Near-Duplicate Document Detection Using Semantic-Based Similarity Measure: A Novel Approach

Rajendra Kumar Roul and Jajati Keshari Sahoo

Abstract Detection of near-duplicate pages, especially based on their semantic content is a relevant concern in information retrieval. It is needed to avoid redundancy in the search results against a query as well as facilitate the ranking of the documents in the order of their semantic similarities. Although much work has been done in near-duplicate page detection based on content similarity (as evident in existing literature), the realm of semantic similarity provides a relatively unexplored pool of opportunities. In this paper, a novel technique is proposed to detect whether two documents belonging to a corpus have near-duplicate semantic content or not and a heuristic method is introduced to rank the documents based on their semantic similarity scores. This objective is achieved by examining the proposed technique for computing semantic-based similarity between two documents and applying an averaging mechanism to associate a similarity score to each document in the corpus. The empirical results on DUC datasets witness the effectiveness of the proposed approach.

Keywords Duplicate · Lemmatization · Near-duplicate · Ranking · Semantic

1 Introduction

Duplicate is an inherent problem of every search engine and it became an interesting problem in the late 1990s [1]. If two documents have similar content then they are considered as duplicates. Similarly, documents having minor modifications but are similar to a maximum range are known to be near-duplicates such as times-

R. K. Roul (✉)
Department of Computer Science, Thapar Institute of Engineering and Technology, Patiala 147004, Punjab, India
e-mail: raj.roul@thapar.edu

J. K. Sahoo
Department of Mathematics, BITS-Pilani, K. K. Birla Goa Campus, Zuarinagar 403726, Goa, India
e-mail: jksahoo@goa.bits-pilani.ac.in

H. S. Behera et al. (eds.), *Computational Intelligence in Data Mining*, Advances in Intelligent Systems and Computing 990,
https://doi.org/10.1007/978-981-13-8676-3_46

tamps, advertisements, and counters. In the recent years, near-duplicate document detection has received more attention due to the overwhelming requirements [2–6]. Near-duplicate documents are not bit-wise similar although they display striking similarities. It has been observed from a study which includes 238,000 hosts that 10% of these hosts are mirrored [7]. There are many standard techniques available which can easily identify the duplicate documents, but identifying the near-duplicate documents is a tough task. Hence, detecting and eliminating near-duplicate documents still remains unanswered [8]. There are various reasons why near-duplicate documents exist on the web and some of the reasons are (i) different interpretation of the same events (ii) while reprinting the documents, people may modify its content which also known as typographical errors (iii) main content of two documents are the same but the forum or blogs are replied differently, for example, two documents having different text size, font, bold types, etc., but having same content (iv) documents having different versions and plagiarism (v) due to reprinted noise. Semantic similarity is a metric defined over a set of documents where different mathematical tools are used to estimate the strength of the semantic relationship between concepts or topics [9]. To compute the similarity between a pair of documents, the semantic content or likeness of their meaning is compared which facilitate the classification of documents into topics. It also helps to rank the documents based on their similarity scores. The notion of distance between a pair of documents is most commonly used to characterize this metric.

Most of the research works have used content-based similarity to detect the near-duplicate documents. However, detection of near-duplicate documents using semantic similarity has much importance due to the semantic nature of the web. Researchers have focused on semantic-based work to detect near-duplicates, but ignore to rank the documents in order to find the percentage of similarities between the documents. The proposed approach not only detects the near-duplicates based on their semantic similarity but also ranks the documents to catch the percentage of similarities with respect to all the documents of a corpus. To achieve this, four well-known semantic techniques (Word2Vec, WordNet, Normalized Google Distance, and Latent Dirichlet Allocation (LDA)) are used for computing the similarity scores between pairs of documents in the corpus. These scores are used as the features for training different traditional classifiers to generate document pairs with semantic similarity scores. A heuristic technique is proposed to rank the documents by taking a weighted average of the features from the classifier and then computing an aggregate score of each document. Experimental results on four categories of DUC datasets show that the proposed approach is promising. Remaining part of the paper is as follows: Sect. 2 describes the detailed mechanism for detecting the near-duplicate documents. Results and analysis of the experiment are carried out in Sect. 3. The paper is concluded in Sect. 4 with some future enhancement.

2 Proposed Approach

The following steps are used to detect the near-duplicate documents and rank them in a given corpus P.

Step 1: *Preprocessing of the documents*:
Consider a corpus P having a set of classes $C = \{c_1, c_2, \ldots, c_n\}$ and documents $D = \{d_1, d_2, \ldots, d_p\}$. The documents are preprocessed which includes removal of stop words, stemming,[1] lemmatizating[2] and converting the documents to a Bag-Of-Words format where each word is stored along with its word count in a dictionary. Documents use different forms of a word, such as *operate, operating, operation* and *operative*. Stemming and lemmatization are done separately because we require the dictionary of stemmed words for Word2Vec and LDA analysis while the lemmatized dictionary of words is needed to perform NGD and WordNet techniques. Stemming collapses related words by a heuristic process that chops off the ends of the words resulting in a term which may not be present in the English dictionary. Lemmatization on the other hand, collapses the different inflectional forms of a word, resulting in a base term of a dictionary word. Then a new corpus of the documents is generated from the given dataset by pooling the preprocessed lists of documents together. Thus, two corpuses are created—one for the stemmed words and another for the lemmatized words.

Step 2: *Calculating the similarity scores*:
The semantic similarity between a pair of documents are calculated using four semantic techniques as discussed below:

i. *Word2Vec*:
It is used to make word embeddings by grouping similar words together in the vector space without human intervention. It captures the linguistic contexts of the words using two-layer neural network architecture. Its input is a text corpus and produces a vector space, i.e., feature vectors for words in the corpus. Word2vec generates the exact meaning of a word based on the usage and content of the dataset assigned to it [10]. These guesses are very useful to us as it can be used to establish associations between words (e.g., "man" map to "king", "woman" map to "queen") or cluster the documents and classify them. The vectors formed by word2vec are called *neural word embeddings*. These vectors are representations of the words which makes natural language computer readable and is shown in the Fig. 1. Word2vec makes word embeddings in two ways, namely *continuous bag of words (CBOW)* and *skip-gram* models [10]. The proposed approach used *CBOW* model and the reason for choosing *CBOW* over *skip-gram* model is that *CBOW* model is several times faster to train than the *skip-gram*, slightly better accuracy for the frequent words. To work on this model, first the word

[1] https://tartarus.org/martin/PorterStemmer/.

[2] https://algorithmia.com/algorithms/StanfordNLP/Lemmatizer.

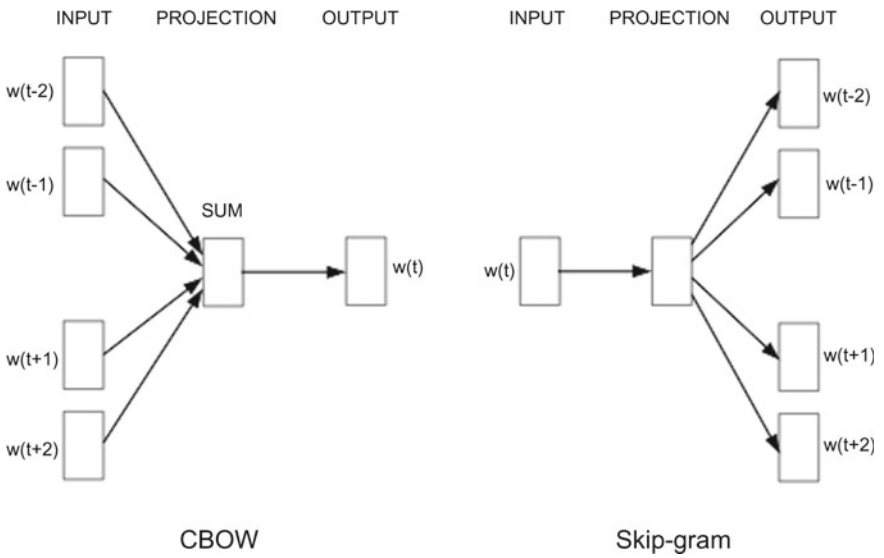

Fig. 1 Word2vec similarity

embeddings are formed using word2vec. The similarity between d_i and d_j is computed by considering the average of the maximum similarity between the words of the two documents d_i and d_j as shown in the Eq. 1.

$$word2vec(d_i, d_j) = \frac{\sum_{x \in d_i} max_{y \in d_j} cosine\_similarity(d_i, d_j)}{length\ of\ d_i} \quad (1)$$

ii. *WordNet*:
It is a huge lexical database of English where nouns, verbs, adverbs, and adjectives are clustered into sets of cognitive synonyms (synsets), each represents a different concept [11]. The main relationship among words in WordNet is synonymy, for example the synonymy that exists between "boat" and "ship" will be captured in WordNet. To calculate the semantic similarity score between two documents using WordNet, first the synsets are found from the words of each document. The synsets are then compared using the similarity measure provide by the WordNet and average of the maximum value for each synset is taken as the similarity score between the pair of documents. The threshold of similarity score between two documents is decided as 0.3 [3] Eq. 2 is used to calculate the WordNet similarity score numerically.

[3] decided experimentally for DUC-2001 dataset on which the better result is obtained and thresholds decided for different DUC datasets are different.

$$wordnet(d_i, d_j) = \frac{\sum_{x \in d_i} \max_{y \in d_j} similarity(synset(d_i), synset(d_j))}{length\ of\ d_i} \tag{2}$$

iii. *Normalized Google Distance (NGD)*:
NGD measures the distance between two terms by comparing the number of web hits returned by the Google search engine [12]. Here, the nouns (noun extraction from a document is done using NLTK[4]) of the documents are used for measuring the NGD value between them. The NGD value between two nouns n_p and n_q is calculated as follows:

$$NGD(n_p, n_q) = \frac{\max(\log H(n_p), \log H(n_q)) - \log H(n_p, n_q)}{\log T - min(\log H(n_p), \log H(n_q))}$$

where T represents the number of web documents retrieved by Google, $H(n)$ and $H(n_p, n_q)$ are the number of hits returned by Google for one noun and two nouns, respectively. The similarity between each pair of nouns is computed by subtracting the NGD from 1 (NGD lies between 0 and 1) as follows:

$$NGD\text{-}similarity(n_p, n_q) = 1 - NGD(n_p, n_q)$$

The similarity between two documents d_i and d_j having total number of nouns M_p and M_q, respectively is computed by counting the noun-noun pair (noun-noun pair is formed by taking a noun from d_i and another noun from d_j) whose NGD similarity values is above a certain threshold α (it is set to 0.61 as decided by experiment for which the better result is obtained) divided by total number of noun-noun pairs as given in the Eq. 3.

$$NGD\text{-}similarity(d_i, d_j) = \frac{count(NGD\text{-}simialrity(n_p, n_q)) > \alpha}{M_p \times M_q} \tag{3}$$

iv. *Latent Dirichlet allocation*:
LDA [13] is a huge lexical database of English where nouns, verbs, adverbs, and adjectives are clustered into sets of cognitive synonyms (synsets), each represent a different concept [14]. Hellinger Distance[5] is used to calculate the distance and similarity (1-Hellinger Distance). Hellinger distance is widely used to compute the distance between two probability distributions and used here for the LDA model.

Step 3: *Training the classifier*:
The document pairs are run on different traditional classification models after obtaining the semantic similarity scores for the above four semantic measures, which are used as the feature vector for the classifier. The dataset categorizes the documents into different folders, each belonging to a partic-

[4] https://www.nltk.org/.

[5] www.encyclopediaofmath.org/index.php?title=Hellinger_distance&oldid=16453.

Table 1 Similarity vectors with class label of a pair of documents

$$\begin{bmatrix} d_{12} =< ngd_{12}, word2vec_{12}, wordnet_{12}, lda_{12}, C_{12} > \\ \\ d_{13} =< ngd_{13}, word2vec_{13}, wordnet_{13}, lda_{13}, C_{13} > \\ \\ \ldots \\ \ldots \\ \ldots \\ d_{ij} =< ngd_{ij}, word2vec_{ij}, wordnet_{ij}, lda_{ij}, C_{ij} > \end{bmatrix}$$

ular topic based on semantic content. Hence, the label for classification is given as "1" if the pair of documents given as input belong to the same folder and "0" otherwise. In other words, different sets are contained in different folders with documents within the same folder as *near-duplicates* to each other while those in two different folders are *not near-duplicates*. The near-duplicate and not near-duplicate documents are classified into class "1" and class "0", respectively. Table 1 shows the similarity vectors between each pair of documents where d_{ij} represents documents d_i and d_j and C_{ij} is the class label for d_i and d_j.

Step 4: *Ranking the near-duplicate documents based on semantic similarity score*: After the classifiers get trained, the decision of that classifier is considered which gives the highest F-measure. The output can be interpreted as a feature matrix, which is a list of feature vectors for each near-duplicate document pair (as decided by the classifier). This feature matrix is used to find the ranking of near-duplicate documents. Now, the aggregate semantic similarity score between the pair of near-duplicate documents is computed by taking an *average (AVG)* of the scores obtained from the four semantic measures used for each document pair and it is shown in Eq. 4.

$$Semantic\text{-}similarity(d_i, d_j)_{AVG} = \frac{\sum_{i=1}^{4} a[i]}{4} \tag{4}$$

where, $a[i]$ is the score of each semantic algorithm between d_i and d_j. Alternatively, a *weighted average* of the scores from the four measures is decided as the aggregate similarity between each pair of documents. These weights are determined from the parameter thresholds set by the classifier for its input. We then compute the semantic similarity score of each document with the entire set of documents in the corpus individually, by averaging the aggregate similarity score of document d_i with all the other documents it is paired with and this gives us the composite similarity score of the document d_i. Equation 5 illustrates this calculation where N is the total number of documents in the corpus P.

$$Semantic\text{-}similarity(d_i)_{AVG} = \frac{\sum_{j \in P} Semantic\text{-}similarityty(d_i, d_j)_{AVG}}{N} \tag{5}$$

The documents are ranked in a descending order based on their semantic similarity scores and then top K documents having the highest similarity scores are picked up. Algorithm 1 illustrates the details.

Algorithm 1: Ranking of near-duplicate documents

Input: Feature matrix (FM) of near-duplicate documents and the corpus P
Output: Ranking of near-duplicate documents
$Semantic_{avg}[] \leftarrow \phi$ //stores the average of semantic similarity between two documents
$Net\text{-}semantic_{avg}[] \leftarrow \phi$ //stores the overall semantic average of a document
$final\_list \leftarrow \phi$
for all (d_i, d_j) pairs $\in$ FM **do**
  $Semantic_{avg}[d_i, d_j] \leftarrow$ avg. of the score of each semantic algorithm between d_i and d_j
end for
for all $d_i \in$ FM **do**
  for all $d_j \in P$ **do**
    $Net\text{-}semantic_{avg}[d_i] \leftarrow$ overall semantic average of d_i with respect to P
  end for
end for
//Ranking of near-duplicates documents
sort all $d_i \in P$ in descending order based on their $Net\text{-}semantic_{avg}[d_i]$ and select top K documents.
$final\_list \leftarrow$ top K documents
return $final\_list$

3 Experimental Analysis

Different DUC[6] (Document Understanding Conference) datasets are used for experimental purpose. DUC datasets follow a set of naming conventions to name their folders and subsequent documents within the folders. For the purposes of convenience, the folders are renamed in incremental order (from 1–30 (in DUC-2000, 2001, and 2002) and 1–50 (in DUC-2005)) and the documents in a similar fashion. The documents of each DUC dataset are preprocessed and two dictionary files are created from the documents, one containing the stemmed words and the other containing the lemmatized form of the words. To find the similarity scores between the documents, four semantic similarity techniques such as Word2Vec, WordNet, NGD, and LDA are used. For WordNet, top 30 terms from each document were selected to represent the document and a similarity score was found based on these 30 terms. WordNet is a computationally intensive task as it deals with a large dataset and it

[6] http://www.duc.nist.gov.

Table 2 Performances on DUC-2000

Classifier	DUC-2000			
	Precision	Recall	F-Measure	Accuracy
SVM	77.43	79.64	**78.52**	**85.68**
*k*NN	54.87	56.49	55.67	67.35
G-NB	60.97	62.38	61.67	73.28
M-NB	66.56	68.01	67.28	76.52
B-NB	60.29	56.47	58.32	65.35
DT	56.73	52.47	54.52	67.67
RF	68.30	70.49	69.38	80.35
GB	66.69	64.00	65.32	76.42
ET	70.56	70.48	70.52	79.37

takes a large amount of time, and hence to speedup the process we take a representative sample of terms. Similarly, NGD also has huge constraints and a mechanism is used to cache the results, enabling us to get the desired results in lesser time. A pickle file is created to run the document terms in batches and get their similarity values to increase the speed of the output. 80% of the documents from each DUC dataset are used as the training set to train the classifier while the remaining 20% is used as the test set. The classifiers used are SVM (Linear), *k*NN, Gaussian Naive-Bayes (G-NB), Multinomial Naive-Bayes (M-NB), Binomial Naive-Bayes (B-NB), Decision Trees (DT), Random Forest (RF), Gradient Boosting (GB), and Extra Trees (ET). From the results, one can notice that the performance of Linear SVM dominated all other classifiers for all DUC datasets. Tables 2, 3, 4 and 5 show the performance of various classifiers on different DUC datasets. The maximum F-measure and accuracy are marked as bold in the Tables. Figures 2 and 3 show the precision and recall of different classifiers on DUC datasets, respectively. Figs. 4 and 5 show the F-measure and accuracy of different classifiers on DUC datasets, respectively.

3.1 Hyper-Parameter Tuning

Linear SVM: Parameters tuned: C = 10
kNN: Parameters tuned: $k = 5$
Naive Bayes: Parameters tuned: Prior Probabilities = [0.75, 0.25]
Random Forest: Parameters tuned: n_estimators = 11, number of classifiers used:10
Gradient Boosting: Parameters: learning_rate = 0.1, n_estimators = 290, max_depth = 4, subsample = 0.4, random_state = 0, number of classifiers used:10
Extra Trees: Parameters: n_estimators = 20, learning_rate = 1, number of classifiers used:10

Table 3 Performances on DUC-2001

Classifier	DUC-2001			
	Precision	Recall	F-Measure	Accuracy
SVM	80.98	82.51	**81.74**	**92.89**
*k*NN	50.65	52.11	51.37	63.46
G-NB	61.27	65.61	63.37	72.71
M-NB	64.54	63.11	63.82	74.75
B-NB	58.34	56.72	57.52	68.32
DT	48.98	49.58	49.28	61.38
RF	65.51	61.64	63.52	73.43
GB	63.65	64.71	64.18	78.52
ET	59.45	63.22	61.28	74.38

Table 4 Performances on DUC-2002

Classifier	DUC-2002			
	Precision	Recall	F-Measure	Accuracy
SVM	81.89	79.86	**80.86**	**93.67**
*k*NN	51.64	55.21	53.37	70.03
G-NB	59.87	60.77	60.32	73.23
M-NB	61.87	62.67	62.27	77.27
B-NB	57.93	55.27	56.57	71.18
DT	55.87	55.25	55.65	72.23
RF	69.40	71.24	70.31	77.46
GB	61.26	63.72	62.47	73.38
ET	67.89	63.59	65.67	75.08

Table 5 Performances on DUC-2005

Classifier	DUC-2005			
	Precision	Recall	F-Measure	Accuracy
SVM	78.43	76.92	**77.67**	**95.46**
*k*NN	54.64	52.15	53.37	64.43
G-NB	65.87	61.05	63.37	68.83
M-NB	63.87	58.81	61.24	77.65
B-NB	54.92	54.44	54.68	70.38
DT	56.87	58.67	57.76	67.23
RF	75.34	73.89	74.61	77.46
GB	64.26	62.83	63.54	83.68
ET	62.39	69.76	65.87	81.48

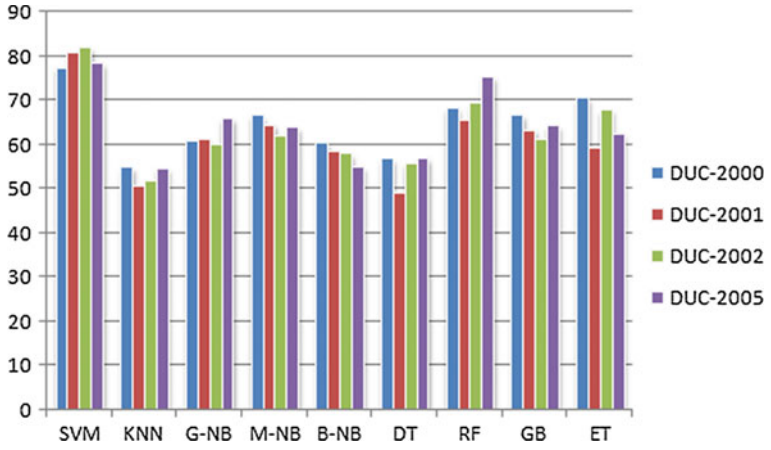

Fig. 2 Precision of different classifiers

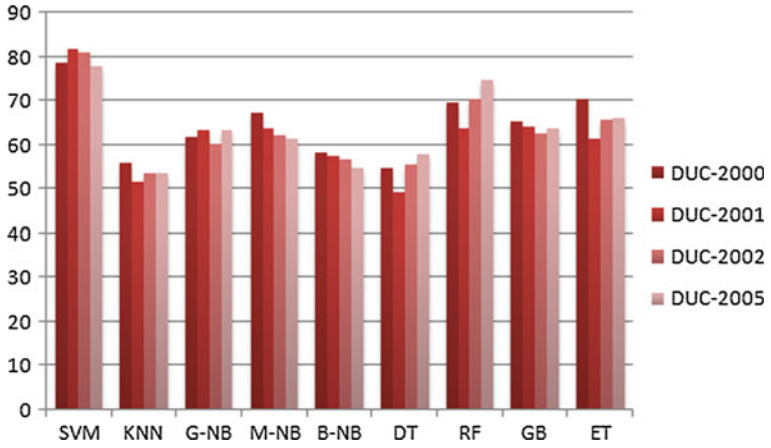

Fig. 3 Recall of different classifiers

3.2 DUC-2000 and DUC-2002 Datasets

For both datasets, the test folder is organized into subfolders (30 in this case). Each of those subfolders contains 9–11 documents having summaries pertaining to a specific topic. The documents within those subfolders contain 100–400 word summaries, and each of those documents have similar content and hence are duplicates. Each set having an average of 10 documents constituting a total of 296 documents and 43956 pairs of documents which are used to detect the near-duplicates. Out of 43956 pairs of documents, 1350 pairs of documents are not near-duplicates as they come from the same clusters (each set having 10 documents and hence, 45 pairs of documents.

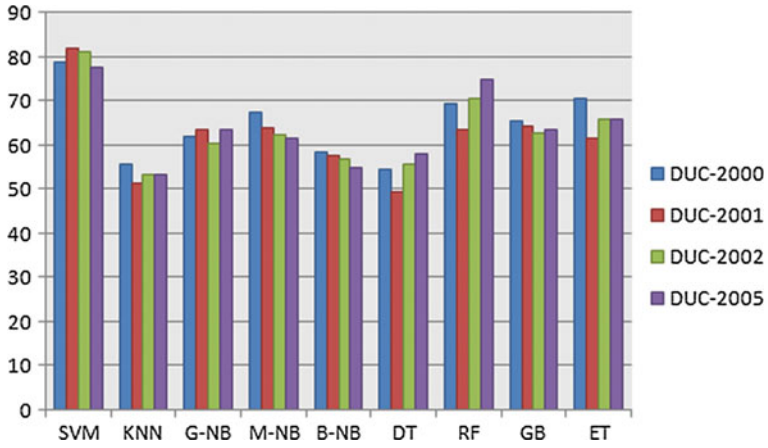

Fig. 4 F-measure of different classifiers

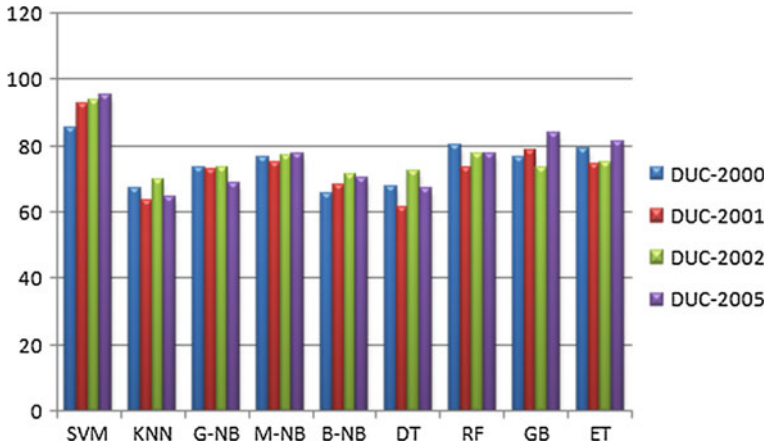

Fig. 5 Accuracy of different classifiers

Thus, 30 document sets have a total of 1350 pairs of documents). The remaining 42606 pairs of documents are used as near-duplicates.

3.3 DUC-2001 Dataset

This dataset has 30 sets of documents with sets defined by different types of criteria such as event set, opinion set, etc. Each set having around 10 documents constituting a total of 299 documents and 44551 pairs of documents which are used to detect the near-duplicates. Out of 44551 pairs of documents, 1350 pairs of documents are not

near-duplicates as they come from same clusters (each set having 10 documents and hence, 45 pairs of documents. Thus, 30 documents set have total of 1350 pairs of documents). The remaining 43201 pairs of documents are used as near-duplicates.

3.4 DUC-2005 Dataset

This dataset has 50 document sets. Each set having 25–50 documents constituting a total of 1503 documents and 1128753 pairs of documents which are used to detect the near-duplicates. Out of 1128753 pairs of documents, 23250 pairs of documents are not near-duplicates as they come from the same sets (each set having on an average 30 documents and hence, 465 pairs of documents. Thus, 50 documents set have total of 23250 pairs of documents). The remaining 1105503 pairs of documents are used as near-duplicates.

3.5 Parameters for Performance Evaluation

The following parameters are used to measure the performance of different classifiers.

i. *Accuracy (a)* is the ratio between the sum of true positive cases, *TP* (number of documents that are near-duplicate and are retrieved by the approach) and true negative cases, *TN* (number of documents that are not near-duplicate and are not retrieved by the approach) with the total number of documents, $N = TP + FP + TN + FN$. Eq. 6 shows how to compute the accuracy of a classifier.

$$a = \frac{TP + TN}{N} \tag{6}$$

where, *FP*: number of documents that are not near-duplicate and are retrieved by the approach and *FN*: number of documents that are near-duplicate and are not retrieved by the approach.

ii. *Precision (p)* is the fraction of the web documents suggested as near-duplicate (C_d) by the approach which are actually near-duplicate (A_d) and is shown in Eq. 7.

$$p = \frac{|A_d| \cap |C_d|}{|C_d|} \tag{7}$$

iii. *Recall (r)* is the fraction of total actual near-duplicate documents in the corpus which are suggested as near-duplicate by the approach and is shown in Eq. 8.

$$r = \frac{|A_d| \cap |C_d|}{|A_d|} \tag{8}$$

Table 6 Similarity scores on DUC-2001

Document name	Similarity score
SJMN91-06184003	0.5808
SJMN91-06184088	0.5909
SJMN91-06187248	0.5566
SJMN91-06254192	0.5626
SJMN91-06290185	0.5783
WSJ910702-0078	0.5385
WSJ910702-0086	0.5582
WSJ910910-0071	0.5457
WSJ911011-0071	0.5757
WSJ911016-0124	0.5466
WSJ911016-0126	0.5747
AP880908-0264	0.5134
AP890616-0192	0.5179
AP901213-0236	0.5133
LA012689-0139	0.5605

Table 7 Top 15 documents of DUC-2001

Rank	Document name	Similarity score
1	LA031790-0064	0.6070
2	WSJ920302-0142	0.4022
3	AP890901-0154	0.4120
4	AP880509-0036	0.4354
5	SJMN91-06130055	0.4139
6	AP900207-0041	0.4871
7	SJMN91-06324032	0.4682
8	AP880714-0187	0.4079
9	AP890123-0150	0.4492
10	FT931-10514	0.4901
11	AP880909-0135	0.4270
12	AP891005-0230	0.4031
13	FBIS4-23474	0.4375
14	AP881109-0149	0.4609
15	LA110490-0034	0.4555

Table 8 Semantic similarity scores on DUC-2001

First document name	Second document name	Word2Vec	WordNet	LDA	NGD
WSJ911004-0115	WSJ880510-0079	0.6557	0.3192	0.4409	0.5431
AP900816-0166	AP890127-0271	0.7493	0.4763	0.3023	0.4561
SJMN91-06267062	WSJ870804-0097	0.6558	0.4812	0.3023	0.3014
AP890527-0001	AP880316-0061	0.4997	0.5111	0.3023	0.3967
FT922-10715	FT924-12193	0.5309	0.4782	0.3023	0.4385
AP891005-0216	LA113089-0118	0.6245	0.3998	0.7415	0.5118
AP900514-0011	LA112389-0104	0.5308	0.4958	0.3828	0. 7103
SJMN91-06041121	SJMN91-06067177	0.7184	0.5008	0.4013	0.3311
SJMN91-06294205	AP890525-0127	0.7183	0.4506	0.4305	0.3431
SJMN91-06254192	AP891125-0090	0.7495	0.4640	0.3023	0.5643
AP880316-0208	AP891125-0090	0.5308	0.4436	0.3023	0.6113
AP891005-0230	WSJ891026-0063	0.2187	0.4237	0.9999	0.4667
SJMN91-06270171	SJMN91-06165057	0.6558	0.4103	0.3009	0.3001
LA102189-0067	AP881022-0011	0.5307	0.4193	0.7087	0.3144
WSJ910620-0075	AP900816-0139	0.6558	0.5466	0.3023	0.6113

For demonstration purposes, we have considered DUC-2001 dataset and shown some near-duplicate documents and their composite similarity scores in Table 6 and the top 15 near-duplicate documents and their composite similarity scores in Table 7. Table 8 shows four semantic similarity scores between the pair of documents of DUC-2001 dataset.

4 Conclusion

The paper proposed a methodology to identify the semantically similar near-duplicate documents of a corpus and a novel ranking approach to retrieve the top K near-duplicate documents from the corpus. Four semantic-based techniques are used in order to find the semantic similarity between two documents. The present work not only detects the near-duplicates based on their semantic similarity but also ranks the documents to catch the percentage of similarities with respect to all documents of a corpus. Experimental results on different DUC datasets show the stability and efficiency of the proposed approach. This approach can be optimized by developing a method to compute the weights from the features of the classifier in a better manner. The techniques used here are mathematically formalized and presented, yet the empirical results may not be the most optimized due to the properties of the dataset. Relying heavily on NGD for a large number of documents in retrospect is not a good idea because NGD requires continuous internet connectivity and has a considerable time overhead. When selecting the most dissimilar documents from the corpus, we need to make sure that this dissimilarity is among the documents included in the ranking and not the whole corpus. Hence, a dynamic threshold is needed to set the similarity of documents being added to the top-ranked list to ensure semantically relevant yet the diverse results. The selection of the threshold value needs to be done by coming up with a good heuristic with the help of domain knowledge. An alternate approach to rank the documents is to generate a query from the semantic content of the documents of the corpus and this query can be used to determine the similarity among the documents. Different techniques can be developed to identify and optimize such query. The proposed methodology also can be extended by introducing different deep learning techniques to identify the near-duplicates in the corpus and comparing their results with shallow learning classifiers.

References

1. Manber, U.: Finding similar files in a large file system. In: Winter USENIX Technical Conference. vol. 94, pp. 1–10 (1994)
2. Roul, R.K., Mittal, S., Joshi, P.: Efficient approach for near duplicate document detection using textual and conceptual based techniques. In: Advanced Computing, Networking and Informatics, Volume 1: Advanced Computing and Informatics Proceedings of the Second International Conference on Advanced Computing, Networking and Informatics (Icacni-2014). vol. 27, pp. 195–203. Springer, Berlin (2014)
3. Zhou, Z., Yang, C.-N., Chen, B., Sun, X., Liu, Q., QM, J.: Effective and efficient image copy detection with resistance to arbitrary rotation. IEICE Trans. Inf. Syst. **99**(6), 1531–1540 (2016)
4. Zhou, Z., Wu, Q.J., Sun, X.: Encoding multiple contextual clues for partial-duplicate image retrieval. Pattern Recognit. Lett. (2017)
5. Zhou, Z., Wu, Q.J., Huang, F., Sun, X.: Fast and accurate near-duplicate image elimination for visual sensor networks. Int. J. Distrib. Sens. Netw. **13**(2), 1–12 (2017)
6. Zhou, Z., Mu, Y., Wu, Q.J.: Coverless image steganography using partial-duplicate image retrieval. Soft Comput. 1–12 (2018)

7. Bharat, K., Broder, A.: Mirror, mirror on the web: A study of host pairs with replicated content. Comput. Netw. **31**(11), 1579–1590 (1999)
8. Manku, G.S., Jain, A., Das Sarma, A.: Detecting near-duplicates for web crawling. In: Proceedings of the 16th International Conference on World Wide Web, pp. 141–150 (2007)
9. Feng, Y., Bagheri, E., Ensan, F., Jovanovic, J.: The state of the art in semantic relatedness: a framework for comparison. Knowl. Eng. Rev. **32** (2017)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceeding of ICLR-2013, pp. 1–12 (2013)
11. Chua, S., Kulathuramaiyer, N.: Semantic feature selection using wordnet. In: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence. IEEE Computer Society, pp. 166–172 (2004)
12. Cilibrasi, R.L., Vitanyi, P.M.: The google similarity distance. IEEE Trans. Knowl. Data Eng. **19**(3), 1–15 (2007)
13. Blei, D.M., Ng, A.Y., Jordan, M.I: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
14. Girolami, M., Kabán, A.: On an equivalence between plsi and lda. In: Proceedings of the 26th annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pp. 433–434. ACM (2003)