ISSN 2007-9737

Sentence Similarity Techniques for Short vs Variable Length Text using Word Embeddings

D. Shashavali, V. Vishwjeet, Rahul Kumar, Gaurav Mathur,
Nikhil Nihal, Siddhartha Mukherjee, Suresh Venkanagouda Patil

Samsung R & D Bangalore,
India

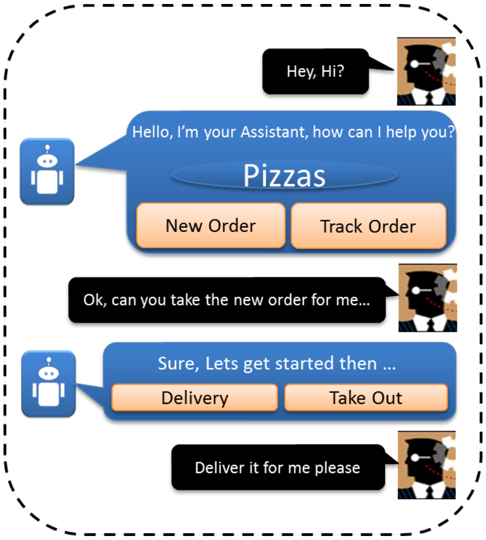
{shasha.d, v.vishwjeet, rahul.k4, gaurav.m4,
nikhil.nihal, siddhartha.m, suresh.patil}@samsung.com

Abstract. In goal-oriented conversational agents like Chatbots, finding the similarity between user input and representative text result is a big challenge. Generally, the conversational agent developers tend to provide a minimal number of utterances per intent, which makes the classification task difficult. The problem becomes more complex when the length of the representative text per action is short and the length of the user input is long. We propose a methodology that derives Sentence Similarity score based on N-gram and Sliding Window and uses the FastText Word Embeddings technique which outperforms the current state-of-the-art Sentence Similarity results. We are also publishing a dataset on the shopping domain, to build conversational agents. And the extensive experiments done on the dataset fetched better results in accuracy, precision and recall by 6%, 2% and 80% respectively. It also evinces that our solution generalizes well on the low corpus and requires no training.

Keywords. Sentence similarity, word embeddings, natural language processing, sliding window, N-grams, text classification.

1 Introduction

The determination of sentence similarity in natural language processing has a wide range of applications. In applications like Chatbots, the uses of sentence similarity include estimating the semantic meaning between the user input and button text. Hence, such applications need to have a robust algorithm to estimate the sentence similarity which can be used across a variety of domains. Well, the main reason we want to infer meaning from raw text is that NLU aims at building systems that understand user utterance and trigger meaningful results based on the user input. Refer Figure 1 for example.



Hey, Hi?

Hello, I'm your Assistant, how can I help you?

Ok, can you take the new order for me...

Sure, Lets get started then ...

Deliver it for me please

Fig. 1. The primary goal of the dialogue systems is to understand the user's input or goal by using NLU techniques, the bot must manage to achieve a goal by showing the appropriate action

Multitask learning [13] schemes along with supervised and unsupervised approaches have lead to the betterment of NLP task results.

A simple approach [6] using WMD (Word Mover's Distance), which measures the dissim-

doi: 10.13053/CyS-23-3-3273

ilarity between two sentences, as the minimum distance that the embedded words of a sentence need to travel to reach the embedded words of other. The recent approach [14] to sentence level semantic similarity technique is based on unsupervised learning from conversational data. This approach process the sentences in a high dimensional space and doesn't fetch better results on short sentences, so it's very hard to learn direct Sentence Embeddings. Also, the most recent Sentence Encoder models [4], Transformer encoder and Deep Averaging Network (DAN) have a trade-off of accuracy and computational resource requirement. Moreover, one needs to build the deep neural networks (DNN) or more sophisticated architectures and train the model with the large corpus.

Here, we propose methods which are based on Cosine similarity calculation along with Sliding window and Weighted N-gram. The proposed approach is fairly simple in architecture and outperforms the latest Universal Sentence Encoder technique [4].

2 Related Work

Sentence similarity has many interesting applications such conversational agent with script strategies [1] and the Internet. The recent work in the area of natural language processing has contributed valuable solutions to calculate the semantic similarity between words and sentences. However, much research has been done on measuring long text similarity, the computation of sentence similarity is far from perfect [7, 5, 8]. We propose to compute sentence similarity between a very short (1-3 words) and lengthy sentences. Bag of word cosine similarity does not take care of word order in a sentence. For example, “Do I not look good?” and “I do not look good.” will have a 100% cosine similarity score. For document similarity, weighted N-Gram over cosine similarity is being suggested in 3.2.2. We took N-Gram weighting formula from the paper [3].

The use of unsupervised word embedding representation of words as vectors, is to preserve semantic information [10]. The Wordwise sum of vectors or average of the vectors also produces a vector with the potential to encode meaning. The mean was used as baseline in [11]. The sum of word embeddings first considered in [10] for short phrases, was found to be an effective model for summarization in [9].

The cosine distance, as is commonly used when comparing distances between embeddings, is invariant between sum and mean of word embeddings. Both sum and mean of word embeddings are computationally inexpensive, given the fact that pre-trained word embeddings are available. Deep learning solutions [12] handle sentence similarity with variable-length but, requires a huge chunk of data to train and is resource heavy to train and maintain.

3 Model Architecture

The proposed methodologies use Word Embeddings and Cosine similarity techniques for word representation and calculating similarity score.

3.1 Word Embedding and Cosine Stacks

Word Embeddings. Word embeddings computed using diverse methods are basic building blocks for Natural Language Processing (NLP) and Information Retrieval (IR). They capture the similarities between words [2]. And as our approach is naturally dependent on a word embedding, we've chosen FastText [3] over other embeddings. Firstly, subword information is taken into consideration in which each word w is represented as a bag of character N-gram. This further signifies that, for previously unseen words (e.g. due to typos), the model can make an educated guess towards its meaning, thus allowing to learn reliable representation for rare words. Inherently, this also allows you to capture meaning for suffixes/prefixes. Second, and most importantly, we notice that the proposed approach provides very good word vectors even when using small training datasets.

Cosine Similarity. The cosine similarity between two vectors (or two sentences on the Vector Space) is a measure to calculate the cosine of the angle between them. This metric is a measurement of orientation and not magnitude. It can be seen as a comparison between sentences on a normalized space because, we're not only taking into consideration the magnitude of each word count (tf-idf) of each document, but also the angle between the sentences. To obtain the equation for cosine similarity, we simply rearrange the equation of dot product between two vectors.

$$\begin{aligned} \vec{a} \cdot \vec{b} &= |\vec{a}| \left|\vec{b}\right| \cos \Theta, \\ \cos \Theta &= \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \left\|\vec{b}\right\|}. \end{aligned} \tag{1}$$

3.2 Approaches

We detail two different methods which are as follows.

3.2.1 Sliding Window with Average Weighted Word Vectors

In language, the meaning of the sentence is reflected by the words in it. Older methods used the weighted average of word embedding to represent the sentence and cosine similarity. But, as we are comparing the similarity between short and long sentences, doing the weighted average on a long sentence doesn't help. Moreover, it reduces the weight of the main action verb in the overall representation, which in turn affects the sentence similarity. To overcome this, we use the sliding window approach (Fig. 2) on a long sentence, so that the main action verb weight will be the same in both inputs.

After applying sliding window on S_2, we get a list of substrings $S_{2'}$. For vector representation of every window, we iterate through the $S_{2'}$ and take the weighted average of word embedding, to find the cosine similarity with S_1. The final similarity score for S_1 and S_2 is taken as the maximum score, obtained from the window comparisons. In Chatbot application, False Positive must be very less for better user experience. We tried the weighted N-gram approach to further reduce false positives.

3.2.2 Weighted N-gram Vectors

N-grams are consecutive strings of N words, for example, trigrams are all possible three word long substrings of a given sentence. To compare two sentences, the sentences are tokenized into unigram, bigram and trigram.

For every unigram of sentence S_1, find similarity with every unigram of sentence S_2 and select the maximum score as match score for that unigram. All the selected unigram scores are averaged over to get a final unigram score:

$$score_1 = \frac{1}{N_1} \sum_{n=1,n'=1}^{N_1 N_2} \underset{n}{max}(similarity(S_1 U_n, S_2 U_{n'})),$$
$$N_1 = number\ of\ unigrams\ in\ S_1,$$
$$N_2 = number\ of\ unigrams\ in\ S_2.$$

Likewise, for every bigram of S_1, find similarity with every bigram of S_2 and select maximum score as match for that bigram. All the selected bigram scores are averaged over to get a final bigram score:

$$score_2 = \frac{1}{N_1} \sum_{n=1,n'_1=1}^{N_1 N_2} \underset{n}{max}(similarity(S_1 B_n, S_2 B_{n'})),$$
$$N_1 = number\ of\ bigrams\ in\ S_1,$$
$$N_2 = number\ of\ bigrams\ in\ S_2.$$

The final similarity score of the sentences is taken as the weighted sum of the final similarity scores of unigrams, bigrams and trigrams:

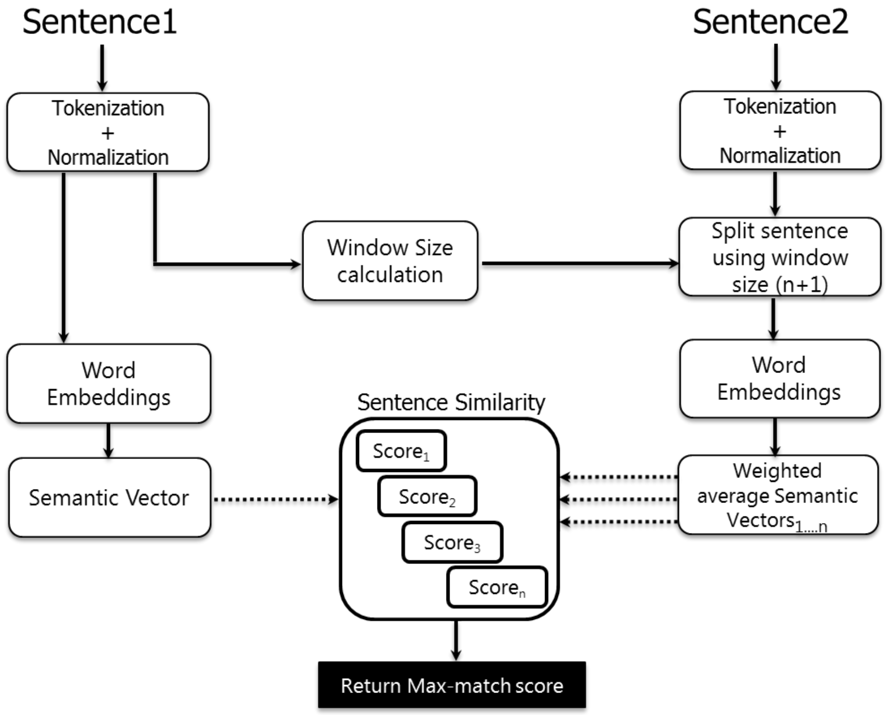
$$sentence\ similarity = \sum_{g=1}^{G} w_g * score_g,$$

where

$$w_g = \frac{g}{\sum_{g=1}^{G} g}.$$

As discussed in section 3.1, we used cosine similarity on averaged word embedding to calculate similarity between N-grams.

doi: 10.13053/CyS-23-3-3273



❖ Sentence1 is of short-length and Sentence2 is of varied-length

Fig. 2. Sliding window approach

4 Results

Here, we describe the data set, which is a conversational data found in Chabot builder based NLP engine environment. We then compare the 3.1 and 3.2 sections with latest Google's Universal Sentence encoder based sentence similarity approach.

4.1 Dataset

Although, many datasets are accessible, there are currently no suitable benchmarks (or even standard text sets) for the evaluation of similarity between long and short sentences. We release a dataset [1] which is very specific to conversational agents problem statement. Here, the dataset has been structured into two columns, first, the long sentence which imitates user input and second, the short sentence which typically resembles the button text in the chat conversation.

Table 1. Sample test dataset

User Input	Button Text
Scrap my order	Cancel Order
Junk my order	
Drop my order	
Display recently viewed items	Show recent items
Open items I just viewed	
Show my last seem items	

4.2 Sentence Similarity

A testing instance is a pair of button text and user input. The similarity score between each user input and button text is calculated. Based on similarity score, comparison is categorized as positive or negative. Comparison between button

[1] https://github.com/shashavali-d/SentenceSimilarity

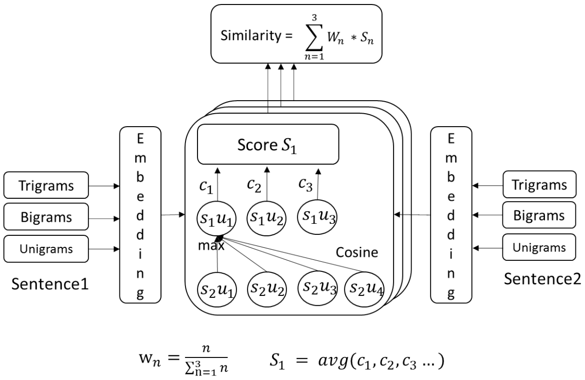


Fig. 3. Weighted N-gram approach

Table 2. Results with our approaches vs Universal Sentence Encoder

Approaches/ Metrics	Google Universal Sentence Encoder	Sliding Window with avg. Weighted Vectors	Weighted N-gram Vectors
Recall	0.0789	0.2593	0.9408
Precision	0.9022	0.6507	0.9226
F1 Score	0.1451	0.3708	0.9316
Accuracy	0.9256	0.9298	0.9880

text and user input is deemed positive, if the similarity score is above threshold (0.9). Similarly, the comparison between button text and user input is deemed negative, if the similarity score is below the threshold (0.9). We used the performance metric precision, F1 Score and recall for evaluating our solution.

Our model outperformed Google's sentence similarity in F1 and Recall, see Table 2.

5 Conclusion

In the development stages of Chatbots, the current bot platforms provided ML solutions and required large training data from developers. And, the platform had to manage multiple data perpetually and the process became complex and expensive to train the model every time.

In this paper, we propose the sliding window with average weighted word vectors and Weighted N-gram vectors for developing the input semantics vector. The proposed method replaces the sentence embedding approach with simple word embedding based sentence representation and also it doesn't need large dataset for training.

We are excited about the execution of our approaches and will apply the same to other text classification tasks in the near future. We plan to improve the word representation using dependency and constituency parsing information and also, to apply other vector Similarity method than cosine, for the betterment of results.

References

1. **Allen, J. (1988).** *Natural Language Understanding*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA.

2. **Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003).** A neural probabilistic language model. *J. Mach. Learn. Res.*, Vol. 3, pp. 1137–1155.

3. **Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016).** Enriching word vectors with subword information. *CoRR*, Vol. abs/1607.04606.

4. **Cer, D., Yang, Y., Kong, S., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., & Kurzweil, R. (2018).** Universal sentence encoder. *CoRR*, Vol. abs/1803.11175.

5. **Hatzivassiloglou, V. & Wiebe, J. M. (2000).** Effects of adjective orientation and gradability on sentence subjectivity. *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING '00, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 299–305.

6. **Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015).** From word embeddings to document distances. *ICML*.

7. **Landauer, T. K., Foltz, P. W., & Laham, D. (1998).** An introduction to latent semantic analysis. *Discourse Processes*, Vol. 25, No. 2-3, pp. 259–284.

8. **Landauer, T. K., Laham, D., Rehder, B., & Schreiner, M. E. (1991).** How well can passage meaning be derived without using word order: A comparison of latent semantic analysis and humans. *Proc. of the 19th annual meeting of the*

Cognitive Science Society, Erlbaum, Mawhwah, NJ, pp. 412–417.

9. **Le, Q. V. & Mikolov, T. (2014).** Distributed representations of sentences and documents. *CoRR*, Vol. abs/1405.4053.

10. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013).** Distributed representations of words and phrases and their compositionality. *CoRR*, Vol. abs/1310.4546.

11. **Pennington, J., Socher, R., & Manning, C. D. (2014).** Glove: Global vectors for word representation. *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

12. **Ramaprabha, J., Das, S., & Mukerjee, P. (2018).** Survey on sentence similarity evaluation using deep learning. *Journal of Physics: Conference Series*, Vol. 1000, pp. 012070.

13. **Subramanian, S., Trischler, A., Bengio, Y., & Pal, C. J. (2018).** Learning general purpose distributed sentence representations via large scale multi-task learning. *International Conference on Learning Representations*.

14. **Yang, Y., Yuan, S., Cer, D., Kong, S., Constant, N., Pilar, P., Ge, H., Sung, Y., Strope, B., & Kurzweil, R. (2018).** Learning semantic textual similarity from conversations. *CoRR*, Vol. abs/1804.07754.

Article received on 26/02/2019; accepted on 04/03/2019.
Corresponding author is D. Shashavali.