Article | DOI: 10.2478/ijssis-2022-0002 Issue 1 | Vol. 15 (2022)

A novel approach to capture the similarity in summarized text using embedded model

Asha Rani Mishra* and
V.K. Panchal

Department of Computer Science,
Al Falah University, Faridabad,
Haryana, India.

*E-mail: asha1.mishra@gmail.com

The article was edited by
Ashutosh Sharma.

Received for publication
October 25, 2021.

Abstract

The presence of near duplicate textual content imposes great challenges while extracting information from it. To handle these challenges, detection of near duplicates is a prime research concern. Existing research mostly uses text clustering, classification and retrieval algorithms for detection of near duplicates. Text summarization, an important tool of text mining, is not explored yet for the detection of near duplicates. Instead of using the whole document, the proposed method uses its summary as it saves both time and storage. Experimental results show that traditional similarity algorithms were able to capture similarity relatedness to a great extent even on the summarized text with a similarity score of 44.685%. Moreover, degree of similarity capture was greater (0.52%) in case of use of embedding models with better text representation as compared to traditional methods. Also, this paper highlights the research status of various similarity measures in terms of concept involved, merits and demerits.

Keywords
Embedding models, Extractive text summarization, Near duplicate, Similarity measures, Text representation.

Introduction

Near duplicate documents are similar but not having identical content i.e., not identical bitwise (Xiao et al., 2008). In order to make searching faster, there is a need to remove duplication of content on the World Wide Web (WWW). The presence of near duplicates in text documents affects performance badly in the performance while integrating data from different sources. Several text extraction techniques—Topic Modeling, Key Phrase extraction and Text Summarization (Mishra et al., 2019) are available to fetch relevant information from unstructured text data. Different text extraction techniques can show different results even if applied in the same document. Text summarization generates concise and coherent summary from large pieces of text without any modification for preserving key contents in the original text. For text documents, the near duplicate detection task is more challenging. Even though there exists a proportion of the same words in two pairs of documents but in different order will not be considered as identical. Synonyms can be another important issue that needs to be addressed.

Traditional techniques like Bag of Words (BOW), Shingling, Hashing (MinHash and SimHash) are good to identify duplicate documents but not efficient for detection of near duplicates. Commonly used approaches for duplicate detection include shown in Table 1.

Available research mostly relates the task of detection of near duplicates as the detection of intermediate level of similarity and mostly similarity estimation is done by using statistical techniques like hashing, singling and signature based. With the help of recent Artificial Intelligence tools like Machine Learning, Deep Learning and Natural Language Processing, text embedding models can be used to generate vectors to capture more semantic similarity during similarity estimation. Text embedding models are used to capture semantics which are not often detected by commonly used approaches like shingling and hashing.

Table 1. Conventional near duplicate detection techniques.

Category	Approach	Characteristics	Merits
Keyword based	BOW (Bag of Words)	Comparing words and frequency of words with respect to other documents	Used in large documents uses Term Frequency -Inverse Document Frequency (TF-IDF) to create fingerprints. Reduces storage space
Fingerprint based	Shingling	Compares short phrases adding context to the word	Fingerprints are created with tokenized documents by using overlapped substrings and consecutive words. Statistical concepts are used to find near duplicates
	SimHash	Generate fixed length hashes for each document which are stored for duplication detection	Obtain ‘f’ bit fingerprint for each document. Used as dimension reduction
Hash based	MinHash	Phrases are hashed into numbers for comparison to identify duplication and content hashes are stored	It stores a small amount of information for each document for effective comparison
	Locality Sensitive Hashing (LSH)	Probabilistic approach to detect similar documents. Hash function generated similar hashes for similar shingles	Search space contains only those documents which tend to be similar which maximizes the probability of collision for similar content

Summary can be used to represent the whole document as it is generated by extracting relevant content, so it can be used for capturing similarity instead of working with the whole document which can save both time and storage.

A text summarization-based near duplicate detection approach with efficient text representation by using text embedded models is presented in this research. In the section “Text embedding in text representation and text similarity”, the role of text embedding in text representation and commonly used text similarity techniques is discussed. In the section “Related work”, related work about near duplicate detection, text embedding models and text summarization is elaborated. In the section “Proposed methodology”, the proposed approach is discussed. In the section “Experimental results and discussion” presents related experimental results followed by conclusion and future scope in the last section.

Text embedding in text representation and text similarity

Text similarity or similarity estimation is one of the active research trends nowadays that acts as a basis of various Natural Language Processing (NLP) tasks and play an important used in many research domains including detection of near duplicates as it plays important role in document matching (Wang and Dong, 2020). In order to label two entities as near duplicate in a quantitative manner, similarity function can used to measure whose value can range between the interval [0, 1]. Higher values of similarity score indicates more similarity. Any text similarity technique will first convert or map the input documents into vectors which contain real valued numbers. Next, suitable similarity measures can be applied on these vectors. Performance of text similarity algorithms considers two aspects—efficient text representation and choice of similarity measure function. Objective of text similarity algorithms is to determine commonness between two input documents as similarity scores are directly proportional to commonness. Traditional similarity measurement methods like statistical, corpus and knowledge based considers only text representation. In the traditional approach, the first way is to divide text into overlapping groups of sequential words called shingles. Similarity is considered or measured by the proportion of identical shingles found in the pair of text documents. In the second way, the vector of words is defined for representing a particular document and then similarity is computed by comparing the vectors. With the growth of modern Artificial Intelligence tools, semantic aspect integration can increase the efficiency of text representation techniques. Various text representation techniques are shown in Table 2.

Text embedding models represent words in the form of numeric values or vectors based on the context and order in a document. These models are used for text representation and can be utilized in finding similarity between documents (Khattak et al., 2019). Text embedding models can detect similarity even when it is mixed or modified. It maps each document to a low dimension and dense vector in a continuous vector space. While word embedding considers only the world, text embedding considers phrases/paragrams. It can be used in several ways while computing text similarity (Tan and Phienthrakul, 2019). Related words are closer in vector space. Various embedding models are listed in Table 3. Commonly used text similarity measurement techniques and various metrics whose value lies in the range of [0, 1] used in this regard are shown in Tables 4 and 5, respectively.

Related work

Pamulaparty et al. (2014): Research work involving initial pre-processing of documents includes stop word removal and stemming. Keywords generated are passed as an input to the Near Duplicate detection algorithm. Using a similar hash (SimHash) function with respect to various thresholds (<60%, 60–70%, 70–80%, > 80%) near duplicate documents are determined.

Pamulapartya et al. (2015): Proposed a framework for near duplicate document detection using machine learning models. In phase 1, fuzzy C means clustering is performed on the document before putting directly to the near duplicate which reduces the scope of comparison of the document. In phase 2 a discriminative function is used for classification exploiting the inherent features present in the documents

Table 2. Text representation techniques.

Text representation method	Concept used	Characteristics	Merits	Demerits
Vector Space Model	Word count/BOW model	It uses the concept of linear algebra to compute similarity	Simple to compute based on the frequency of words	Ignore the importance of rare words
Document vectors	TF-IDF vectors	It also computes the count of documents in which a particular word is present along its significance	It does not give importance to most frequent words in the document which does not contribute much in similarity computation	Does not consider the semantic aspect
Embedding model	Word embedding	These are the high dimensional representations of words	Handle words having similar meaning i.e., synonyms. Does not require any feature engineering	It cannot be applied directly in the computation of text similarity
Topic modeling	Latent Dirichlet Allocation (LDA)	Documents are represented by inherent latent topics where each topic can be drawn as probability of distribution of words	Probabilistic model, for defining feature matrix of a document based on semantics	Requires prior knowledge of the number of and it does not capture correlation

Table 3. Different embedding models for text representation (Khattak et al., 2019; Mishra et al., 2020).

Embedding model	Characteristics	Merits	Demerits	Variants
One hot encoding	Maps each word from vocabulary to unique index in vector space	Learn dense representation of words	Dependent on corpus knowledge	–
Word2Vec	Maps each word to a point in vector space E.g. Continuous Bag of Words (CBOW), Skip Gram	Used in Neural networks for predicting focus words as prediction-based models	Dimension is between 50 and 500. Context window is between 5 and 10	Doc2Vec paragraph2vec e.g., Distributed Memory Model of Paragraph Vectors (PV-DM), Paragraph Vector Continuous Bag of words (PV-CBOW)
GloVe	Term co-occurrence matrix based on vocabulary size is used	Minimized reconstruction error, captures larger dependency due to larger context window, Count based model	Order of dependencies are not preserved; performance depends on data type	GloVe with skip gram window
FastText	Sub words are also considered	Extends the functionality of Word2Vec skip gram to handle out of vocabulary (OOV) words	Longer time to train	Probabilistic FastText
Embedding from Language Models (ELMo)	Captures context at both word and character level. Same word can be used for different contexts	Performs sentence level embedding by using bidirectional Recurrent Neural Networks (RNN), can be used in transfer learning	Unable to use left to right and right to left context at the same time	–
Bidirectional Encoder Representations from Transformers (BERT)	Considers n bidirectional representations in unsupervised mode	It can be pre trained using one extra output layer	Random sentence is replaced by special tokens('Mask') to consider both left to right and right to left information at the same time	Robustly Optimized BERT Pre Training Approach (RoBERTa), A lite version of BERT(ALBERT), Encoder that Classifies Token Replacement Accurately'(LECTRA), Generalized Autoregressive Pre Training for Language Understanding (XLNet), Distilled version of BERT (DistilBERT), BERT for Summarization (BERTSUM)

Table 4. Categorization of Text similarity measurement techniques.

Text similarity measure	Category	Considers semantic?	Approach used	Characteristics
String based	Character based	No	Hamming Distance, Levenshtein distance, Damerau-Levenshtein, Needleman-Wunsch, Longest Common Subsequence, Smith-Waterman, Jaro, Jaro-Winkler and N-gram	Used to find typographical mistakes but less efficient text analytics and computationally less effective for large text documents. Used in String matching approximation
	Token/term based	No	Jaccard similarity Dice's coefficient Cosine similarity Manhattan distance and Euclidean distance	Useful in case of recognition of term rearrangement
Statistics based	Corpus/knowledge base	Yes	TF-IDF, (Latent Semantic Indexing (LSI)word2Vec, GloVe, Bidirectional Encoder Representations from Transformers (BERT), Latent Semantic Analysis (LSA), LDA	It uses only text representation and does not consider distance between texts

computed as weighted terms. A decision is made by function verifying the similarity vector created from features.

Yung-Shen et al. (2013): Proposed a method for detecting duplicate documents using three key components. First pre-processing on input document for feature selection. Highly weighted features are selected. Second similarity measure metrics are used for finding similarity degree between input and set of all pairs of documents. Third component is to learn a discriminant function using the Support Vector Machine (SVM) classifier.

Gali et al. (2016): Evaluated 21 measures to find similarity between two titles. Damerau-Levenshtein performed well by detecting changes in character/ token and real data. Smith-Waterman performed well in case of character change while Bi-Jaccard worked well for both character/token and real data.

Hassanian-esfahania and Karga (2018): Due to the unordered nature of sets, the MinHash algorithm does not cover all near duplication properties. Even though the count of shared attributes in the documents is more, position of attribute also matters. A MinHash algorithm (min-wise) is proposed to enhance the data structures of traditional MinHash algorithms for better representation of near duplications. This approach showed an unbiased estimate of Jaccard coefficient with less variance.

Feng and Wu (2015): In this paper, authors improved the work of Wang and Chang (2009) by using a suffix tree for comparing two documents instead of using fixed sized sliding windows. By using the suffix tree all possible pairs of identical sentences were found. Also, they add a validation step by comparing selected terms at specified patterns in all matched sentences. The algorithm "SL + ST" (sentence length +Suffix tree) is compared with SpotSigs and 3 Shingles.

Rodier and Carter (2020): In this paper, authors proposed an online system to detect near duplicate documents on the dataset of web-based news articles by adapting the shingling algorithm (Broder, 2000). Further they used this system in an application where situational awareness tool to increase the efficiency of human analysts. This system works in two phases- In the first phase, it determines whether a new document is near duplicate of previously processed document. Each document is represented as a sketch consisting of a set of 8-byte numbers. For two similar documents, it will generate sets of 8-byte numbers that overlap proportionality to their similarity. This method results in very high precision scores with increased recall and F1 scores.

Hajishirzi et al. (2010): In this paper, authors proposed an algorithm for near duplicate document detection in which each document is represented as a k-gram (sparse) vector. Weight of the vector is learned

Table 5. Popular Text similarity metrics (Pamulaparty et al., 2014, 2015; Gali et al., 2016; Yung-Shen et al., 2013).

Similarity measurement method	Highlights
Euclidean distance	Consider the distance of text in vector form. Uses frequency of tokens to generate feature vectors
Cosine	Consider the angle between two vectors. Fails to capture variations of the representation for unstructured/semi structured text
Manhattan	Consider the distance between two real vectors
Hamming	Consider the count of positions in which two bits are different. Binary strings must be of the same length
Jaccard distance	Compute's length of two strings and then finds common characters to indicate the presence in near locations. Transposition in reverse order is performed to find matching characters between two strings
Jaro Winkler	It extends the Jaro distance metric by a prefix value (p = 0.1). This provides a higher value of weights to the strings having common prefix length whose value lies in the range of (Xiao et al., 2008; Khattak et al., 2019)
Cosine similarity with k shingles/k gram	Shingling the document means considering consecutive words and grouping as a single entity. A more general approach is to shingle the document. This takes consecutive words and groups them as a single object. In general, the set of all 1-shingles represents the' bag of words' model
TF-IDF	Based on the concept of term frequency (TF) which is the count of occurrence of a token in a document. The inverse document frequency (IDF) is the way to find the relevance of unique or odd words. Cosine similarity with TF-IDF is used to find similarity scores
Normalized Levenshtein	Based on the minimum number of edit operations
Soft-TFIDF	TF-IDF and Jaro Winkler are combined to measure similarity. First Jaro Winkler finds pairs of tokens common to both strings and then TF-IDF is used to find similarity scores exceeding the suitable value of threshold set in Jaro Winkler

to optimize for similarity functions (cosine or Jaccard coefficient) which are further mapped to hash values by using the technique of locality sensitive hashing. These hash values are used as document signatures and contribute to calculating similarity. News articles and email messages are used as target domains. This method was found to be more accurate than Shingles and I match.

Arun and Sumesh (2015): In this paper, four phase sentence level features, word mapping technique, term document weighting scheme and modified similarity technique is used which gives improved precision and recall.

Yandrapally et al. (2020): A study of near duplicate algorithms based on state pairs is presented for web app model inference. Webpages were divided into three categories-clone, near duplicate and distinct. Threshold values were systematically computed and used by 10 near duplicate detection techniques for three different domains.

Pamulaparty et al. (2017): Proposed random forest method random forest- Streaming Random Forest (SRF) and Oblique Random Forest (ORF) showed better accuracy as compared to other algorithms while detecting near duplicates in context of web crawling. Keyword extraction, URL indexing and similarity computation were the three phases to distinguish between near duplicate and non-duplicate web pages.

Do and LongVan (2015): Proposed an algorithm for detection of near duplicates in articles by extracting key phrases based on ontology and matching signatures. Similarity is calculated between extracted key phrases. A set of characteristic key phrases present in the articles were used to find near duplicates. Proposed algorithm showed good precision and recall.

Al-Subaihin et al. (2019): Analysed different text representation techniques for mobile application in order to describe textual content, Vector Space Model (VSM) using TF-IDF with frequency weighting combined with Latent semantic Indexing (LSI) were used. This was compared with other text feature extraction techniques like topic modeling. Results showed that cluster quality by topic modelling approach were more favourable as it captures more similarity.

Jain et al. (2017): Proposed a text summarization approach in which extractive text summary is generated by calculating similarity score between the abstractive summary and original sentences of text data using neural network approach for feature extraction.

El-Kassas et al. (2021): Explained different applications, approaches (Extractive, Abstractive and Hybrid), methods used in these approaches, building blocks—text summarization operations, text representation models and statistical and linguistic features. Also, it discusses various datasets, automatic evaluation tools.

Hendre et al. (2021): Highlights the relevance of semantic similarity while analysing text data by using the approach of a neural embedding model for text data representation. Sentence embedding models-Elmo, Glove and Google Sentence Encoder were used to combine with TF-IDF and Jaccard similarity for experimental purpose. ELMO and Google Sentence Encoder showed best results by capturing maximum similarity.

Albalawi et al. (2020): Provides a detailed description of applications, methodology and tools for topic modelling which is used for finding important topics present in the short text like comments, reviews and short length text messages. A comparison of five topic modelling methods-Latent Semantic Analysis (LSA), LDA, Non-Matrix Factorization (NMF), Principal Component Analysis (PCA) and Random projection on the basis of standard statistical evaluation metrics -Precision, Recall, F Score and topic coherent were established on two textual datasets. LDA and NMF topic modelling methods produced valuable output by extracting more meaningful topics.

Alqahtani et al. (2021): In order to generate patterns from text efficiently several processes like text mining, clustering, natural language processing and text similarity are involved. String based tools are suitable for lexical similarity. LCS, Jaro, and N-gram, Damerau Levenshtein (character-based algorithm) and the Cosine similarity, Euclidean Distance, Jaccard similarity, Block Distance, and Matching Coefficient (term-based algorithm) are popular techniques to measure lexical similarity. LSA is a popular corpus-based technique which is not suitable for nonlinear text distribution. WordNet is based on a semantic network and is based on a knowledge tool.

Chandrasekaran and Mago (2021): Semantic textual similarity is one of the most challenged NLP tasks. Measuring semantic similarity techniques can be knowledge, corpus, and deep neural network or can use hybrid-based techniques. Knowledge based include Edge counting methods (LCS), Feature based method (WordNet), Information content, Word embedding based (GloVe, FastText, BERT, word2vec), corpus based include LSA, Hyperspace Analogue to Language (HAL), Explicit Semantic Analysis (ESA), Word-Alignment models, Latent Dirichlet Allocation (LDA), Normalised Google Distance (NGD), Dependency-based models, Kernel-based models, In addition to this methods, deep learning based model includes Convolutional Neural Networks (CNN), Long Short Term Memory (LSTM), Bidirectional Long Short Term Memory (Bi-LSTM), and Recursive Tree LSTM which can be used to measure semantic similarity.

Roul and Sahoo (2020): Semantic content based near duplicate detection is one of the relevant research aspects in information retrieval as it avoids redundancy in the search results during query processing and removal of near duplicate pages improves page ranking. Authors proposed a novel method for the detection of near duplicate documents in a corpus on the semantic similarity score. A heuristic based method is used to rank the documents according to their semantic similarity scores. This has been achieved by applying an averaging method on DUC datasets which associates a similarity score to each individual document in the corpus based on semantic content. Effectiveness of the proposed method was concluded based on the empirical results performed. To achieve this, Word2Vec, WordNet, Normalized Google Distance, and Latent Dirichlet Allocation (LDA)) are used for computing the similarity scores between pairs of documents in the corpus. The computed score is used as features for training classifiers to generate document semantic similarity scores for document pairs. Experiments showed improved performance on DUC datasets.

Mansoor et al. (2020): Proposed a deep learning-based method to compute semantic similarity by using Long Term Short Memory (LSTM) which is an explicit type of Recurrent Neural Networks (RNN) to capture sequence among different elements in a sentence

Table 6. Recent research studies on text similarity and representation.

Concept/algorithm/ method used	Author(s)	Usage
Text similarity (SimHash, MinHash), Text clustering	Pamulaparty et al., 2014, 2015, 2017) Hassanian-esfahania and Karga (2018)	Near Duplicate detection on the basis of keywords generated from text, Fuzzy C means clustering with discriminant function, Random forest method for classification of near duplicates
Text similarity	Yung-Shen et al. (2013) Gali et al. (2016)	Near Duplicate detection on the basis of 21 similarity metrics computation between a pair of documents or two titles
Signature based text similarity measurement	Mohammadi and Khasteh, 2020 (Hajishirzi et al., 2010)	Reference texts are generated using genetic algorithms to obtain signatures for text documents as a sequence of 3 grams for detection of duplicate and near duplicate documents. For generating signature cosine text similarity measure is used on the datasets on CiteseerX, Enron and Gold Set of Near-duplicate News Articles
Text similarity	Do and LongVan (2015)	Near Duplicate detection by applying signatures generated based on ontology on extracted key phrases
Text representation methods	Al-Subaihin et al. (2019), Mishra (2019)	TF-IDF combined with LSI for topic modeling, spam classification
Text mining, clustering, natural language processing and text similarity	Alqahtani et al. (2021)	Text matching methods
Semantic similarity	Chandrasekaran and Mago (2021)	Any NLP task which involves semantic textual similarity
Semantic similarity	Roul and Sahoo (2020)	Near Duplicate detection of web pages on DUC dataset
Deep learning based semantic similarity	Mansoor et al. (2020)	Sentence similarity using LSTM and CNN per trained with word2vec on Quora dataset
Text representation using ELMo model	Peters et al. (2018)	Question answering, Textual entailment, semantic role labelling, Named entity extraction, sentiment analysis
Text representation using FastText model	Shashavali et al. (2019)	In goal oriented conversational agents (Chabot)
Text similarity based on distance	Stefanovič et al. (2019)	Plagiarism detection
Semantic similarity for short text based on corpus, knowledge and deep learning model	Han et al. (2021)	Text classification and text clustering, sentiment analysis, information retrieval, social networks plagiarism detection on the dataset
Text classification based on text embedding method	Li and Gong (2021)	Deep Learning Text classification on the dataset Sohu news dataset
Text Similarity based on text distance and text representation	Wang and Dong (2020)	Information retrieval, Machine translation, question answering, machine, document matching

Text representation using BERT model	Wang et al. (2019)	Extractive-Abstractive Text summarization with BERT embedding model with Reinforcement Learning on CNN/Daily Mail dataset and DUC2002
Word Embedding Model, Text classification, Word tagging	Ajees et al. (2021) Alqrainy and Alawairdhi (2021)	SVM classification to classify animate nouns for Malayalam text, comprehensive tag for Arabic language
Lexical Taxonomy	Nazar et al. (2021)	Elimination of incorrect hypernym links, taxonomy with new relations in Spanish, English and French

combined with Convolutional Neural Networks (CNN) for extracting local features. Proposed model used word2vec for text representation. Experiments carried out in Quora dataset showed better F score, precision and recall as compared to traditional text similarity methods (Naïve Bayes, Decision Tree, CNN, LSTM with word2vec and LSTM with GloVe).

Peters et al. (2018): Introduced a deep context-based learning model for word representation. Word vectors are internal states of a deep bidirectional model. In this model, each token acts as a function for the entire input sentence with the help of bidirectional LSTM and ELMo model. Word representation using ELMo model where higher-level LSTM states capture context aspects of words while lower-level state model aspects of syntax Performance of the model was analysed across six NLP challenging tasks including question answering, which showed reduction of relative error in a range of 6–20% over other models.

Shashavali et al. (2019): Proposed a method for measuring sentence similarity score using weighted N-gram, sliding window, cosine similarity and FastText embedding techniques. Improved results with accuracy, precision and recall by 6%, 2% and 80%, respectively, were obtained as compared to Universal Sentence encoder technique. Proposed work performs well for small training dataset. Concept of sliding windows were used as cosine similarity with weighted average word embedding does not perform well while computing sentence similarity between short and long sentences.

Stefanovič et al. (2019): Proposed a method to calculate similarity between two texts using word level n-gram to form a bag of n-gram combined with self-organising map (SOM). For evaluation Dice, Cosine, Overlap and extended Jaccard similarity measures were considered. N gram frequency is used to generate a frequency matrix of a dataset (A corpus of plagiarized short answers). Highest similarity was captured by using overlap measure.

Han et al. (2021): Presented a survey based on semantic similarity measurement for short text. The study categorizes the techniques into three categories–Corpus based (LSA, LDA, word2Vec, para2Vec, VSM), knowledge based (shortest path, Resnik, ESA) and deep learning based (CNN, LSTM, BERT).

Li and Gong, (2021): Used four embedding models i.e., word2Vec, doc2Vec, TF-IDF and embedding layer for text classification on Chinese news dataset. Deep Learning models (CNN, LSTM, GRU, MLP, 2 layer GRU, CNNGRU and CNNGRU_Merge, TextCNN) are used for classification purposes. The 2-layer GRU model with word2Vec embedding showed highest accuracy.

Wang et al. (2019): Proposed text summarization technique combining both extractive and abstractive approaches. In order to capture semantic features, a BERT text embedding model is used. Important sentences are selected from the input sentences (corpus). Next abstractive based summary is used for generating summary. For this, two sub models (both extractive and abstractive) and for updating in end-to-end training, reinforce learning. Proposed method achieved better accuracy.

Ajees et al. (2021): A machine learning based deep level tagging is used to provide more context to each noun and verb words for any Malayalam words. Two methods are combined for this-word embedding which uses word2Vec with skip gram variant and suffix stripping SVM classification to identify animate noun identification. This method exploits morphological features of the input text document.

Table 6 highlights various recent research studies for text mining tasks including near duplicate detection which uses text similarity measurement techniques and text embedding models for text representation.

A critical look at the available literature reveals that the following issues need to address:

1. Need to reduce the summarized latency in text summarization tasks.
2. Need to generate an open summarized framework since existing work is mostly domain specific.
3. Need to increase the accuracy of framework for capturing similarity with the help of emerging AI tools.

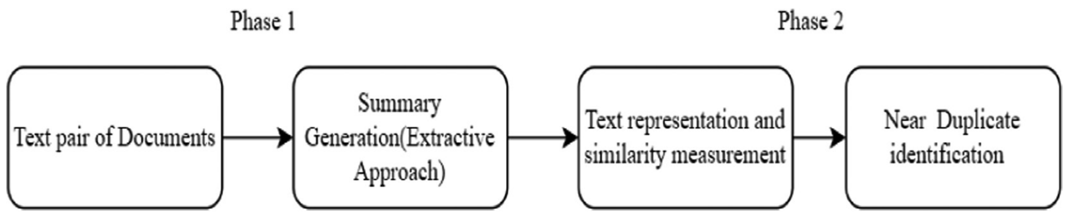
Phase 1
Phase 2
Text pair of Documents
Summary Generation(Extractive Approach)
Text representation and similarity measurement
Near Duplicate identification

Figure 1: Block diagram for proposed approach.

4. Since efficient summary can be generated with proper feature representation and better semantic understanding with the help of advanced AI tools, it can play an important role for detection of near duplicates by taking summarized text as input with an objective of reducing both time and storage.

Proposed methodology

In the proposed approach, similarity metrics are applied to find the degree of relatedness on summarization. For generating text summary, the LSA method as an extractive text summarizer is considered. For better semantic aspect, text embedding models are used for better vector representation. Extractive text summarization is a technique used in various domains of text analytics to extract meaningful textual content by keeping only important sentences without any modification in the original content. Figure 1 shows a generic approach for detecting near duplicates in two input pairs of text. For better utilization of time and storage while performing near duplicate detection the first summary of original content is generated. Moreover, to capture semantic similarity, a text embedding model is applied on a summary generated before applying a suitable text similarity algorithm for calculating similarity scores on the vector representation of text. Detailed working approach is shown with the help flowchart in Figure 2.

Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm 4 presents complete details about the various phases and sequence of concepts involved in the proposed method.

Algorithm 1: Near duplicate detection using summarized text

```
1. document_set := {Text 1, Text 2}, threshold := ø // Initialize
2. function Near_Duplicate_Detection(document_set)
      Input: Pair of text documents
      returns labeled documents as near duplicate or non-duplicate
3. output_set=Generate_ Summary(document_set) ; // Phase 1: Generation of summary
4. vector_set = Generate_ vector(output_set) ; // Phase 2: Text representation
5. similarity_score=calculate_similarity_score(vector_set; // Similarity score calculation
6. if similarity_score > ø then // comparison with threshold
7.      label 'Near Duplicate'
8. else
9.      label ' Non Duplicate'
10. end function
```

Algorithm 2: Generation of summary for the input documents present in document_set using Extractive approach

```
1. function Generate_ Summary(document_set)
      Input: pair of text documents
      returns generated summary
2. forall text document in document_set do
3. Pre-processing: Block level breaking of text into key phrases or sentences, Tokenization (sentences), Lemmatization, stemming, stop word removal, POS tagging, Named Entity Recognition
4. Identification of interrelated sentences: Similarity measuring functions are used to find related sentences to be included in the summary
5. Weighting and ranking of selected sentences: Numeric values are assigned to find important features. Higher ranked sentences are selected for summary
6. output_set:= {text 1_summary, text 2_summary};
7. return output_set // pair of summarized text
```

Algorithm 3: Text representation using embedding model to generate vectors

```
1. function Generate_ vector(output_set)
       Input: Pair of summarized text documents
       returns vector representation for input document pairs
2. forall summarized text document in output_set do
3. vector_set = embedding_model(output_set);
4. vector_set={$V_{Text1}$, $V_{Text2}$};
5. return vector_set // pair of vectors
```

Algorithm 4: Similarity score calculation for summarized text vectors

```
1. function calculate_similarity_score (vector_set)
       Input: pair of vectors
       returns similarity scores of the summarized text documents
2. similarity_score = similarity_function(vector_set)
3. return similarity_score
```

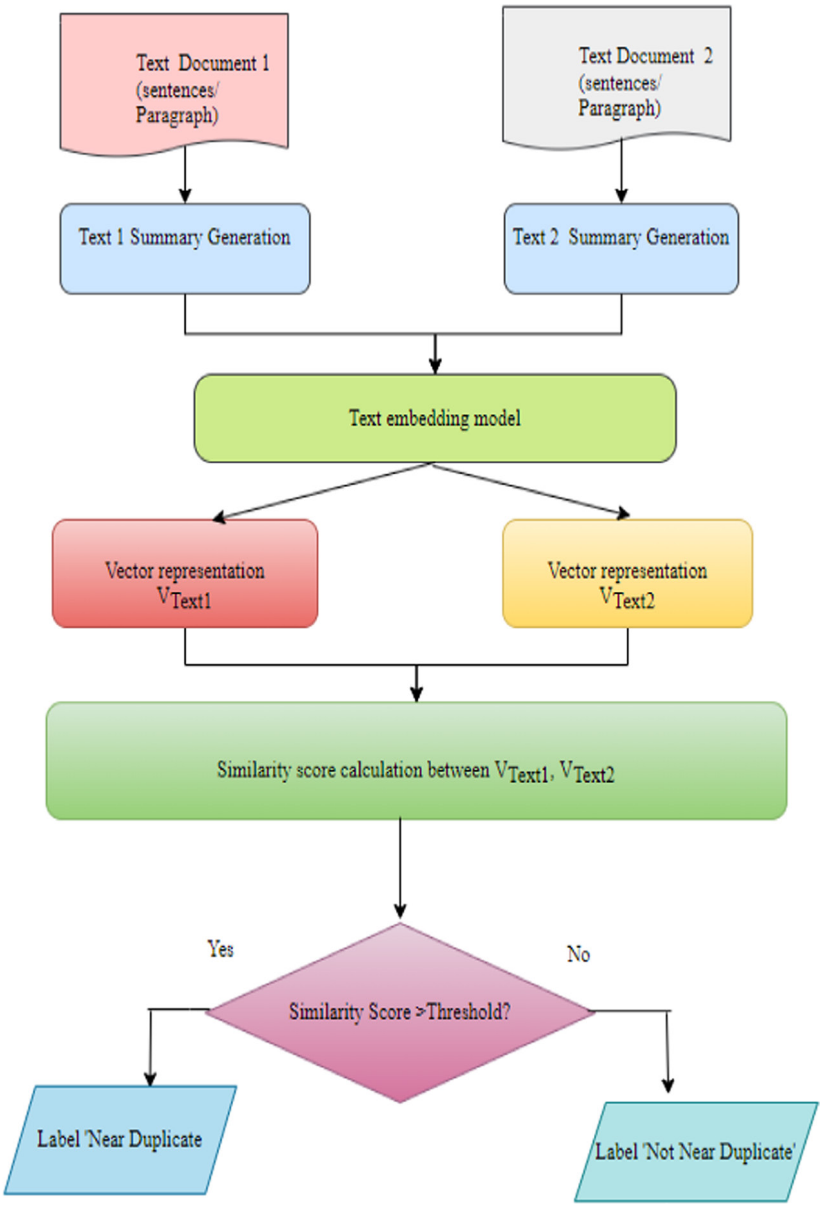
Text Document 1 (sentences/ Paragraph)
Text Document 2 (sentences/ Paragraph)
Text 1 Summary Generation
Text 2 Summary Generation
Text embedding model
Vector representation V_{Text1}
Vector representation V_{Text2}
Similarity score calculation between V_{Text1}, V_{Text2}
Yes
No
Similarity Score >Threshold?
Label 'Near Duplicate
Label 'Not Near Duplicate'

Figure 2: Workflow of proposed approach of near duplicate detection.

Experimental results and discussion

For experimental purposes, abstracts of research articles (Elrefaiy et al., 2018; Mishra et al., 2019) as Text 1 and Text 2 are considered as shown in Table 7. Table 8 shows the result of text summarization technique which is applied for generation of text summary of input documents, LSA which is based

Table 7. Input texts.

Input	Original text
Text 1	“Everyday large volume of data is gathered from different sources and are stored since they contain valuable piece of information. The storage of data must be done in efficient manner since it leads in difficulty during retrieval. Text data are available in the form of large documents. Understanding large text documents and extracting meaningful information out of it is time-consuming tasks. To overcome these challenges, text documents are summarized in with an objective to getrelated information from a large document or a collection of documents. Text mining can be used for this purpose. Summarized text will have reduced size as compare to original one. In this review, we have tried to evaluate and compare different techniques of Text summarization.”
Text 2	“In the view of a significant increase in the burden of information over and over the limit by the amount of information available on the internet, there is a huge increase in the amount of information overloading and redundancy contained in each document Extracting important information in a summarized format would help a number of users. It is therefore necessary to have proper and properly prepared summaries. Subsequently, many research papers are proposed continuously to develop new approaches to automatically summarize the text. ''Automatic Text Summarization" is a process to create a shorter version of the original text (one or more documents) which conveys information present in the documents. In general, the summary of the text can be categorized into two types: Extractive-based and Abstractive-based. Abstractive-based methods are very complicated as they need to address a huge-scale natural language. Therefore, research communities are focusing on extractive summaries, attempting to achieve more consistent, non-recurring and meaningful summaries. This review provides an elaborative survey of extractive text summarization techniques. Specifically, it focuses on unsupervised techniques, providing recent efforts and advances on them and list their strengths and weaknesses points in a comparative tabular manner. In addition, this review highlights efforts made in the evaluation techniques of the summaries and finally deduces some possible”

Table 8. Text summarization on original text.

Text summarization (using LSA method) on	Generated summary
Text 1	“Everyday large volume of data is gathered from different sources and are stored since they contain valuable piece of information. The storage of data must be done in efficient manner since it leads in difficulty during retrieval. To overcome these challenges, text documents are summarized in with an objective to get related information from a large document or a collection of documents.”
Text 2	“In the view of a significant increase in the burden of information over and over the limit by the amount of information available on the internet, there is a huge increase in the amount of information overloading and redundancy contained in each document. Specifically, it focuses on unsupervised techniques, providing recent efforts and advances on them and list their strengths and weaknesses points in a comparative tabular manner. In addition, this review highlights efforts made in the evaluation techniques of the summaries and finally dedtices some possible future trends.”

on extractive text summarization is used. For better vector representation of text, text embedding models are used which act as function parameters for similarity calculation. For analysing the performance of text similarity functions with embedding models, we have considered 6 models—Word2Vec,Universal Sentence Encoder, FastText, ELMo, GloVe and BERT. Similarity score is calculated using various similarity functions on both original and summarized text with and without embedding models. To get detailed insights more, similarity functions are applied on both original and summarized text on other text extraction strategies like topic modelling and key phrase extraction. Table 9 and Table 10 show topics generated when LDA method is applied on both original and summarized text pair respectively. It can be easily interpreted that high weighted topics from original text are included as topics in the summary also. Table 11 shows key phrases generated using TF-IDF method.

Table 12 shows values of similarity scores generated when various text similarity functions based on various traditional distance based metrics are applied on original pair of text, topics modelling, key phrase extraction and summary. Figure 3 shows similarity values generated by extractive approaches

Table 9. Topic modeling on original text.

Topic modelling (using LDA method) on	Topics with weights
Text 1	Topic #1 [(‘different’, 1.06), (‘since’, 1.03), (‘data’, 0.97), (‘try’, 0.88), (‘evaluate’, 0.88), (‘technique’, 0.88), (‘review’, 0.88), (‘summarization’, 0.88)]
	Topic #2 (‘text’, 1.42), (‘document’, 1.39), (‘large’, 1.16), (‘form’, 1.01), (‘available’, 1.01), (‘summarize’, 0.91), (‘information’, 0.9), (‘meaningful’, 0.85)]
Text 2	Topic #1 [(‘information’, 1.24), (‘summary’, 1.1), (‘summarize’, 1.05), (‘research’, 1.0), (‘amount’, 0.9), (‘increase’, 0.9), (‘help’, 0.84), (‘would’, 0.84)]
	Topic #2 [(‘text’, 1.36), (‘based’, 1.34), (‘provide’, 1.08), (‘extractive’, 1.07), (‘summarization’, 1.07), (‘abstractive’, 1.06), (‘technique’, 1.02), (‘summary’, 1.01)]

Table 10. Topic modeling on summary of original text.

Topic modelling (using LDA method) applied on	Topics with weights
Text 1 Summary	Topic #1 [(‘document’,0.091),(‘data’,0.065),(‘information’,0.065), (‘piece’,0. 039)’,’(‘contain’,0.039), (’summarize’,0.039), (’manner’, 0.039), (‘do’,0.039), (‘must’,0.039), (‘large’, 0.039)]
	Topic #2 [(‘document’,0.044), (‘information’, 0.044), (‘data’,0.044), (‘source’, 0.044), (‘different,’0.043), (‘valuable’,0.043), (‘lead’, 0.043), (‘challenge’, 0.043), (‘collection’, 0.043), (‘relate’, 0.043]
Text 2 Summary	Topic #1 [(‘information’,0.056),(‘increase’,0.040),(‘effort’,0.040), (‘amount’,0.040), (‘technique’,0.040),(‘specifically‘,0.024), (‘unsupervised‘,0.024),(‘future’,0.024), (‘overload’,0.024),(‘comparative’, 0.024)]
	Topic #2 [(‘information’,0.027), (‘technique’, 0.027), (‘amount’,0.027), (‘effort’, 0.026), (‘increase’,026), (‘possible’, 0.026), (‘redundancy’,0.026), (‘make’,0.026), (‘summary’,0.026), (‘strength’, 0.026)]

Table 11. Key phrase extraction on Text 1 and Text 2 using weighted TF-IDF method.

Key phrase extraction method applied on	Key phrases with weights
Text 1	[('form', 0.5769999999999996), ('large documents', 0.5769999999999996), ('text data', 0.5769999999999996),('large text documents', 0.5769999999999996), ('meaningful information', 0.5769999999999996), ('time-consuming tasks', 0.5769999999999996), ('different techniques', 0.5769999999999996), ('review', 0.5769999999999996), ('text summarization', 0.5769999999999996), ('different sources', 0.4759999999999998)]
Text 2	[('prepared summaries', 1.0), ('abstractive-based methods', 0.7069999999999996), ('huge-scale natural language', 0.7069999999999996), ('documents', 0.6670000000000004), ('summary', 0.6320000000000001), ('types', 0.6320000000000001), ('elaborative survey', 0.5769999999999996), ('extractive text summarization techniques', 0.5769999999999996), ('review', 0.5769999999999996), ('many research papers', 0.5340000000000003)]

Table 12. Similarity scores using traditional similarity metrics on original texts, topics, keyword extracted and summary.

Text similarity measure	Similarity score (in %) between Text 1 and Text 2	Similarity score (in %) between topics of Text 1 and Text 2	Similarity score (in %) between key word extracted of Text 1 and Text 2	Similarity score (in %) between Text 1 and Text 2 Summary
Euclidean distance [ED]	23.70	22.40	20.03	15.36
Normalized Levenshtein [NL]	27.80	26.43	33.69	29.08
Hamming Distance [HD]	40.0	7.14	10.8	27.0
Term Frequency-Inverse Document Frequency [TF-IDF]	53.71	55.90	38.86	41.11
Jaccard Distance [JD]	56.23	38.2	42.75	48.97
Cosine Similarity [CS]	63.0	29.46	30.15	41.86
Jaro Winkler [JW]	68.0	76.8	70.0	72.80
Cosine similarity with k shingles [CS_kshingles]	89.0	62.5	61.92	81.30

almost matches the scores when same algorithm is applied in original text.

Table 13 shows results generated when text embedding models are used to generate vectors for similarity calculation. Figure 4 shows better text representations resulting in better similarity score even when it is applied on summarized text. Figures 5 and 6 show graphical comparison and similarity distribution based on similarity scores using both traditional and embedding model approaches respectively on both original text pair and its summary.

From the above experimental details, it can be seen that in traditional similarity measures Jaro Winkler performs best in all three-text extraction

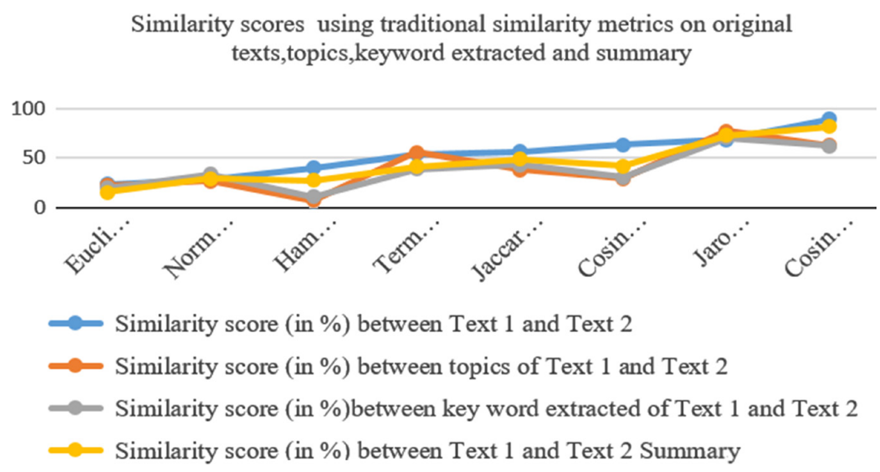

Figure 3: Similarity scores for the various text extraction methods.

Table 13. Similarity scores using text embedding models on original and summarized document.

Embedding model	Similarity score (in %) between Text 1 and Text 2	Similarity score (in %) between Text 1 summary and Text 2 summary
Word2Vec	5.28	14.26
Universal Sentence Encoder [USE]	81.36	69.39
FastText with soft cosine similarity [FT_SoftCS]	81.76	92.40
ELMo with cosine similarity (ELMo_CS)	88.59	76.32
Glove with cosine similarity (GloVe_CS)	97.89	95.60
BERT with cosine similarity (BERT_CS)	72.28	82.29

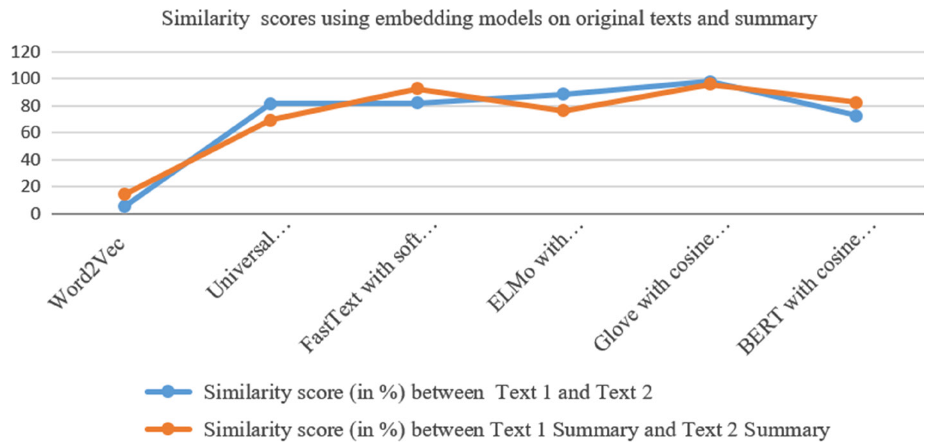

Figure 4: Impact of text representation on similarity calculation.

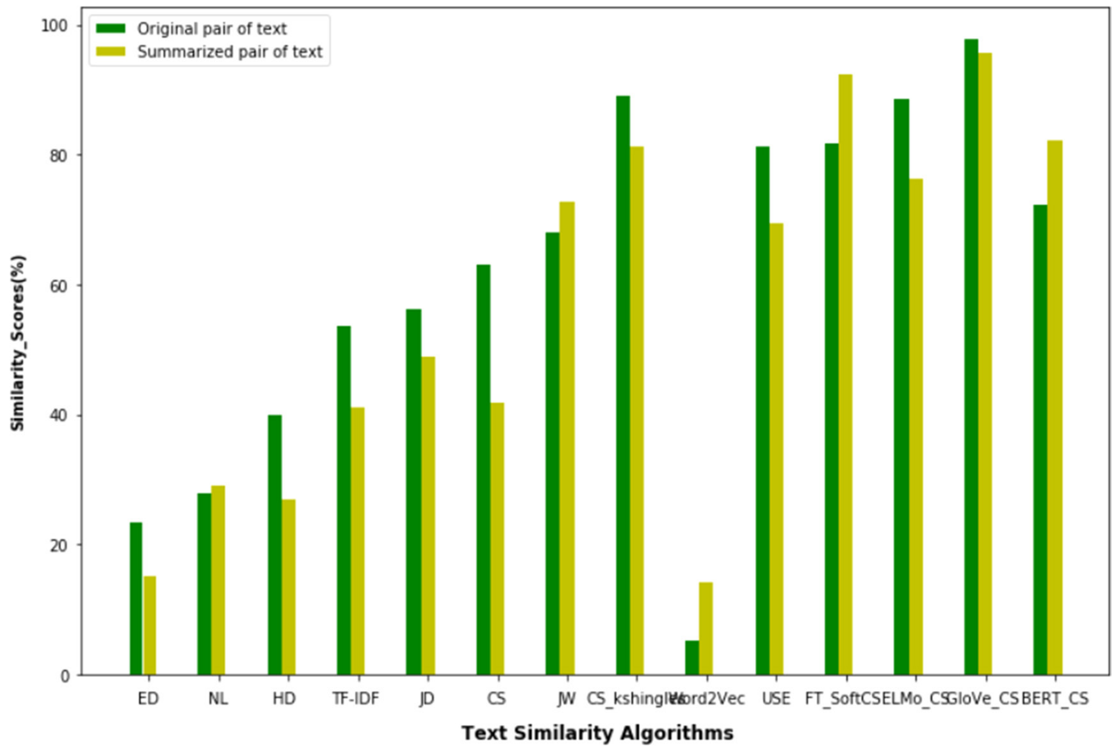

Figure 5: Graphical representation of similarity scores using various similarity measure techniques.

approaches i.e., topic modelling, keyword extraction and text summary generation as compared to other approaches shown in Table 12. Use of embedding models provides efficient text representation which leads to enhancing the performance of similarity algorithms as shown in Table 13. Soft cosine similarity using FastText [SoftCS_FT] performs best as text representation technique summarized text, while GloVe with cosine similarity captures the highest degree of similarity in both original and summarized text shown

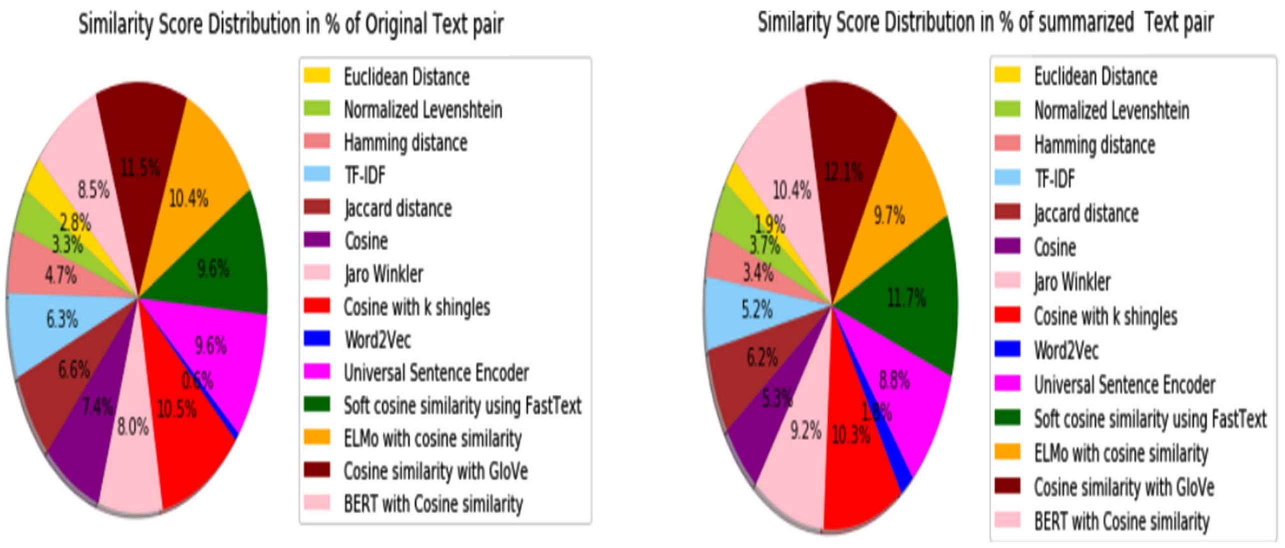

Figure 6: Similarity Score distribution using Various Similarity Search Techniques on original and summarized text.

Table 14. Result analysis.

	Similarity function	Original text	Summarized text
Without embedding model	Jaro Winkler [JW]	68	72.80
	Cosine similarity with k shingles [CS_kshingles]	89.0	81.30
With embedding model	Soft cosine similarity using FastText [FT_SoftCS]	81.76	92.40
	Cosine similarity with GloVe (GloVe_CS)	97.89	95.60

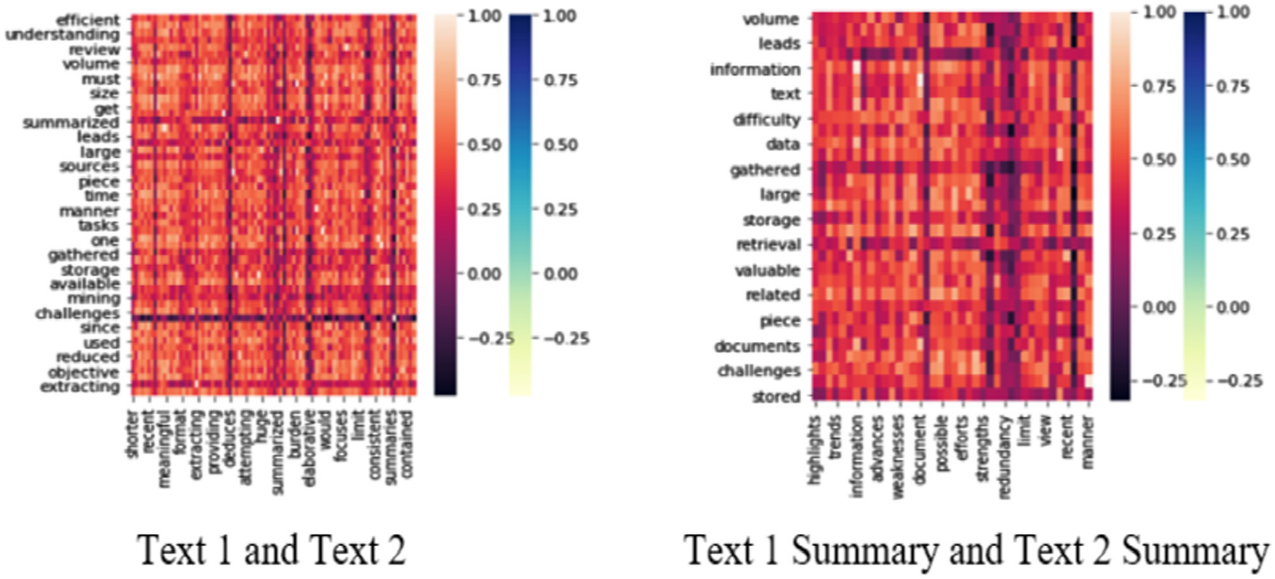

Figure 7: Heat map (GloVe) using both approaches.

Table 15. Analysis of impact of embedding models on Text similarity measurement.

No. of Text similarity algorithms	Approach used	Average similarity score (in %) between Text 1 and Text 2	Average similarity score (in %) between Text 1 summary and Text 2 summary	Difference (in %)
8	Without text embedding models	52.68	44.685	7.995
6	With text embedding models	71.19	71.71	0.52

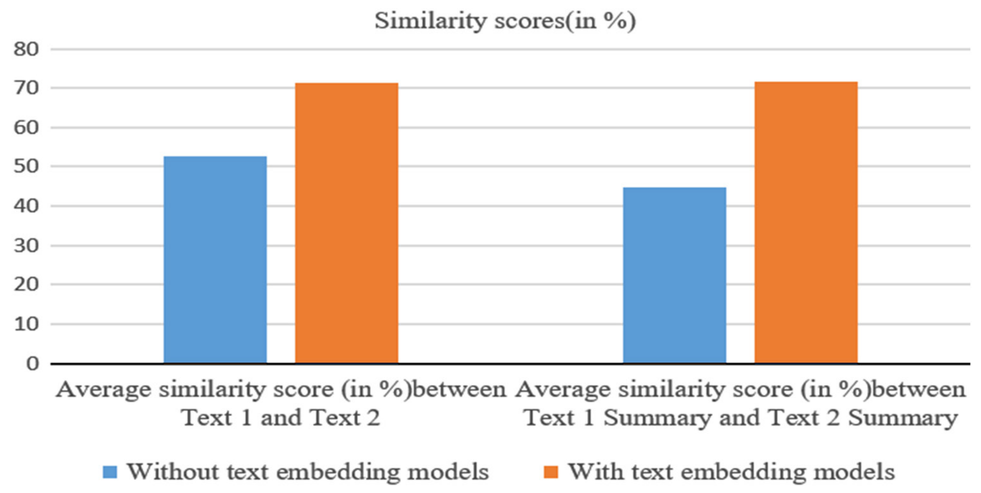

Figure 8: Comparison of similarity score of original vs. summarized text.

in Table 14. Heat map for GloVe embedding model is graphically represented in Figure 7. Table 15 highlights overall analysis for the proposed methodology. Figure 8 shows a graphical comparison of similarity scores for both original and summarized text combined with and without text embedding model.

Conclusion and future scope

Extractive approach of text summary generation is used to make the proposed approach independent of domain knowledge. So, in this paper an attempt has been made to use this concept to design and develop a near duplicate detection algorithm. Proposed approach performs reasonably well even for a higher value of threshold (>50%). Based on results obtained by the proposed method, it is possible to consider summary instead of whole document along with text embedding's to capture better similarity, as results shows average similarity score of 6 summarized embedded text results in an increase of 0.52%. By using a suitable embedding model this percentage can increase by considerable value as word2Vec performance was poor.

The functionality of the text summarization algorithm can be increased by adding other coherent elements such as synonym, antonymy, collocation, calculation, similarity and the element of transformation. In terms of results, the syntax of sentences to work more efficiently should be more mathematical and linguistic. The integration method consists of Grammatical and Lexical Linking within the text as well as a sentence containing a sentence and provides important details. In future operations alternatives may be used in an invisible way which creates an internal semantic representation and use of native language generation strategies for making a summary. In the future, Deep Learning can be used for developing generalized text embedding models to handle insufficient data and adding a deeper level context to POS tagging. Also abstractive text summarization can be used which generates summary on the basis of hidden text.

Literature Cited

Ajees, A. P., Abrar, K. J., Sumam, M. I. and Sreenathan, M. 2021. A deep level tagger for malayalam, a morphologically rich language. *Journal of Intelligent Systems* 30(1): 115–129.

Albalawi, R., Yeap, T. H. and Benyoucef, M. 2020. Using topic modeling methods for short-text data: a comparative analysis. *Frontiers in Artificial Intelligence* 3. Available at: https://doi.org/10.3389/frai.2020.00042.

Alqahtani, A., Alhakami, H., Alsubait, T. and Baz, A. 2021. A survey of text matching techniques. *Engineering, Technology & Applied Science Research* 11(1): 6656–6661. doi: 10.48084/etasr.3968.[1].

Alqrainy, S. and Alawairdhi, M. 2021. Towards developing a comprehensive tag set for the arabic language. *Journal of Intelligent Systems* 30(1): 287–296.

Al-Subaihin, A., Sarro, F. and Black, S. 2019. Empirical comparison of text-based mobile apps similarity measurement techniques. *Empirical Software Engineering* 24: 3290–3315.

Arun, P. R. and Sumesh, M. S. 2015. Near-duplicate web page detection by enhanced TDW and simHash technique. 2015 *International Conference on Computing and Network Communications (CoCoNet'15)*, December 16–19, Trivandrum.

Broder, A. 2000. Identifying and Filtering Near-Duplicate Documents. In Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, Montreal, Canada, pp. 1–10.

Chandrasekaran, D. and Mago, V. 2021. Evolution of semantic similarity—a survey. *ACM Computing Surveys* 54(2): 1–37, doi: 10.1145/3440755.[2].

Do, N. and LongVan, H. 2015. Domain-specific key-phrase extraction and near-duplicate article detection based on ontology. *The 2015 IEEE RIVF International Conference on Computing & Communication Technologies—Research, Innovation, and Vision for Future (RIVF)*, pp. 123–126, doi: 10.1109/RIVF.2015.7049886.

El-Kassas, W. S., Salama, C. R., Rafea, A. A. and Mohamed, H. K. 2021. Automatic text summarization: a comprehensive survey. *Expert Systems with Applications* 165: 113679.

Elrefaiy, A., Abas, A. R. and Elhenawy, I. 2018. Review of recent techniques for extractive text summarization. *Journal of Theoretical and Applied Information Technology* 96(23): 7739–7759.

Feng, J. and Wu, S. 2015. "Detecting near-duplicate documents using sentence level features", In Chen, Q., et al. (Eds), *DEXA 2015, Part II, LNCS 9262* Switzerland: Springer International Publishing; pp. 195–204, doi: 10.1007/978-3-319-22852-5_17.

Gali, N., Mariescu-Istodor, R. and Fränti, P. 2016. Similarity measures for title matching. *2016 23rd International Conference on Pattern Recognition (ICPR) Cancún Centre*, Cancún, December 4–8.

Han, M., Zhang, X., Yuan, X., Jiang, J., Yun, W. and Gao, C. 2021. A survey on the techniques, applications, and performance of short text semantic similarity. *Concurrency and Computation: Practice and Experience* 33(5), doi: 10.1002/cpe.5971.

Hajishirzi, H., Yih, W. and Kołcz, A. 2010. Adaptive near-duplicate detection via similarity learning. SIGIR'10, Geneva, July 19–23.

Hassanian-esfahania, R. and Kargar, M. -J. 2018. Sectional MinHash for near-duplicate detection. *Expert Systems with Applications* 99: 203–212.

Hendre, M., Mukherjee, P., Godse, M. 2021. Utility of neural embeddings in semantic similarity of text data. In Bhateja, V., Peng, S. L., Satapathy, S. C. and Zhang, Y. D. (Eds), *Evolution in Computational Intelligence. Advances in Intelligent Systems and Computing* 1176. Springer, Singapore, Available at: https://doi.org/10.1007/978-981-15-5788-0_21.

Jain, A., Bhatia, D. and Thakur, M. K. 2017. Extractive text summarization using word vector embedding. 2017 International Conference on Machine Learning and Data Science (MLDS), pp. 51–55, doi: 10.1109/MLDS.2017.12.

Khattak, F. K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C. and Rudzicz, F. 2019. A survey of word embeddings for clinical text. *Journal of Biomedical Informatics X* 4:100057.

Li, S. and Gong, B. 2021. Word embedding and text classification based on deep learning methods. *MATEC Web of Conferences* 336(3): 06022, doi: 10.1051/matecconf/202133606022.

Mansoor, M., Ur Rehman, Z., Shaheen, M., Khan, M. A. and Habib, M. 2020. Deep learning based semantic similarity detection using text data. *Information Technology and Control* 49(4): 495–510, doi: 10.5755/j01.itc.49.4.27118.

Mishra, A. R. 2019. Impact of feature representation on supervised classifiers—A comparative analysis. *Global Sci-Tech* 11(2): 69–74.

Mishra, A. R., Panchal, V. K. and Kumar, P. 2019. Extractive text summarization—an effective approach to extract information from Text. *2019 International Conference on contemporary Computing and Informatics (IC3I)*, Singapore, pp. 252–255, doi: 10.1109/IC3I46837.2019.9055636.

Mishra, A. R., Panchal, V. K. and Kumar, P. 2020. "Similarity Search based on Text Embedding Model for detection of Near Duplicates". *International Journal of Grid and Distributed Computing* 13(2): 1871–1881.

Mohammadi, H. and Khasteh, S. H. 2020. A fast text similarity measure for large document collections using multireference cosine and genetic algorithm. *Turkish Journal of Electrical Engineering Computer Sciences* 28(2): 999–1013.

Nazar, R., Balvet, A., Ferraro, G., Marín, R. and Renau, I. 2021. Pruning and repopulating a lexical taxonomy: experiments in Spanish, English and French. *Journal of Intelligent Systems* 30(1): 376–394.

Pamulaparty, L., Rao, C. V. G. and Rao, M. S. 2014. A near duplicate detection algorithm to facilitate document clustering. *International Journal of Data Mining & Knowledge Management Process (IJDKP)* 4(6): 39–49, doi: 10.5121/ijdkp.2014.4604 39.

Pamulapartya, L., Rao, C. V. G. and Rao, M. S. 2015. XNDDF: towards a framework for flexible near-duplicate document detection using supervised and unsupervised learning. *International Conference on Intelligent Computing, Communication & Convergence (ICCC-2014), Procedia Computer Science* 48: 228–235.

Pamulaparty, L., Rao, C. V. G. and Rao, M. S. 2017. Critical review of various near-duplicate detection methods in web crawl and their prospective application in drug discovery. *International Journal of Biomedical Engineering and Technology* 25(2/3/4): 212–226.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. 2018. Deep contextualized word representations. arXiv:1802.05365.

Rodier, S. and Carter, D. 2020. Online near-duplicate detection of news article. *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, 11–16 c European Language Resources Association (ELRA), Marseille, May 11–16, pp. 1242–1249, licensed under CC-BY-NC.

Roul, R. K. and Sahoo, J. K. 2020. Near-duplicate document detection using semantic-based similarity measure: a novel approach. *Advances in Intelligent Systems and Computing* 990: 543–558.

Shashavali, D., Vishwjeet, V., Kumar, R., Mathur, G., Nihal, N., Mukherjee, S. and Patil, S. V. 2019. Sentence similarity techniques for short vs variable length text using word embeddings. *Computación y Sistemas* 23(3): 999–1004.

Stefanovič, P., Kurasova, O. and Štrimaitis, R. 2019. The N-grams based text similarity detection approach using self-organizing maps and similarity measures. *Applied Sciences (Switzerland)* 9(9): 1870, doi: 10.3390/app9091870.

Tan, T. and Phienthrakul, T. 2019. Sentiment classification using document embeddings trained with cosine similarity. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pp. 407–414.

Wang, J. H. and Chang, H. C. 2009. Exploiting Sentence-level Features for Near-duplicate Document Detection. In Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology (AIRS09), Sapporo, Japan, Springer: Berlin/Heidelberg, Germany, pp. 205–217.

Wang, J. and Dong, Y. 2020. Measurement of text similarity: a survey. *Information* 11(9): 421.

Wang, Q., Liu, P., Zhu, Z., Yin, H., Zhang, Q. and Zhang, L. 2019. A text abstraction summary model based on BERT word embedding and reinforcement learning. *Applied Sciences (Switzerland)* 9(21): 4701, doi: 10.3390/app9214701.

Xiao, C., Wang, W., Lin, X. and Yu, J. X. 2008. Efficient Similarity Joins for Near DuplicateDetection" WWW2008, April 21–25, Beijing, ACM 78-1-60558-085-2/08.

Yandrapally, R. K., Stocco, A. and Mesbah, A. 2020. Near-duplicate detection in web app model inference. ICSE '20, May 23–29, Seoul, Republic of Korea, ACM, New York, NY, May 23–29, 12pp. Available at: https://doi.org/10.1145/3377811.3380416.

Yung-Shen, L., Ting-Yi, L. and Shie-Jue, L. 2013. Detecting near-duplicate documents using sentence-level features and supervised learning. *Expert Systems with Applications* 40(5): 1467–1476.