Sentence Similarity Estimation for Text Summarization Using Deep Learning

Sheikh Abujar, Mahmudul Hasan and Syed Akhter Hossain

Abstract One of the key challenges of natural language processing (NLP) is to identify the meaning of any text. Text summarization is one of the most challenging applications in the field of NLP where appropriate analysis is needed of given input text. Identifying the degree of relationship among input sentences will help to reduce the inclusion of insignificant sentences in summarized text. Result of summarized text always may not identify by optimal functions, rather a better summarized result could be found by measuring sentence similarities. The current sentence similarity measuring methods only find out the similarity between words and sentences. These methods state only syntactic information of every sentence. There are two major problems to identify similarities between sentences. These problems were never addressed by previous strategies provided the ultimate meaning of the sentence and added the word order, approximately. In this paper, the main objective was tried to measure sentence similarities, which will help to summarize text of any language, but we considered English and Bengali here. Our proposed methods were extensively tested by using several English and Bengali texts, collected from several online news portals, blogs, etc. In all cases, the proposed sentence similarity measures mentioned here was proven effective and satisfactory.

Keywords Sentence similarity · Lexical analysis · Semantic analysis
Text summarization · Bengali summarization · Deep learning

S. Abujar (✉) · S. A. Hossain
Department of Computer Science and Engineering, Daffodil International University, Dhanomondi, Dhaka 1205, Bangladesh
e-mail: sheikh.cse@diu.edu.bd

S. A. Hossain
e-mail: aktarhossain@daffodilvarsity.edu.bd

M. Hasan
Department of Computer Science and Engineering,
Comilla University, Comilla 3506, Bangladesh
e-mail: mhasanraju@gmail.com

A. J. Kulkarni et al. (eds.), *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, Advances in Intelligent Systems and Computing 828, https://doi.org/10.1007/978-981-13-1610-4_16

1 Introduction

Text summarization is a tool that attempts to provide a gist or summary of any given text automatically. It helps to understand any large document in a very short time, by getting the main idea and/or information of entire text from a summarized text. To produce the proper summarization, there are several steps to follow, i.e., lexical analysis, semantic analysis, and syntactic analysis. Possible methods and research findings regarding sentence similarity are stated in this paper. Bengali language has very different sentence structure and analyzing those Bengali alphabets may found difficult in various programming platforms. The best way of starting for preprocessing both Bengali and English sentences, initially need to convert into Unicode [2]. Sentence could be identified in a standard form, and it will help to identify sentence or words structure as needed. The degree of measuring sentence similarity is being measured by method of identifying sentence similarity as well as large and short text similarity. Sentence similarity measures should state information like: If two or more sentences are either fully matched in lexical form or in semantic form, sentence could be matched partially or we could found any leading sentence. Identifying centroid sentence is one of the major tasks to accomplish [1]. Few sentences can contain some major or important words which may not be identified by words frequency. So, only depending on word frequency may not always provide the expected output, though several times most frequent words may relate to the topic models. Meaningfully same but structurally different sentences have to avoid while preparing a better text summarizer [3]. But related or supporting sentences may add a value to the leading sentences [4]. Finally, most leading sentence and relationship between sentences could be determined.

In this paper, we have discussed several important factors regarding assessing sentence and text similarity. Major findings are mentioned in details, and more importantly potential deep learning methods and models were stated here. Several experimental results were stated and explained with necessary measures.

2 Literature Review

The basic feature of text summarization would be either abstractive or extractive approach. Extractive method applies several manipulation rules over word, sentence, or paragraph. Based on weighted values or other measures, extractive approach chooses appropriate sentence. Abstractive summarization requires several weights like sentence fusion, constriction, and basic reformulation [5].

Oliva et al. [6] introduced a model SyMSS, which measure sentence similarity by assessing, how two different sentences systaltic structure influence each other. Syntactic dependence tree help to identify the rooted sentence as well as the similar sentence. These methods state that every word in a sentence has some syntactic connections and this will create a meaning of every sentence. The combination of LSA

[7] and WordNet [9] to access the sentence similarity in between every word was proposed in Han et al. [8]. They have proposed two different methods to measure sentence similarity. First one makes a group of words—known as the align-and-penalize approach, and the second one is known as SVM approach, where the method applies different similarity measures using n-gram and by using support vector regression (SVR), and they use LIBSVM [10] as another similarity measure.

A threshold-based model always returns the similarity value between 0 and 1. Mihalcea et al. [11] represent all sentences as a list of bag of words vector, and they consider first sentence as a main sentence. To identify word-to-word similarity measure, they have used highest semantic similarity measures in between main sentence and next sentence. The process will continue repeated times until the second main sentence could be found, during this process period. Das and Smith introduced a probabilistic model which states syntax and semantic-based analysis. Heilman and Smith [12, 13] introduce as new method of editing tree, which will contain syntactic relations between input sentences. It will identify paraphrases also. To identify sentence-based dissimilarity, a supervised two-phase framework has been represented using semantic triples [14]. Support vector machine (SVM) can combine distributional, shallow textual, and knowledge-based models using support vector regression model.

3 Proposed Method

This section represents a new proposed sentence similarity measuring model for English and Bengali language. The assessing methods, sentence representation, and degree of sentence similarity have been explained in detail. The necessary steps required especially for Bangla language have been considered while developing the proposed model. This model will work for measuring English and Bengali sentence similarity. The sentence structure and lexical form are very different for Bangla language. The semantic and syntactic measures also can add more values in this regard. The concept of working with all those necessary steps will help to produce better output, in every aspect. In this research, lexical methods have been applied and untimely a perfect expected result has been found.

3.1 Lexical Layer Analysis

The lexical layer has few major functions to perform, such as lexical representation and lexical similarity. Both of these layers have several other states to perform. Fig. 1 is the proposed model for lexical layer.

Figure 1 introduces the sentence similarity measures for lexical analysis. Different sentences will be added into a token. A word-to-word and sentence-to-sentence analyzer will perform together. An order vector will add all those words and/or sentences

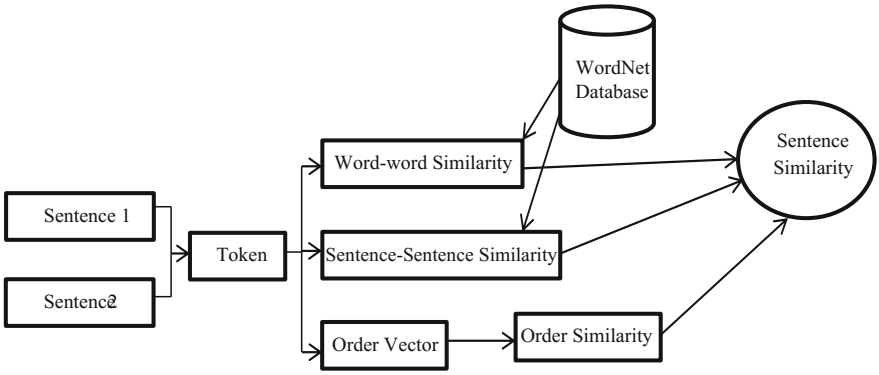

Fig. 1 Lexical layer analysis model introduces different layers of proposed methods

order in a sequence based on similarity measures. With the reference of weighted sum, the order of words and sentence will be privileged. A WordNet database will send lexical resources to word-to-word and sentence-to-sentence processes. Ultimately based on the order preference, the values from three different states (word–word similarity, sentence–sentence similarity, and order similarity) will generate the similar sentence output. The methods were followed by one of the popular deep learning algorithm—text rank.

(a) *Lexical Analysis*: This state splits sentence and words into different tokens for further processing.
(b) *Stop Words Removal*: Several values hold representative information such as article, pronoun. These types of words could be removed while considering text analysis.
(c) *Lemmatization*: This is a step to convert and/or translates each and every token into a basic form, and exactly from where it belongs to the very same verb form in the initial form.
(d) *Stemming*: Stemming is the state of word analysis. Word–word and sentence-to-sentence both methods need all their contents (text/word) in a unique form. Here every word will be treated as a rooted word such as play, player—both words are different as word though in deep meaning those words could be considered as branch words of the word "Play." By using a stemmer, we could have found all those texts in a unique form before further processing. The confusion of getting different words in structure but same in inner meaning will reduce. So, it is a very basic part of text preprocessing modules.

Figure 2 states how lexical steps had been processed with appropriate example. All the necessary processes as lexical analysis, stop words removal, and stemming had been done as per the mentioned process. Those sentences will be used for further experiments in this paper.

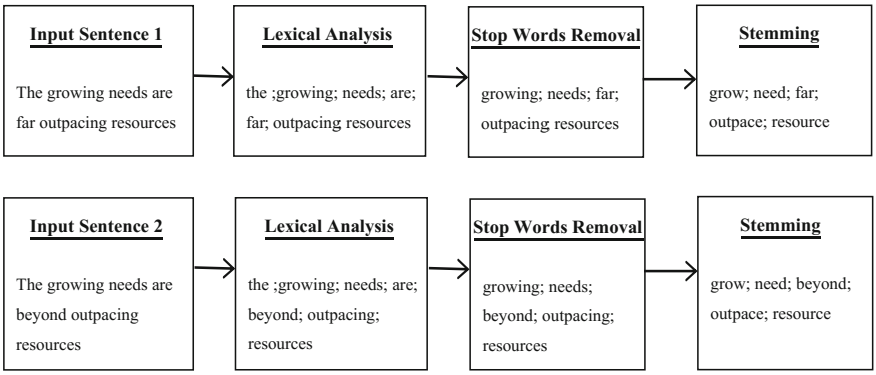

Fig. 2 Lexical layer processing of input sentences. It was clearly shown how multiple process handle single input data

3.2 Sentence Similarity

Path measure helps to sense the relatedness of words from the hierarchies of WordNet. It calculates and replies to the path distance between two words. Path measure will be used to identify similarity scores between two words. Path measure could be calculated through Eq. (1).

$$Path\_measure(\text{token1}, \text{token2}) = 1/Path\_length(\text{token1}, \text{token2}) \tag{1}$$

Path_measure will send two different tokens as token1 and token2. Both tokens are assigned the value of a single sentence after splitting. *Path_length* will return the distance of two different concepts from WordNet.

Levenshtein Distance (Lev.) algorithm has been used to identify the similarity matrix in between of words. To identify the sentence similarity, measuring words similarly pay more importance. Lev. counts the minimum number of similarity required for the operation of insertion, deletion, and modification of every character which may require transforming from a sentence to another sentence. Here it was used to identify distance and/or similarity measure between words. Longest common subsequences (LCS) have also implemented though expected output was found using Lev. Here LCS does not allow substitutions. The distance of sentences followed by Lev. will be calculated based on Eq. (2).

$$\text{LevSim} = 1.0 - (\text{Lev.Distance}(W1, W2)/\text{maxLength}(W1, W2)) \tag{2}$$

The degree of relationship helps to produce a better text summarizer by analyzing text similarity. The degree of measurement could be word–word, word–sentence, sentence–word, and sentence–sentence. In this research, we had discussed the similarity between two different words. Such as there are a set of words (after splitting

Table 1 Similarity score between words using path measure and LevSim

	Grow	Need	Far	Outpace	Resource
Grow	1.00	0.25	0.00	0.14	0.00
Need	0.25	1.00	0.11	0.17	0.14
Far	0.00	0.11	1.00	0.00	0.09
Outpace	0.14	0.17	0.00	1.00	0.00
Resource	0.00	0.14	0.09	0.00	1.00

every individual sentence): $W = \{W1, W2, W3, W4, \ldots, Wn\}$. Lev. distance calculates the distance between two words: $W1$ and $W2$, and max length will reply the score of maximum character found in between $W1$ and $W2$. Only similarity will be checked between two different words. The similarity between words could be measured by Algorithm 1.

Algorithm 1 Similarity between Words

```
W1= Sentence1.Split(" ")
W2= Sentence2.Split(" ")
        if Path_measure(W1,W2) < 0.1 then
                W_similarity= LevSim(W1,W2)
else
        W_similarity = Path_measure(W1,W2)
 end if
```

In Algorithm 1, the value of path will be dependent of distance values and Lev Similarity (LevSim) value could be found from Eq. 1. The words similarity score less than 0.1 will be calculated through the LevSim method, else the score will be accepted form the path measure algorithm. W_similarity will receive similarity score of between two words. The range of maximum and minimum score is in between $\{0.00 - 1.00\}$. Table 1 represents the similarity value of words from sentence 1.

Wu and Palmer measure (**WP**) use the WordNet taxonomy to identify the global depth measures (relatedness) of two similar or different concepts or words by measuring edge distance as well as will calculate the depth of longest common subsequences (LCS) value of those two inputs. Based on Eq. (3), WP will return a relatedness score if any relation and/or path exist in between on those two words, else if no path exist—it will return a negative number. If the two inputs are similar, then the output from synset will only be 1.

$$\text{WP\_Score} = 2 * \text{Depth(LCS)} \big/ (\text{depth}(t1) + \text{depth}(t2)) \tag{3}$$

In Eq. 3, $t1$ and $t2$ are token of sentence 1 and sentence 2. Table 2 states the WP similarity values of given input (as mentioned in Fig. 2).

Lin measure (**Lin**.) will calculate the relativeness of words or concepts based on information content. Only due to lack of information or data, output could become zero. Ideally, the value of Lin would be zero when the synset value is the rooted node.

Table 2 Similarity score between words using Wu and Palmer measure (WP)

	Grow	Need	Far	Outpace	Resource
Grow	1.00	0.40	0.00	0.25	0.00
Need	0.40	1.00	0.43	0.29	0.57
Far	0.00	0.43	1.00	0.00	0.38
Outpace	0.25	0.29	0.00	1.00	0.00
Resource	0.00	0.57	0.38	0.00	1.00

Table 3 Similarity score between words using Lin measure (Lin.)

	Grow	Need	Far	Outpace	Resource
Grow	1.00	0.40	0.00	0.25	0.00
Need	0.40	1.00	0.43	0.29	0.57
Far	0.00	0.43	1.00	0.00	0.38
Outpace	0.25	0.29	0.00	1.00	0.00
Resource	0.00	0.57	0.38	0.00	1.00

But if the frequency of the synset is zero, then the result will also be zero but the reason will be considered as lack of information or data. Equation (4) will be used to measure the Lin. value, and Table 3 will state the output values after implementing the input sentences on Lin. measures.

$$\text{Lin\_Score} = 2 * \text{IC(LCS)} \Big/ (\text{IC}(t1) + \text{IC}(t2)) \tag{4}$$

In Eq. 4, IC is the information content.

A new similarity measure algorithm was experimented where all those mentioned algorithm and/or methods will be used. Equation (5) states the new similarity measure process.

$$\text{total\_Sim}(t1, t2) = (\text{Lev\_Sim}\ (t1, t2) + \text{WP\_Score}(t1, t2) + \text{Lin\_Score}(t1, t2)) \Big/ 3 \tag{5}$$

In Eq. 5, a new total similarity values will be generated based on all mentioned lexical and semantic analysis. Edge distance, global depth measure, and analysis of information content are very much essential. In that purpose, this method has applied and experimented out is shown in Table 4.

Algorithm 2 A proposed similarity algorithm

1: *matrix=newmatrix(size(X)*size(Y))*
2: *total_sim=0*
3: *i=0*
4: *j=0*
5: *for i∈ A do*

Table 4 New similarity score

	Grow	Need	Far	Outpace	Resource
Grow	1.00	0.21	0.00	0.13	0.00
Need	0.21	1.00	0.18	0.15	0.34
Far	0.00	0.18	1.00	0.00	0.15
Outpace	0.13	0.15	0.00	1.00	0.00
Resource	0.00	0.34	0.15	0.00	1.00

6: *for* $j \in B$ *do*
7: *matrix(i, j)=similarity_token(t1,t2)*
8: *end for*
9: *end for*
10: *for has_line(matrix) and has_column(matrix) do*
11: *total_Sim=(Lev_Sim(matrix)+WP_Score(matrix)+Lin_Score(matrix))/3*
12: *end for*
13: *return total_Sim*

The Algorithm 2 receives the token on two different X, Y as input text. Then it will create a matrix representation of $m * n$ dimensions. Variable total_sim (total similarity) and i, j (which are the values for iteration purpose) will initially become 0. Initially, matrix(i, j) will generate the token matrix, where values will be added. The variable total_sim will record and update calculate the similarity of pair of sentences based on token matrix—matrix(i, j).

4 Experimental Results and Discussion

Several English and Bengali texts were tested though the proposed lexical layer to find out the sentence similarity measure. Texts are being collected from online resource, for example, wwwo.prothom-alo.com, bdnews24.com, etc. Our python Web crawler initially saved all those Web (html content) data into notepad file. We have used Python—Natural Language Toolkit (NLTK: Version–3) and WS4J (a Java API, especially developed for WordNet use). All the experimented results are stated in this section.

Tables 1, 2, and 3 state that the experimented result of similarity measure by using path measure and LevSim, Wu and Palmer measure (WP), and Lin measure (Lin.) consecutively. All those methods are either applied in lexical analysis or semantic analysis. In this research article, the proposed method of identifying sentence similarity using a hybrid model is being stated in Table 4.

This method was also applied in Bengali language using Bengali WordNet. Experimented results are shown in Table 5.

Table 5 New similarity score (applied in Bengali sentence)

	ট্রেন	সিট	ভাড়া	গন্তব্য
ট্রেন	1	0.78	0.88	0.16
সিট	0.78	1	0.31	0.24
ভাড়া	0.88	0.31	1	0.23
গন্তব্য	0.16	0.24	0.23	1

5 Conclusion and Further Work

This paper has presented sentence similarity measure using lexical and semantic similarity. Degree of similarity was mentioned and implemented in the proposed method. There are few resources available for Bengali language. More development on Bengali language is just more than essential. Bengali WordNet is not stable as like other WordNet available for English language. This research found suitable output in the unsupervised approach, though a huge dataset will be required to implement the supervised learning methods. There are other sentence similarity measures and could be done by more semantic analysis and syntactic analysis. Both of these analyses if could be done together including lexical similarities, a better result could be found. More importantly, for a better text summarizer, we need to identify the leading sentences. Centroid sentences could optimize the analysis of post-processing of text summarization. Evaluating system developed summarizer before publishing as final form is more important. Backtracking methods could possibly be a good solution in his regards.

Acknowledgements We would like to thanks Department of Computer Science and Engineering of two universities: Daffodil International University and Comilla University, Bangladesh, for facilitating such joint research. Special thanks to DIU-NLP and Machine Learning Research Laboratory for providing research facilities.

References

1. Ferreira R et al (2013) Assessing sentence scoring techniques for extractive text summarization. Expert Syst Appl 40:5755–5764 (Elsevier Ltd.)
2. Abujar S, Hasan M (2016) A comprehensive text analysis for Bengali TTS using unicode. In: 5th IEEE international conference on informatics, electronics and vision (ICIEV), Dhaka, Bangladesh, 13–14 May 2016
3. Abujar S, Hasan M, Shahin MSI, Hossain SA (2017) A heuristic approach of text summarization for Bengali documentation. In: 8th IEEE ICCCNT 2017, IIT Delhi, Delhi, India, 3–5 July 2017
4. Lee Ming Che (2011) A novel sentence similarity measure for semantic-based expert systems. Expert Syst Appl 38(5):6392–6399
5. Mani I, Maybury MT (eds) (1999) Advances in automatic text summarization, vol 293. MIT Press, Cambridge, MA

6. Oliva, J et al (2011) SyMSS: a syntax-based measure for short-text semantic similarity. Data Knowl Eng 70(4):390–405
7. Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
8. Han L, Kashyap AL, Finin T, Mayfield J, Weese J (2013) UMBC EBIQUITY-CORE: semantic textual similarity systems. Volume 1, Semantic textual similarity, Association for Computational Linguistics, Atlanta, Georgia, USA, June, pp 44–52
9. Miller GA (1995) Wordnet: a lexical database for English. Commun ACM 38:39–41
10. Chang C-C, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27, 1–27
11. Mihalcea R, Corley C, Strapparava C (2006) Corpus-based and knowledge-based measures of text semantic similarity. National conference on artificial intelligence, vol 1. AAAI Press, Boston, MA, pp 775–780
12. Heilman M, Smith NA (2010) Tree edits models for recognizing textual entailments, paraphrases, and answers to questions. In: Annual conference of the North American chapter of the association for computational linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, pp 1011–1019
13. Heilman M, Smith NA (2010) Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In: Human Language Technologies, Stroudsburg, PA, USA, pp 1011–1019
14. Qiu L, Kan M-Y, Chua T-S, (2006) Paraphrase recognition via dissimilarity significance classification, EMNLP. Association for Computational Linguistics, Stroudsburg, PA, USA, pp 18–26