Sentence similarity techniques for short vs variable length text using Word Embeddings

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

Samsung R & D Bangalore - India Pvt. Ltd.
shasha.d@samsung.com, v.vishwjeet@samsung.com, rahul.k4@samsung.com, gaurav.m4@samsung.com, nikhil.nihal@samsung.com, siddhartha.m@samsung.com, suresh.patil@samsung.com

Abstract. In goal-oriented conversational agents like Chatbots, finding the similarity between predefined text per action and user input of variable length is a hard NLP problem. Usually, the Conversational agent developers often tend to provide minimal number of sentences (less than eight) per intent which makes the classification task difficult. The problem becomes more complex when length of the representative or predefined text per action is short (less than four words) and the length of the user input is long. We propose a methodology which derives Sentence Similarity score based on N-gram and Sliding Window approaches using FastText Word Embeddings which outperforms the current state-of-the-art sentence similarity techniques. We referred a dataset related to shopping domain to build Conversational agents. Extensive experiments on the dataset achieved improvement of 6% in accuracy, 2% in precision and 80% in recall in Classification task based on the existing sentence similarity techniques. It also shows that our solution generalizes well on low corpus and requires no training.

Key words: Sentence Similarity, Word Embeddings, Natural Language Processing, Sliding Window, N-grams, Text Classification.

1 Introduction

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 text input and button text. Hence, such applications need to have a robust algorithm to estimate the sentence similarity which can be used across variety of domains. Well there are several reasons we want to infer meaning from raw text. One reason is that the field of NLU aims at building systems that understand what you say or write to them, trigger actions based on that and convey back meaningful information. Refer Figure 1 for example.

There are currently many competing schemes for learning sentence embeddings [1]. While simple baselines like averaging word embeddings works consistently. A few novel unsupervised and supervised approaches, as well as multitask learning [2] schemes have emerged and lead to interesting improvements.

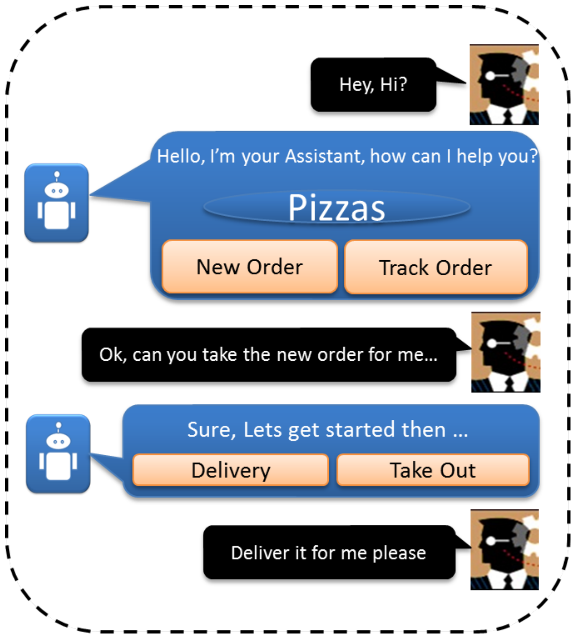
Hey, Hi?
Hello, I'm your Assistant, how can I help you?
Pizzas
New Order
Track Order
Ok, can you take the new order for me...
Sure, Lets get started then ...
Delivery
Take Out
Deliver it for me please

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.

A simple approach [3] using WMD (Word Mover's Distance), which measures the dissimilarity between two sentences as the minimum amount of distance that the embedded words of one sentence need to travel to reach the embedded words of another sentence. The most recent approach [4] to sentence level semantic similarity is based on unsupervised learning from conversational data. These approaches process the sentences in a high dimensional space and doesn't work on small sentences so it's very hard to learn direct sentence embeddings. And the most recent Sentence Encoder models [5] Transformer encoder and Deep Averaging Network (DAN) have a trade-off of accuracy and computational resource requirement. And moreover you need to build the deep neural networks (DNN) or more sophisticated architectures and train the model with 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 [5].

2 Related Work

Sentence similarity has many interesting applications. Examples include conversational agent with script strategies [6] 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. Although much research has been done on measuring long text similarity, the computation of sentence similarity is far from perfect [7, 8, 9]. We propose to compute sentence similarity between a very short sentence (1-3 words) and a variable-length sentence.

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 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 [10].

The use of unsupervised word embedding representation of words as vectors to preserve semantic information [11]. Word wise 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 [12]. The sum of word embeddings first considered in [11] for short phrases, it was found to be an effective model for summarization in [13]. 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 cheap models, particularly given pretrained word embeddings are available. Deep learning solutions [14] handles 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 for word representation and calculate similarity score respectively.

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 [15] and as our approach is naturally dependent on a word embedding we've chosen FastText [10] over other embeddings because first, it takes in to account subword information i.e., each word w is represented as a bag of character n-gram. That means that even 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 very small training datasets.

Cosine similarity. The cosine similarity between two vectors (or two sentences on the Vector Space) is a measure that calculates 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 taking into the consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the sentences. In order to obtain the equation for cosine similarity we simply rearrange the equation of dot product between two vectors.

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

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 the sentence. Previously established methods to estimate the semantic similarity between sentences, use the weighted average of word embedding to represent the sentence and cosine similarity to find the sentence similarity. But as we are comparing the short and long sentences similarity, doing the weighted average on long sentence doesn't help, moreover it reduces the weightage of main action verb in the overall representation which finally affects the sentence similarity. To overcome this we use sliding window approach on long sentence, so that main action verb weightage will be same in both inputs.

After applying sliding window on S2 we get a list of substrings S2'. We iterate through the S2', for vector representation of every window take the weighted average of word embedding and find the cosine similarity for every window with S1. The final similarity score for S1 and S2 is taken as the maximum score obtained from the window comparisons. In Chatbot application False Positive must be very less for better user experience. To further reduce false positives, we tried weighted N-Gram approach.

3.2.2 Weighted n-gram vectors. N-grams are consecutive string 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 S1 find similarity with every unigram of sentence S2 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.

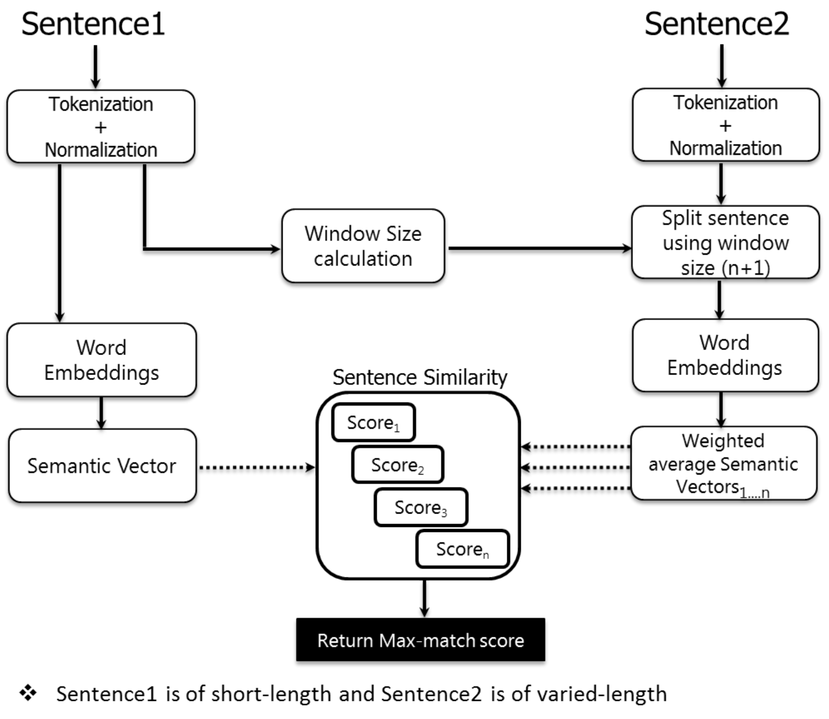
Sentence1
Sentence2
Tokenization + Normalization
Tokenization + Normalization
Window Size calculation
Split sentence using window size (n+1)
Word Embeddings
Word Embeddings
Sentence Similarity
Score$_1$
Score$_2$
Score$_3$
Score$_n$
Semantic Vector
Weighted average Semantic Vectors$_{1,...,n}$
Return Max-match score
Sentence1 is of short-length and Sentence2 is of varied-length

Fig. 2. Sliding window approach.

$$score_1 = \frac{1}{N_1} \sum_{n=1,n'1=1}^{N_1 N_2} \max_n(similarity(S_1U_n, S_2U_{n'}))$$
$$N_1 = number\ of\ unigrams\ in\ S_1$$
$$N_2 = number\ of\ unigrams\ in\ S_2$$

Similarly, for every bigram of S1 find similarity with every bigram of S2 and select maximum score as match for that bigram. All the selected bigram scores are averaged over to get a final bigram score.

$$score_2 = \frac{1}{N_1} \sum_{n=1,n'1=1}^{N_1 N_2} \max_n(similarity(S_1B_n, S_2B_{n'}))$$
$$N_1 = number\ of\ bigrams\ in\ S_1$$
$$N_2 = number\ of\ bigrams\ in\ S_2$$

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

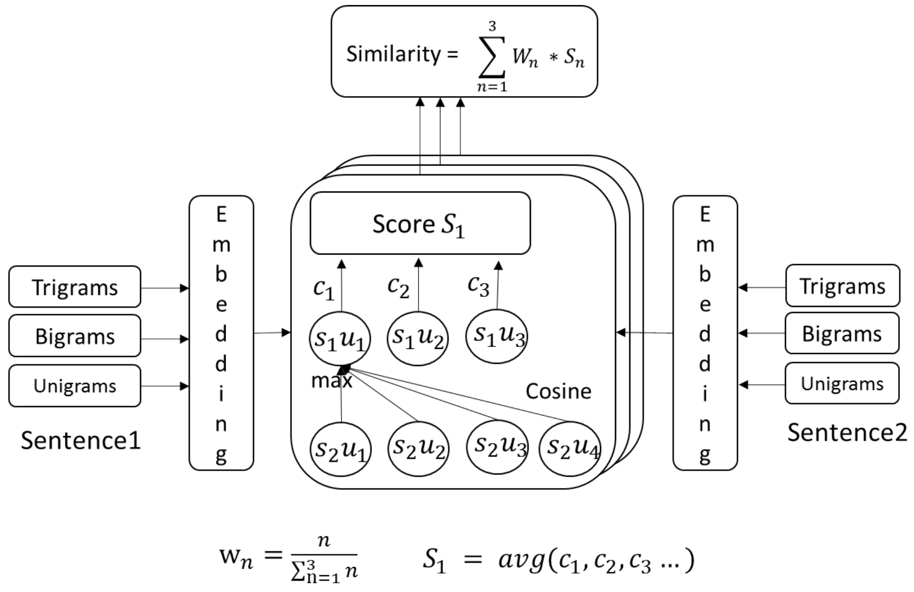

Fig. 3. Weighted n-gram approach.

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

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

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 a few related studies have been published, there are currently no suitable benchmarks datasets (or even standard text sets) for the evaluation of similarity between long and short sentences. Dataset has been generated for the evaluation of long and short sentence similarity scenarios which is very specific to conversational agents. Here, the dataset has been structured in to two columns, first the long sentence which imitates user input and the second have short text which is typically resembles the button text in the chat conversation.

Table 1. Sample test dataset.

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

4.2 Sentence Similarity

A testing instance is a pair of button text and its user input, the similarity between each user input and button text is calculated. Based on similarity score each comparison is categorized as positive or negative. Comparison between button text and its user input is deemed positive if the similarity score is above threshold (0.9). Similarly, the comparison between button text and other user input is deemed positive if the similarity score is below the threshold (0.9). We used the performance metric precision, F1 Score and recall for evaluating our solution.

Table 2. Results with our approaches vs Universal Sentence Encoder.

Metrics/Approaches	Google Universal Sentence Encoder	Sliding Window with Average Weighted Vectors	Weighted N-gram Vectors
Recall	0.078944153	0.259318293	0.94086506
Precision	0.902234637	0.650769231	0.922677565
F1 Score	0.145184852	0.370857443	0.931682561
Accuracy	0.925640274	0.929853372	0.988025415

Our model performed better compared to Google's sentence similarity in terms of F1 and Recall.

5 Conclusion

In this work, we proposed the sliding window with average weighted word vectors and Weighted n-gram vectors for developing the input semantics vector. Proposed method replaces the sentence embedding approach with simple word embedding based sentence representation.

For sentence similarity of long vs short sentence, our approaches do not need large dataset for training. The recent conversational agent platforms provides ML solutions as a service and developers provides dynamic data for training, the platform needs to manage multiple developers data at a time. And, it's very complex and costly task to train model every time. We referred a dataset[1] because the problem doesn't have any suitable benchmarks. On Sentence Similarity tasks, we achieved much improved results than latest Google's solution and also it outperformed many previously reported ensembles.

We are excited about the future use of our approaches and plan to apply for text classification tasks.We plan to improve the word representation using dependency parsing and constituency parsing information. And, we are also, planning to apply other vector Similarity method than cosine.

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

Bibliography

[1] Rücklé, A., Eger, S., Peyrard, M., Gurevych, I.: Concatenated p-mean word embeddings as universal cross-lingual sentence representations. CoRR **abs/1803.01400** (2018)

[2] Subramanian, S., Trischler, A., Bengio, Y., Pal, C.J.: Learning general purpose distributed sentence representations via large scale multi-task learning. In: International Conference on Learning Representations. (2018)

[3] Kusner, M.J., Sun, Y., Kolkin, N.I., Weinberger, K.Q.: From word embeddings to document distances. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, JMLR.org (2015) 957–966

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

[5] 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.: Universal sentence encoder. CoRR **abs/1803.11175** (2018)

[6] Allen, J.: Natural Language Understanding. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA (1988)

[7] Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse Processes **25** (1998) 259–284

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

[9] Landauer, T.K., Laham, D., Rehder, B., Schreiner, M.E.: How well can passage meaning be derived without using word order ? a comparison of latent semantic analysis and humans. (1997)

[10] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. CoRR **abs/1607.04606** (2016)

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

[12] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). (2014) 1532–1543

[13] Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR **abs/1405.4053** (2014)

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

[15] Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3** (2003) 1137–1155