A Comparison of Semantic Similarity Methods for Maximum Human Interpretability

Pinky Sitikhu[1], Kritish Pahi[2], Pujan Thapa[3], Subarna Shakya[4]
Department of Electronics and Computer Engineering, Tribhuwan University
Pulchowk Campus, IOE, Nepal
[1]pinky.sitikhu524@gmail.com
[2]kritishpahi@gmail.com
[3]pujanthapa88.63@gmail.com
[4]drss@ioe.edu.np

***Abstract*—The inclusion of semantic information in any similarity measures improves the efficiency of the similarity measure and provides human interpretable result. This paper presents three different methods to compute semantic similarities between short news texts. These methods are based on corpus-based and knowledge-based methods: cosine similarity using tf-idf vectors, cosine similarity using word embeddings and soft cosine similarity using word embeddings. As a result, cosine similarity using tf-idf vectors performed best among three in finding similarities between short news texts.**

***Index Terms*—semantic similarity, cosine similarity, soft cosine similarity, word embeddings**

I. Introduction

Text similarity has been one of the most important applications of Natural Language Processing. Two texts are said to be similar if they infer the same meaning and have similar words or have surface closeness. Semantic similarity measures the degree of semantic equivalence between two linguistic items, be they concepts, sentences, or documents. A proper method which computes semantic similarities between documents, will have a great impact upon different NLP applications like document classification, document clustering, information retrieval, machine translation and automatic text summarization. There are different approaches to compute similarities between documents that uses lexical matching, linguistic analysis, or semantic features. In the context of short texts, methods for lexical matching might work for trivial cases. But it is arguably not an efficient method because this method considers whether the words of short texts look alike eg. in terms of distances, lexical overlap [1] or largest common substring [2]. Generally, linguistic tools like parsers and syntactic trees are used for short text similarity, however, all the texts such as tweets might not be parsable. A linguistically valid interpretation might be very far from the intended meaning. So, semantic features are quite important in text mining. For semantic features, external sources of structured semantic knowledge such as Wikipedia, WordNet, or word embeddings are used [3].

The main objective of this paper is to compare the semantic similarities between short texts to maximize human interpretability. The basic idea to compute text similarities is by determining feature vectors of the documents and then calculating the distance between these features. A small distance between these features means high degree of similarity, whereas a large distance means low degree of similarity. Euclidean distance, Cosine distance, Jensen Shannon Distance, Word Mover distance [4] are some of the distance metrices used in the computation of text similarity. This paper presents two different methods to generate features from the documents or corpus: (1) using Tf-idf vectors and (2) using Word Embeddings and presents three different methods to compute semantic similarities between short texts: (1) Cosine similarity with tf-idf vectors (2) Cosine similarity with word2vec vectors (3) Soft cosine similarity with word2vec vectors.

II. Related Work

N. Shibata et. al [14] performed a comparative study to measure the semantic similarity between academic papers and patents. They compared the structures of citation network of scientific papers with patents by citation analysis, measured similarity between scientific papers and patents by comparing three semantic similarity methods: Jaccard coefficient, cosine similarity of tf-idf vector, and cosine similarity of log-tf-idf vectors, and also discussed the validity of the results with experts.

S. Zhang et. al [15] provided a survey of similarity measure methods including semantic similarity between concepts and semantic textual similarity. This paper classified methods of semantic similarity into four categories based on background information resource used. It showed how similarity computation methods assist in many aspects of social network analysis.

H. Pu et. al [16] provided a short text semantic similarity calculation method that combines both knowledge-based method and corpus-based method. A large corpus was used to compare and analyze several words and text semantic algorithms in this experiment. The method used was based on improved word semantic similarity calculation method that combines two word semantic similarity by some strategies.

Wael H. Gomaa [17] discussed three text similarity approaches: string-based, corpus-based and knowledge-based similarities. The string based measure works on sequences of string and the composition of characters. Corpus based measure determines the similarity between words using the

information gained from corpus, and knowledge based measure is based on identifying the degree of similarity between words using information from semantic networks.

In this experiment, we introduced three different semantic similarity measures based on corpus-based and knowledge-based similarity. The cosine similarity using tf-idf vectors is corpus-based method in which tf-idf vectors were used to gain the information about the corpus or document. The cosine similarity using word embeddings and soft-cosine similarity using word embeddings are knowledge-based similarity method, in which word embeddings from pre-trained "conceptnet-numberbatch" dataset was used to find the degree of similarity between the words.

III. Methodology

A. Dataset

The dataset used in this experiment is the popular and publicly available short news articles called AG's news topic classification.

1) AG's News Topic Classification Dataset: This news corpus consists of news articles from the AG's corpus of news articles [5] on the web and have 4 largest classes. Each class consists of 30,000 training samples and 1900 testing samples. This dataset is provided by the academic community for research purposes in data mining, information retrieval, etc.

B. Data Cleaning and Preprocessing

Basic preprocessing steps like lemmatization, stop-word removal, and filtering characters, punctuations and numbers were done. WordNetLemmatizer [6] was used to lemmatize the words and the English stop-words were taken from NLTK library [7].

C. Feature Vectors

In machine learning, a feature vector is a n-dimensional vector of numerical features that represent some object or context. It holds an important place in computing semantic similarities between texts. In this experiment, two different methods have been used to compute feature vectors. They are:

1) Tf-idf Vectors: TF-IDF is an abbreviation for Term Frequency-Inverse Document Frequency and is a very common algorithm to transform a text into a meaningful representation of numbers. Tf-idf weight is a statistical measure that evaluates the importance of a particular word to a document.

Mathematically,

$$tfidfweight = \sum_{i \in_d} tf_{i,d} * log(\frac{N}{df_i}) \tag{1}$$

where, $tf_{i,d}$ is the number of occurrences of i^{th} term in document d, df_i is the number of documents containing i^{th} term, N is the total number of documents.

A tf-idf model was created using sklearn vectorizer model. This model was fitted using the documents and a set of tf-idf vectors containing tf-idf weight of each words of the documents were created. Now, these tf-idf vectors were used as a feature vectors for measuring similarities between the news dataset.

D. Word Embeddings

Word embeddings are vector representation of a word obtained by training a neural network on a large corpus. It has been widely used in text classification using semantic similarity. Word2vec is one of the most widely used forms of word embeddings. The word2vec takes text corpus as input and produces word vectors as output, which can be further used to train any other word to obtain its corresponding vector value. This word2vec model uses continuous skip-gram model [8], based on distributional hypothesis. An open-source library, Gensim [9] provides different pre-trained word2vec models trained on different kinds of datasets like GloVe, google-news, ConceptNet, Wikipedia, twitter [10]. A pre-trained word embeddings named ConceptNet Numberbatch 19.08 [11] was used as a word2vec model to create feature vectors for the dataset in this experiment.

E. Similarity Measures

Similarity function or measure is a real-valued function that quantifies the similarity between two objects. This experiment presents two similarity measures: cosine similarity and soft-cosine similarity.

1) Cosine Similarity: It is a similarity measure which measures the cosine of the angle between two vectors projected in a multi-dimensional plane. It is the judgment based on orientation rather than magnitude.

Given two vectors of attributes, A and B, the cosine similarity is presented using a dot product and magnitude as:

$$similarity = cos(\theta) = \frac{A.B}{|A||B|} \tag{2}$$

2) Soft Cosine Similarity: A soft cosine or soft similarity between two vectors generalize the concept of cosine similarity and considers similarities between pairs of features [12]. The traditional cosine similarity considers the Vector Space Model (VSM) features as independent or completely different, while soft cosine measure considers the similarity of features in VSM as well. The similarity between a pair of features can be calculated by computing Levenshtein distance, WordNet similarity or other similarity measures. For example: words like "cook" and "food" are different words and are mapped to different points in VSM. But, semantically they are related with each other.

Given two N-dimensional vectors a and b, the soft cosine similarity can be calculated as follows:

$$softcosine(a,b) = \frac{\sum_{i,j}^{N} s_{ij} a_i b_j}{\sqrt{\sum_{i,j}^{N} s_{ij} a_i a_j} \sqrt{\sum_{i,j}^{N} s_{ij} b_i b_j}} \tag{3}$$

where s_{ij} = similarity(feature, feature)

The matrix s represents the similarity between features. If there is no similarity between features (s_{ii} = 1, s_{ij} = 0 for i $\neq$ j) then, this equation is equivalent to cosine similarity.

IV. Proposed Method

On the basis of methods to compute feature vector and similarity measures, this paper presents three methods to compute similarities between short text data.

A. Cosine Similarity using TF-IDF Vectors

The pre-processed documents were converted into tf-idf vectors by using vectorized tf-idf model. The obtained vectors were sparse matrix containing tf-idf weights for each words of each document having size of [number of documents * number of features(unique words)]. Now, these tf-idf weights from the matrix were used as a feature for each document, and similarity between documents are computed using cosine similarity. The inbuilt cosine similarity module from sklearn was used to compute the similarity.

B. Cosine Similarity using Word2Vec Vectors

In this method, the pre-trained word2vec model was loaded using gensim [9]. This word2vec model was used to compute the vector values or word embeddings of each word of all the preprocessed documents. The word2vec vectors for the words that do not exist in the vocabulary of the pre-trained model were set to zero. An average of all the word vectors of a document was computed and the resultant vector was used as a feature word2vec vector for that document. So, all the documents were vectorized from n-gram vectors into word2vec vectors. These word2vec vectors were then used as feature vectors to compute the cosine similarity within the documents.

C. Soft Cosine Similarity using Word2Vec Vectors

This method also used the word embeddings from same pre-trained word2vec model. These word2vec vectors were used to construct a term similarity index which computes cosine similarities between these word2vec vectors from the pre-trained model.

The preprocessed documents were converted into a dictionary or a bag of words model and the formed dictionary was converted into a corpus (a sparse vector containing unique word ids and its number of occurrences). Since soft cosine measure uses a similarity matrix between pair of features, the dictionary (feature of the dataset) and term similarity index (features of word2vec model) were used to compute a term similarity matrix. Finally, soft cosine similarity between the documents were computed. The modules provided by gensim were used to compute this similarity measure [13].

V. Result and Analysis

About 2000 news articles from the dataset were selected randomly for the experiment. The similarity of each news article was computed against itself and all the other articles. A huge matrix of size (number of news articles * number of news articles) ie. 2000 * 2000 was created. We know that, self-similarity is always 1. So, all the diagonal values of the similarity matrix was replaced by 0 because we wanted to find the most similar document for any document beside itself. The most similar news article for a news article was computed by finding out the maximum similarity value in the similarity matrix.

These figures show the illustration of the procedure explained above. D1, D2, D3, D4, D5 represents five documents.

	D1	D2	D3	D4	D5
D1	1	0.2236	0.1884	0.2674	0.1722
D2	0.2236	1	0.135	0.245	0.2308
D3	0.1884	0.135	1	0.1817	0.284
D4	0.2674	0.245	0.1817	1	0.2495
D5	0.1722	0.2308	0.284	0.2495	1

Fig. 1. Similarity Matrix of five documents with each other.

	D1	D2	D3	D4	D5
D1	0	0.2236	0.1884	0.2674	0.1722
D2	0.2236	0	0.135	0.245	0.2308
D3	0.1884	0.135	0	0.1817	0.284
D4	0.2674	0.245	0.1817	0	0.2495
D5	0.1722	0.2308	0.284	0.2495	0

Fig. 2. Replacing all diagonal values (self-similarity values) to 0.

	D1	D2	D3	D4	D5
D1	0	0.2236	0.1884	0.2674	0.1722
D2	0.2236	0	0.135	0.245	0.2308
D3	0.1884	0.135	0	0.1817	0.284
D4	0.2674	0.245	0.1817	0	0.2495
D5	0.1722	0.2308	0.284	0.2495	0

Fig. 3. Finding out the most similar document on the basis of similarity value.

The analysis of the performance of these three methods was done by retrospective analysis by measuring the top-1 accuracy. Top-1 accuracy is the conventional accuracy in which the answer from the model must be the expected answer. So, the newsgroup or news category of each news article and its respective most similar news article were checked and compared. If an article and its most similar article are in same category or newsgroup, then it is noted as correctly classified, otherwise not. The accuracy of each method was calculated on the basis of this rule, ie.

$$accuracy = \frac{\text{total no. of correctly classified news articles}}{\text{total number of news articles}} * 100 \quad (4)$$

The table below shows the top-1 accuracy of each method using the dataset.

This table shows that all of the three methods performed quite well with great accuracy. But, among these three methods, cosine similarity using tf-idf showed greater accuracy. Cosine similarity with word2vec had relatively low accuracy

TABLE I
TABLE SHOWING ACCURACY OF EACH METHOD

Methods used to calculate text similarity	***Top-1 Accuracy (in %)***
Cosine Similarity using tf-idf Vectors	76.8
Cosine Similarity using Word2Vec Vectors	75.9
Soft Cosine Similarity using Word2Vec Vectors	76.05

among all three methods. The reason behind this is the fact that the document vector is computed as an average of all word vectors in the document and the assignment of zero value for the words, that are not available in word2vec vocabulary. Soft cosine similarity performed better than cosine similarity with word2vec as this method used a similarity matrix with pre-trained word embeddings and dictionary or BoW of the documents as features.

VI. CONCLUSION

In this research paper, we performed a comparison between three different approaches for measuring the semantic similarity between two short text news articles. AG's news-related data sets was used to verify the experiment. The three approaches are: Cosine similarity with tf-idf vectors, cosine similarity with word2vec vectors, soft cosine similarity with word2vec vectors. All of these three methods had shown promising results. Among these three vectors, cosine with tf-idf had the highest accuracy when the results were cross-validated and the newsgroup of a news article and its corresponding most similar article were compared. These methods can be very useful in text clustering and other text mining applications.

ACKNOWLEDGMENT

We would like to express our deepest gratitude to Prof. Dr. Subarna Shakya for supervising and motivating us for this research work. We are immensely grateful to all the people who have directly or indirectly supported us in this work.

REFERENCES

[1] V. Jijkoun and M. De Rijke, "Recognizing Textual Entailment Using Lexical Similarity," 2005.
[2] Islam, A. and Inkpen, D. 2008. "Semantic text similarity using corpus-based word similarity and string similarity". ACM Trans. Knowl. Discov. Data. 2, 2, Article 10 (July 2008), 25 pages.
[3] T. Kenter and M. De Rijke, "Short Text Similarity with Word Embeddings", 2015.
[4] Finding similar documents with Word2Vec WMD, Available.
[5] A. Gulli, "AG's corpus of news articles",(2004) http://www.di.unipi.it/~gulli/AG-corpus-of-news-articles.html .
[6] WordNet Lemmatizer, https://www.nltk.org/api/nltk.stem.html#module-nltk.stem.wordnet.
[7] NLTk English Stopwords, Available: https://www.nltk.org/book/ch02.html
[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", arXiv:1310.4546v1 [cs.CL][2013].
[9] Gensim Word2Vec, https://radimrehurek.com/gensim/models/word2vec.html.
[10] Gensim Data, https://github.com/RaRe-Technologies/gensim-data
[11] ConceptNet Numberbatch, https://github.com/commonsense/conceptnet-numberbatch.
[12] G. Sidorov, A. Gelbukh, H. Gomez-Adorno, D. Pinto, "Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model", Computacion y Sistemas Vol. 18, No. 3, 2014, pp. 491-504.
[13] "Finding similar documents with Word2Vec and Soft Cosine Measure", https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/soft-cosine-tutorial.ipynb.
[14] N. Shibata, Y. Kajikawa, I. Sakata, "How to measure the semantic similarities between scientific papers and patents in order to discover uncommercialized research fonts: A case study of solar cells", (2010).
[15] S. Zhang, X. Zheng, C. Hu, "A Survey of Semantic Similarity and its Application to Social Network Analysis", (2015).
[16] H. Pu, G. Fei, H. Zhao, G. Hu, C. Jiao, Z. Xu, "Short Text Similarity Calculation Using Semantic Information", (2017).
[17] W.H. Gomaa and A. A. Fahmy, "A Survey of Text Similarity Approaches", IJCA(0975-8887), Volume68-No.13, (2013).