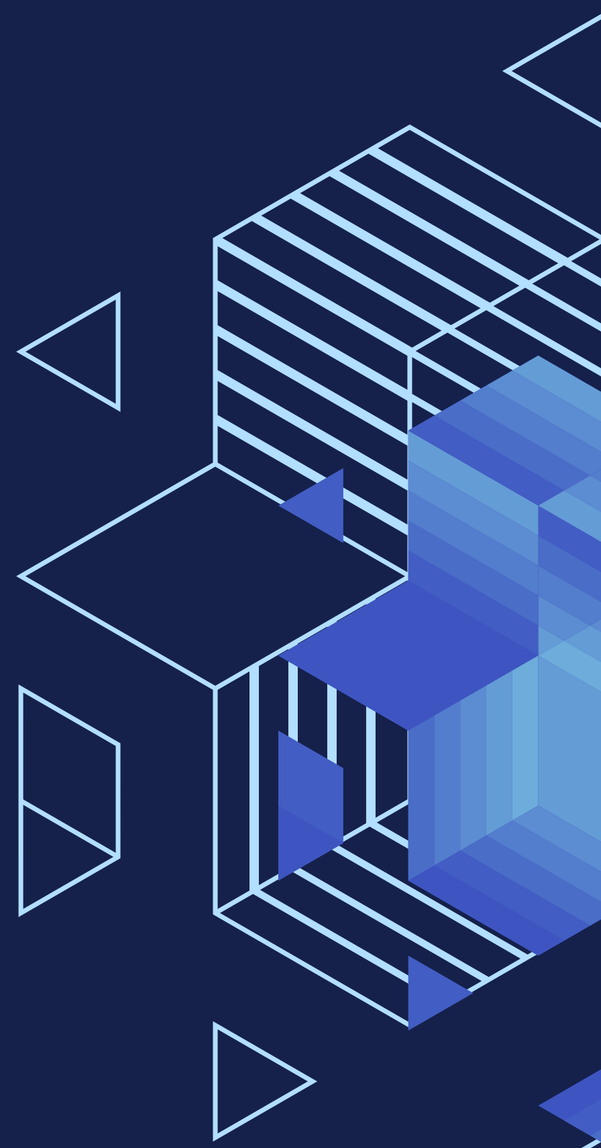
PreCog Rectruitment Tasks

-By Mohd Zeeshan

Table Of Content



Programming Task

SynCluster

clusters are all you need

A semantic vector clustering technique for pair-wise word-embedding similarity scoring using lexical databases

Introduction

This project introduces an unsupervised word embedding technique that leverages existing lexical databases, more specifically WordNet[1] database. WordNet contains sets of words attached to every word within its corpus known as Synsets[2] which stands for synonym sets. These sets are words with similar meaning as the target word and are used in a variety of different contexts. Similarly, WordNet also contains antonyms for the target word(s).

The main idea behind this technique is to strategically cluster the target word along with its synonyms in the vector space based on its occurence within other synset clusters. The same is done for antonyms but in a different vector space. We then use the euclidean distances from both to calculate a final similarity score for the word-pairs.

[1] https://wordnet.princeton.edu/documentation
[2] https://www.geeksforgeeks.org/nlp-synsets-for-a-word-in-wordnet/

Corpus/Dataset

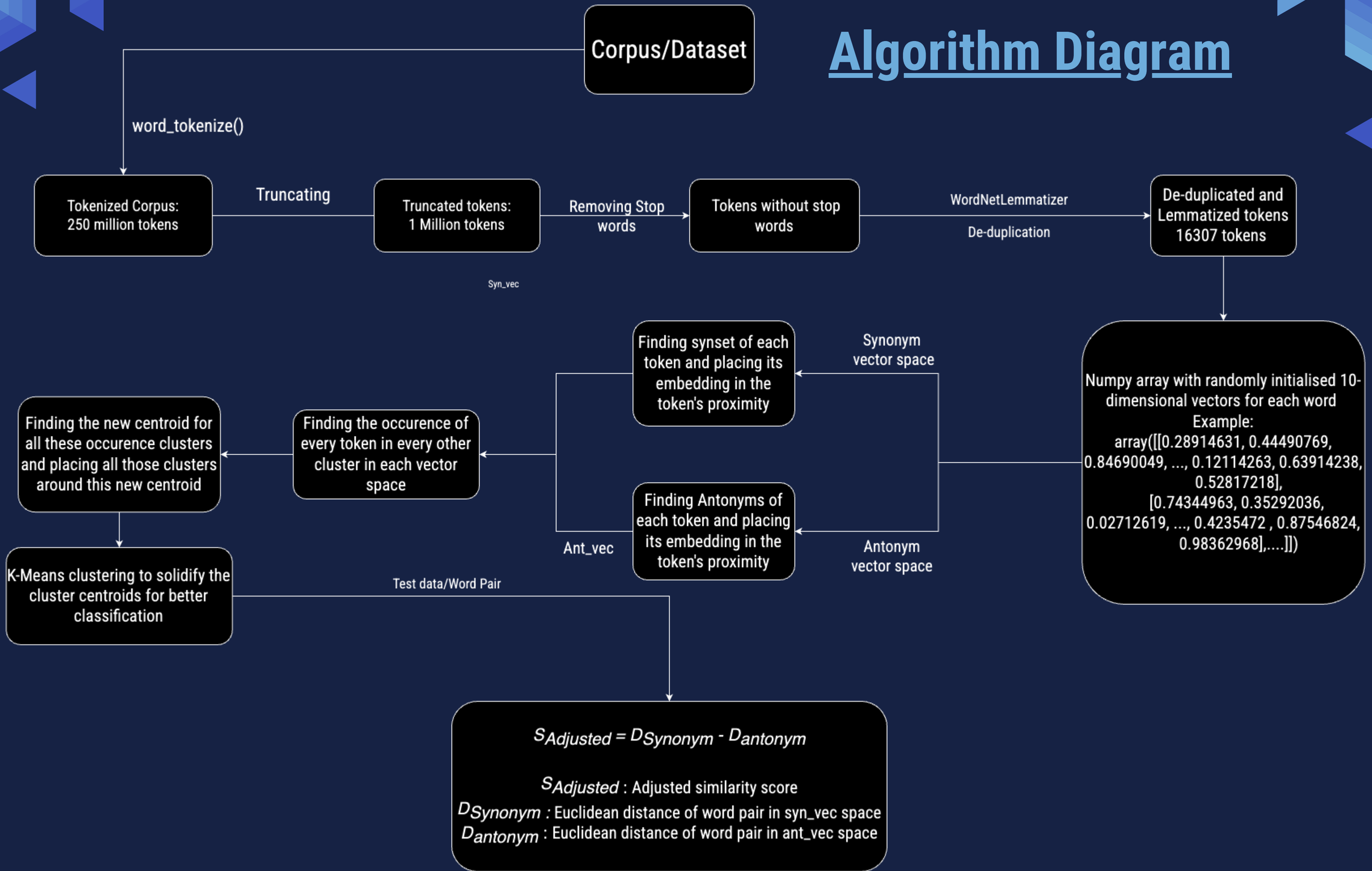
Amazon Fine Food Reviews Dataset

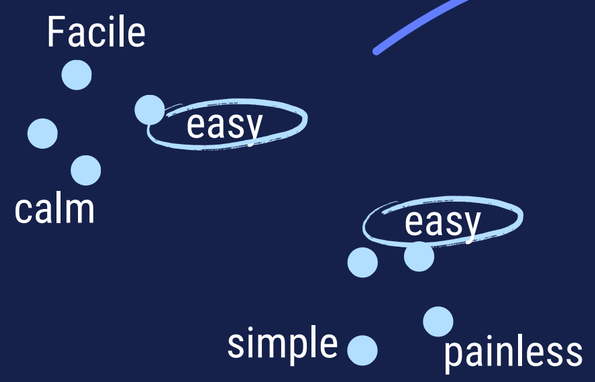
For this project, the dataset used was Amazon's Fine food reviews[3] which contains 568,454 unique user written reviews. This dataset was chosen due to its large amounts of regularly used vocabulary since the content is completely user-generated. This would likely cover all the common words within the bell curve of english language.

The total number of tokens was about 250 million. Since the task is to do it within 1 million tokens, I truncated the token list to 1 million at maximum.

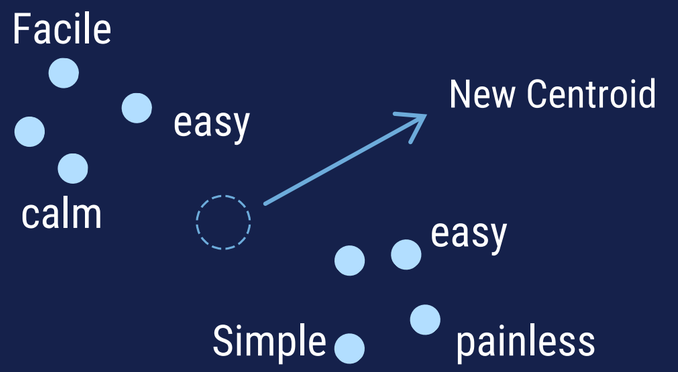
[3] https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

Algorithm Diagram

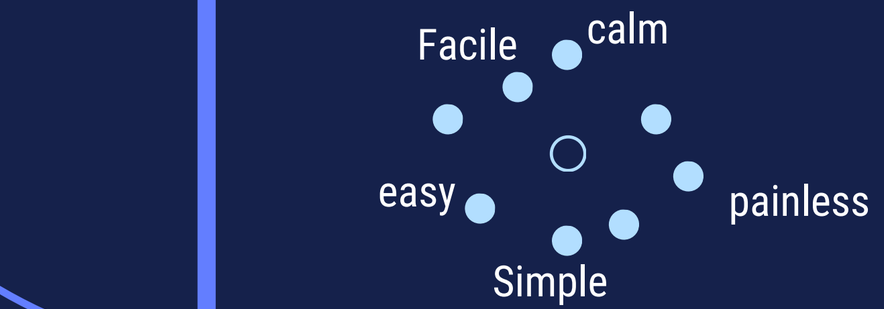




Step 1: Same word in a different cluster detected



Step 2: New centroid calculated from old cluster’s centroids



Step 3: Vectors of both clusters placed in proximity to the new centroid

Step 4: Repeat

Result analysis

- The model's training time is about 20 minutes for this dataset and increases exponentially with corpus size. This is due to the fact that the time complexity of K-means clustering increases with size in input data and class size.
- The test dataset is the word-pairs from SimLex-999 dataset. It is an evaluation dataset for word similarity models consisting of 999 word pairs with human-labelled similarity scores for each word pair
- We first pre-process the data by removing the rows where either of the words from the pair are missing from our vocabulary. Then we iterated through each word-pair and calculated the euclidean distances for each pair. Also, the distance is normalised to be between 0 and 1. This is the similarity score predicted.
- Then, MSE(mean square error) of each predicted similarity score and scaled+normalized SimLex scores is calculated, averaged and displayed in percentage

The final accuracy was :
83.997%

Note: this test was done only on the synonym vector space scores and not with antonym adjusted scores.

Phrase and Sentence similarity Detection

Phrase-pair and sentence-pair similarity detection using pre-trained GloVe word embeddings

Dataset(Phrase)

- For this task, the dataset provided was: PiC(phrase in context) phrase similarity[4] from HuggingFace.
- It is a human labelled dataset of 7000+ phrase pairs for similarity detection.
- This dataset has 6 columns: phrase1, phrase2, sentence1, sentence2, label and idx.
- idx is the id of the phrase pair.
- label is either 0 or 1 with 0 meaning not similar and 1 meaning similar.
- Phrase1 and phrase2 are the word pairs
- Sentence1 and sentence2 are the same sentences with the phrases put in the same place to show how in-context or out-context each one sounds.

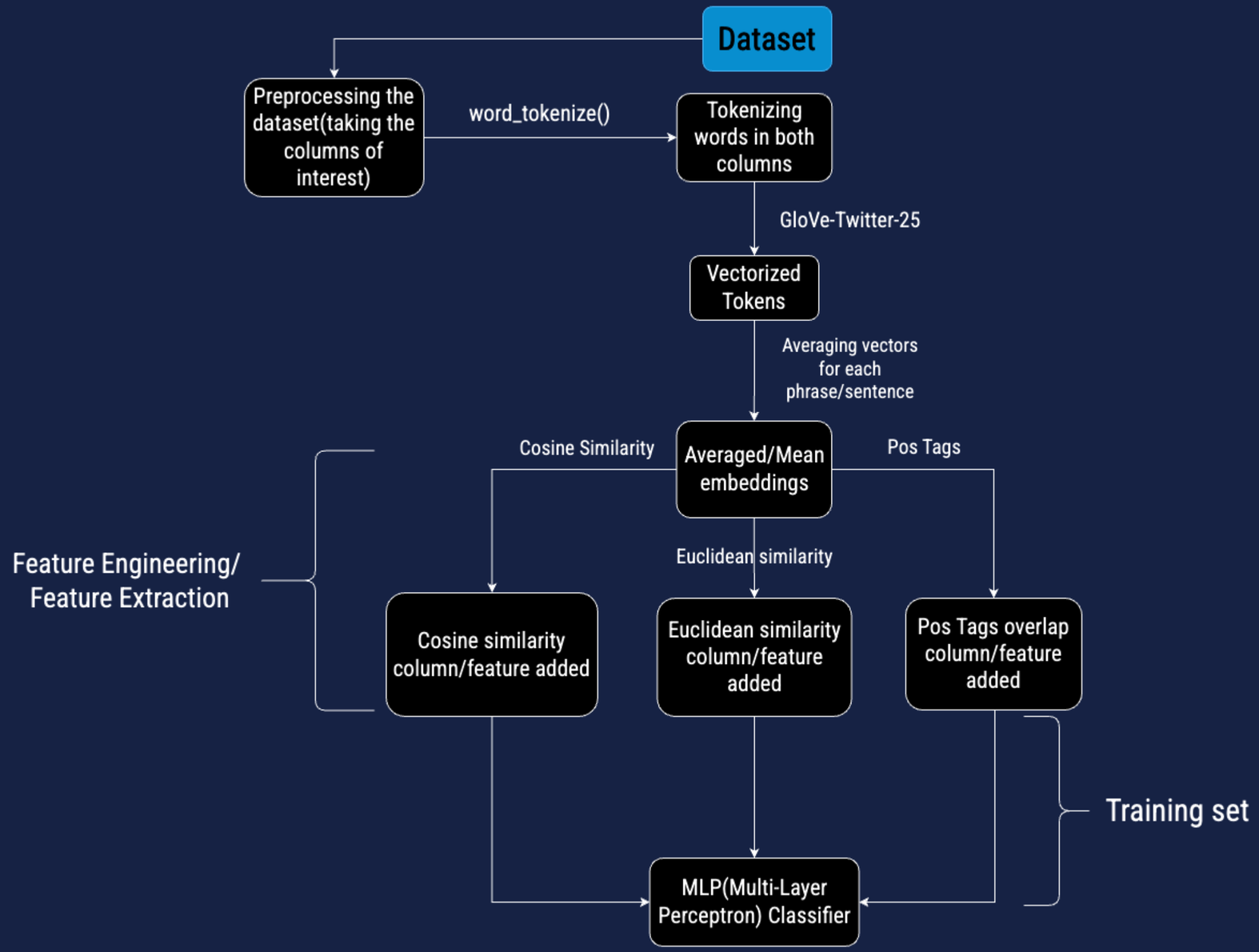
[4] https://huggingface.co/datasets/PiC/phrase_similarity

Dataset(Sentence)

- For this task, the dataset provided was: PAWS(Paraphrase Adversaries from Word Scrambling)[5] from HuggingFace.
- This dataset contains 108,463 human-labeled and 656k noisily labeled pairs that feature the importance of modeling structure, context, and word order information for the problem of paraphrase identification
- The dataset has two subsets, one based on Wikipedia and the other one based on the Quora Question Pairs (QQP) dataset.
- Dataset has 4 columns: id, sentence1, sentence2 and label
- sentence1 and sentence2 are the target sentence pairs that are paraphrased such that the semantic meaning is preserved.

[5] https://huggingface.co/datasets/paws

Algorithm Diagram(For Both Phrase and Sentence similarity)

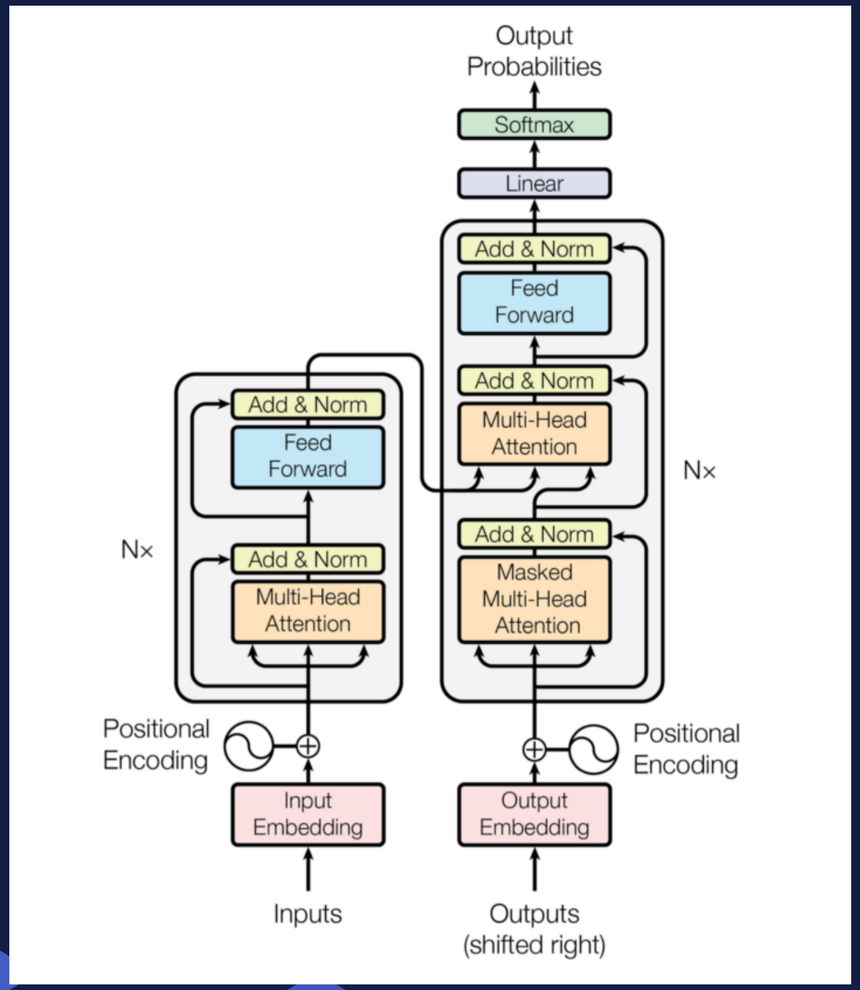


Result analysis

- The phrases/sentences were first tokenized and vectorized using GloVe-Twitter-25 embeddings
- GloVe(Global Vectors) uses word-word global co-occurence matrices to learn the contextual and semantic relatedness of words across the corpus rather than focusing on the local context.
- The reason for choosing GloVe vectors is to ensure that words in sentence show more relatedness in meaning rather than local context as the sentences are being paraphrased, meaning they are not losing their inherent meaning but instead they have minimal change in terms of semantics like order of words or use of synonyms.
- I took the average of each phrase/sentence's word vectors to give each phrase/sentence one single vector representation
- Then I extracted various features for each pair like cosine-similarity, euclidean distance and POS-tag overlaps
- Using these, features I trained various classifiers like RandomForests, SVC, SGD, KNN, etc and used GridSearchCV for best hyperparameters.
- After testing, I found out that MLPClassifier with 1e-7 lr performs the best with 0.5017 as accuracy score on phrase similarity and 0.5615 on sentence similarity.

Bonus Task: SIMBArt

Finetuning BERT-base-uncased for phrase-pair and sentence-pair similarity detection task



Introduction

SIMBArt(Similarity Biderectional Encoder Represention from Transformers) is a fine-tune of BERT-Base-Uncased[6] model, which is a pre-trained Transformer base model which can be fine-tuned for downstream tasks, and in our case that is Similarity detection between phrases/sentence pairs.

The reason I used BERT was because it is perfect for context-focused tasks, which, in our case was Phrase-pair and Sentence-pair similarity detection

[6] https://huggingface.co/bert-base-uncased
[2] https://www.geeksforgeeks.org/nlp-synsets-for-a-word-in-wordnet/

Transformer Architecture

- Transformer Architecture is a neural network architecture that was introduced in the paper titled “Attention is all you need”[7], Published by Vaswani et al. in 2017(Google Research) during the NeurIPS 2017 conference.
- Transformers in simple words, consist of a Encoder-Decoder system which uses Self-attention and a Multi-head attention mechanism to encode input tokens into Self-attention vectors which capture immense contextual meaning compared to previous Neural Network architectures.
- This makes Transformers extremely powerful for large natural language processing tasks like summarization, sentiment analysis, text classification, machine translation and most recently, for text-text, text-image and text-audio generation like ChatGPT, MidJourney and ElevenAI

[7] A. Vaswani et al., “Attention Is All You Need.” arXiv, 2017. doi: 10.48550/ARXIV.1706.03762.

Result analysis

- For phrase similarity detection, I preprocessed the data to create a single column of text like this: [phrase1] [phrase2] [sentence1] [sentence2]. This way, the model can recognize patterns within the paraphrases and learn from them
- The model was trained with AdamW(Adam with optimization weight decay) as the optimizer and loss function as sparse categorical cross entropy.
- Learning rate was set to 1e-5 and trained for 10 epochs.
- The accuracy achieved on the test set was 0.6490. which is slightly better compared to the original method I used with MLP classifier.
- This marginal improvement is likely due to the small number of training samples.
- For sentence similarity Detection, I preprocessed the data in the same way like this: [sentencce1] [sentence2]
- The rest of the training hyperparameters were same except it was trained on 3 epochs only
- The model performed significantly better than the phrase similarity with an accuracy of 0.9016.
- This is was likely due to the large number(40k+) of high quality training samples(data is king).

Paper-reading Task

BERTScore: Evaluating Text Generation with BERT

Introduction

- The paper “BERTScore: Evaluating Text Generation with BERT” introduces a new metric for evaluating the quality of text generation models. The metric, called BERTScore, is based on the BERT language model and uses contextual embeddings to compare the similarity between a generated text and a reference text.
- BERTScore is shown to have outperformed other commonly used metrics, such as ROUGE and BLEU, in capturing human judgments of text quality.
- The authors of the paper also provide an open-source implementation of BERTScore, which can be easily integrated into existing evaluation pipelines, Making it accessible to anyone who wants to use it to evaluate their own text generation models.

BERTScore: Evaluating Text Generation with BERT

Existing Evaluation Metrics

- The paper compares BERTScores with various existing evaluation metrics such as ROUGE, BLEU, and METEOR.
- Through tests, it was found that BERTScore outperformed the other metrics in terms of correlation with human judgement of text similarity and quality.
- It uses importance weight which alows it to give higher importance to words that are more semantically important in the sentence.
- This makes BERTScore more effective in evaluating text similarity and quality, especially in cases where simple word overlap or n-gram matching may not accurately capture the true meaning of the text.
- The authors employed the user of a pre-trained BERT model to compute the contextual embeddings of the input text and the reference text.
- This allowed BERTScore to capture the nuances and complexities of the language used in the text, which in turn improved its performance in evaluating text similarity and quality.
- It used recall BERT, precesion BERT and F1 FBERT to evaluate each pair. And through tests, the authors recommended the users/readers to use the FBERT for any practical purposes.

BERTScore: Evaluating Text Generation with BERT

Robustness Analysis

- The authors evaluated the robustness of BERTSCORE in the context of adversarial paraphrase classification, using the Quora Question Pair corpus (QQP) and the Paraphrase Adversaries from Word Scrambling dataset (PAWS).
- The QQP dataset distinguishes between real duplicate and related yet distinct questions, while PAWS generates adversarial examples through word swapping.
- BERTScore performs extremely well for both datasets as the authors have shown in the table
- Overall, BERTScore is task-agnostic and doesn’t depend on distant-dependencies, n-grams or even the language of text generation making it superior to the majority of the human-annotated metrics out there.

BERTScore: Evaluating Text Generation with BERT

Strengths

Semantic Understanding:

- The BERTScore paper demonstrates that using pre-trained BERT embeddings allows the metric to capture semantic information and contextual nuances. The authors show examples where BERTScore outperforms older metrics in tasks requiring a deeper understanding of meaning and context. It also uses the IDF scores to find the rarely used words to weight the importance.

Language Agnostic:

- BERTScore is designed to be language-agnostic and the authors showcase its effectiveness across multiple languages. They provide results and evaluations for diverse language pairs showing the versatility of the metric in cross-lingual tasks.

Correlation with Human Judgments:

- The paper presents correlation analysis between BERTScore and human judgments in various evaluation tasks. The authors conduct experiments and comparisons demonstrating that BERTScore aligns well with human assessments of the quality of generated text.

BERTScore: Evaluating Text Generation with BERT

Weaknesses

Language Limitation:

- In the paper, the authors demonstrated the solution’s robustness on various languages. But, these were widely used languages like English, Chinese, Turkish, etc. The authors failed to show its effectiveness on lesser used languages like Telugu or Kannada. This could prove to be a barrier especially in an age where startups right now are competitively training foundational LLMs in local languages.

Sensitivity to sentence length:

- The model heavily depends on the size of sentences to check its similarity with the paraphrase as it takes the contextual embeddings of each word and the IDF weighting to determine the semantic meaning of the sentence. Shorter sentence could fail to provide this luxury. For example the model would struggle to find the similarity between: "The complexity of modern urban dynamics is shaped by various socioeconomic factors, impacting employment and essential service accessibility." and "Various factors affect urban dynamics, from jobs to essential services." Both the sentences mean the same thing but have varying lengths.

BERTScore: Evaluating Text Generation with BERT

Improvements

- Making BERTScore more adaptable and flexible to work with local languages can be a good improvement. Since a lot of these languages do not have much training resources/data, few-shot training examples should be enough to make the model understand the vocabulary.
- Another important improvement could be to quantize the model since BERT is a transformer model and it extremely resource intensive. This could allow the model to run with lower resource and hence have a large-scale adoption from companies that train large-foundational models
- BERTScore can be used for RLAIF(Reinforcement Learning using AI Feedback)[8] where LLMs can be trained by getting its outputs constantly scored by this “supervisor” BERT model so that the LLM can improve its understanding without having a large available training corpus.

[8] H. Lee et al., “RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback.” arXiv, 2023. doi: 10.48550/ARXIV.2309.00267.

THANK YOU!