Semantic Textual Similarity in Bengali Text

Conference Paper · September 2018

DOI: 10.1109/ICBSLP.2018.8554940

2 authors:

Md Shajalal
Hajee Mohammad Danesh Science and Technology University
12 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Masaki Aono
Toyohashi University of Technology
133 PUBLICATIONS **2,436** CITATIONS
SEE PROFILE

International Conference on Bangla Speech and Language Processing (ICBSLP), 21-22 September, 2018

Semantic Textual Similarity in Bengali Text

Md Shajalal
Department of Computer Science & Engineering
Bangladesh Army University of Science and Technology
Saidpur, Nilphamari, Bangladesh
shajalal@baust.edu.bd

Masaki Aono
Department of Computer Science & Engineering
Toyohashi University of Technology
Toyohashi, Aichi, Japan
aono@tut.jp

***Abstract*—Measuring the textual similarity is indispensable in many information retrieval applications. Researchers proposed numerous similarity measures to compute the semantic similarity between texts for monolingual and multilingual texts. But methods for measuring similarity for Bengali text segments are not so commonly available. In this paper, we propose an approach to estimate the semantic similarity between Bengali text segments. The similarity score is computed with the help of word-level semantics from a pre-trained word-embedding model trained on Bengali Wikipedia texts. In this regard, we employ an algorithm to measure the semantic similarity of Bengali texts. To test the performance of our method, we conducted experiments on a dataset for semantic textural similarity for Bengali texts. We prepare the dataset using the same approach as SemEval applied in the STS 2017. The experimental results in terms of Pearson correlation coefficient conclude that our method achieves a state-of-the-art performance for semantic textual similarity in Bengali texts.**

Key–Words: Bengali Textual Similarity;Semantic Similarity; Word-level Semantics; Word-embedding.

I. Introduction

The textual similarity between texts is an important and mandatory task in many applications in information retrieval. The performance of many natural language processing (NLP) applications such as text summarization, machine translation, plagiarism detection, sentiment analysis etc. is also dependent on the textual and semantic similarity. There are also some other applications that used the similarity such as relevance feedback, text classification, word sense disambiguation, subtopic mining, web search and so on [1]–[3]. The similarity measures for many languages like English, Arabic, Spanish are available and there are some research tasks to compute the similarity between multilingual and monolingual texts organized by SemEval STS organizer [4]–[6].

One of the typical approaches to compute the similarity is lexical matching between texts. The similarity score is computed based on the number of terms belong to both text segments. But these measures are not able to compute the similarity beyond a trivial level. Moreover, this matching can only estimate the textual similarity but not semantic. Let us consider two texts, "তিনি তোমাদের বাংলা শিক্ষক"(meaning, *He is your Bengali teacher*) and "তিনি তোমাদের বাংলা শিক্ষককে পিটিয়েছেন" (meaning, *He has beaten your Bengali teacher*). According to the lexical matching, there are four lemmatized terms ("তিনি", "তোমাদের", "বাংলা", and "শিক্ষক") exist in both sentences. That means the similarity is nearly 0.80(on a scale of 1.0). But there is no semantic connection between these two texts. If we consider another example sentence-pair, "আমার একটি পোষা প্রানী আছে" (meaning, *I have a pet*) and "আমি বিড়াল পুষি"(meaning, *I own cat*). There is no single terms exist in these two sentences but there is an obvious semantic similarity. The table I depicts the scenario. However, these two examples conclude that the lexical measures are not enough to capture the similarity.

TABLE I: The weekness of lexical matching in capturing semantic similarity

Sentence 1	Sentence 2	Similarity
তিনি তোমাদের বাংলা শিক্ষক	তিনি তোমাদের বাংলা শিক্ষককে পিটিয়েছেন	Lexically similar but **not semantically**
আমার একটি পোষা প্রানী আছে	আমি বিড়াল পুষি	Semantically similar but **not lexically**

Estimating similarity between Bengali texts is more challenging as compared to the English texts. One of the main reasons is that the resources for the Bengali language are not even comparable with the resources of English language. To preprocess English texts there are well known tokenizer, stemmer, lemmatizer, that are applied in almost every NLP task and their performance is even arguably better. But on contrary, these kind of tools are not so commonly available to canonicalize Bengali texts. Furthermore, there are also well-organized resources such as WordNet, NLP POS tagger etc. that amplify the performance of any similarity estimation method. Therefore the methods for Bengali texts lack such kind of tools and resources.

In this paper, we try to overcome the challenges in capturing the semantic similarity between Bengali text-pair. We introduced a method for measuring the semantic similarity for Bengali texts. In this regard, an efficient Bengali semantic textual similarity measuring algorithm is employed based on a pre-trained word-embedding model that is trained on Wikipedia texts. We also prepare a the dataset that can be used to test the performance of semantic similarity measure in Bengali texts. We follow the same procedure as the SemEval STS 2017 task [6]. The experimental results clearly demonstrate that our method is effective to measure the semantic similarity in Bengali texts. The contributions of this research are:

1) A semantic textual similarity estimation algorithm for Bengali texts, and

978-1-5386-8207-4/18/$31.00 ©2018 IEEE

2) A dataset that can be used for measuring the performance of any Bengali semantic similarity measure.

The rest of the paper is structured as follows: In **Section II**, we present the working principle of word-embedding. **Section III** summarizes the related work on semantic textual similarity. In **Section IV**, we present our proposed method to incorporate the challenges on semantic similarity for Bengali texts. The experiments and evaluation results are presented to show the effectiveness of our proposed method in **Section V**. Some concluded remarks and future directions are described in **Section VI**.

II. Word-Embedding

The word-embedding (*word2vec*) can predict a word in a certain context for a set of given words. The framework for learning word vectors is shown in the Fig. 1. Here the context of three words ("the", "cat", and "sat") is used to predict the word ("on"). The input words are mapped to columns of the matrix W to predict the output word [7].

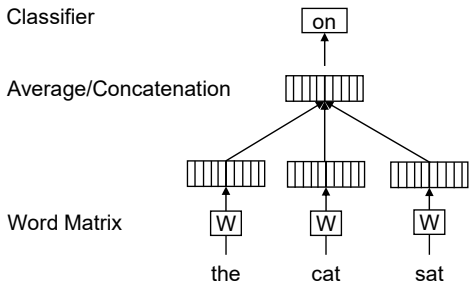

Fig. 1: A framework for learning word vectors [7].

Every word has a unique vector representation where the vector is represented by a column in a matrix W. The column is indexed by the position of the word in the vocabulary. To predict the next word in a sentence, the concatenation or sum of the vectors is employed as features [7]. Given a sequence of training words $w_1, w_2, w_3, ...W_T$, the main objective of *word2vec* model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=k}^{T-k} \log\, p(w_t|w_{t-k}, ..., w_{t+k})$$

A multiclass classifier is used for the prediction task, such as softmax [7].

$$p(w_t|w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

where each of y_i is un-normalized log-probability for each output i, computed as

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W)$$

where U, b are the softmax parameters. h is constructed by a concatenation or average of word vectors extracted from W.

The word vector based on neural networks are usually trained using stochastic gradient descent where the gradient obtained by backpropagation. An algorithm for training word vectors is publicly available at *code.google.com/p/word2vec* [8]. In a trained word vector model, words with similar meaning are mapped to a similar position in a vector space.

A pre-trained word-embedding model[1] trained in Bengali Wikipedia texts [22] is used in our method to estimate the similarity. The dimension of the feature vector per word is 300. The other parameters to train the model are explain in [22].

III. Related Work

The similarity measures to compute the semantic textual similarity between multilingual and monolingual texts have been proposed in recent past [4]–[6], [9]–[11]. But the similarity measures for Bengali texts are not so commonly available. Researchers proposed different methods and techniques to capture the semantic similarity between text segments using different resources [4]–[6], [9]–[11]. But these methods are mostly for English textual similarity. SemEval Semantic Textual Similarity (STS) tasks were organized for monolingual and multilingual texts [4]–[6]. The participating methods employed a large number of features using a wide variety of resources [11], [12]. Additionally, they applied some handcrafted rules that deal with currency values, negation, compounds, number overlap and literal matching [4]–[6]. Different multiple resources such as WordNet, Wikipedia, a dependency parser, NER tools, lemmatizer, POS tagger, stop word list etc were leveraged to extract features. Based on the content information of the text segments and external resources multiple syntactic, semantic, and structural features were also used to capture the similarity [13].

However, the similarity measures for Bengali texts are hardly available. Sinha et al. [14] proposed a new lexicon and similarity measure for measuring the similarity between Bengali texts. A distinct lexical organization using the semantic association between Bengali words than can be accessed efficiently by different applications. Rudrapal et al. [15] proposed a method for measuring the semantic similarity of Bengali tweets using WordNet. Mihalcea et al. [9] suggested a method for measuring the semantic similarity of texts by exploiting the information that can be drawn from the similarity of the component words. Specifically, the researchers used numerous corpus-based and knowledge-based measures of word semantic similarity and used them to derive a text-to-text similarity metric.

[1] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Some structured semantic knowledge like Wikipedia and WordNet are also employed to estimate the similarity. In some prior works [16]–[18], the methods are very similar to one another, using pairings of words and WordNet-based measures for semantic similarity. Researchers also used corpus-based methods combining with WordNet-based measures [9], [19]. In [9], they introduced an IDF-weighted alignment approach, based on WordNet-based and corpus-based similarities. They applied the similarities between words which are identical in terms of their part-of-speech (POS) tag. Then a single score is calculated using the average over the maximum similarities. In [19], the similarity score has been measured by combining the word order score and a WordNet similarity measure. Recently, researchers tried word-embedding based techniques for semantic similarity [20], [21].

IV. Semantic Textual Similarity Measuring Algorithm for Bengali Texts

This section presents our introduced algorithm for measuring the semantic similarity between two sentences $S1$ and $S2$. The **Algorithm 1** presents the pseudo-code. The table **II** summarizes the basic notation used in our proposed algorithm.

TABLE II: Basic notation used in **Algorithm 1**

Symbol	Description
$S\_terms[\,]$	List of words after processing sentence S
AVS	300 dimensional average feature vector for each sentence S
tc_S	Total number of words contain in vocabulary of $w2v\_model$ for S
$w2v\_model$	Trained word-embedding model
$vocab(w2v\_model)$	Vocabulary of $w2v\_model$
$add(t, \mathbf{AVS}, w2v\_model)$	Adding the 300 dimensional vector for term t with **AVS**
$divide(\mathbf{AVS}, tc\_S)$	Dividing each value of **AVS** with tc_S
sim	The semantic textual similarity score between $S1$ and $S2$

For a given pair of sentences $S1$ and $S2$, we first remove different types of punctuation marks, Bengali digits, etc. The preprocessed sentences are split into the list of words. The list is denoted by $S\_terms[\,]$. Two list of words $S1\_terms[\,]$, and $S2\_terms[\,]$(in step 1 & 2 in **Algorithm 1**) are used to compute the similarity between two corresponding sentences $S1$ and $S2$, respectively. Then we compute the average feature vector **AVS** for each sentence. Word-embedding model returns a 300-dimensional vector for each term. Therefore we retrieve the feature vector for each term t belongs to $S\_terms[\,]$. The vectors are computed only for words those belong to the vocabulary of the Word2Vec model, $vocab(w2v\_model)$. The feature vectors for each word belongs to a particular sentence are then added.

This addition is done in scope (from step 7 to step 12) and scope (from step 14 to step 19) for $S1$ and $S2$, respectively. The vectors after the addition are stored in **AVS1** and **AVS2** for the two sentences. Each values in **AVS1** and **AVS2** are then divided by total number of words tc_S1 and tc_S2, respectively for corresponding sentences $S1$ and $S2$. The division is done in step 13 and 20, respectively. The average feature vectors **AVS1** and **AVS2** are then used to calculate the similarity. We applied the cosine similarity to compute the similarity score between $S1$ and $S2$. The cosine similarity is computed in step 21. The equation of cosine similarity can be elaborated as follows:

$$sim(S1, S2) = \frac{\mathbf{AVS1} \cdot \mathbf{AVS2}}{||\mathbf{AVS1}|| \cdot ||\mathbf{AVS2}||} = \frac{\sum_{i=0}^{300} AVS1_i \cdot AVS2_i}{\sqrt{\sum_{i=0}^{300} AVS1_i^2}\sqrt{\sum_{i=0}^{300} AVS2_i^2}}$$

where $AVS1_i$ and $AVS2_i$ denote the i-th feature value of vector **AVS1** and **AVS2**, respectively.

Algorithm 1: Semantic textual similarity estimation algorithm for Bengali texts: BSTS($S1$, $S2$, $w2v\_model$)

Input: Sentence1 $S1$, Sentence2 $S2$, and Word2Vec model, $w2v\_model$
Output: Similarity score, $sim(S1, S2)$ between $S1$ and $S2$

1 $S1\_terms[\,] \leftarrow Preprocess(S1)$
2 $S2\_terms[\,] \leftarrow Preprocess(S2)$
3 **AVS1** $\leftarrow [0, \cdots, 0]$
4 **AVS2** $\leftarrow [0, \cdots, 0]$
5 $tc\_S1 \leftarrow 0$
6 $tc\_S2 \leftarrow 0$
7 **for** *for each term,* $t \in S1\_terms$ **do**
8 **if** t *in vocab(w2v_model)* **then**
9 **AVS1** $\leftarrow add(t, \mathbf{AVS1}, w2v\_model)$
10 tc_S1++
11 **end**
12 **end**
13 **AVS1** $\leftarrow divide(\mathbf{AVS1}, tc\_S1)$
14 **for** *for each term,* $t \in S2\_terms$ **do**
15 **if** t *in vocab(w2v_model)* **then**
16 **AVS2** $\leftarrow add(t, \mathbf{AVS2}, w2v\_model)$
17 tc_S2++
18 **end**
19 **end**
20 **AVS2** $\leftarrow divide(\mathbf{AVS2}, tc\_S2)$
21 $sim(S1, S2) \leftarrow \frac{\mathbf{AVS1} \cdot \mathbf{AVS2}}{||\mathbf{AVS1}|| \cdot ||\mathbf{AVS2}||}$

V. Experiments and Evaluation

A. Dataset Preparation

We prepared the dataset following the same approach is done by SemEval Semantic Textual Similarity task, STS2017 [6]. In the STS2017 task, the organizers provided 250 pairs of sentences. We translated those sentences manually by the human assessor. The STS2017 organizers provided the similarity score per sentence-pair that are calculated by human assessors' judgments. We employed their provided gold-standard judgment as a ground truth in this research. Their human assessors have given the similarity score using the following similarity label ranges from $[0, 5]$.

- **Label 0**: On different topics

- **Label 1**: Not similar but share few common details
- **Label 2**: Not similar but share some common details
- **Label 3**: Roughly similar
- **Label 4**: Similar
- **Label 5**: Completely similar

The distribution of the similarity labels after the annotation is mentioned elsewhere in [6]. The human assessors are instructed to assign the labels as followings [6]:

1) Assign labels as precisely as possible according to the underlying meaning of the two sentences rather than their superficial similarities or differences.
2) Be careful of wording differences that have an impact on what is being said or described.
3) Ignore the grammatical errors and awkward wording as long as they do not obscure what is being conveyed.
4) Avoid over labeling pairs with middle range scores.
5) Be careful of over-reliance on an extreme score like 0 or 5.

We applied a pre-trained model which trained on Bengali Wikipedia texts [22]. The dimension of the feature vector per words in the model is 300. Continuous-bag-of-words, *cbow* algorithm is used to train the model. The other parameters are described elsewhere in [22]. The model was trained on whole Bengali Wikipedia texts [22].

B. Evaluation Metric

The performance of our method has been tested based on *Pearson Correlation Coefficient*[2]. This evaluation metric has also been used as an official metric to test the performance of a method in SemEval STS2017 [6].

Let $X = \{x_1, x_2, x_3...x_n\}$ and $Y = \{y_1, y_2, y_3...y_n\}$ be the two sets of scores for n pairs of sentences generated by the system and human assessors' judgment, respectively. Each element x_i or y_i in set X and Y, respectively represents the semantic textual similarity between i-th sentence-pair. The *Pearson Coefficient Correlation* r is defined as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where n is the number of sentence pairs and x_i, y_i are the similarity scores given by participant and human assessors, respectively indexed with i. The arithmetic mean of the elements of X is defined by $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and analogously for $\bar{y}$.

[2]https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

C. Experimental Results

We conducted experiments with different experimental settings. We first employed the edit distance (using the terms as the lexical unit) based lexical similarity between sentences as the baseline. We then applied our proposed algorithm (**Algorithm 1**), BSTS to compute the semantic similarity for Bengali texts. The experimental results are summarized in Fig. 2.

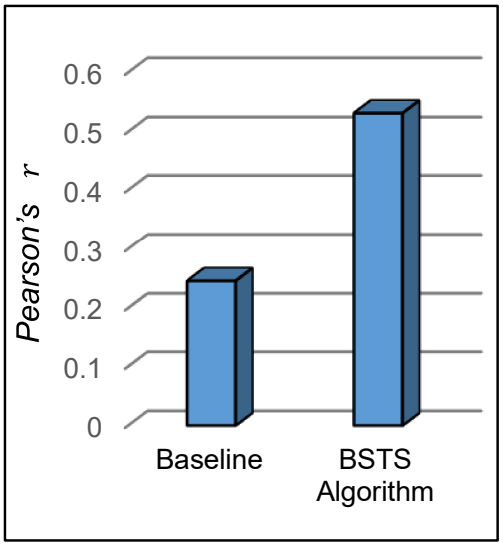

Fig. 2: The performance of our porpoised algorithm and the baseline method in terms of *Pearson's Correlation Coefficient* r.

The Fig. 2 illustrated that our method can capture a far better semantic and textual similarity as compared to the traditional lexical similarity. The figure also indicates that the vector representation of each word in the word-embedding model is useful and effective to measure the semantic similarity between texts.

TABLE III: Walk-trough example

Sentence 1	Sentence 2	Lexical	BSTS
তিনি তোমাদের বাংলা শিক্ষক	তিনি তোমাদের বাংলা শিক্ষককে পিটিয়েছেন	0.8000	**0.3377**
আমার একটি পোষা প্রানী আছে	আমি বিড়াল পুষি	0.0000	**0.6138**

Table III illustrates the performance of our proposed algorithm and the lexical similarity between the two example sentences-pairs (given in table I). As we noted that the traditional lexical matching has given 80% similarity for first sentence-pair ("তিনি তোমাদের বাংলা শিক্ষক" and "তিনি তোমাদের বাংলা শিক্ষককে পিটিয়েছেন") but, the performance of our method concludes that they are 33% similar. Based on the semantic meaning of these two sentences, they are not similar but share few common information. Our proposed algorithm has given less similarity as compared to the lexical similarity. On contrary, for the second example ("আমার একটি পোষা প্রানী আছে" and "আমি বিড়াল পুষি"), the lexical