Semantic Textual Similarity

Team KLM

Beeraka Krupa Kiranmai (2021201022)

Padigala Lakshman Sai (2021201069)

Nalluru Maneesh Gupta (2021201041)

INTRODUCTION

Semantic Textual Similarity (STS) is a dynamic subfield within the realm of Natural Language Processing (NLP), which aims to determine the degree of semantic similarity or relatedness between pairs of sentences or texts. This branch of NLP involves the creation of algorithms and models that can effectively capture the semantic content of text and use it to compute similarity scores between pairs of sentences. Such scores are useful in a wide range of applications, including text classification, information retrieval, machine translation, and question answering. STS methods make use of various linguistic and machine learning techniques to capture the meaning of words, phrases, and sentences, and represent them in a high-dimensional vector space. By calculating the distance or similarity between the vector representations of sentences, the degree of semantic similarity between them can be determined. STS is a rapidly growing field with a plethora of applications in numerous areas of NLP and beyond.

PROBLEM STATEMENT

The primary objective of this project is to develop methods for assessing the degree of similarity between pairs of sentences using Semantic Textual Similarity (STS) measures. The focus of the project is on two specific sub-tasks: **monolingual assessment in English-English and Spanish-Spanish**, and semantic relatedness classification. In the monolingual sub-task, the project aims to evaluate sentence pairs in a given language, whereas the semantic relatedness classification sub-task is designed to determine whether sentence pairs are related or not, using a **Gold Standard scale ranging from 0 to 5**. A score of 5 indicates that the input sentences convey the same meaning, while a score of 0 indicates that the sentences have no semantic similarity. Specifically, the monolingual task will involve assessing the similarity between English-English and Spanish-Spanish sentences. The successful completion of this project will contribute to the development of accurate and efficient methods for comparing the meaning of text fragments in natural language processing.

Example 1:

Sentence 1 : The bird is bathing in the sink.

Sentence 2 : Birdie is washing itself in the water basin.

STS score: 5.0

Example 2:

Sentence 1 : They flew out of the nest in groups.

Sentence 2 : They flew into the nest together.

STS score: 2.0

Literature Review

In order to gain a comprehensive understanding of the problem of Semantic Textual Similarity (STS) and its associated sub-tasks of monolingual assessment and semantic relatedness classification, a literature review was conducted. The review focused on a set of papers that provide solutions to the aforementioned problems. The aim of the review was to identify the key concepts, methods, and techniques that have been used in previous research, and to evaluate their effectiveness in achieving the desired outcomes.
This section presents an overview of the relevant literature, highlighting the strengths and limitations of previous approaches, and identifying areas for further research.

1. The paper "A Unified Framework for Semantic Similarity in Textual, Graph, and Knowledge-Based Spaces" by Rada Mihalcea et al. (2011) presents a comprehensive approach to Semantic Textual Similarity (STS) that incorporates multiple types of semantic representations to measure similarity between sentences. The authors propose a unified framework that includes methods for measuring similarity based on various features such as lexical, syntactic, semantic, and knowledge-based information. The framework is designed to accommodate different types of semantic spaces, including textual, graph, and knowledge-based representations. The authors demonstrate the effectiveness of their approach by evaluating it on several benchmark datasets and comparing its performance to other state-of-the-art methods. The paper highlights the importance of incorporating multiple types of semantic information to accurately capture the meaning of text, and provides a valuable contribution to the field of STS research.

2. The paper "SemEval-2012 Task 6: A Pilot on Semantic Textual Similarity" by Eneko Agirre et al. (2012) describes a methodology for measuring the degree of semantic similarity between pairs of sentences from different domains and languages. The authors also explored the use of combination methods, where they combined supervised and unsupervised techniques to compute similarity scores. They used different types of features such as lexical, syntactic, and semantic features and combined them in various ways to improve the performance of their methods.

3. "Distributed Representations of Words and Phrases and their Compositionality" by Tomas Mikolov et al. (2013): This paper introduces the word2vec algorithm, which learns distributed representations of words and phrases from large amounts of text data. The paper shows that these representations can capture semantic relationships between words and can be used to compute similarity between sentences.

4. "STS Benchmark: A Benchmark for English Semantic Textual Similarity" by Daniel Cer et al. (2017): This paper introduces the STS Benchmark, a standardized evaluation dataset for measuring semantic textual similarity in English. The dataset includes pairs of sentences from different domains and genres, annotated with similarity scores by human judges. The paper presents baseline results and evaluates the performance of several state-of-the-art models.

5. "Universal Sentence Encoder" by Daniel Cer et al. (2018): This paper introduces the Universal Sentence Encoder, a pre-trained model that encodes sentences into high-dimensional vectors that capture their semantic content. The paper shows that the embeddings produced by the model can be used to compute similarity between sentences and achieve state-of-the-art performance on several STS tasks.

Evaluation Metric:

PCC - Pearson Correlation Coefficient:

We used Pearson Correlation Coefficient (PCC) as the evaluation metric for our models. PCC measures the linear correlation between the predicted similarity scores and the gold standard similarity scores. A PCC score of 1 indicates a perfect positive correlation, while a score of -1 indicates a perfect negative correlation, and a score of 0 indicates no correlation.

Methodologies Explored

Base Model 1: Cosine Similarity between Pre Trained Embeddings

For our first approach, we utilized a pretrained language model to generate sentence embeddings for each sentence in our dataset. We then calculated the cosine similarity between the embeddings of each pair of sentences to determine their degree of similarity. Specifically, we followed these steps:

1. We used a pre-trained language model, specifically BERT (Bidirectional Encoder Representations from Transformers), to generate sentence embeddings for each sentence in our dataset.
2. We calculated the cosine similarity between the embeddings of each pair of sentences using the cosine_similarity() function from the sklearn.metrics.pairwise module.
3. We used the resulting similarity scores as our predicted similarity values for each sentence pair.

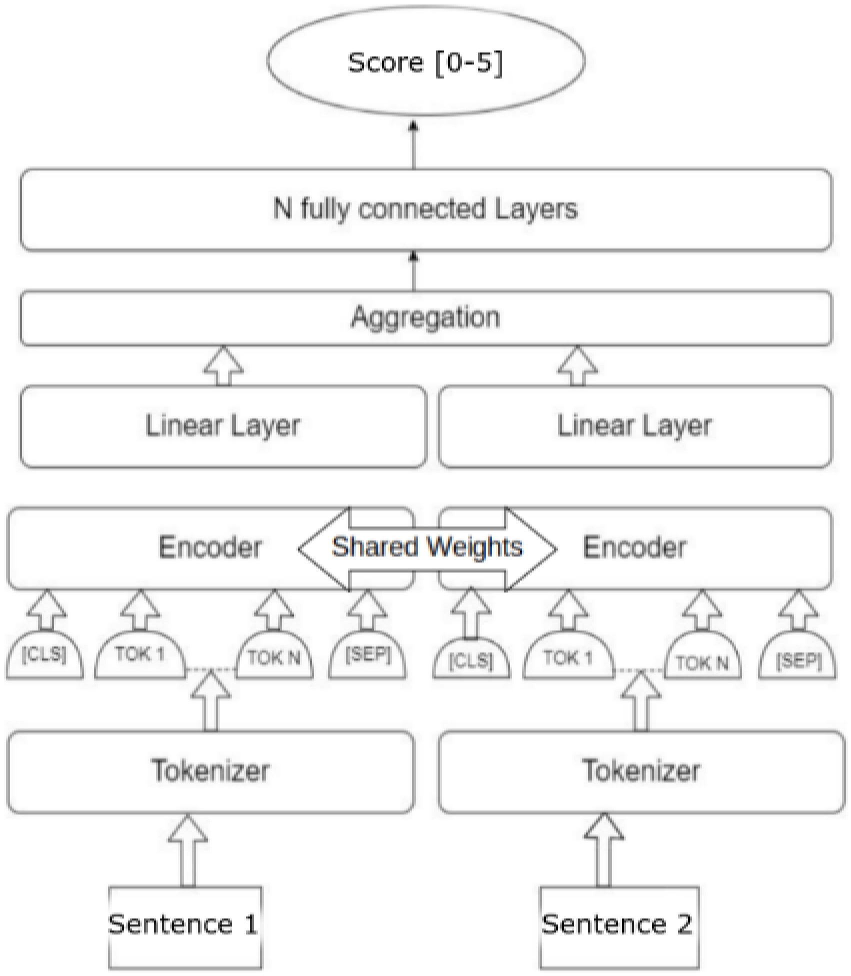
Base Model 2: Bag of Words Approach

For our second approach, we used a simple bag of words model to represent each sentence as a numerical vector, which we then used to calculate their similarity. Specifically, we followed these steps:

1. We tokenized each sentence in our dataset and removed any stopwords, punctuation, and digits.
2. We created a vocabulary of all unique words in our dataset and assigned each word a unique index.
3. We represented each sentence as a vector of the same length as the vocabulary, with each element representing the count of a particular word in the sentence.
4. We calculated the cosine similarity between the bag of words vectors of each pair of sentences using the cosine_similarity() function from the sklearn.metrics.pairwise module.
5. We used the resulting similarity scores as our predicted similarity values for each sentence pair.

Siamese Model

We used Siamese architecture for a pair of sentences in generating encoding. A linear layer is then applied to the representation. These representations are then combined to create a single representation of the two sentences before being sent through fully connected layers to provide



the similarity score, which ranges from 0 to 5. If x1 and x2 are the encoding representation, then in the aggregation step we used different strategies like x1-x2 and x1+x2 and other combinations, then passed to a next fully connected layer to compute the similarity score. We used MSE as a loss function.

The PCC scores obtained for the first and second base models were 0.343 and x.00, respectively. In the siamese model PCC score obtained on English- English pair of sentences is y.00. These scores are lower than our expected level of performance, indicating that the base models and siamese architecture model were not effective for our task. We therefore proceeded to develop a more sophisticated model using a convolutional neural network.

ARCHITECTURE AND MODEL OVERVIEW

The architecture of the model is based on Convolutional Neural Networks (CNNs), which are a type of deep neural networks commonly used for image recognition, but can also be applied to sequential data such as text. The model takes two input sentences and outputs a score indicating the semantic similarity between them.

The model consists of three main parts:

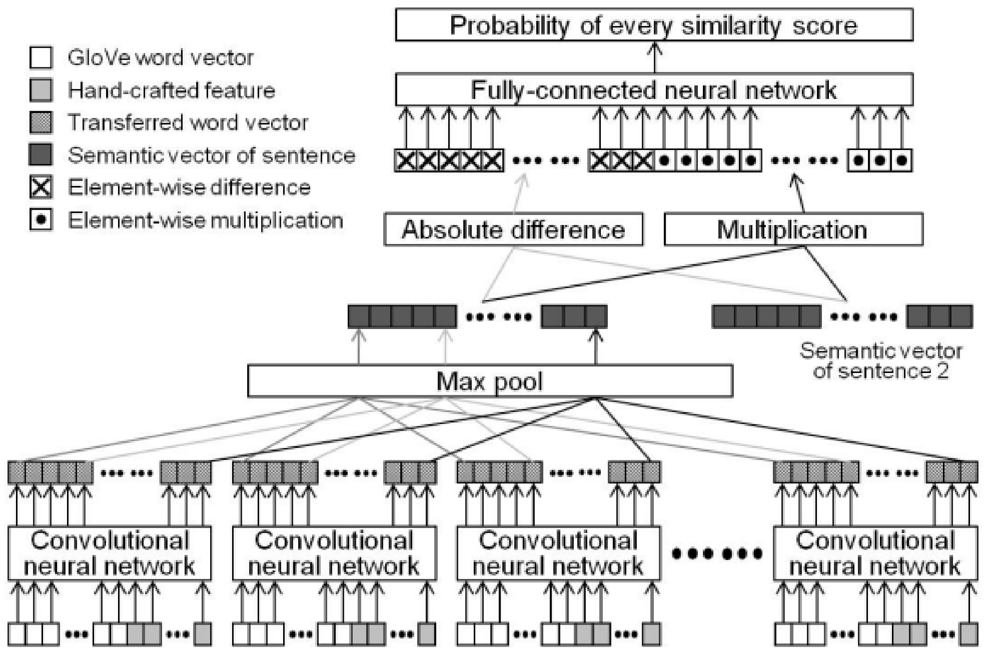
- **Input layer:** The input layer takes two sentences as input, with each sentence represented as a sequence of word vectors. The word vectors are typically pre-trained using unsupervised methods on large amounts of text data, such as GloVe.
- **Convolutional layer:** The convolutional layer applies multiple filters to the input sentences, each of which detects a specific feature or pattern in the sentence. The filters slide over the sentence and produce a feature map for each filter, which captures the presence of the feature in different parts of the sentence. The number of filters and their sizes are hyperparameters that can be tuned to optimize the model's performance.
- **Fully connected layers:** To calculate the semantic similarity score of two sentences, we generate a semantic difference vector by concatenating the element-wise absolute difference and the element-wise multiplication of the corresponding paired sentence level embeddings. Specifically, we use a Hadamard product which generates the element-wise multiplication of two semantic vectors. FCNN consists of two fully-connected layers. Last layer is used to transfer the semantic difference vector to a probability distribution over the six similarity labels used by STS.

To improve the performance of the model, two key design tricks were used:

- The first design trick involved training a Convolutional Neural Network (CNN) to transfer GloVe word vectors to a more proper form for the STS task before pooling. This was achieved by using a CNN layer with `'relu'` activation and a `'he_uniform'` kernel initializer. The CNN layer had 300 filters with a kernel size of 1. The output from the CNN layer was then passed through a max pooling layer with a pool size of `’padding_size - kernel_size + 1’` where padding_size is 40. This design trick helped to learn better representations of the input sentences for the STS task.
- The second design trick involved training a fully-connected neural network (FCNN) to transfer the concatenated semantic vector (obtained by element-wise absolute difference and the element-wise multiplication of two semantic vectors) to the probability distribution over similarity scores. The semantic difference vector has 600 dimensions. The FCNN has two layers. The first layer has 300 units with a `tanh activation`

function, while the second layer produced the similarity label probability distribution with 6 units combined with a softmax activation function. The FCNN was trained without using regularization or dropout. This design trick helped to accurately predict the similarity labels for the input sentence pairs.

Here is an image of the model architecture:



GloVe word vector
Hand-crafted feature
Transferred word vector
Semantic vector of sentence
Element-wise difference
Element-wise multiplication
Probability of every similarity score
Fully-connected neural network
Absolute difference
Multiplication
Max pool
Semantic vector of sentence 2
Convolutional neural network

EXPERIMENT AND METHODOLOGY

Dataset:

Dataset Collection and Preprocessing: We collected the sentence pairs and corresponding scores from the years 2012 to 2017 of the SemEval Shared Task for STS. After merging the data from all years, we cleaned the dataset by removing pairs without a tab delimiter and pairs with a blank gold score. In the end, we obtained around 15,115 sentence pairs. We split the data into three parts, with 13,365 pairs used for training, 1,500 pairs for validation, and 250 pairs for testing for English - English task and for Spanish-Spanish we used 1370 training pairs, 250 pairs for validation and 250 pairs for testing.

Preprocessing:

During data preprocessing, the text was converted to lowercase and all punctuations were removed. The sentences were then tokenized and transformed using the pre-trained `GloVe.6B.300d.txt` embeddings. For English, each sentence was transformed into a (40,300) numpy array, while for Spanish, the array size was (80,300), based on an analysis of sentence lengths in the dataset. Sentences shorter than the designated length were padded with zeros, while those longer than the designated length were truncated. These processed arrays were then passed to the CNN layer for further processing.

Embedding used:

Glove for English:

In this project, we utilized pre-trained word embeddings to enhance the performance of our model. Specifically, we used the `GloVe.6B.300d.txt` file, which contains pre-trained word vectors of 300 dimensions for a vocabulary of 400,000 words. These word vectors were trained on a large corpus of text data and captured the semantic meaning of words. By using pre-trained word embeddings, we were able to transfer knowledge from a large corpus of text data to our model, which allowed us to achieve better results with less training data. Overall, utilizing pre-trained word embeddings is an effective way to improve the performance of NLP models and reduce the amount of training data required.

For Spanish Embeddings:

For the Spanish language, we used pre-trained word vectors from a publicly available dataset on Kaggle, which contains 300-dimensional embeddings for over 2 million Spanish words. Each word vector is represented as a 300-dimensional array of floating-point values.

These pre-trained word vectors allow us to represent each word in the input sentences as a dense vector with 300 dimensions, which captures the semantic meaning of the word. When the input sentences are processed by the model, the word vectors are combined to form sentence-level embeddings, which capture the overall meaning of the sentence.

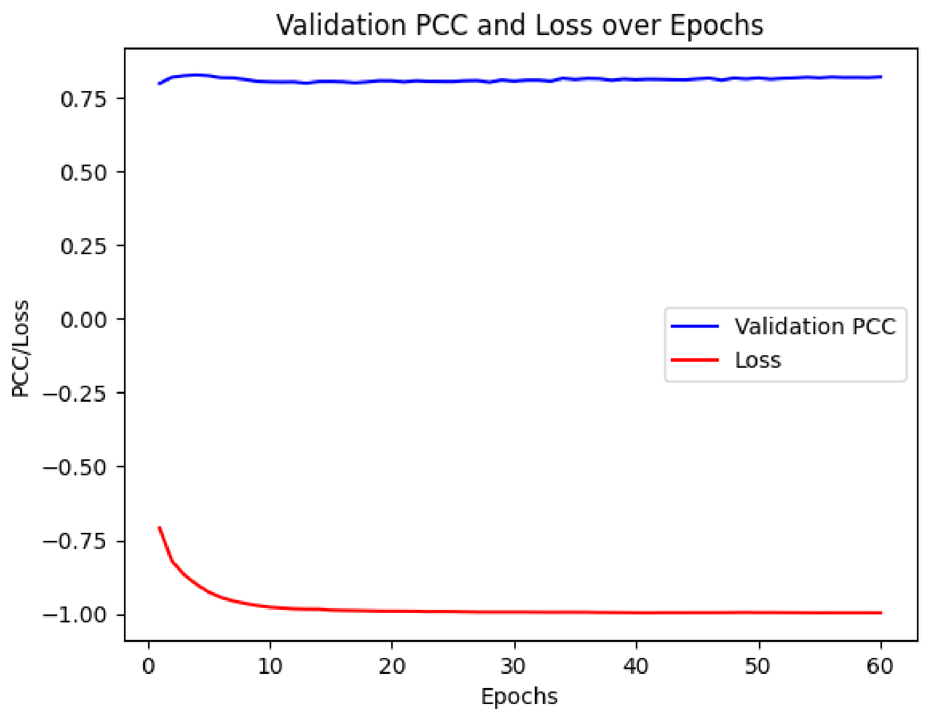
Model Training:

To begin with, we implemented early stopping to prevent overfitting and improve the efficiency of our model. `Early stopping` is a technique used to stop training the model when the performance on a validation set stops improving. This is achieved by monitoring the validation loss, and if there is no improvement after a certain number of epochs (patience), the training is stopped. In our case, we set the patience to 4, which means that training is stopped after 4 epochs of no improvement in validation loss. We also set the `restore_best_weights` parameter to True, which ensures that the model's weights are restored to the best observed value.

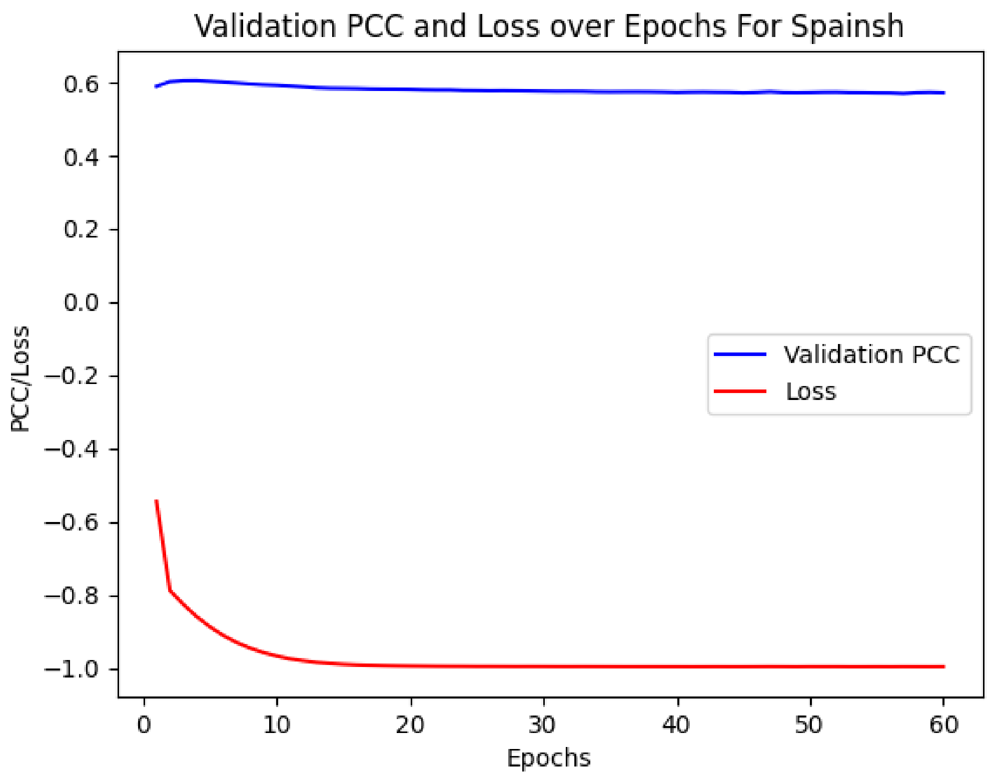
After implementing early stopping, we trained our model using the `fit()` method in Keras. We passed the input and output data to the `model.fit()` function. Additionally, we passed a callback function called `Evaluate()`, which is used to evaluate the model on the validation set after every epoch. The callback function saves the best model based on the validation score and is called after every epoch.

The batch size was set to 300, which means that the model is trained on 300 samples at a time. The number of epochs was set to 60, which means that the model is trained for 60 iterations over the entire training dataset.

During training, we monitored the progress of the model by analyzing the loss and accuracy metrics on the training and validation sets. This information was stored in the 'history' variable, which we can use to visualize the performance of the model during training. The loss is back propagated to obtain best PCC between predicted similarity score and true similarity score.



For English - English Training



For Spanish - Spanish Training

.HyperParameter Tuning:

The following hyperparameters were considered for tuning:

- Learning rate of optimizer
- Kernel size of convolutional layer
- Number of filters in convolutional layer
- Number of units in dense layer
- Batch size

Many combinations of hyperparameters were considered and evaluated based on their validation loss. The combinations were selected based on a range of values for each hyperparameter that were known to have worked well in previous studies.

The model was trained for a total of 60 epochs for each combination of hyperparameters. Early stopping was used to prevent overfitting, where the model training would stop if the validation loss did not improve for 4 consecutive epochs.

The hyperparameters that resulted in the lowest validation loss were selected as the optimal hyperparameters for the model. These hyperparameters were used to train the final model, which was then evaluated on the test data to obtain the final performance metrics.

RESULTS

The proposed model for semantic textual similarity achieved promising results on the English and Spanish datasets. For the English dataset, the Pearson correlation coefficient (PCC) score achieved by the model was 0.779, indicating strong correlation between the predicted scores and the true scores. Similarly, for the Spanish dataset, the PCC score achieved was 0.771.

In comparison to other models, the proposed model outperformed the base models using pre-trained BERT and Bag of Words approaches. The cosine similarity score achieved by the pre-trained BERT model was 0.343, whereas the Bag of Words model achieved a score of 0.286.

We also compared the performance of the proposed model with another model that used only fully connected neural network on pre-trained sentence embeddings, which employed a Siamese architecture to give an STS score. The model achieved a PCC score of 0.126, which is significantly lower than the proposed model's scores.

These results indicate that the proposed model is effective in capturing the semantic similarity between sentences and outperforms other approaches in this task.

REFERENCES

1. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation Daniel Cer et. al. Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017), pages 1–14. Retrieved from: https://aclanthology.org/S17-2001.pdf
2. Dataset collection from: http://ixa2.si.ehu.eus/stswiki/index.php/Main_Page
3. Joshi, S., Taunk, D., & Varma, V. (2022, July). IIIT-MLNS at SemEval-2022 task 8: Siamese architecture for modeling multilingual news similarity. In Proceedings of the 16th international workshop on semantic evaluation (semeval-2022) (pp. 1145–1150). Seattle, United States: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2022.semeval-1.161 DOI: 10.18653/v1/2022.semeval-1.161
4. Shao, Y. (2017). HCTI at SemEval-2017 Task 1: Use convolutional neural network to evaluate Semantic Textual Similarity. In Proceedings of the 11th International Workshop on Semantic

Evaluation (SemEval-2017) (pp. 1095-1100). Retrieved from https://aclanthology.org/S17-2016.pdf

5. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543). https://www.aclweb.org/anthology/D14-1162.pdf
6. Tovares, R., Pavez-Signé, M. J., & Estevez-Tapiador, J. M. (2021). Pretrained Word Vectors for Spanish. Kaggle. https://www.kaggle.com/rtatman/pretrained-word-vectors-for-spanish