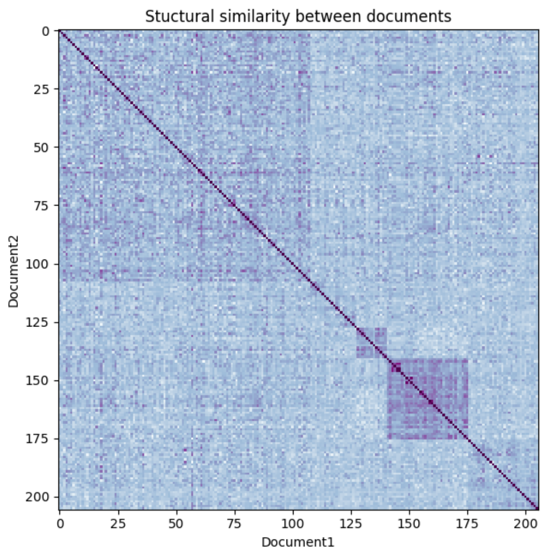
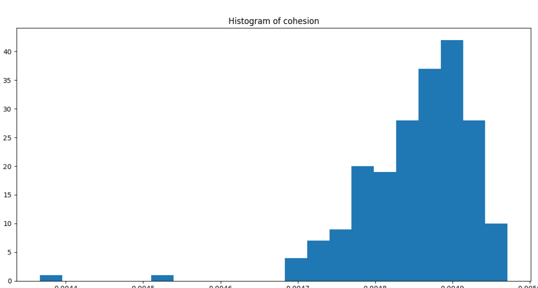


Fig 3 For each term, calculate the cohesion with the other terms based on the values in the SIM-MATRIX, with the term as the cluster head. This gives us a score for every term (which will act as a representative of the cluster) and then we rearrange the terms based on this score in descending order, and thus we can rank them. Cohesion is the harmonic value of similarity of a document with all the other documents in the corpus. This value is a measure of how unique a document is, thus how much a document contributes to the corpus. The cohesion values are concentrated in small range, which indicates that most documents are similar and contribute almost equally to the corpus.

Fig. 3. Cohesion



IV. Conclusion

Our results on DUC-2001 indicate all cohesion values in range of 0.0044-0.005, all the values are concentrated in this small space, which indicates that no particular document contributes significantly to corpus. The analysis of results indicate that given dataset has several near duplicate document pairs. There are several similar documents in the corpus corresponding to every document. All documents contribute almost equally to the corpus, thus, no distinctly unique documents are present in corpus. This also means there are various near duplicate documents present in the corpus. In this report, harmonic mean of 19 content based similarity values is considered to rank the contribution of document to dataset. This also identifies the relative uniqueness of documents. A diversified use of different measures is made to produce effective results. 10 vector similarity measures, 8 boolean similarity measures and a structural similarity based approach is used to rank the documents. The results can be used to detect near duplication, plagiarism or find content uniqueness of documents. The approach may produce better results if different weights are assigned to the similarity measures based on their relevance. Inclusion of semantic similarity measures can be tested to produce more promising results.