NLP Class Project 2025

(https://github.com/AntoLemp/nlp_exercise)

Team: Antonios Evangelos Lempesis (P22086)
Iakovos Prekas (P22147)

Contents

Project Description 2
Tools 2
The importance of Semantic reconstruction and NLP's role in the process 3
Methodology 3
Part A 3
Part B 4
Part C 5
Experiments & Results 6
NLPAnalyzer 7
CustomNLP 10
Food for Thought & Conclusions 13
Sources 14

Project Description

In this project, we will work with the two provided texts and aim to reconstruct them to enhance their semantic clarity and, where possible, improve their grammatical accuracy as well.

We will reconstruct 1 sentence from each text on our own. And use 3 different Language Models on each text to paraphrase them.

Finally, we are going to ask **ChatGPT** **4.0** to paraphrase them and use that as our golden standard since we believe that right now, no matter what **Language Model** we use on our local machine, there is no way we are getting a semantically better text than that of ChatGPT.

Text1 = 'Today is our dragon boat festival, in our Chinese culture, to celebrate it with all safe and great in our lives. Hope you too, to enjoy it as my deepest wishes. Thank your message to show our words to the doctor, as his next contract checking, to all of us. I got this message to see the approved message. In fact, I have received the message from the professor, to show me, this, a couple of days ago. I am very appreciated the full support of the professor, for our Springer proceedings publication'

Text2 = 'During our final discuss, I told him about the new submission — the one we were waiting since last autumn, but the updates was confusing as it not included the full feedback from reviewer or maybe editor? Anyway, I believe the team, although bit delay and less communication at recent days, they really tried best for paper and cooperation. We should be grateful, I mean all of us, for the acceptance and efforts until the Springer link came finally last week, I think. Also, kindly remind me please, if the doctor still plan for the acknowledgments section edit before he sending again. Because I didn't see that part final yet, or maybe I missed, I apologize if so. Overall, let us make sure all are safe and celebrate the outcome with strong coffee and future targets'

Tools

To run our codebase, we created a virtual environment using conda and added our dependencies using poetry to better organize our libraries and make it possible for others to clone our repository and use it without having to download anything extra .

Version: Python >= 3.10
Dependency Manager: Poetry
Libraries: pandas, numpy, matplotlib, torch, scikit-learn, nltk, transformers, sentencepiece, protobuf, seaborn, sentence-transformers, genism
Environment Considerations: Conda

The importance of Semantic reconstruction and NLP's role in the process

Semantic reconstruction is the process of restoring or refining the meaning of text while preserving its intent, often by correcting errors, disambiguating words, or optimizing structure. NLPs are used for fixing grammatical errors while retaining semantics, paraphrasing to make the sentences clearer, being able to differentiate the meaning before the same word inside different contexts.

Methodology

Part A

In this part we are going to try and reconstruct 1 sentence from each text.

From **text 1** we chose: Thank your message to show our words to the doctor, as his next contract checking, to all of us. While from **text 2** we chose: Anyway, I believe the team, although bit delay and less communication at recent days, they really tried best for paper and cooperation.

We chose these sentences because they aren't grammatically correct, but we can mostly understand them. So, we will be able to make them semantically and grammatically correct by mostly moving around and fixing the grammatical form of some words.

First, we will need to load them from their respective files. As you can see, from sentence1.txt, we have split the complex sentence in smaller ones so we can work on replacing/fixing small parts of it.

Using the nltk library, we created **context free grammar** that we are going to use to parse the sentences to show that it is at least grammatically correct. Please note that the grammar follows the Cambridge Dictionary but we have made a few omissions just so it doesn't expand too much.

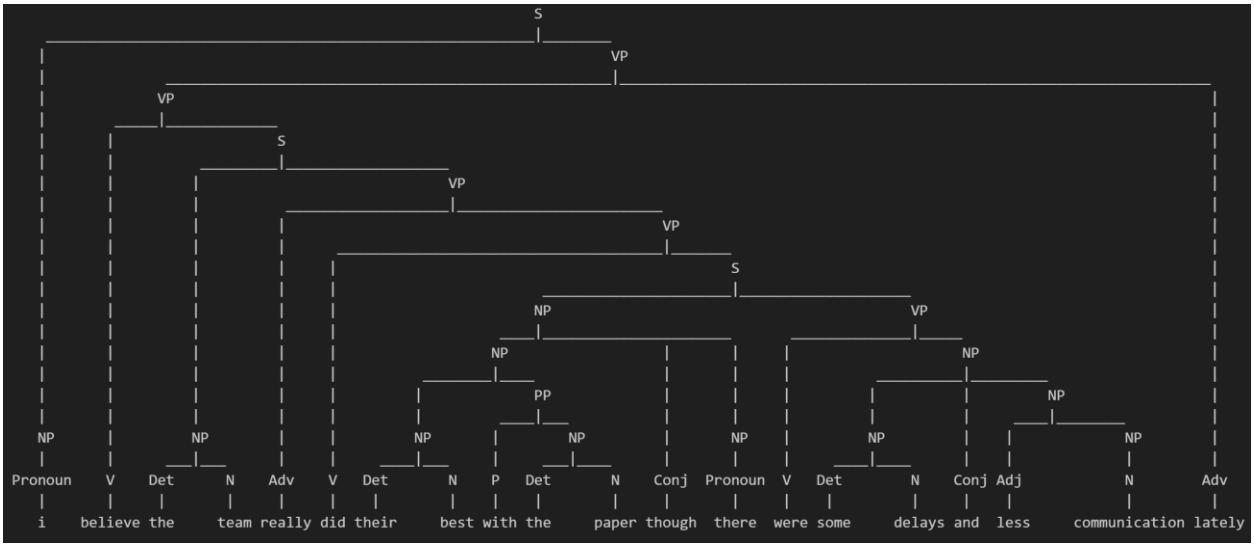
In the **function sentence_fixer(sentence)** we decided to rewrite each sentence by replacing specific phrases with more polished or clearer alternatives. The sentences we ended up with are:

Fixed_sentence1: thank you for your message, it helped convey our thoughts to the doctor, who will assist all of us with checking the next contract.

Fixed_sentence2: i believe the team really did their best with the paper, though there were some delays and less communication lately.

We believe that both sentences are more grammatically and semantically correct than the original. We can prove this by using a **parser** with the grammar we created. First we will

need to tokenize each word and remove any stopwords and then using the parser, create a parsing tree.



Example 1 – Parsed tree for second fixed sentence. (You can find the rest in our code)

Part B

Now we are going to use **3 Language models** to paraphrase and reconstruct the whole text. We will split each text into sentences to we get better results overall since we believe that the paraphrasers work better with smaller sentences.
All the models used were from the https://huggingface.co website and you can find more information about each one by going to **https://huggingface.co/<name-of-model>** where <name-of-model> is the full name which you can find next to the following bullets.

Language Models:

- humarin/chatgpt_paraphraser_on_T5_base
  This model is based on the T5-base model while its dataset is based on the Quora paraphrase question, texts from the SQUAD 2.0 and the CNN news dataset. It uses "transfer learning" to get the model to generate paraphrases.
- tuner007/pegasus_paraphrase
  This model is based on the Pegasus library. You can find more info about it here: arXiv:1912.08777v3 [cs.CL].
- stanford-oval/paraphraser-bart-large
  This model is based on the bart-large model, and is an extension of facebook/bart-large, it is better described in the following paper arXiv:2010.04806v2 [cs.CL].

For some reason, even though the laptops have onboard graphic cards we only managed to use the “cpu” for the models, but at the end of the day, it doesn’t really matter since it is

quick enough and managed to create sentences that we feel really manager to fix the semantic and grammatical errors from the original sentences.

First, we had to use the respective **tokenizer and seq2seq mode**l libraries suggested by the authors. Then following the deployment example providing in the humarin/chatgpt_paraphraser_on_T5_base, we made the **paraphrase function** and tuned its variable to better work with all 3 models.

We ended up deciding to only generate one sentence for each sentence of each text so everything is done automatically, and we don't have to pick manually which paraphrased sentence is the best semantically and grammatically.

We then saved each sentence to a txt file with the following format:
<model>-paraphrased-text<1 or 2>.txt

```
Final Paraphrased Text1:
Our Chinese culture has a dragon boat festival that is celebrated today.
Hope you enjoy it as I wish.
Thank you for your message, which will be shown to the doctor.
I received this message to see the approved one.
I received a message from the professor, to show me what he was talking about.
The professor supported the Springer proceedings publication.
```

Example 2 – Pegasus model paraphrasing text1

Part C

For this part, as mentioned at the start, we used a ChatGPT generated paraphrase, as a benchmark, for all our sentences. This way we will be able to tell how close each of the paraphrased sentences is to the actual "golden standard".

Benchmark Text1: Today is the Dragon Boat Festival in our Chinese culture—a time to celebrate safety, happiness, and well-being in our lives.I also wish you the very best during this special occasion. Thank you for your message and for helping convey our thoughts to the doctor regarding his upcoming contract review on behalf of all of us. I have seen the approved message. In fact, I received it from the professor a few days ago. I truly appreciate the professor's full support for our Springer proceedings publication.

Benchmark Text2: During our final discussion, I mentioned the new submission—the one we had been waiting on since last autumn. However, the update was a bit confusing, as it didn't seem to include the full feedback from the reviewer or perhaps the editor. In any case, I believe the team—despite some delays and reduced communication recently—really did their best with the paper and the overall collaboration. We should all be grateful

for the acceptance and the efforts made, especially now that the Springer link finally came through last week. Also, please kindly remind me if the doctor still plans to edit the acknowledgments section before resending the paper. I haven't seen the final version of that part yet—or I may have missed it. If so, I apologize. Overall, let's ensure everything is in order and take a moment to celebrate this achievement—with a strong coffee and new goals ahead!

Just by reading this, we can see that the sentences are now semantically and grammatically correct, meaning we can use them to compare them with the rest of the paraphrased sentences.

We can make the first simple comparison by using TfidfVectorizer (Term Frequency–Inverse Document Frequency vectors) and cosine_similarity from the sklearn library. We define the function compute_cosine_similarity(text1, text2) in which we create the **tfidf_matrix** where each dimension corresponds to a word in the combined vocabulary of text1 and text2 and then we compute the **cosine similarity** between the two vectors. We will discuss the results in the following section.

Experiments & Results

Results from the previous comparison.

Original vs. Benchmark

- **Text1: 0.6790**, **Text2: 0.7174**
    - These show **moderate to high similarity**, indicating that the benchmark text closely reflects the original one.
    - This is expected since the Benchmark is simply paraphrasing it but at the same time might be using more complicated words/phrases that aren't part of the original one
- **Sentence1: 0.4202**, **Sentence2: 0.3301**
    - Sentence-level similarity is significantly lower.
    - Suggests the specific sentences in the benchmark differ more in **word choice or structure** from the original one than the full texts do.

Model-Generated vs. Benchmark

- **ChatGPT-T5**:
    - **Text1: 0.6003**, **Text2: 0.7301**
    - Close to or higher than the original-to-benchmark similarity (especially for **Text2)**, suggesting T5 generated a version quite aligned with the benchmark (might be because both used ChatGPT for their training).
- **Pegasus**:
    - **Text1: 0.5105**, **Text2: 0.6726**
    - Lower than ChatGPT-T5 in both cases. Still captures substantial similarity but likely uses more **divergent phrasing**.

- **BART**:
    - **Text1: 0.6018**, **Text2: 0.6827**
    - BART performed comparably to ChatGPT-T5, especially on **Text2**, and slightly better than Pegasus.

Fixed Sentences vs. Benchmark Sentences

- **Sentence1: 0.5103**, **Sentence2: 0.5920**
    - Higher than the original sentences similarity to the benchmark (0.42, 0.33).
    - Indicates our **manual fixes improved alignment** with the benchmark text.

Overall, we can see the similarity between the original texts and the benchmarks is around 0.7, meaning that even though we reconstructed it to make it more understandable, it kept its original meaning.
Another thing we could point out is that the reconstructed Text2 is way closer to Benchmark Text2, than the reconstructed Text1 to Benchmark Text1. This could be happening because Text2 was already semantically okay compared to Text1 making the reconstruction a lot easier.

NLPAnalyzer

Class performs NLP tasks with Sentence-BERT (sentence-transformers) to compute semantic similarity between texts using cosine similarity of embeddings.

It uses the pre-trained sentence transformer model (all-MiniLM-L6-v2), which provides high-quality embeding without the need for additional training.

It provides the following functions:

- **__init__(self, model_name='all-MiniLM-L6-v2'):** Initializes the Model turns sentences into dense vector representations.
- **compute_similarity(self, text1, text2):** Takes two sentences and computes cosine similarity between their embeddings.
- **preprocess_text(self, text):** Currently only does simple lowercasing and trimming whitespace.
- **visualize_embeddings(self, texts,labels):** Encodes all texts into embeddings while reducing the dimension to 2D so they can be used for PCA (linear overview) and t-SNE (non-linear)
- **analyze_texts(self, texts):** Preprocesses all texts, runs pairwise cosine similarity on every unique pair in the list and then results as a list of tuples.
- **run_analysis(self, texts,labels):** Pretty much the start function. It calls analyze_texts() to compute all similarities and runs visualize_embeddings().

Results

Original vs. Benchmark

- **Text1: 0.9724, Text2: 0.9320**

Model-Generated vs. Benchmark

- **ChatGPT-T5**:
  - **Text1: 0.9631, Text2: 0.9332**
- **Pegasus**:
  - **Text1: 0.9165, Text2: 0.9052**
- **BART**:
  - **Text1: 0.9283, Text2: 0.8868**

Fixed Sentences vs. Benchmark Sentences

- **Sentence1: 0.9054, Sentence2: 0.8726**

Now we have way higher scores than any TF-IDF because Sentence-BERT understands semantics, meaning it analyzes the whole sentence and doesn't just compare word co-occurrence. So even if two sentences don't share many exact words, if their meanings are similar, the cosine similarity as we can see stays high.

What we can understand from these results is that we were correct in believing that the sentences created were semantically and grammatically correct. Plus the lowest scores are still the ones involving inner sentences which make sense since those are the hardest to compare and might still not have the exact meaning.

We can better see these scores by seeing the PCA and t-SNE visualizations

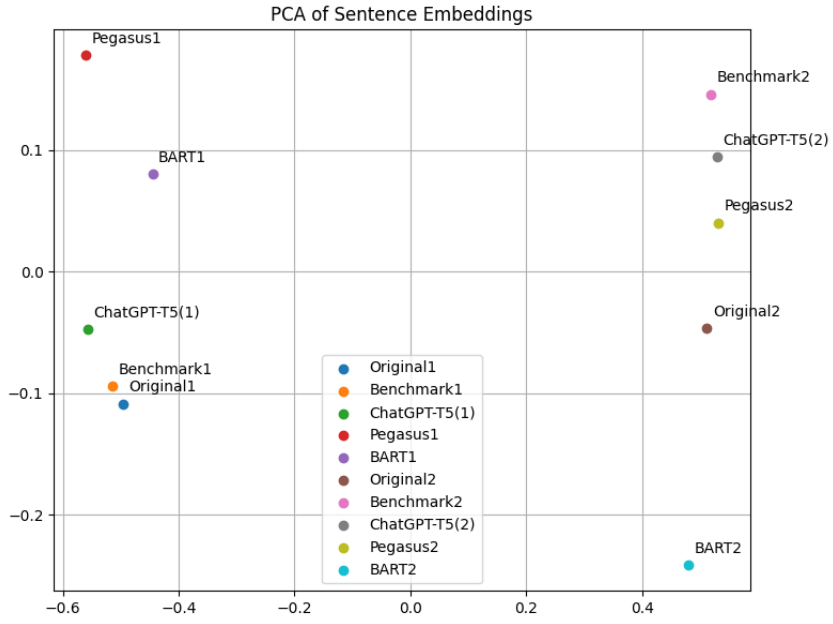


Image – PCA sentence embedding (NLPAnalyzer)

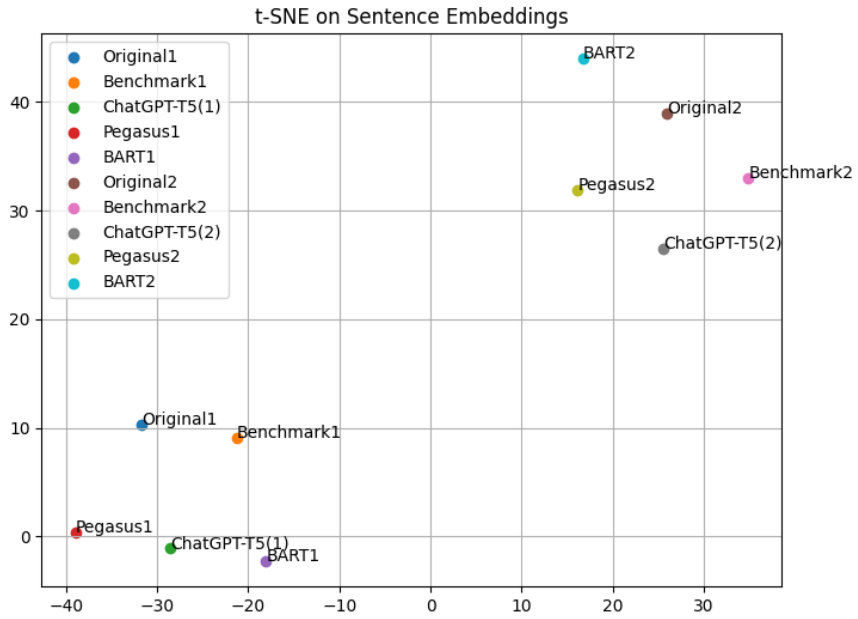


Image – t-SNE sentence embedding (NLPAnalyzer)

As we can see both PCA and t-SNE, the sentences are in two groups, one around Original1 and the other around Original2.
For PCA specifically, we can see that most semantically similar sentences aren't always ranked the same for both groups. In the first group, the benchmark and ChatGPT-T5 are much closer to the Original1 meaning they are similar to it and probably kept some of its characteristics. On the other hand, Bart1 and Pegasus1 are almost 0.1 points away meaning their different types of computation didn't help them that much in this case.

Group2 is a different story, here we can see that none of them are really close to the Original2 but BART2 is by far the worst being the furthest from its original text yet again.

For t-SNE, we are comparing the local structure, here we can see that the sentences are somewhat equally distanced from each other. Meaning that even though their structure is really like each other in both cases, none of the LM's gave us the same exact answer.
Also, we can see that the BART models again are the worst ones.

CustomNLP

Performs advanced NLP tasks, including:

- preprocessing,
- word embedding training (Word2Vec),
- semantic clustering,
- visualization.

Unlike the previous NLPAnalyzer (which used pre-trained sentence embeddings), this implementation builds its own word embeddings from scratch and applies them to analyze text similarity and clustering.

It provides the following functions:

- **__init__(self, embedding_dim=100):** Initializes the process, it is keeping a lemmatizer, stop_words, vocab, word2vec_model and word_embeddings
- **preprocess(self, text):** Tokenizes, lowercases, removes punctuation and stopwords, keeps nouns, verbs, and adjectives, lemmatizes them and then returns a unique set of tokens.
- **build_vocabulary(self, texts):** Preprocesses each text and the returned tokens are saved in a vocabulary
- **train_word2vec(self, tokenized_texts):** Uses gensim's skip-gram Word2Vec and stores the vectors created in word_embeddings
- **get_embedding(self, word):** Returns the embedding for a specific word
- **build_concept_clusters(self, n_clusters=20):** Reduces word vectors with PCA, clusters words into n_clusters using KMeans and then it returns them.
- **def visualize_clusters(self, clusters):** Uses NetworkX to create a graph of word pairs inside clusters.
- **sentence_embedding_weighted(self, sentence):** Computes the weighted average of word vectors in a sentence and returns a single sentence embedding.
- **visualize_embeddings(self, texts, labels):** Creates both PCA and t-SNE plots of sentence vectors.

Pretty much what this class does is build a domain-specific understanding via Word2Vec trained on your texts, it lets you examine semantic structure without relying on prebuilt embeddings and using clusters and graphs it shows how things are organized.

Results

Original vs. Benchmark

- **Text1: 0.9440**, **Text2: 0.9806**

Model-Generated vs. Benchmark

- **ChatGPT-T5**:
  - **Text1: 0.9251**, **Text2: 0.9667**
- **Pegasus**:
  - **Text1: 0.9068**, **Text2: 0.9710**
- **BART**:
  - **Text1: 0.9331**, **Text2: 0.9730**

Fixed Sentences vs. Benchmark Sentences

- **Sentence1: 0.8078**, **Sentence2: 0.8466**

In comparison with NLPAnalyzer they pretty much get the same results when comparing the entire texts. We could say that CustomNLP gets much better results for Text 2 but at the same time we can't really say when it comes to sentence vs sentence the results for CustomNLP are much lower, this might be because Word2Vec struggles with big differences and isn't as good as NLPAnalyzer when it comes to paraphrasing and generally getting the same meaning from two different sentences. But overall, it did a fantastic job in showing that the paraphrased sentences are pretty close to the original one.

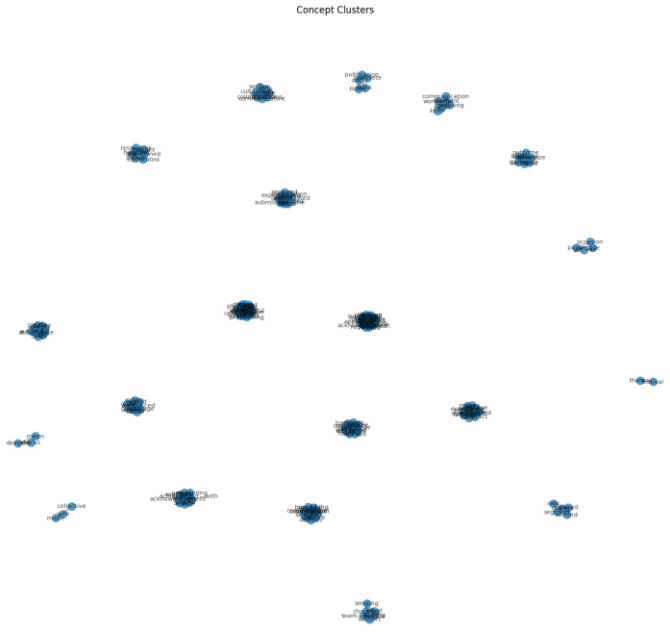


Image- Concept Clusters (CustomNLP)

The image shows that CustomNLP (mostly) successfully identified and organized semantically related words into coherent clusters. (e.g., "communication", "teamwork", "submitted") It also successfully removed any unnecessary words.

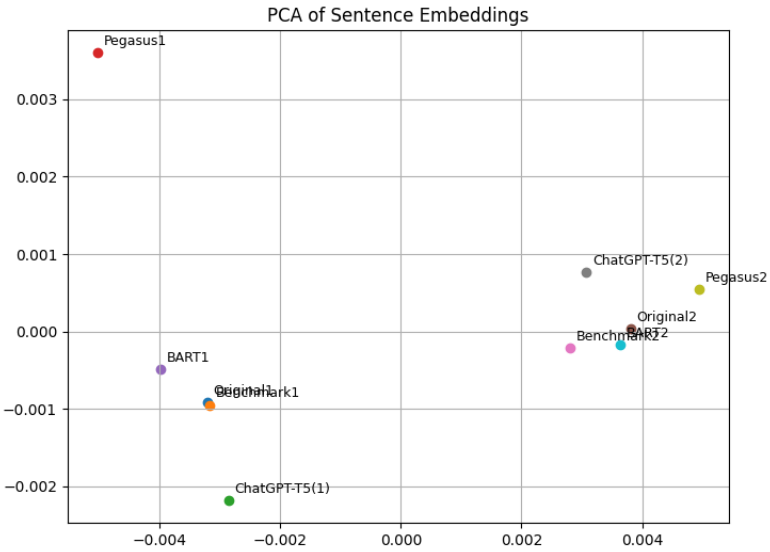


Image – PCA for Sentence Embeddings (CustomNLP)

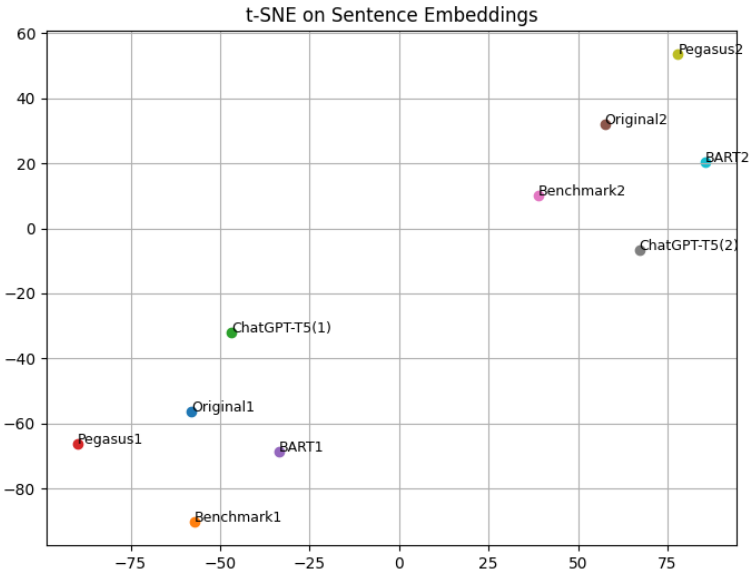


Image – t-SNE on Sentence Embeddings (CustomNLP)

We pretty much see the same results as NLPAnalyzer although an interesting thing to notice is that on PCA everything is extremely close to each other except Pegasus1, meaning that we overall Pegasus1 might be the worst paraphrased sentence out of all them.

Food for Thought & Conclusions

Concept Clusters: The concept clustering graph shows clearly defined semantic groupings, (e.g., "communication", "teamwork", "submitted"). This indicates that the Word2Vec model was able to learn meaningful vector spaces where semantically similar terms exist. But at the same time, we have words that aren't that similar and still got grouped into the same cluster, which in projects like this, they don't change much but they could be detrimental if the decision between 2 sentences depends on that small variable.

Sentence Similarity Scores: Cosine similarities between generated and benchmark texts are mostly above 0.90, showing that sentence embeddings were able to approximate the original meaning. But at the same time, they could be lexically distant since our models don't really take that into consideration.

(PCA/t-SNE): The PCA and t-SNE plots show that original, benchmark, and generated sentences stay in tight clusters, especially in Text2. That shows the embeddings stay semantically close.

The biggest challenges in reconstructions were finding out how to find paraphraser models that worked for our intended purposes. If you look at the Hugging Face website, you will see that most models are based on T5 models, so we ended up using models that don't have many downloads nor quite extensive documentation. Maybe if we tried using different websites we could find more widely used Language Models for Seq2Seq.

Now, if we are talking about more technical challenges, we could say that optimizing the paraphraser to accept all types of Models and get actually good results from it was a challenge since most of the time, we wouldn't get Seq2Seq answers but translations or even answers to our input.

As we show from our results this could be semi-automated, NLP can paraphrase correctly, but because they don't always consider the actual meaning of the word, they could be lexically distant. Meaning that a human still needs to take a look before using it.

- ChatGPT-T5 is the closest one to the actual Benchmark (possibly because the same sentences were used for the training for both). It is generally paraphrased correctly.
- BART the sentences aren't as clear as T5, it struggles with long sentences which are badly written and don't have a clear meaning from the start.
- Pegasus strong paraphrasing capabilities but cuts out a lot of details.
- CustomNLP, good and efficient but it ignores syntactic nuance.
- NLPAnalyzer same problem as CustomNLP but its embedding and accuracy is a bit better.

Overall, we believe that none of the techniques are perfect but all of them can produce a produce a mostly positive result, meaning a semantically and grammatically correct sentence.

If we could start from all over again, I think we would be searching for more types of Language Models that could get a different result or try to compare T5 models against each other to see which one fits our case better.

Sources

- AutoQA: From Databases To QA Semantic Parsers With Only Synthetic Training Data arXiv:2010.04806v2 [cs.CL]
- PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization arXiv:1912.08777v3 [cs.CL]
- Hugging Face
- Transformers
- Sentencepiece
- SBERT
- gensim
- Cambridge Dictionary