Rosetta Stone project
Group 21
Team member: Elena del Rosal Pérez (E04827)

1. Introduction:

This machine learning project has the principal objective of analyzing the semantic similitude between the sentences discovered by an alien civilization on an ancient text file on the abandoned Earth. This project was at first complicated for me since I'm alone but then with the class material I finally managed to work on it. It is a work that puts into practice what we have seen in class, especially in this case the transformers, as it has been the main model that I have used in the work. I have focused the work on comparing two different models to see how they perform with the same task and to see their differences in a practical way.

2. Methods

I start importing the dataset and importing the libraries we will need. It is important to highlight that for this project, due to hardware limitations, I was unable to process the entire dataset and had to work with a sample instead. Although using a subset of the data allowed me to experiment and train models within a reasonable time, it is important to note that working with the full dataset would likely produce more robust and reliable results. A larger dataset typically helps models generalize better by exposing them to a wider variety of linguistic patterns, reducing the risk of overfitting, and improving overall performance, especially in tasks like sentence similarity where subtle differences in phrasing matter. Therefore, using the full dataset is generally recommended for more accurate and meaningful outcomes but I couldn't.

Now, I am going to analyze and process the dataset in order to be able to work with it in perfect conditions. First I see if there are outliers in the dataset, which, as we see, there are not.
Now, I make stemming which is basically cutting words to their root form (for example, from “buying” to “buy” or from from uppercase to lowercase. I first used “PorterStemmer” but then I noticed that this model is designed only with english words so I then used “clean_text” because my dataset contains multilingual sentences. Using “PorterStemmer” could distort words in other languages. In contrast, “Clean_text” applies simple, language-agnostic cleaning—lowercasing which preserves the original sentence structure.

Now, I decided to translate all the sentences since the Parrot library only works in english so I thought it was better to translate before paraphrasing than after. For this I used GoogleTranslator. GoogleTranslator is a neural machine translation tool that automatically detects the source language and provides accurate translations. Since Parrot only works effectively with English input, translating the multilingual data ensures consistency and allows for high-quality paraphrasing.

Now, to improve the model I'm going to paraphrase using a parrot model. The parrot model is a transformer-based model that is specifically designed to generate paraphrases of natural language sentences, that is, to reformulate a sentence maintaining its original meaning but with different words or structures.

Now, in this part of the analysis, which for me is the most important, I am using the SentenceTransformer library with the multilingual BaBSE model to generate semantic embeddings for the translated with and without the paraphrased layer. (It is very important to do this since ML models only work with numbers so what this technique is doing is to convert all the words from each sentence as a point in space, where is represented by a dense vector (made out of numbers). SenterTransformer is a pre-trained BERT-type model optimised for semantic similarity. It is important to highlight that this model already tokenized the phrases (which consist of splitting our text into words, sentences, or subwords. It is necessary to do this since computers are not able to process raw text as we humans do).

What does this SentenceTransformer do (which is implicit in the code) ?

- Tokenization: Convert text to tokens as BERT expects.
- Padding layer: Matches lengths automatically.
- Attention layers : Uses self-attention as all transformer.
- Transformer layers: Uses BERT, LaBSE, etc. type models.
- Final pooling: Gives you a single vector representing the whole sentence.

With this, I am now able to take the similarities between the sentences. Which I did with the non-paraphrased and with the paraphrased to the, see the difference by comparing them. After this, I made some comparisons and graphics, which are explained in the next section.

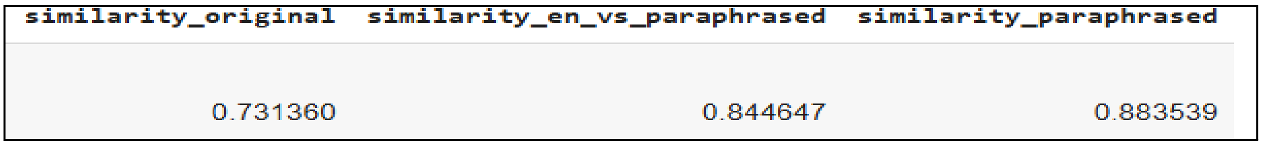
Then, I created a SpaCy model with the en_core_web_md model to compute the semantic similarity between English sentence pairs (sentence1_en and sentence2_en). This model uses pre-trained word vectors (GloVe): these are 300-dimensional embeddings trained on a large corpus .It processes each sentence into a Doc object containing tokens and vector representations, and then computes the cosine similarity between the vector averages of both sentences without taking into account the context of them.

3. Experimental design

Is paraphrasing as important as we think?

Purpose:

To see and check if paraphrasing is as useful as they say. So then I generated three types of similitudes.



similarity_original	similarity_en_vs_paraphrased	similarity_paraphrased
0.731360	0.844647	0.883539

Baseline:

Using SentenceTransformers I thought it would be interesting to compare similarities between sentences paraphrased and between sentences non-paraphrased.

Evaluation Metric:

similarity original: Is telling me how semantically similar the original translated sentences are to each other

similarity_en_vs_paraphrased: Is telling me If the paraphrasing model (Parrot) has managed to keep the same meaning.

If this score is high, the paraphrasing has been faithful.

similarity_paraphrased: If after going through the paraphrasing model, the two sentences are still equivalent.

This score reflects whether the overall meaning between the two sentences has been preserved after transforming both sentences and so the if it is higher than the first one, paraphrasing is useful.

Comparison between Transformers and SpaCy

Purpose:

The purpose of this comparison is because spaCy model uses word vectors (GloVe embeddings) which are based in average word vectors while the LaBSE model (transformers) are based in contexts so understands better what the phrase signifies, allowing for far more accurate and nuanced similarity comparisons between phrases.

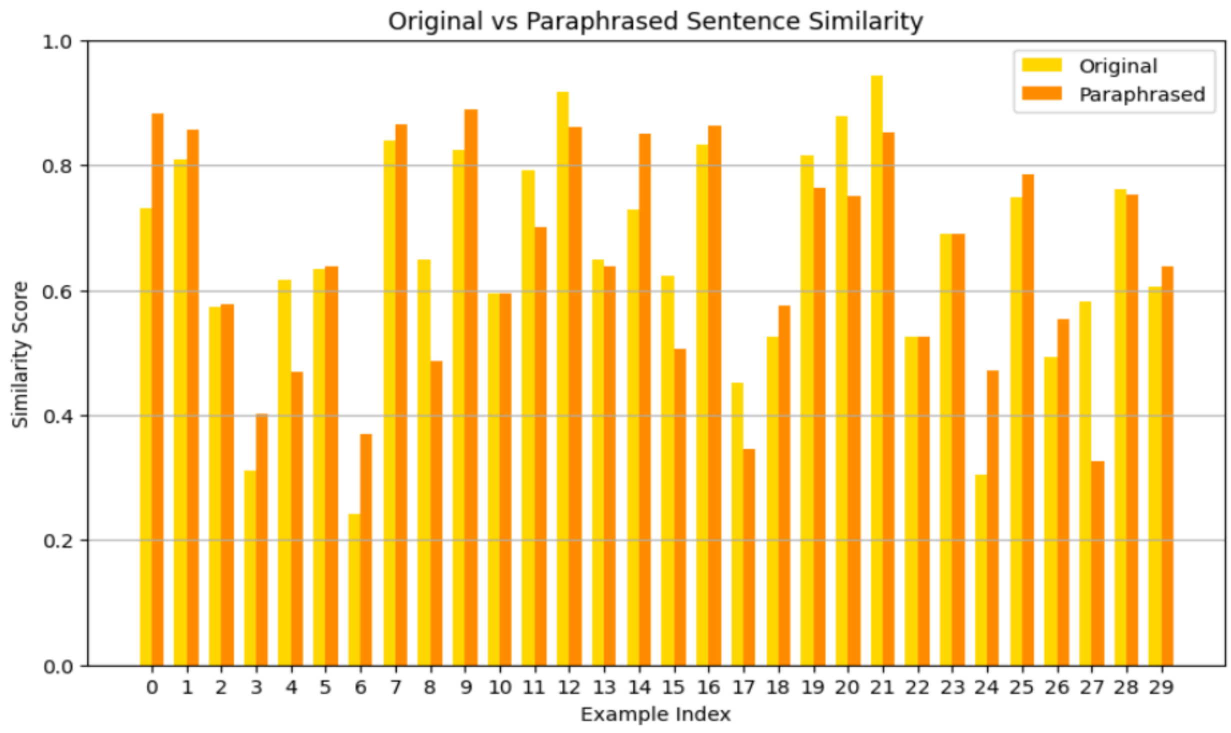
Baseline:

I decided to use the SpaCy library as Anna told me in the call we had. What I wanted to do is calculate the semantic similarity between two English sentences using the spaCy en_core_web_md model, which is based on word vectors (GloVe embeddings trained in English), and then do the same but using Transformers.

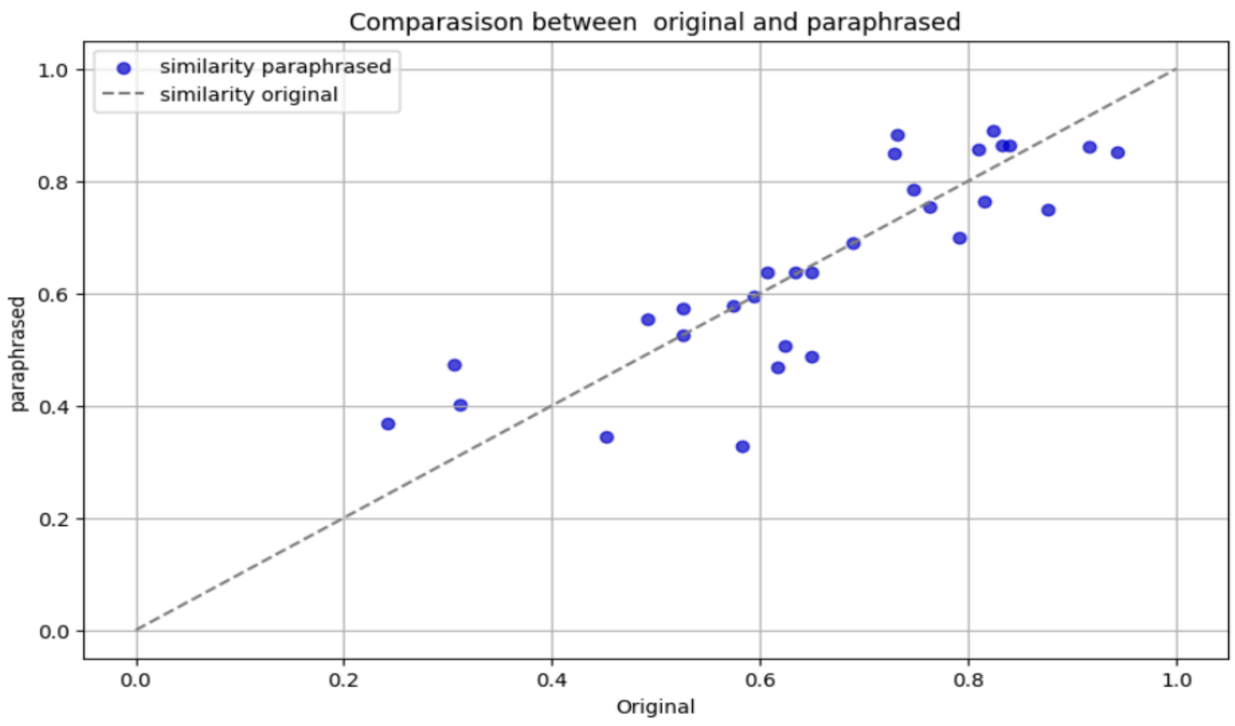
Evaluation metric:

Finally I compared my transformers model solutions with the score on the original dataset to see the performance.

4. Results



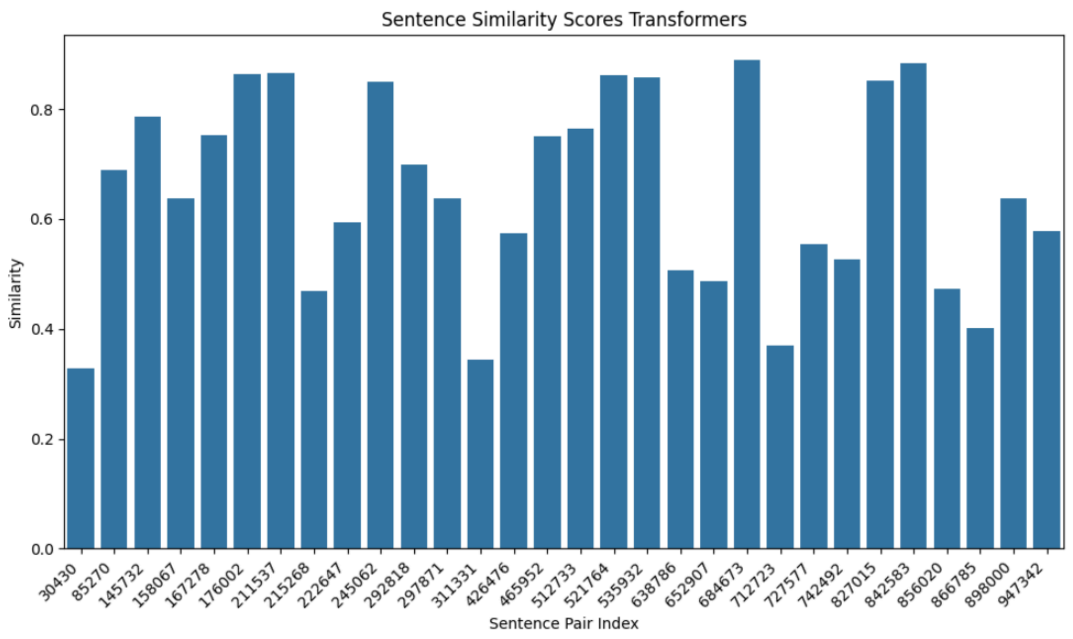
We can see in this graph which compares original vs paraphrased sentences, that in general paraphrased similarity is higher so it supports the idea that the use of paraphrasing is useful.

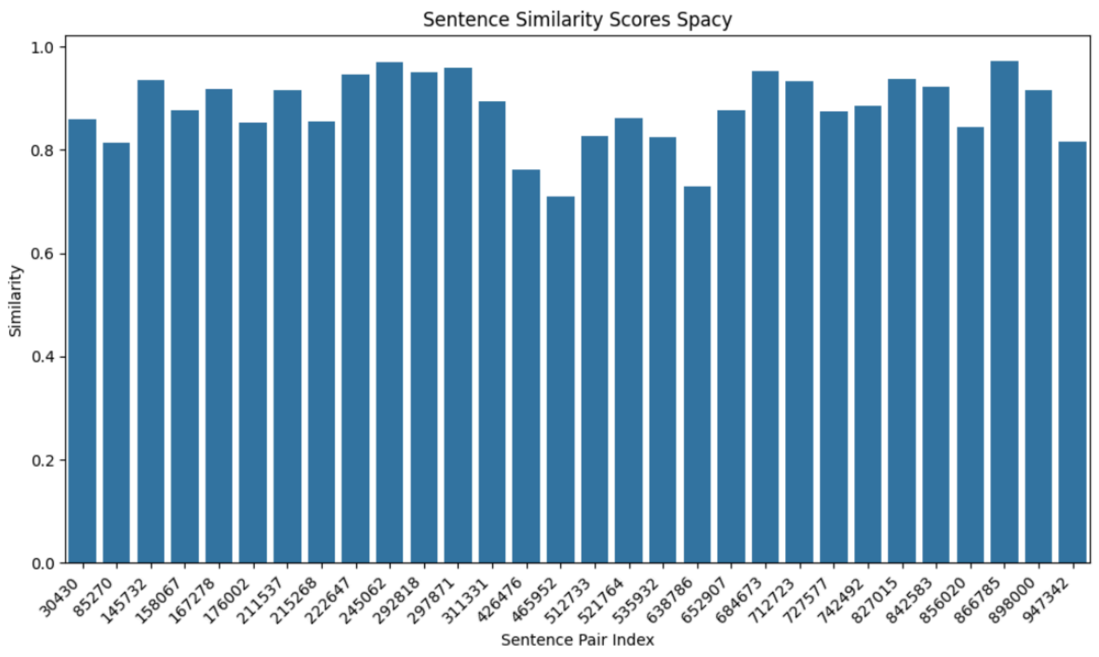


As we can see in this scatter plot, where the straight line represents the similarity between the phrases non-paraphrased and the dots represent the paraphrased ones, most of the

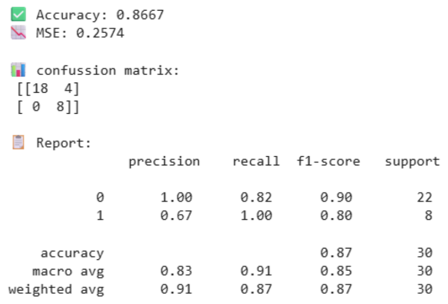
points are close to or below the line, indicating that paraphrasing has maintained equivalence in most cases, although it sometimes reduces it.
There are no extreme deviations, indicating good semantic stability in the process.

By comparing both SpaCy and transformers and with these two tables, I was able to demonstrate the limitations of surface-level similarity and highlight the advantages of using sentence embeddings for accurate semantic evaluation. As we see the values on the first graph are smaller because Transformers are more accurate and precise when calculating similarities between the sentences since it takes into account the context, not only the semantic structure, as SpaCy does, so as we can see (2nd graph), similarities are higher.





Now, to see how the sentences were in general similar between the, I took the accuracy and the mean squared error. As we see, the model achieved an an accuracy of 0.8667 which means that it correctly classified 87% of the sentence pairs based on my selected threshold (0.8). Additionally, the MSE was 0.2574, indicating a relative low average difference between the predicted similarity scores and the true labels, confirming that the models predictions are close to the expected values.



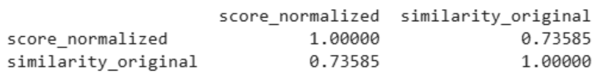
```
✅ Accuracy: 0.8667
📉 MSE: 0.2574

📊 confussion matrix:
 [[18  4]
 [ 0  8]]

📋 Report:
               precision    recall  f1-score   support

           0       1.00      0.82      0.90        22
           1       0.67      1.00      0.80         8

    accuracy                           0.87        30
   macro avg       0.83      0.91      0.85        30
weighted avg       0.91      0.87      0.87        30
```

Then, I also made a confusion matrix which resulted in very good results gor my model.

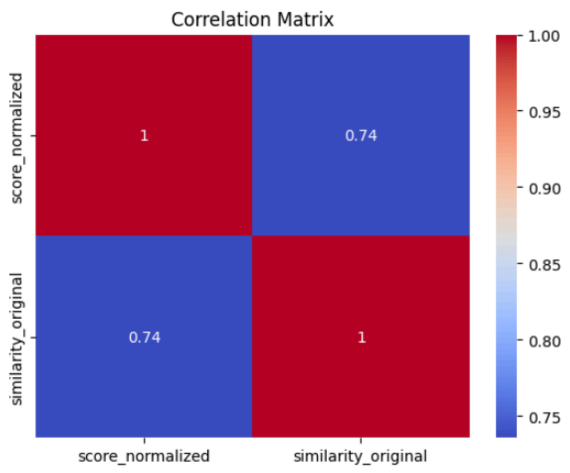
- 8 true negatives (TN): the model correctly predicted that the similarity was below the threshold (not equivalent).
- 18 true positives (TP): the model correctly identified sentence pairs as semantically equivalent.
- 4 false positives (FP): the model predicted equivalence when it shouldn't have.
- 0 false negatives (FN): the model never missed a truly equivalent pair.

For the final result, to see how my model performs I am going to compare it with the score on the original dataset. For that I made a correlation matrix between the score and the similarity_paraphrased (which is the one fron the transformers model).

This was the result:



```
                     score_normalized  similarity_original
score_normalized              1.00000              0.73585
similarity_original           0.73585              1.00000
```



So, the correlation between the original dataset score (normalized) and the semantic similarity computed using LaBSE was 0.74, indicating a strong positive relationship. This suggests that the transformer-based model effectively captures semantic equivalence in line with the reference scores provided in the dataset.

5. Conclusion

After having worked on this project and since I was alone and I had not used python as much as I used it in this project, I have learned a lot. It was difficult but I think it was worth it. This project has given me a much better understanding of the different types of models and their differences. It is always important to see what our precise objective is in order to use one model or another. In this case I have realized that it is better to use the transformers model because it is the one that takes into account the context of the sentences. Moreover, its structure is more complex and complete, so the results are more accurate. Finally, my model has turned out to be quite effective since the correlation score is 0.74, which is very good. This indicates that the transformer is capturing the semantic meaning between the phrases very well and that its similarity measure aligns with the original label of the dataset.