# PurdueNLP at SemEval-2017 Task 1: Predicting Semantic Textual Similarity with Paraphrase and Event Embeddings

**I-Ta Lee, Mahak Goindani, Chang Li, Di Jin, Kristen Marie Johnson**
**Xiao Zhang, Maria Leonor Pacheco, Dan Goldwasser**
Department of Computer Science, Purdue University
West Lafayette, IN 47907
{lee2226,mgoindan,li1873,jind,john1187,
zhang923,pachecog,dgoldwas}@purdue.edu

## Abstract

This paper describes our proposed solution for SemEval 2017 Task 1: Semantic Textual Similarity (Daniel Cer and Specia, 2017). The task aims at measuring the degree of equivalence between sentences given in English. Performance is evaluated by computing Pearson Correlation scores between the predicted scores and human judgements. Our proposed system consists of two subsystems and one regression model for predicting STS scores. The two subsystems are designed to learn Paraphrase and Event Embeddings that can take the consideration of paraphrasing characteristics and sentence structures into our system. The regression model associates these embeddings to make the final predictions. The experimental result shows that our system acquires 0.8 of Pearson Correlation Scores in this task.

## 1 Introduction

The SemEval Semantic Textual Similarity (STS) task (Daniel Cer and Specia, 2017) is to assess the degree of similarity between two given sentences and assign a score on a scale from 0 to 5. A score of 0 indicates that the two sentences are completely dissimilar, while a score of 5 indicates that the sentences have the same meaning. Predicting the similarity between pieces of texts finds utility in many NLP tasks such as question-answering, and plagiarism detection.

In this paper, we proposed a system to facilitate STS task. Our system includes training two types of embeddings–Paraphrase Embeddings (PE) and Event Embeddings (EE)–as features to assess STS. For the first type of embeddings, PE, we exploit two crucial properties for measuring sentence similarity: paraphrasing characteristics and sentence structures. The paraphrasing characteristics help identifying if two sentences share the same meaning. Our system incorporates it using an unsupervised learning step over the Paraphrase Database (PPDB; Ganitkevitch et al. 2013), which is inspired by Wieting et al. 2015a. The sentence structure, on the other hand, can detect structural differences, which reflect different aspects of the similarity between the input sentences. Our system employs a Convolutional Neural Network (CNN) to strengthen the embedding by including the sentence structure into our representation. The second type of embeddings, EE, conveys the distributional semantics of events in a narrative setting, associating a vector with each event.

In the last part of our system, we build a regression model that associates the two distributed representations and predicts the similarity scores.

## 2 System Description

Our system builds two types of embedding models, Paraphrase Embeddings (PE) and Event Embeddings (EE), and trains a regression model for predicting the similarity score between two sentences, which is described in this Section 2.3.

### 2.1 Paraphrase Embeddings

The Paraphrase Database (PPDB) is a large scale database containing millions of automatically extracted paraphrases. Wieting et al. 2015a show that by training word embeddings on PPDB (Ganitkevitch et al., 2013), paraphrase information can be captured by the embeddings, which is very useful for the STS task. Their system works well when word overlaps reflect sentence similarities, which is the most common case in the STS dataset. We extend their work by introducing a Convolutional Neural Network (CNN) model, because it better accounts for sentence structure.

*Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 198–202,
Vancouver, Canada, August 3 - 4, 2017. ©2017 Association for Computational Linguistics

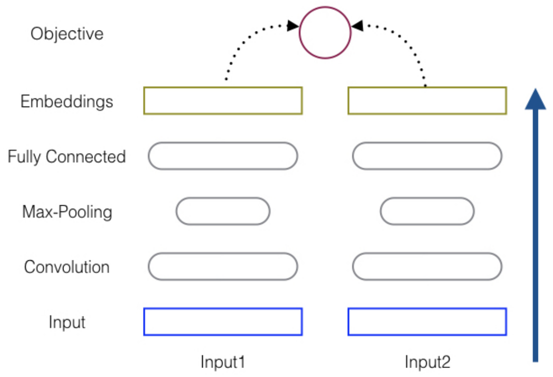

Figure 1: The convolutional neural network architecture consists of two networks that share the network parameters. The networks are constructed by a convolutional layer, a max-pooling layer, and two fully connected layers.

Figure 1 describes our network architecture. Each input example consists of a pair of sentences/phrases. The initial input representation for each sentence is created by averaging the word vectors of the words in the sentence. The initial word vectors can rely on pre-trained word embeddings, such as Word2Vec (Mikolov et al., 2013) or Glove (Pennington et al., 2014).

This input layer is followed by a convolutional layer, a max-pooling layer, and two fully connected layers. The projected outputs (the embeddings layer in Figure 1) comprise the PE that will later be used for regression. Note that the two networks in Figure 1 share the network parameters. During training, the errors back-propagate not only to the network, but also to the embeddings. To train PE, we adopt a 2-step framework inspired by Wieting et al. 2015a and initialize our word embedding look-up table with the best performing embeddings they released–*PARAGRAM-PHRASE XXL*. In the first step, we train the CNN on PPDB 2.0 (Pavlick et al., 2015) and aim at making PE a quality representation for paraphrase-related tasks. The objective function here is a margin-based ranking loss (Wieting et al., 2015a):

$$\begin{aligned}\min_{W_c,W_w} \Bigg( \sum_{<x_1,x_2>\in X} & max(0, \delta - cos(g(x_1), g(x_2)) \\ & + cos(g(x_1), g(t_1))) \\ & + max(0, \delta - cos(g(x_1), g(x_2))) \\ & + cos(g(x_2), g(t_2))\Bigg) \\ & + \lambda_c ||W_c||^2 + \lambda_w ||W_{init} - W_w||^2,\end{aligned}$$

where $X$ is all the positive paraphrasing pairs; $\delta$ is the margin [1]; $g(\cdot)$ is the functional representation of CNN; $\lambda_c$ and $\lambda_w$ are two hyperparameters for L2-regularization; $W_c$ is the parameters to be trained; $W_w$ is the most recent word embeddings; $W_{init}$ is the initial word embeddings; and $t_1$ and $t_2$ are negative examples. The negative examples are randomly and uniformly selected from other examples. That is, for $x_1$, we randomly select a phrase $t_1$ from the corpus, which is nearly unlikely to be a paraphrase to $x_1$. The same strategy is also applied to select $t_2$ for $x_2$.

In the second step, we fine-tune the PE by fitting it to SemEval STS data. This is a supervised regression task, with an objective function that considers both the distances and angles of the two projected embeddings. This regression objective is the same as the one that we will describe in Section 2.3. Although the objective function used here and in Section 2.3 are the same, they are used differently. The intention of using it in this step is to adjust the PE representations, while the regression model in Section 2.3 is used for combining different embeddings for regression. More details will be discussed in Section 2.3.

## 2.2 Event Embeddings

Word embeddings capture distributional semantics. It is a function that maps a word to a dense, low-dimension vector. With the same concept in mind, we can infer event semantics by exploring its contextual events to build EE. Similar ideas have be explored in several recent works (Granroth-Wilding and Clark, 2016; Pichotta and Mooney, 2016; Pacheco et al., 2016).

Our EE model is constructed as follows: first, we extract event tokens, similar to narrative scripts construction (Chambers and Jurafsky, 2008). We resolve co-referent entities and run a dependency parser on all documents [2]. For each entity in a co-reference chain, we represent an event token $e$ by its predicate $p(e)$, a dependency relation to the entity $d(e)$, and animacy of the entity $a(e)$; resulting in a triplet $((p(e), d(e), a(e)))$. An event chain thus can be constructed by corresponding all the entities in a co-reference chain to event tokens.

We extend the definition of the event predicate $p(e)$ to include lemmatized verbs and predicative adjectives. These extensions are useful as they

[1] $\delta$ is tuned over $\{0.4, 1\}$ in our evaluation.

[2] we use Stanford CoreNLP library (Manning et al., 2014)

capture important information about the state of the entity. For example, “Jim was hungry. He ate a sub”. The word “hungry” captures meaningful narrative information that should be included in the event chain of the entity “Jim”, so the resulting chain here should be: *(hungry, subj, animate), (eat, subj, animate)*. Moreover, relying on verb predicates alone is sometimes insufficient, when the verbs are too ambiguous on their own, e.g., verbs like *go*, *get*, and *have*. For such weak verbs, we include their particles and clausal complement (xcomp) in the predicates, e.g., “have to sleep” will be represented as one predicate–*have_to_sleep*. Lastly, negations to the predicate matter a lot to event semantics, so we also include it as a part of predicates. For instance, “did not sleep” will be represented as *not_sleep*.

For dependencies $d(e)$, we only consider subjects, objects, and indirect objects in the dependency tree. Argument animacy information $a(e)$ is also included, because the entity’s animacy often changes the event semantics. For instance, the difference in meaning of the phrases “killed a joke” and “killed a person” is hard to identify without including the object’s animacy information. There are three possible animacy types that are represented in our triplet: *animate*, *inanimate*, or *unknown*.

The Skip-Gram model (Mikolov et al., 2013), which predicts contextual tokens given a current token, is then used for training EE. The model treats each event token as a word and each event chain as a sentence, and learns EE by optimizing the following objective:

$$p(C(e)|e) = \prod_{e' \in C(e)} P(e'|e)$$
$$= \prod_{e' \in C(e)} \frac{exp(v_{e'}, v_e)}{\sum_{e^* \in E} exp(v_{e^*}, v_e)},$$

where $e$ is the current event, $C(e)$ is the contextual events of $e$, and $v_e$ is the embedding representation of $e$.

To make the computation feasible, the negative sampling strategy is again used here. For each pair of event tokens in a sliding window, we sample $k$ negative tokens. Other optimizing strategies for improve embedding quality used by Mikolov et al. 2013 are also applied here, such as subsampling for high-frequency tokens and filtering low-frequency tokens. The followings are the hyperparameters related to PE that are used in our system: the sub-sampling rate is empirically set to 0.001; the minimum count of tokens is set to 5; the sliding window size and $k$ are set to 5; and the vector dimension is set to 300.

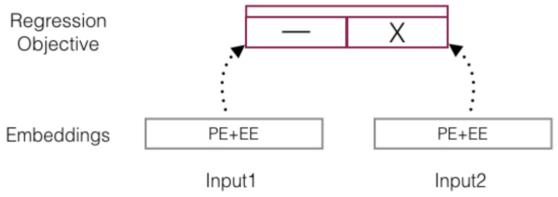

Figure 2: The regression model that considers the distance and angle between the two inputs.

### 2.3 Regression

In this section, we discuss how to fuse the different embedding representations in the final regression model that predicts a similarity score between the two input sentences. The objective function is shown below:

$$h_* = v_{e1} \otimes v_{e2} \quad (1)$$
$$h_\triangle = |v_{e1} - v_{e2}| \quad (2)$$
$$h = \tanh(W_* \cdot h_* + W_\triangle \cdot h_\triangle) \quad (3)$$
$$p = \text{softmax}(W \cdot h), \quad (4)$$

where $v_{e1}$ and $v_{e2}$ are vector representations of input 1 and input 2 respectively; $W_* \in R^{d \times k}$, $W_\triangle \in R^{d \times k}$, and $W \in R^{6 \times k}$ are the parameters to be trained; $d$ is the total dimension of PE and EE; $k$ is a hyperparameter of hidden layer size (the 6 in the first dimension of $W$ is from the softmax layer outputs which account for the probabilities of integer scores between 0-5). The final score is calculated by taking the mean of the 6 softmax outputs. This regression model is visualized in Figure 2. The PE and EE are concatenated to represent each input. They are fixed representations that will not be updated during the regression. The ”X” and ”-” shown in Figure 2 are element-wise products and element-wise differences between two input representations (Equation (1) and (2)). They represent the angles and distances between the input sentences. This regression objective has been shown to be very useful in text similarity tasks (Tai et al., 2015).

## 3 Evaluation

We train PE using two datasets, PPDB 2.0 (Pavlick et al., 2015) and SemEval STS data. These are

| | Train | Dev. | Test |
|---|---|---|---|
| W2V | 0.3060 | 0.2442 | 0.2641 |
| EE | 0.2491 | 0.2458 | 0.3545 |
| paragram-small | 0.6723 | 0.5446 | 0.6989 |
| paragram-XXL | 0.6639 | 0.6610 | 0.7322 |
| PE | 0.8138 | 0.6896 | 0.7979 |
| PE+W2V | 0.8214 | 0.6879 | 0.7961 |
| PE+EE (official) | - | - | 0.7928 |
| PE+EE | 0.8243 | **0.6932** | **0.8015** |
| Winner STS2017 | - | - | 0.8547 |

Table 1: Pearson Correlation Scores for the models we tested, where the Train data is STS2012-2015, Dev. data is STS2016, Test data is STS2017. The best scores of our model are in bold fonts.

used in the first (Section 2.1) and second steps (Section 2.2) respectively. We used the New York Times (NYT) section of the Gigaword corpus (Parker et al., 2011) for training EE and our baselines. The SemEval STS data is also used in training the final regression model. The data splits are as follows: SemEval STS2012-2015 was used as the training set, STS2016 data was used as the development set, and STS2017 was used as the test set. After the development stage was finished, the training and development sets were both used to train a final model with the best hyperparameters.

To update the parameters, Mini-batch Stochastic Gradient Descent is used for optimizing the parameters and Adagrad (Duchi et al., 2011) is used to update the learning rate while training. The batch size is set to 100 and the number of epochs is set to 10. $L2$-regularization is included in all the objective functions and the $\lambda$ is tuned over $\{1e{-}5, 1e{-}6, 1e{-}7, 1e{-}8\}$. Both PE and EE's dimensions are set to 300.

The first baseline we compare with is the Word2Vec Skip-Gram (W2V; Mikolov et al. 2013) model, one of the most popular universal word embeddings. It was trained over the same corpus as EE (NYT section of Gigaword). The second baseline (paragram-small) and third baseline (paragram-XXL) are the best performing word embeddings for STS tasks shown in Wieting et al. 2015b,a. In order to represent the input sentences with the word embeddings, we average the word embeddings based on the words in the input sentences. This approach has been shown to be effective in Wieting et al. 2015a,

Table 1 lists the Pearson Correlation Score of SemEval 2017 STS tasks. We can see that the general embedding models, (W2V and EE), do not perform well as their general purpose representation does not fit the textual similarity task. On the other hand, paragram-small and paragram-XXL which were trained with the textual-similarity-related data (PPDB and STS data) perform reasonably well. The PE model, which takes paragram-XXL as the initial embeddings and tunes all the parameters on a CNN, gets higher score in both development and test sets. The performance further increases as we introduce EE to be parts of input representations (PE+EE), while the W2V does not provide such improvement (PE+W2V).

PE is specifically designed for identifying paraphrasing characteristics and sentence structures, which we believe are the keys to STS task, resulting in the strongest feature set in our system. We do not expect that using EE alone will give high performance, since considerable amounts of information are filtered out during event chain extraction. In addition, EE does not use any STS-related data during training. However, it is still helpful for capturing high-level event semantics, which can be a complement to our PE.

The official result of PE+EE is also included in Table 1. Our best results improve on it, by fine tuning the model's hyperparameters. In addition, the best performing system of SemEval STS2017 acquires the score of 0.8547, outperforming our model. However, it is not clear that what external resources or hand-crafted features were used in their work. Our system, nevertheless, can accommodate additional resources and features. We believe that our results can be further improved by including such information and we will look into it in the future.

## 4 Conclusion

In this paper, we describe our system for SemEval 2017 STS task which consists three key components relevant to this task: paraphrasing characteristics, sentence structures, and event-level semantics. To incorporate the first two ideas into the system, PE–a CNN model trained with a paraphrase database–is used. It measures sentence similarity in terms of paraphrasing and structure similarities. We capture event semantics using EE, and include it in our system. It complements the PE and further boosts performance. Our full system was able to achieve a 0.8 of Pearson Correlation Score.

## References

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*. Citeseer, volume 94305, pages 789–797.

Mona Diab Eneko Agirre Inigo Lopez-Gazpio Daniel Cer and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. *Proceedings of SemEval* .

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*. Association for Computational Linguistics, Atlanta, Georgia, pages 758–764. http://cs.jhu.edu/ ccb/publications/ppdb.pdf.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *AAAI*. pages 2727–2733.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Maria Leonor Pacheco, I-Ta Lee, Xiao Zhang, Abdullah Khan Zehady, Pranjal Daga, Di Jin, Ayush Parolia, and Dan Goldwasser. 2016. Adapting event embedding for implicit discourse relation recognition. *ACL 2016* page 136.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword. *Linguistic Data Consortium* .

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*. pages 2800–2806.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198* .

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015b. From paraphrase database to compositional paraphrase model and back. *arXiv preprint arXiv:1506.03487* .