Content Based Document Similarity and Near Duplicate Page Detection

Aman Chhajed
2014B4A70608G

***Abstract*—This paper discusses the findings of the study project done on content based document similarity and near duplicate page detection. Retrieval of information has become increasingly faster and much more efficient over the years as search engines continue to optimize their algorithms and test for more useful signals for measuring noise. However, the internet continues to face major challenges in tackling rampant redundancy of content. Implementation of certain techniques like data filtering algorithms can eliminate duplicate and near duplicate documents to make searches more relevant and time saving. The identification of similar or near-duplicate pairs in a large collection is a significant problem with wide-spread applications.**

***Index Terms*—Correlation, SVM, MLPC, F-measure**

I. Introduction

The indexed web contains at least 4.5 billion web-pages (November, 2017). There is a need to use this big volume of information efficiently for effectively satisfying the information need of the user on the Web. Among users looking for information on the Web, 85% of all information requests reach search engines. Search engines in response to a user's query typically produces the list of documents ranked according to closest to the user's request. The general trend shows users continue to rely more and more on these search results. It's also important to note loss of attention span in the internet age. This means it will become increasingly unlikely for a user to look at the next set of results after every set that is seen. These documents are presented to the user for examination and evaluation. Web users have to go through the long list and inspect the titles, and snippets sequentially to recognize the required results. Filtering these results manually consumes effort and time especially when a lot of near duplicate documents are retrieved. The efficient identification of finding potentially irrelevant content is an important challenge in many applications especially when it has a large amount of data and there is the necessity to save data from diverse sources that needs to be addressed. Though near duplicate documents display striking similarities, they are not bit wise similar. Google researchers estimate the total potentially duplicate content to be around one-fourth the size of the index-able pages. Thus algorithms for recognition for removal of these pages become inevitable.

II. Proposed Approach

The following block diagram illustrates the general workflow of our proposed approach for this task (see Fig 1). All those steps are further elaborated here.

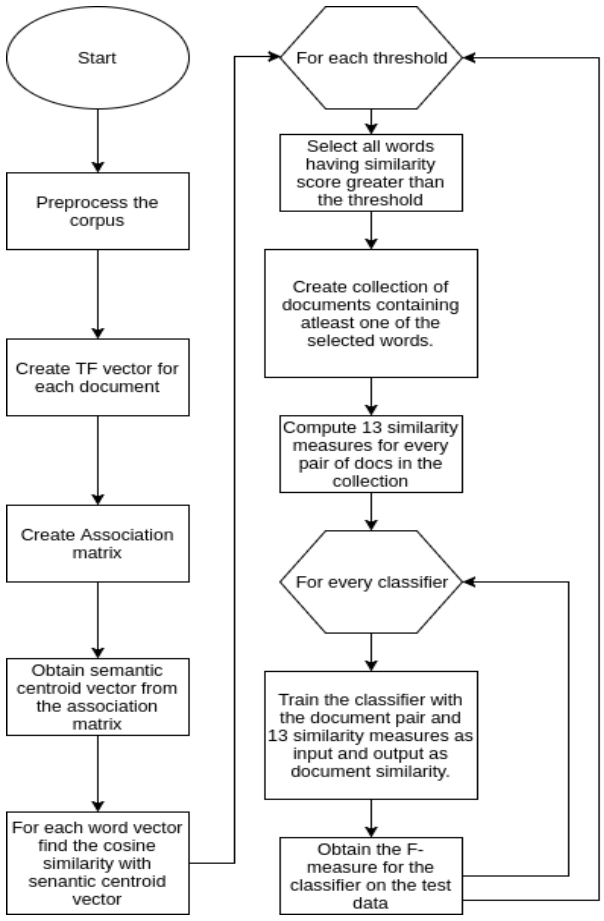

Fig. 1. Block Diagram of proposed approach

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 termdocument matrix is constructed using the vector space

model, where term frequency values are used to measure the weight of the terms (t) in their respective document (t_{ij}).

B. Correlation based keyword Selection

1. To select a subset of corpus we used correlation based keyword selection. Assuming there are n number of keywords, the frequency based correlation factor between keyword i and keyword j is computed using the following equation:

$$C_{ij} = \sum_{k=1}^{D} f_{ik} * f_{jk}$$

where f_{ik} and f_{jk} are the frequency of i^{th} and j^{th} keyword in the k^{th} document respectively, where k is varying over all the documents.

2. After finding the frequency based correlation factors between every pair of keywords in the corpus, an association matrix is formed where each entry in the i^{th} row and j^{th} column is the correlation value between keyword W_i and W_j, and each row i represent a keyword vector corresponding to W_i.

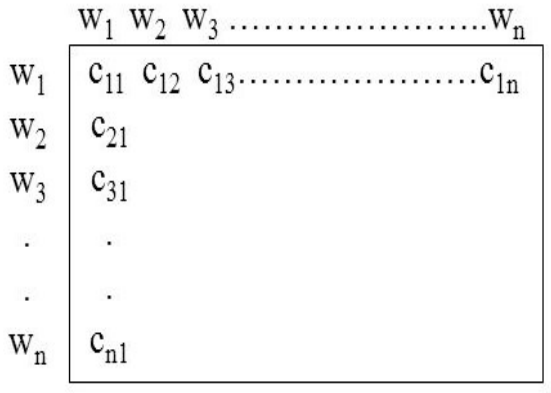

Fig. 2. Association Matrix

3. Next, the correlation values of the association matrix are normalized in order to keep all the correlation values between 0 and 1 and it can be done using the following equation:

$$S_{ij} = \frac{C_{ij}}{C_{ii} + C_{jj} - C_{ij}}$$

Normalized score is 1 if the 2 keywords have the same frequency in all the documents.

4. For each row of above matrix S_{ij}, the mean of all correlation values belonging to S_{ij}, are calculated which generates the semantic centroid vector $\vec{sc}$ of n dimensions where each sc_i can be calculated using the following equation:

$$sc_i = \frac{\sum_{j=1}^{n} S_{ij}}{n}$$

5. Next, cosine similarity between each keyword $\vec{W}_i$ and $\vec{sc}$ are computed. This way, every keyword of D obtains a cosine similarity score. Finally top k keywords are selected based on the threshold we put on cosine similarity scores, and the number of irrelevant words are reduced from the corpus and now every document in D contains only these top k words (Example of top 10 words shown in Table 1). Any document which have none of these top k words are discarded from the corpus.

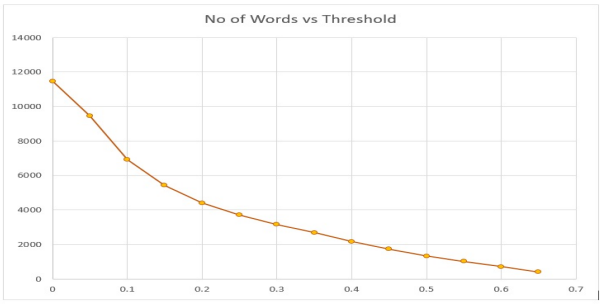

Fig. 3. No of words vs Threshold

TABLE I
TOP 10 MOST SIGNIFICANT WORDS IN THE CORPUS AND THEIR SIMILARITY SCORES

words	word_similarity
acknowledg	0.7829435709
eventu	0.7733762703
despit	0.7676976484
mind	0.7666764223
take	0.756232907
instanc	0.7553865337
singl	0.7549114179
assert	0.7547403013
realli	0.7543047923
short	0.7529377259

TABLE II
LENGTH OF VOCABULARY AT DIFFERENT THRESHOLDS

Threshold	No of Words
0.0	11476
0.05	9460
0.1	6909
0.15	5433
0.2	4407
0.25	3711
0.3	3154
0.35	2693
0.4	2162
0.45	1721
0.5	1329
0.55	1009
0.60	703
0.65	397

C. Similarity Measures

1) Euclidean Distance: Euclidean distance gives the linear distance between two vector points in N-dimensional vector

space.

$$Distance(\vec{t_a}, \vec{t_b}) = \sqrt{\sum_{t=1}^{n} |w_{t,a} - w_{t,b}|^2}$$

2) Cosine Similarity: The similarity measure quantify the similarity between two documents as cosine of angle between their term vector. Mathematically, it can be representation as

$$Score_{cosine}(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}| \times |\vec{t_b}|}$$

3) Jaccard Coefficient: Let A and B be two sets,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

4) Extended Jaccard Coefficient: It is ratio of intersection and union of the two documents.

$$Coff_{Jaccard}(\vec{t_a}, \vec{t_b}) = \frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}|^2 + |\vec{t_b}|^2 - \vec{t_a} \cdot \vec{t_b}}$$

5) Pearson Correlation coefficient: It is the coeffcient of correlation between two vectors.

$$Coff_p(\vec{t_a}, \vec{t_b}) = \frac{n \sum_{t=1}^{n} w_{t,a} \times w_{t,b} - T_a \times T_b}{\sqrt{[n \sum_{t=1}^{n} w_{t,a}^2 - T_a^2][n \sum_{t=1}^{n} w_{t,b}^2 - T_b^2]}}$$

where $T_a = \sum_{t=1}^{n} w_{t,a}$ and $T_b = \sum_{t=1}^{n} w_{t,b}$

6) Manhattan Distance: Manhattan distance computes the distance between two vectors as the absolute sum of the differences of their corresponding components.

$$Dist_{man}(\vec{t_a}, \vec{t_b}) = \sum_{t=1}^{n} |w_{t,a} - w_{t,b}|$$

7) Bray Curtis Distance: This distance is an optimisation over the manhattan distance by using normalisation. It can be used when values are in positive range. This parameter can be helpful in various context which involve skewed data.

$$Dist_{bray}(\vec{t_a}, \vec{t_b}) = \sum_{t=1}^{n} \frac{|w_{t,a} - w_{t,b}|}{(w_{t,a} + w_{t,b})}$$

8) Canberra Distance: It is a weighted form of Manhattan Distance.

$$Dist_{canberra}(\vec{t_a}, \vec{t_b}) = \sum_{t=1}^{n} \frac{|w_{t,a} - w_{t,b}|}{|w_{t,a}| + |w_{t,b}|}$$

9) Chebyshev Distance: It is defined as the max distance along any coordinate direction of given vector.

$$Dist_{cheby}(\vec{t_a}, \vec{t_b}) = \max_t(|w_{t,a} - w_{t,b}|)$$

10) Hamming Distance: The Hamming distance between 1-D arrays u and v, is simply the proportion of disagreeing components in u and v. If u and v are boolean vectors, the Hamming distance is

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

where c_{ij} is the number of occurences of u[k]=i and v[k] = j, for k ¡ n.

11) Overlap Coefficient: Overlap coefficient for two vectors is defined as the dot product of 2 vectors divided by minimum of square of magnitude of 2 vectors.

$$Dist_{overlap}(t_a, t_b) = \frac{|t_a \cdot t_b|}{\min(|t_a|^2 |t_b|^2)}$$

12) Minkowski Distance: It is the generic form of all the distances defined as the inverse root of powered summation.

$$Dist_{minkowski}(\vec{t_a}, \vec{t_b}) = \left(\sum_{t=1}^{n} |\vec{t_a} - \vec{t_b}|^p \right)^{1/p}$$

13) Dice Coefficient: Another similarity measure highly related to the extended Jaccard is the Dice coefficient.The Dice coefficient can be obtained from the extended Jaccard coefficient by adding $\vec{t_a} \cdot \vec{t_b}$ to both the numerator and denominator.

$$Coff_{Jaccard}(\vec{t_a}, \vec{t_b}) = 2\frac{\vec{t_a} \cdot \vec{t_b}}{|\vec{t_a}|^2 + |\vec{t_b}|^2}$$

D. Generating the training feature vector

1) : Select top k words obtained in II-B which have the similarity score above the given threshold.

2) : Discard the documents which does not have any of these top k words.

3) : Construct a term document matrix from these reduced document set and top k words using the Vector Space Model where TF-IDF value is used to measure the weight of the term t_i in its respective document d_j.

4) : Each document is associated with a label which denoted the class set of the document.

5) : We compute the 13 similarity scores using the similarity measures mentioned above in section II-C for each possible pair of documents using their respective TF-IDF vectors. These similarity scores act as the feature vector for training the classifiers.

6) : Each document pair is associated with a same_class attribute which is 1 if both the documents in the pair belongs to same class and 0 if they belong to different class. This attribute act as the target output while training the classifiers.

7) : Performance metrics of different classifiers, at different thresholds are measured using the test data set. Results are discussed in the Experiment Section.

III. Experimental Analysis

Various steps used in our experiment are concisely given in the form of block diagram in Fig 1. All those steps are further elaborated here.

A. Experimental Setup

Dataset used for the study is the DUC2001 Summarization Documents which was compiled together by National Institute of Standards and Technology. NIST produced 60 reference sets, 30 for training and 30 for testing. Each set contained documents, per-document summaries, and multi-document summaries, with sets defined by different types of criteria such as event sets, opinion sets, etc. The training set contained 299 documents and testing set contained a total of 309 documents. We extracted the data from these documents and formed a Dataframe using these data. Documents belonging to the same set are assigned same numeric label. Pre-processing was done on these documents in various steps mentioned below:

1) Useful information is extracted from the HTML tags.
2) Punctuation's and other symbols are eliminated.
3) Stop words are cleared for better computational efficiency and effectiveness of results.
4) Finally, terms are stemmed, parsed and vectorized corresponding to each document in pandas dataframe.
5) Correspondingly, column vector for label is also created.

	docs	label	doc_vec_count
0	\n\n sjmn \n \n us; president; appointment;...	1	[0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, ...
1	\n\n sjmn \n \n photo; photo: clarence thom...	1	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2	\n\n sjmn \n \n photos ; photo: knight ri...	1	[0, 0, 1, 2, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, ...
3	\n\n sjmn \n \n photo; photo: associated pr...	1	[0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ...
4	\n\n sjmn \n \n photos chart; photo: as...	1	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, ...
5	\n\n\nwsj \n\n\n .\n\n\n clarence thomas on ...	1	[0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, ...
6	\n\n\nwsj \n\n\n .\n\n\n the marshall seat:\...	1	[0, 0, 0, 0, 2, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, ...
7	\n\n\nwsj \n\n\n .\n\n\n apply marshall s pr...	1	[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ...
8	\n\n\nwsj \n\n\n .\n\n\n the showdown:\n t...	1	[2, 0, 1, 0, 0, 0, 0, 1, 0, 2, 0, 0, 1, 0, 0, ...
9	\n\n\nwsj \n\n\n .\n\n\n battle scars:\n c...	1	[0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, ...
10	\n\n\nwsj \n\n\n .\n\n\n review & outlook e...	1	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ...
11	\n\n ap \nap nr edt\nr f pm milkenprofile ...	2	[0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, ...
12	\n\n ap \nap nr est\nr f pm milken bjt ...	2	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
13	\n\n ap \nap ny est\nr f pm columbia milke...	2	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
14	\n\n la \n \n\n\njanuary thursday home ...	2	[0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, ...
15	\n\n la \n \n\n\nmarch thursday home ed...	2	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ...

Fig. 4. Dataframe Formed after Preprocessing.

Total vocabulary in the training data set after stemming and removal of stop words is 11476. The length of vocabulary at different thresholds is shown in Table 2. After considering every possible document pair, we have 44551 training examples for which we compute 13 different similarity measures.

B. Classifiers

We used different classifiers to find a suitable classifier for our model and also to validate the accuracy of our results. Following is the list of classifiers we used for training and testing.

1) SVM linear classifier
2) Linear SVC
3) K Nearest Neighbour
4) Gaussian Naive Bayes
5) Random Forest Classifier
6) Decision Tree CLassifier

Doc Ids	Same Class	Cosine similarity	Euclidean Distance	Extended jaccard	Pearson Correlation	Dice	Manhattan distance	Jaccard Similarity	Bray_Curtis Distance	Canberra Distance	Chebyshev Distance	Hamming Distance	Overlap Coefficient	Minkowski Distance
(0, 1)	1	0.718303	0.750596	0.560431	0.714404	0.718303	13.343942	0.317739	0.533058	386.135264	0.170893	350	0.718303	0.338297
(0, 2)	1	0.503697	0.996296	0.336627	0.494819	0.503697	21.009320	0.203106	0.716746	716.652488	0.273625	667	0.503697	0.462387
(0, 3)	1	0.588505	0.907188	0.416938	0.583246	0.588505	15.743881	0.230634	0.652051	467.733681	0.290810	437	0.588505	0.452239
(0, 4)	1	0.473682	1.025981	0.310343	0.464523	0.473682	21.348743	0.166667	0.746064	656.433019	0.229390	625	0.473682	0.458712
(0, 5)	1	0.445863	1.052746	0.286888	0.436538	0.445863	20.392934	0.193333	0.730214	646.775006	0.327589	605	0.445863	0.530985
(0, 6)	1	0.626784	0.863963	0.456435	0.620978	0.626784	16.700179	0.247826	0.624698	562.289037	0.261767	519	0.626784	0.413494
(0, 7)	1	0.468684	1.030841	0.306066	0.461894	0.468684	17.662097	0.162252	0.732134	530.147158	0.312242	506	0.468684	0.523131
(0, 8)	1	0.402881	1.092812	0.252254	0.393376	0.402881	20.555447	0.172330	0.762328	725.778631	0.356515	682	0.402881	0.558398
(0, 9)	1	0.519623	0.980180	0.351007	0.512087	0.519623	19.973295	0.159302	0.745530	768.595335	0.336554	723	0.519623	0.481419
(0, 10)	1	0.428669	1.068954	0.272806	0.420270	0.428669	19.860511	0.151361	0.769483	523.910965	0.242160	499	0.428669	0.488316
(0, 11)	0	0.039431	1.386051	0.020112	0.027401	0.039431	21.888862	0.094703	0.917237	578.903408	0.648514	564	0.039431	0.803044
(0, 12)	0	0.035745	1.388708	0.018198	0.027114	0.035745	18.884008	0.098214	0.912770	517.379930	0.677164	505	0.035745	0.854626
(0, 13)	0	0.058996	1.371863	0.030395	0.050453	0.058996	19.253936	0.082569	0.924734	513.154585	0.502619	500	0.058996	0.768751

Fig. 5. Similarity Measures between pairs of documents

7) Ada Boosting
8) Multi Layer Perceptron Classifier
9) Gradient Boosting Classifier
10) Extra Trees Classifier

C. Parameters for performance measurement

$$Precision(p) = \frac{relevant_{documents} \cap retrieved_{documents}}{retrieved_{documents}}$$

$$= \frac{\#(TruePositive)}{\#(TruePositive + FalsePositive)}$$

$$Recall(r) = \frac{relevant_{documents} \cap retrieved_{documents}}{relevant_{documents}}$$

$$= \frac{\#(TruePositive)}{\#(TruePositive + FalseNegative)}$$

$$F - Measure = 2\frac{pr}{p+r}$$

D. Discussion

Table III, IV and V show F-measures for different classifiers at different threshold values. Table VI shows the extent to which words can be discarded while preserving good F1-scores for each classifier. A good F1-score is a score around $.8 \pm .02$. A higher value of threshold would discard greater number of terms in favour of fewer more associated terms (for creating document vectors for measuring similarity scores). The highest F-measures (relevancy measures) were those trained via SVM (Support Vector Machines (kernel = linear)), MLPC (Multi-Layer Perceptron Classifier) and Random Forest classification algorithms - each at around 0.84 at threshold value of 0 (allowing all terms to contribute to similarity scores). To reduce the number of variables involved, default values were chosen for the classifiers, from the scikit-learn library. LinearSVC is the only classifier which is seen to increase its F measure with increase in threshold (till 0.15) (decrease in word-bag size), probably because of removal of

less-associated words, in our context, noise - for this strong classifier. Also, SVMs took longer than other classifiers to model over training corpus, though it was best able to detect duplicate documents. In SVM, cosine similarity, pearson's correlation, dice coefficient, overlap coefficient, euclidean distance, minkowsky distance (p = 3), that is, 6 out of the 13 similarity measures had the most impact on deciding if two documents are dissimilar or duplicate.

Ensemble methods continue to perform at par if not better than the normal approach - Extra Trees, Random Forest, AdaBoost, Gradient Boosting continue to produce good F1-measures (more than 0.80) even at greater threshold values, suggesting their ability to adapt to lesser information associated with a smaller term-bag size. Another interesting note is inferior classification of K-NN classifier (k = 5), presumably because of the sparsity of the document vectors in the n-dimensional space (here, n is word-bag size) reflecting it's impact over each of the 13 measures of similarity. Almost all classifiers have an increased F1-score at a threshold of 0.05 compared to the f1 scores at 0.0. Moreover, these measures homogeneously decrease for each classifier with an increase in threshold value after 0.05. From this, it can be inferred that all keywords whose cosine similarity scores (w.r.t. semantic centroid) is less than 0.05 are basically behaving like noise in our corpus and seem needful to be discarded. At threshold of 0.2, the vocabulary size is decreased significantly by a factor of 2.6 and the classifiers like SVM (linear kernel), AdaBoost, Random Forest and MLPC still gave F-measures of around 0.8 as compared to other classifiers which performed quite poorly. It seems likely to aver the top K words chosen using threshold value of 0.2 are the most meaningful words and these words can be used to decrease number of documents used in training classifiers. This reduction in number of words as terms, and number of documents can deliver massive computational boosts since irrelevant and redundant words are discarded in favour of gaining more control over speed of retrieval and storage, and hence training corpus size decreases. Reduction in number of words also makes the computation of similarity scores between documents more efficient.

IV. Conclusions

In this paper, a novel near duplicate document detection approach is illustrated along with observations made on the same. The approach combined the correlation between words (key-words or terms) in the corpus and 13 of the best document similarity measures in the world defined to select top features among these measures which affect detection of possibly near duplicate documents. By using the correlation based keyword selection, all the irrelevant and redundant words are removed from the corpus and only the most significant keywords are used to generate a more scalable term vector for each document. Using different document similarity measures, we are able to optimize all these measures to values which provide a much more meaningful measure of similarity between documents which are fed into the classifier.

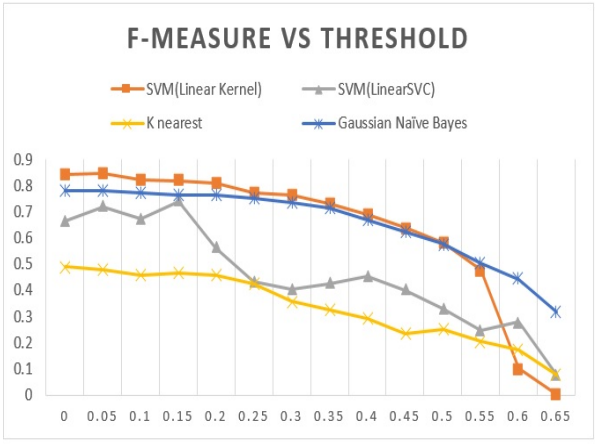

Fig. 6. F-measure vs Threshold-I

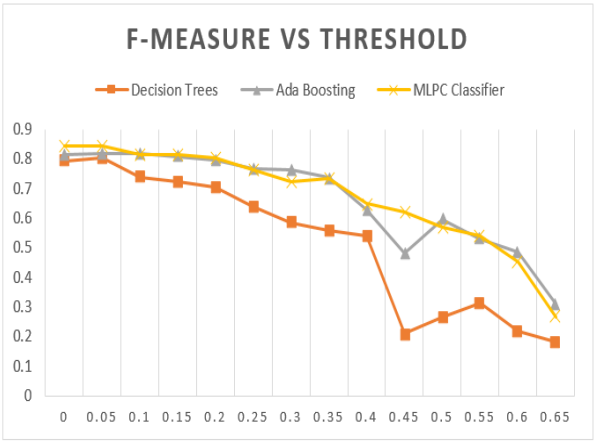

Fig. 7. F-measure vs Threshold-II

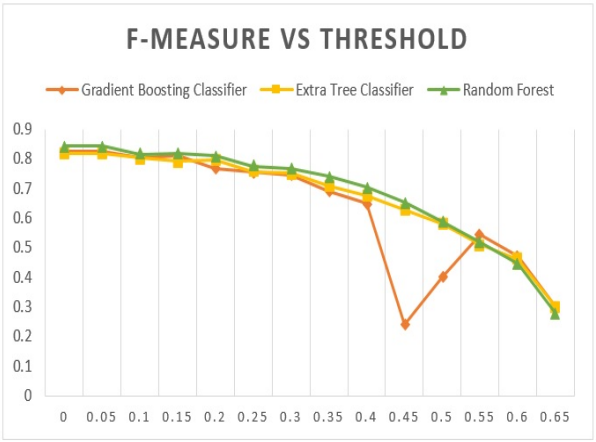

Fig. 8. F-measure vs Threshold-III

Moreover, increasing threshold also results in decreasing overall computations in measuring similarity scores because a higher threshold means fewer terms considered for measuring similarity between two documents.

This work can be further extended by testing with more similarity measures as features to our classifiers. Combining different conventional classifiers to detect near duplicate documents could also aid relevancy. Semantic and structural based similarity measure scores could also help our classifiers in better understanding similarities between documents of the input corpus.

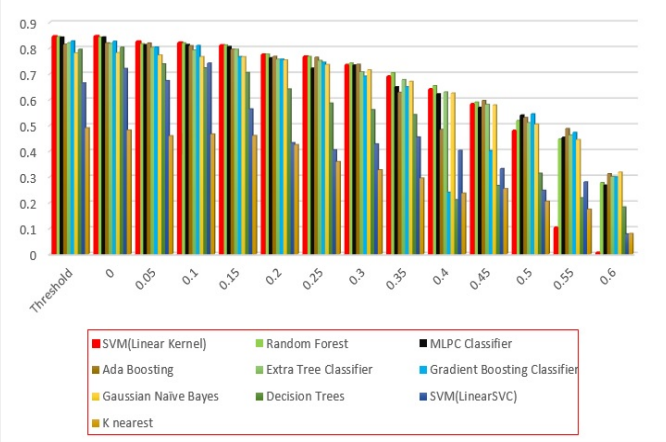

Fig. 9. F-measure vs Threshold for all classifiers

TABLE III
F1-SCORE AT DIFFERENT THRESHOLDS FOR DIFFERENT CLASSIFIERS

Classifier	Threshold				
	0.0	0.05	0.10	0.15	0.20
SVM(Linear)	0.8441	0.8453	0.82393	0.8197	0.8091
LinearSVC	0.6638	0.7200	0.6734	0.7410	0.5638
MLPC	0.8421	0.8424	0.8128	0.8133	0.8053
Random Forest	0.8439	0.8421	0.8162	0.8195	0.8113
Extra Trees	0.8200	0.8175	0.8020	0.7916	0.7945
Gaussian NB	0.7805	0.7813	0.7722	0.7663	0.7655
Decision Trees	0.7944	0.8025	0.7380	0.7224	0.7040
KNN	0.4894	0.4807	0.4587	0.4657	0.4595
Ada Boost	0.8140	0.8196	0.8187	0.8083	0.7948
Gradient Boost	0.8271	0.8257	0.8030	0.8091	0.7661

REFERENCES

[1] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schtze, An Introduction to Information Retrieval, Cambridge University Press, 2009

[2] Modern Information Retrieval, Ricardo Baeza-Yates and Berthier Ribeiro-Neto, Addison-Wesley, 2000.

[3] Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. (Eds.), Recommender Systems Handbook. 1st Edition., 2011, 845 p. 20 illus., Hardcover, ISBN: 978-0-387- 85819-7.

[4] Cross-Language Information Retrieval by By Jian-Yun Nie Morgan & Claypool Publisher series 2010.

TABLE IV
F1-SCORE AT DIFFERENT THRESHOLDS FOR DIFFERENT CLASSIFIERS

Classifier	Threshold				
	0.25	0.30	0.35	0.40	0.45
SVM(Linear)	0.7731	0.7655	0.7331	0.6887	0.6390
LinearSVC	0.4321	0.4041	0.4273	0.4541	0.4020
MLPC	0.7616	0.7213	0.7336	0.6487	0.6216
Random Forest	0.7758	0.7667	0.7411	0.7034	0.6534
Extra Trees	0.7561	0.7513	0.7070	0.6769	0.6289
Gaussian NB	0.7530	0.7356	0.7148	0.6699	0.6243
Decision Trees	0.6403	0.5854	0.5601	0.5409	0.2105
KNN	0.4241	0.3583	0.3265	0.2945	0.2360
Ada Boost	0.7670	0.7636	0.7365	0.6274	0.4832
Gradient Boost	0.7558	0.7442	0.6901	0.6489	0.2392

TABLE V
F1-SCORE AT DIFFERENT THRESHOLDS FOR DIFFERENT CLASSIFIERS

Classifier	Threshold			
	0.50	0.55	0.60	0.65
SVM(Linear)	0.5812	0.4777	0.1013	0.0039
LinearSVC	0.3313	0.2470	0.2797	0.0773
MLPC	0.5690	0.5394	0.4530	0.2670
Random Forest	0.5894	0.5179	0.4463	0.2767
Extra Trees	0.5807	0.5099	0.4614	0.3001
Gaussian NB	0.5782	0.5037	0.4438	0.3177
Decision Trees	0.2655	0.3132	0.2177	0.1820
KNN	0.2533	0.2045	0.1736	0.0788
Ada Boost	0.5957	0.5298	0.4867	0.3112
Gradient Boost	0.4007	0.5435	0.4718	0.2989

TABLE VI
MAXIMUM THRESHOLD FOR EACH CLASSIFIER AND CORRESPONDING F MEASURE

Classifier	**Threshold**	**F-measure**
SVM(Linear Kernel)	0.2	0.8091
Linear SVC	0.15	0.7410
Gaussian Naive Bayes	0.2	0.7655
K nearest	0.2	0.4595
MLPC	0.2	0.8053
Decision Tree	0.05	0.8025
AdaBoosting	0.2	0.7948
Random Forest	0.2	0.8113
Extra Tree	0.2	0.7945
Gradient Boosting	0.15	0.8091