INTRODUCTION TO NLP

Final Report Submission

Team No. - 59

Team Name - Team API

Team Members :

1. Amogha Halhalli (Roll No. - 2021101007)
2. Ishit Bansal (Roll No. - 2021101083)
3. Pranav Gupta (Roll No. - 2021101095)

Introduction

Semantic Textual Similarity (STS) assesses the degree to which two sentences are semantically equivalent to each other. The STS task is motivated by the observation that accurately modeling the meaning similarity of sentences is a foundational language understanding problem relevant to numerous applications including: machine translation (MT), summarization, generation, question answering (QA), short answer grading, semantic search, dialog and conversational systems.

STS is the assessment of pairs of sentences according to their degree of semantic similarity. The task involves producing real-valued similarity scores for sentence pairs. Performance is measured by the Pearson correlation of machine scores with human judgments.

As a part of the Final Project Submission, we trained embeddings of the words in the dataset against several models, including the Word2Vec Model, Doc2Vec Model, BERT Model, RoBERTa Model, Sentence Transformers, Siamese BiLSTMs, Universal Sentence Encoders. Then, we evaluated the Performance of the Model using the Pearson Correlation Function to compare the various embeddings which are generated by different Models which would at the end, lead to generation of higher semantic similarity scores.

The scale spans from 0 denoting no shared meaning to 5 indicating complete equivalence in meaning. Intermediate values signify varying degrees of partial meaning overlap. Semantic tasks within STS encompass textual entailment, semantic relatedness, and paraphrase identification. Unlike textual entailment and paraphrase detection, STS gauges nuanced levels of meaning overlap rather than binary classifications. Semantic relatedness also measures graded semantic connections but remains nonspecific about the relationship's nature, allowing contradictory content to potentially receive a high score (e.g., "night" and "day" are highly related despite their dissimilarity).

Datasets used

1. **Mono-lingual SemEval Datasets(Train, Validation & Test)**

   About the Dataset - All the Files are in the form of .jsonl extension. Data is loaded from each of the files, train.jsonl, val.jsonl, test.jsonl and then data is cleaned from each of the files. The Training Set consists of 5749 pairs of sentences, the validation set consists of 1500 pairs of sentences while the testing set consists of 1379 such samples. The sentences are then broken down to words and converted to embeddings so that these can be used later for comparison of the 2 sentences.

2. **Cross-Lingual SemEval Datasets (Training, Validation and Testing Sets)**

   About the Dataset - The Datasets involve 1st sentence in English Language and the 2nd sentence in Spanish. All the Files are in the form of .parquet extension initially. Firstly, general Preprocessing is done and they are converted to .csv file extension in the format that they can directly be fed to the model. Data is loaded from each of the files, train.csv, val.csv, test.csv. The Training Set consists of 5749 pairs of sentences, the validation set consists of 1500 pairs of sentences while the testing set consists of 1379 such samples. The sentences are then broken down to words and converted to embeddings so that these can be used later for comparison of the 2 sentences.

Data Preprocessing

Tokenization:

Applied following preprocessing steps as part of tokenization:

1. Removed punctuation
2. Replaced numbers by ‘number’
3. Replaced URLs found by ‘url’
4. Replaced Hashtags by ‘hashtag’
5. Replaced Emails by ‘email’
6. Replaced Mentions by ‘mention’
7. Converted Text to Lowercase
8. Split the Sentences into Words.
9. Removed Stopwords from text.
10. Applied lemmatizer on the resultant words list
11. Finally used Stemming to obtain output

After performing these steps, the vocabulary was found to be 8258.

There are 2 files which perform Data Preprocessing for both Mono-lingual and Cross-lingual. Each of them have their separate preprocess.ipynb. In preprocess.ipynb, pre-processing as specified above is performed on the SemEval Datasets for both Mono and Cross-Lingual similarity as mentioned above.

Mono-lingual

Implementation of Mono-lingual models

1. **Word2Vec:**

   We loaded pre-trained Word2Vec embeddings, computed the mean Word2Vec embeddings for each sentence and calculated cosine similarity score to determine the semantic similarity between 2 sentences.

   After this we used different models such as GRU and BiLSTM to contextualize embeddings and evaluated the model on train, test and validation data, and then reported the value metrics.

2. **Bidirectional Encoder Representations from Transformer (BERT):**

   We used the pre-trained BERT Model from the inbuilt transformers module in Python. We first loaded the model and passed the sentences through the model in order to generate Contextual Embeddings. Then we calculated the similarity scores for the untrained model. After this, we fine-tuned the model using MSE Loss (because generation of Similarity Scores is continuous and hence a regression task), evaluated the model on train, test and validation data, and then reported the value metrics.

3. RoBERTa:

RoBERTa is an extension of the BERT model trained on larger data. We used the pre-trained RoBERTa Model from the inbuilt transformers module in Python. We first loaded the model and passed the sentences through the model in order to generate Contextual Embeddings. Then, we calculated similarity scores for the untrained model. After this, we fine-tuned the model using MSE loss (as generation of similarity scores is continuous and hence a regression task), evaluated the model on train, test and valuation data, and then reported the value metrics.

4. Doc2Vec:

For this purpose, the inbuilt Doc2Vec model was loaded from the inbuilt gensim.models module in Python. This proceeds by first tagging all the given input documents of both the sentences. Then, the document was converted into vectorial representation by passing through the model and the Pearson coefficients computed. Then, the model was fine-tuned by passing the embeddings by a bidirectional LSTM to generate contextual embeddings. We evaluated the model on train, test and validation data, and then reported the value metrics.

5. Universal Sentence Encoder (USE):

We used the pre-trained model from the link below. https://tfhub.dev/google/universal-sentence-encoder/4

Since Universal Sentence Encoder is not available on Pytorch, we used tensorflow for this model. We first loaded the model and passed the sentences through the model in order to generate Contextual Embeddings. Then, we calculated similarity scores for the untrained model. After this, we fine-tuned the model using MSE loss (as generation of similarity scores is continuous and hence a regression task), evaluated the model on train, test and validation data, and then reported the value metrics.

6. Sentence Transformers:

We used the pre-trained Sentence Transformer model “paraphrase-MiniLM-L6-v2” from the inbuilt sentence-transformers module in Python. We first loaded the model and passed the sentences through the Model in order to generate Contextual Embeddings (basically encoded the input sentences into embeddings). The cosine similarity was evaluated using the generated embeddings. After this, we fine-tuned the model using MSE Loss (because generation of Similarity Scores is continuous and hence a regression task), evaluated the model on train, test and valuation data, and then reported the value metrics. It is simply a

BERT Model with a Pooling Layer on top. It serves as an effective way of reducing the dimensionality of the output, capturing the global information in the text while at the same time speeding up the process.

7. **Siamese BiLSTM Network:**

In the beginning, word-vectors of pre-trained Word2Vec Model trained on Google News Dataset were loaded and taken as the initial embeddings of words in the dataset. Following this, Siamese BiLSTM Architecture with Attention Mechanism was trained using an Optimizer and MSE Loss. After this, Contextual Embeddings were generated by passing through the model and the cosine similarity scores were evaluated between the Embeddings which leads to evaluation of Pearson metric on train, val and test datasets.

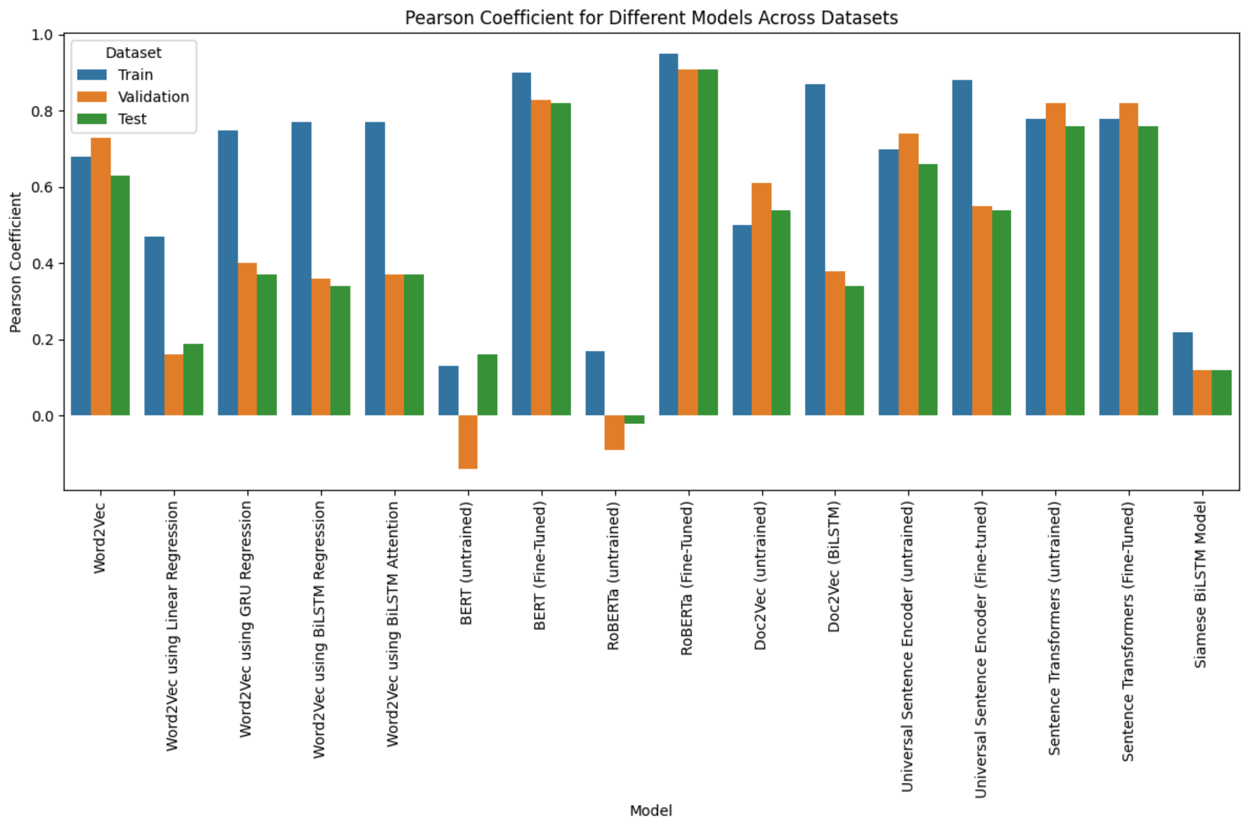
Results obtained for Mono-lingual data

S.No	Model Name	Loss Function	Train Data Pearson Coefficient	Validation Data Pearson Coefficient	Testing Data Pearson Coefficient
1.	Word2Vec (untrained)	None	0.68	0.73	0.63
2.	Word2Vec using Linear Regression	MSE	0.47	0.16	0.19

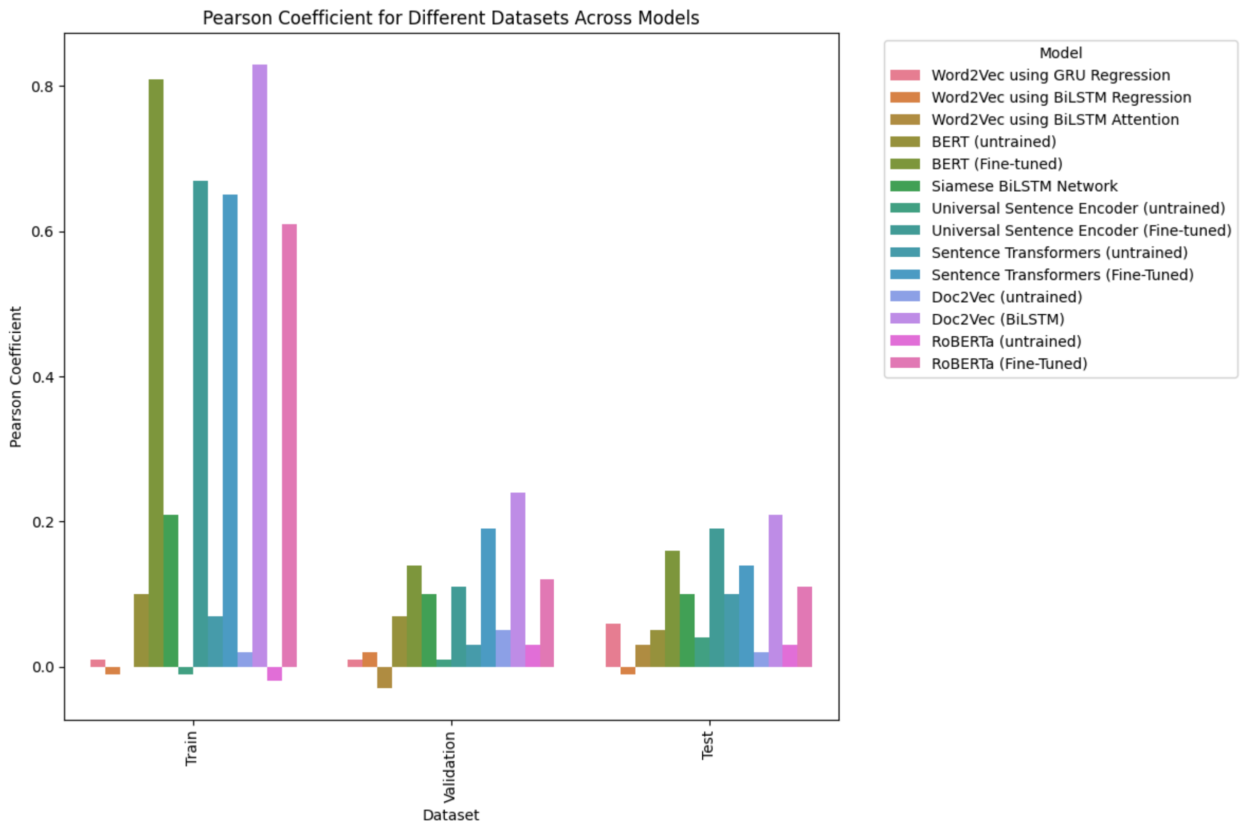
3.	Word2Vec using GRU Regression	MSE	0.75	0.40	0.37
4.	Word2Vec using BiLSTM Regression	MSE	0.77	0.36	0.34
5.	Word2Vec using BiLSTM Attention	MSE	0.77	0.37	0.37
6.	BERT (untrained)	None	0.13	-0.14	0.16
7.	BERT (Fine-Tuned)	MSE	0.90	0.83	0.82
8.	RoBERTa (untrained)	None	0.17	-0.09	-0.02
9.	RoBERTa (Fine-Tuned)	MSE	0.95	0.91	0.91
10.	Doc2Vec (untrained)	None	0.50	0.61	0.54
11.	Doc2Vec (BiLSTM)	MSE	0.87	0.38	0.34
12.	Universal Sentence	None	0.70	0.74	0.66

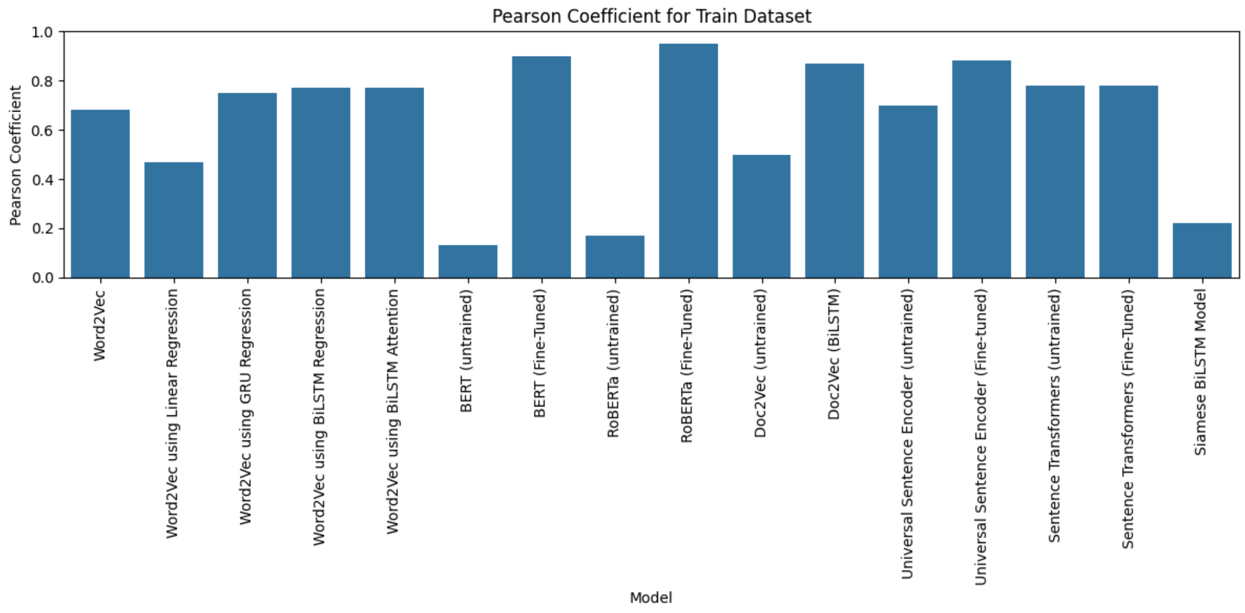
	Encoder (untrained)				
13.	Universal Sentence Encoder (Fine-tuned)	MSE	0.88	0.55	0.54
14.	Sentence Transformers (untrained)	None	0.78	0.82	0.76
15.	Sentence Transformers (Fine-Tuned)	MSE	0.78	0.82	0.76
16.	Siamese BiLSTM Model	MSE	0.22	0.12	0.12

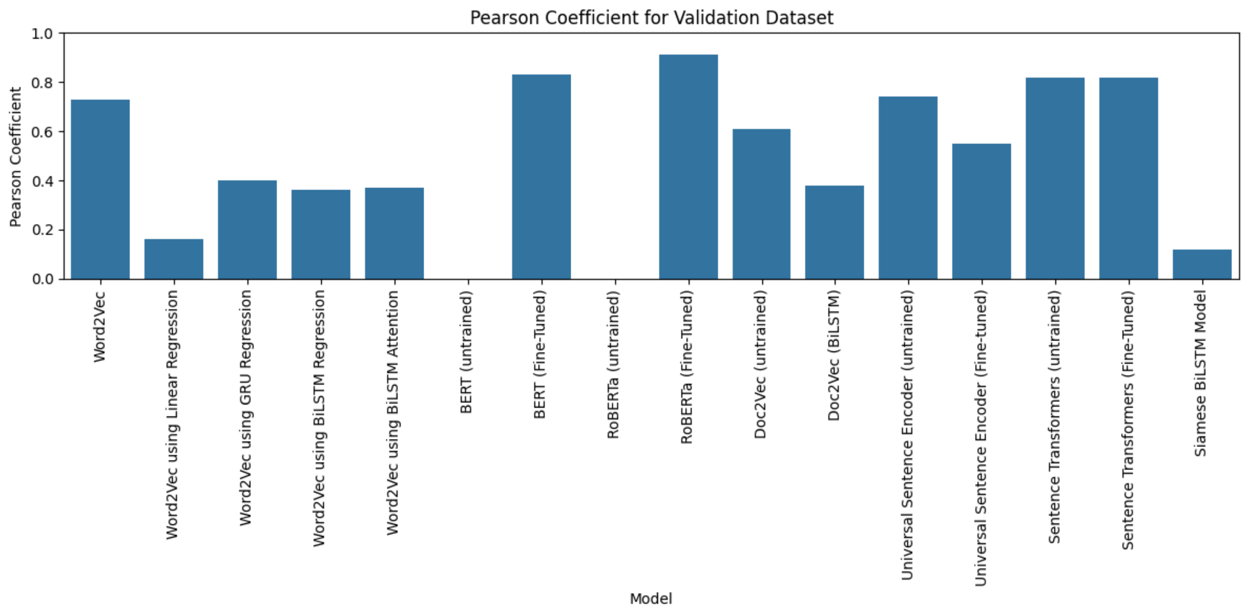
Visualizations

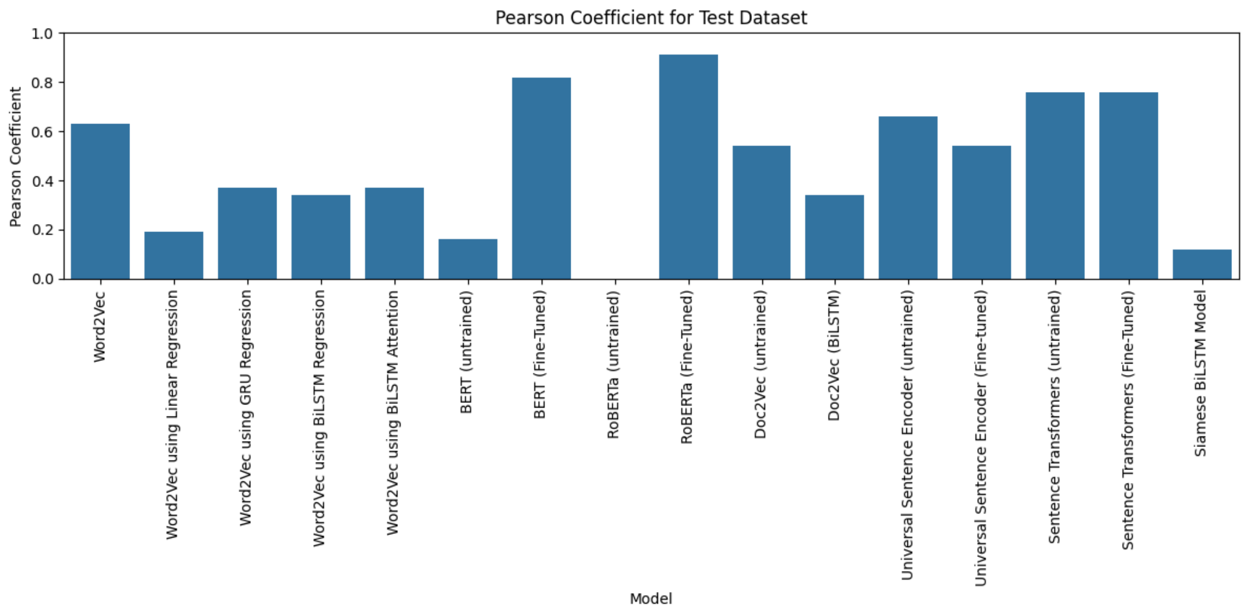


Pearson Coefficient for Different Models Across Datasets
Dataset
Train
Validation
Test
Pearson Coefficient
1.0
0.8
0.6
0.4
0.2
0.0
Model
Word2Vec
Word2Vec using Linear Regression
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-Tuned)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Siamese BiLSTM Model


Pearson Coefficient for Different Datasets Across Models
Pearson Coefficient
Dataset
Train
Validation
Test
Model
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-tuned)
Siamese BiLSTM Network
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)


Pearson Coefficient for Train Dataset
Pearson Coefficient
Model
Word2Vec
Word2Vec using Linear Regression
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-Tuned)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Siamese BiLSTM Model


Pearson Coefficient for Validation Dataset
Pearson Coefficient
Model
Word2Vec
Word2Vec using Linear Regression
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-Tuned)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Siamese BiLSTM Model


Pearson Coefficient for Test Dataset
Pearson Coefficient
Model
Word2Vec
Word2Vec using Linear Regression
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-Tuned)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Siamese BiLSTM Model

Conclusion

In analyzing the results obtained for mono-lingual data across various models, several trends and insights emerge.

Firstly, in the context of Word2Vec embeddings, utilizing pre-trained embeddings without further training yields moderate performance across all datasets, suggesting the usefulness of Word2Vec's semantic representations. However, employing linear regression (Model 2) or recurrent neural networks such as GRU (Model 3) and BiLSTM (Models 4 and 5) for regression tasks on top of Word2Vec embeddings does not significantly improve performance and may even degrade it, indicating that more sophisticated models may not always lead to better results when using basic word embeddings.

Moving to contextualized embeddings like BERT and RoBERTa, while untrained versions (Models 6 and 8) perform poorly on all datasets, fine-tuning them for regression tasks (Models 7 and 9) leads to substantial improvements, particularly evident in the high Pearson coefficients across all datasets. This underscores the importance of fine-tuning contextualized embeddings to specific tasks to harness their full potential.

Doc2Vec embeddings (Models 10 and 11) showcase promising performance, especially when integrated with a BiLSTM model. Universal Sentence Encoder (Models 12 and 13) exhibits robust performance even in its untrained version, with significant improvements upon fine-tuning. Sentence Transformers (Models 14 and 15), whether untrained or fine-tuned, consistently demonstrate strong performance across all datasets.

Lastly, the Siamese BiLSTM Model (Model 16) delivers moderate results, suggesting its potential for semantic similarity tasks but also room for improvement compared to other approaches.

In conclusion, the choice of embedding method and model architecture significantly impacts the performance of mono-lingual semantic similarity tasks. While traditional methods like Word2Vec show moderate success, contextualized embeddings like BERT and RoBERTa, when fine-tuned, outperform them substantially. Additionally, models leveraging advanced architectures like BiLSTM and Universal Sentence Encoder also demonstrate competitive performance, underlining the importance of choosing appropriate embedding methods and model architectures tailored to specific tasks.

<u>Cross-lingual</u>

Implementation of Cross-lingual models

1. **Word2Vec:**

   We loaded pre-trained Word2Vec embeddings, computed the mean Word2Vec embeddings for each sentence and calculated cosine similarity score to determine the semantic similarity between 2 sentences. After this we used different models such as GRU and BiLSTM to contextualize embeddings and then reported the value metrics.

2. Bidirectional Encoder Representations from Transformer (BERT):

We used the pre-trained BERT Model from the inbuilt transformers module in Python. We first loaded the model and passed the sentences (both English language and Spanish language sentences) through the model in order to generate Contextual Embeddings. Then we calculated the similarity scores for the untrained model. After this, we fine-tuned the model using MSE Loss (because generation of Similarity Scores is continuous and hence a regression task), evaluated the model on train, test and validation data, and then reported the value metrics.

3. RoBERTa:

RoBERTa is an extension of the BERT model trained on larger data. We used the pre-trained RoBERTa Model from the inbuilt transformers module in Python. We first loaded the model and passed the sentences through the model in order to generate Contextual Embeddings. Then, we calculated similarity scores for the untrained model. After this, we fine-tuned the model using MSE loss (as generation of similarity scores is continuous and hence a regression task), evaluated the model on train, test and valuation data, and then reported the value metrics.

4. Doc2Vec:

For this purpose, the inbuilt Doc2Vec model was loaded from the inbuilt gensim.models module in Python. This proceeds by first tagging all the given input documents of both the E. Then, the document was converted into vectorial representation by passing through the model and the Pearson coefficients computed. Then, the model was fine-tuned by passing the embeddings by a bidirectional LSTM to generate contextual embeddings. We evaluated the model on train, test and validation data, and then reported the value metrics.

5. Universal Sentence Encoder (USE):

We used the pre-trained model from the link below.

https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3

Since Universal Sentence Encoder is not available on Pytorch, we used tensorflow for this model. We first loaded the model and passed the sentences through the model in order to generate Contextual Embeddings. Then, we calculated similarity scores for the untrained model. After this, we fine-tuned the model using MSE loss (as generation of similarity scores is continuous and hence a regression task), evaluated the model on train, test and validation data, and then reported the value metrics.

6. Sentence Transformers:

We used the pre-trained Sentence Transformer model “paraphrase-MiniLM-L6-v2” from the inbuilt sentence-transformers module in Python. We first loaded the model and passed the sentences through the Model in order to generate Contextual Embeddings (basically encoded the input sentences into embeddings). The cosine similarity was evaluated using the generated embeddings. After this, we fine-tuned the model using MSE Loss (because generation of Similarity Scores is continuous and hence a regression task), evaluated the model on train, test and valuation data, and then reported the value metrics. It is simply a BERT Model with a Pooling Layer on top. It serves as an effective way of reducing the dimensionality of the output, capturing the global information in the text while at the same time speeding up the process.

7. Siamese BiLSTM Network:

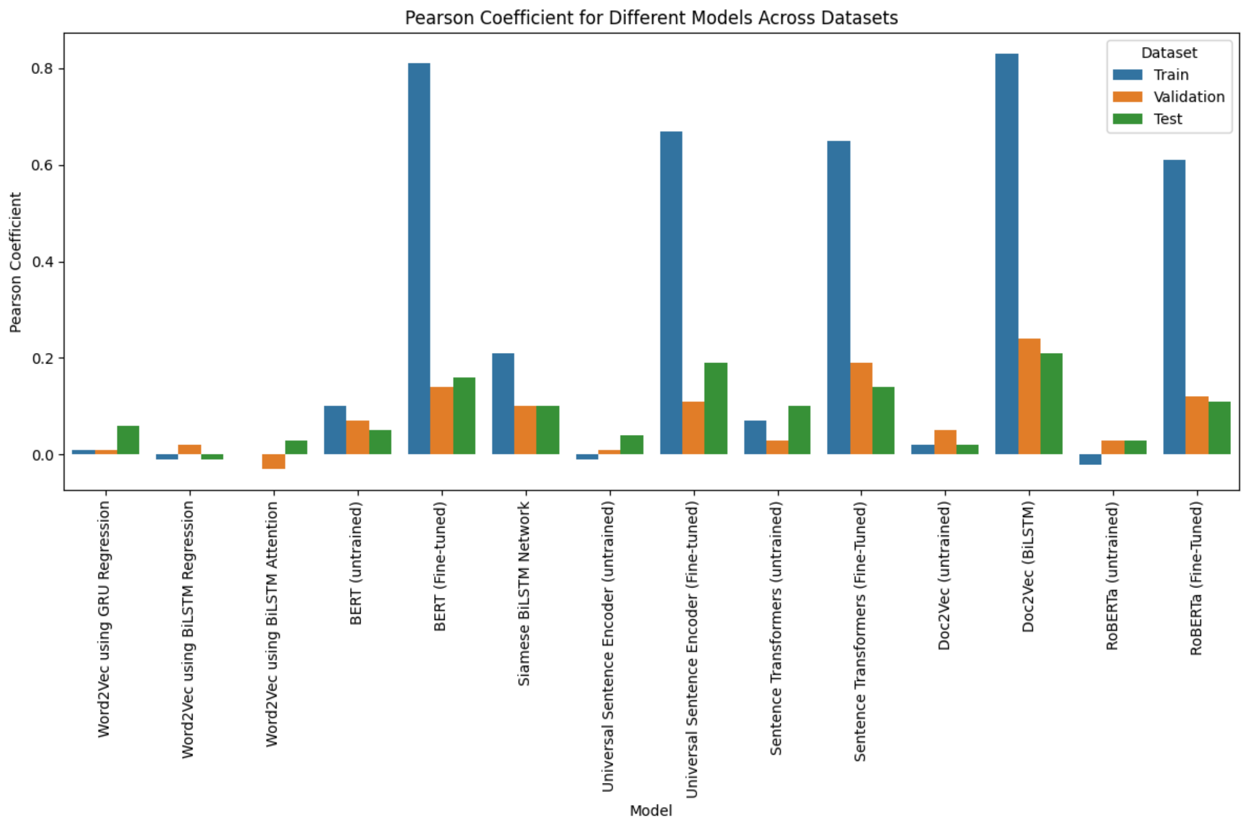
In the beginning, word-vectors of pre-trained Word2Vec Model trained on Google News Dataset were loaded for English language sentences and SBM(Spanish Billion Corpus) Word Vectors were loaded for the Spanish language sentences and taken as the initial embeddings of words of both English and Spanish sentences in the dataset. Following this, Siamese BiLSTM Architecture with Attention Mechanism was trained using an Optimizer and MSE Loss. After this, Contextual Embeddings were generated by passing through the model and the cosine similarity scores were evaluated between the Embeddings which leads to evaluation of Pearson metric on train, val and test datasets.

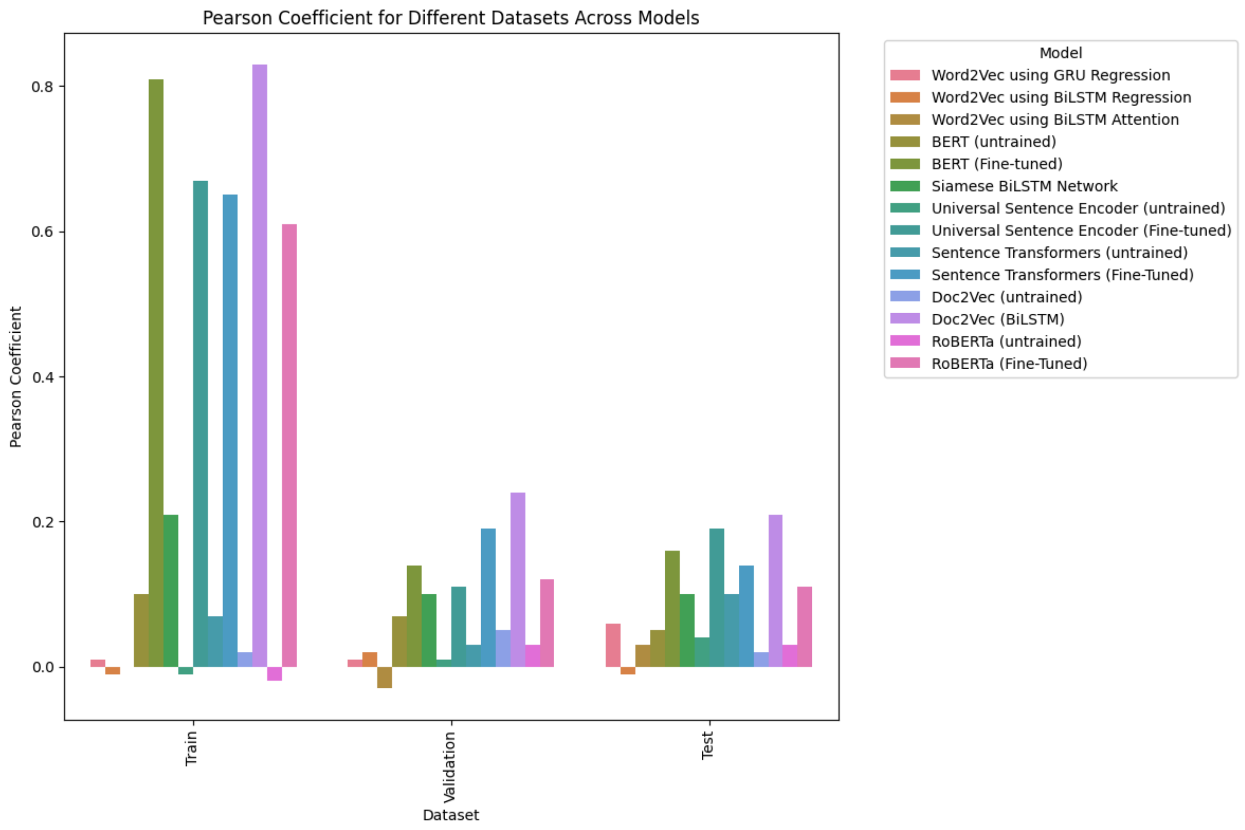
Results obtained for Cross-lingual data

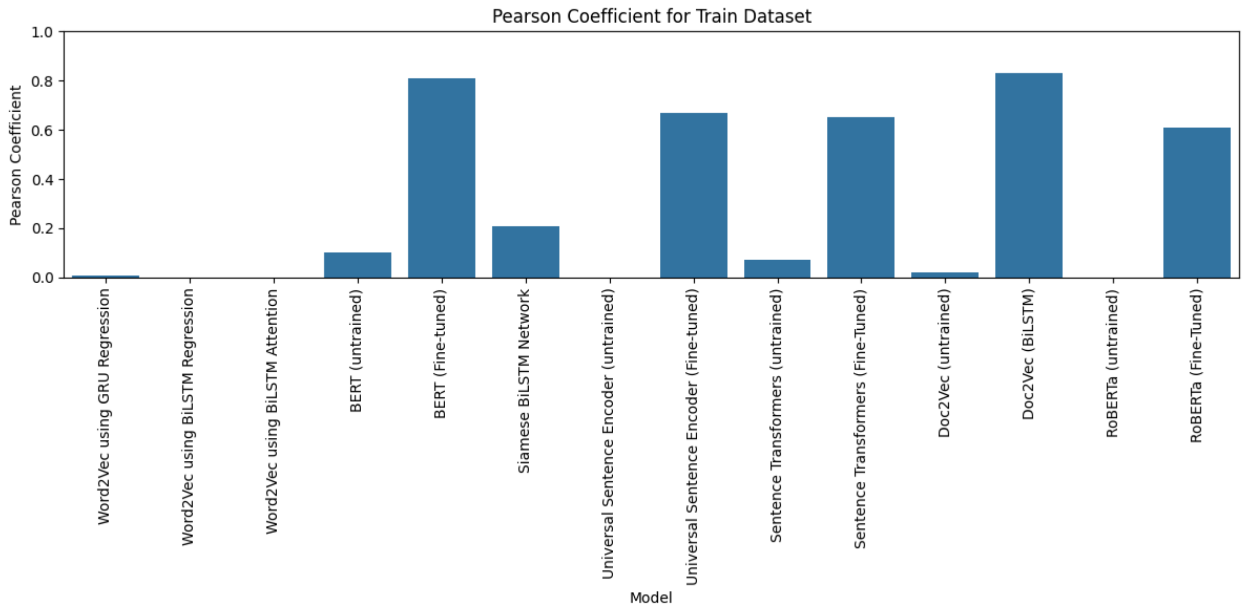
S. No	Model Name	Loss Function	Train Data Pearson Coefficient	Validation Data Pearson Coefficient	Testing Data Pearson Coefficient
1.	Word2Vec using GRU Regression	MSE	0.01	0.01	0.06
2.	Word2Vec using BiLSTM Regression	MSE	-0.01	0.02	-0.01
3.	Word2Vec using BiLSTM Attention	MSE	0	-0.03	0.03
4.	BERT (untrained)	None	0.10	0.07	0.05
5.	BERT (Fine-tuned)	MSE	0.81	0.14	0.16
6.	Siamese BiLSTM Network	MSE	0.21	0.10	0.10

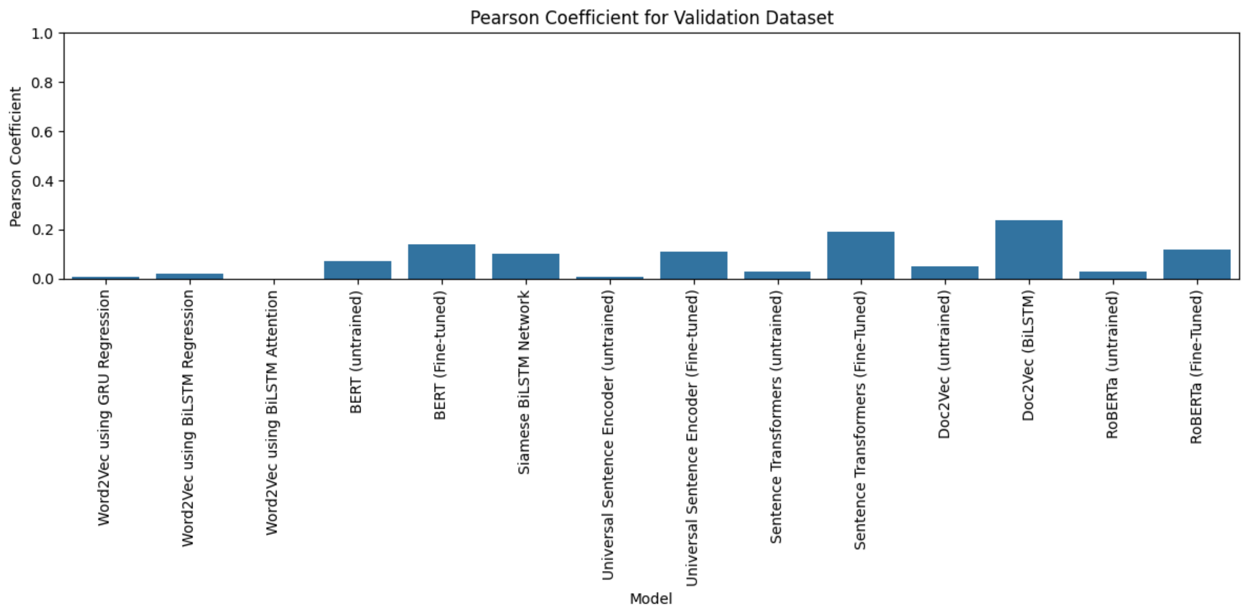
7.	Universal Sentence Encoder (untrained)	None	-0.01	0.01	0.04
8.	Universal Sentence Encoder (Fine-tuned)	MSE	0.67	0.11	0.19
9.	Sentence Transformers (untrained)	None	0.07	0.03	0.10
10.	Sentence Transformers (Fine-Tuned)	MSE	0.65	0.19	0.14
11.	Doc2Vec (untrained)	None	0.02	0.05	0.02
12.	Doc2Vec (BiLSTM)	MSE	0.83	0.24	0.21
13.	RoBERTa (untrained)	None	-0.02	0.03	0.03
14.	RoBERTa (Fine-Tuned)	MSE	0.61	0.12	0.11

Visualizations

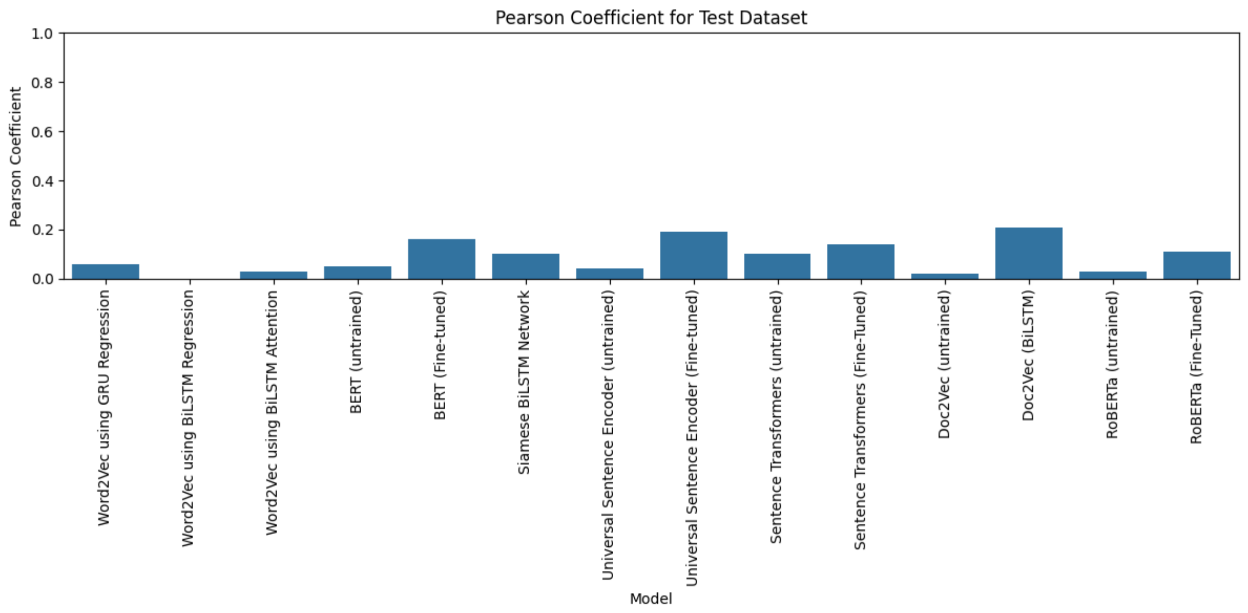

Pearson Coefficient for Different Models Across Datasets
Dataset
Train
Validation
Test
Pearson Coefficient
0.8
0.6
0.4
0.2
0.0
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-tuned)
Siamese BiLSTM Network
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Model


Pearson Coefficient for Different Datasets Across Models
Pearson Coefficient
0.0
0.2
0.4
0.6
0.8
Train
Validation
Test
Dataset
Model
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-tuned)
Siamese BiLSTM Network
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)


Pearson Coefficient for Train Dataset
Pearson Coefficient
0.0
0.2
0.4
0.6
0.8
1.0
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-tuned)
Siamese BiLSTM Network
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Model



Pearson Coefficient for Validation Dataset
Pearson Coefficient
1.0
0.8
0.6
0.4
0.2
0.0
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-tuned)
Siamese BiLSTM Network
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Model



Pearson Coefficient for Test Dataset
Pearson Coefficient
1.0
0.8
0.6
0.4
0.2
0.0
Word2Vec using GRU Regression
Word2Vec using BiLSTM Regression
Word2Vec using BiLSTM Attention
BERT (untrained)
BERT (Fine-tuned)
Siamese BiLSTM Network
Universal Sentence Encoder (untrained)
Universal Sentence Encoder (Fine-tuned)
Sentence Transformers (untrained)
Sentence Transformers (Fine-Tuned)
Doc2Vec (untrained)
Doc2Vec (BiLSTM)
RoBERTa (untrained)
RoBERTa (Fine-Tuned)
Model

Conclusion

Analyzing the results obtained for cross-lingual data reveals several noteworthy observations and trends.

Word2Vec-based models with GRU regression (Model 1), BiLSTM regression (Model 2), and BiLSTM attention (Model 3) exhibit limited performance, with generally low Pearson coefficients across all datasets. This suggests that Word2Vec embeddings alone may not adequately capture the nuances of cross-lingual semantic similarity tasks.

BERT, both in its untrained (Model 4) and fine-tuned (Model 5) forms, displays modest performance, with fine-tuning notably improving results but still falling short of achieving high Pearson coefficients, particularly on the validation and testing datasets.

The Siamese BiLSTM Network (Model 6) demonstrates moderate performance, suggesting its potential for cross-lingual semantic similarity tasks, albeit with room for improvement.

Universal Sentence Encoder, whether untrained (Model 7) or fine-tuned (Model 8), shows mixed results across datasets, indicating its sensitivity to training data and task-specific fine-tuning.

Similarly, Sentence Transformers in both untrained (Model 9) and fine-tuned (Model 10) states exhibit varying performance, with fine-tuning offering noticeable improvements on certain datasets.

Doc2Vec embeddings, particularly when integrated with a BiLSTM model (Model 12), showcase promising performance, outperforming many other models and suggesting their suitability for cross-lingual semantic similarity tasks.

RoBERTa-based models, whether untrained (Model 13) or fine-tuned (Model 14), exhibit mixed results, with fine-tuning offering slight improvements but still falling short of achieving high Pearson coefficients on all datasets.

In summary, while certain models such as BERT and RoBERTa show potential for cross-lingual semantic similarity tasks, their performance is generally modest and may benefit from further fine-tuning. Models leveraging Doc2Vec embeddings integrated with BiLSTM architectures demonstrate promising performance, indicating their effectiveness for capturing cross-lingual semantic similarity. However, there remains a need for more robust and effective approaches to address the challenges posed by cross-lingual semantic similarity tasks.

References

1. https://aclanthology.org/S17-2001.pdf

2. https://www.sbert.net/examples/training/sts/README.html

3. Datasets

https://nlpprogress.com/english/semantic_textual_similarity.html

4. STS References

https://medium.com/@Mustafa77/semantic-textual-similarity-with-bert-e10355ed6afa

5. Mono-Lingual Dataset:
https://huggingface.co/datasets/mteb/stsbenchmark-sts

6. Cross-Lingual Dataset:
https://huggingface.co/datasets/PhilipMay/stsb_multi_mt/tree/main

7. Google News Dataset:

https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?resourcekey=0-wjGZdNAUop6WykTtMip30g

8. Spanish Word Vectors:

https://github.com/dccuchile/spanish-word-embeddings