LLM Fine-Tuning for Semantic Similarity - ELEC-E5550 SNLP

Carlos Rivera 102179473,
Eduardo Fernandes 101419053,
Priyanshi Pal 101623962

19 April 2024

Contents

1 Introduction

Semantic Similarity task is a complex and an ever evolving problem in Natural Language Processing where the aim is to find how similar two pieces of texts are. While the straightforward cases involve overlapping words in the two texts yielding high similarity, our ultimate objective is to fine-tune an LLM, such that it can accurately classifying sentences as similar, even when they lack overlapping words, yet convey the same meaning or idea. Hence, the task is binary classification of semantic similarity.

The challenge of semantic similarity in natural language processing has historically been addressed through various methodologies, including traditional techniques like Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words (BoW), which represent text as numerical vectors but often overlook the intricate nuances of language. Knowledge-based semantic similarity methods, such as edge-counting, feature-based, information content-based, and combined techniques, leverage ontologies and knowledge bases for assessment. Corpus-based approaches, like Latent Semantic Analysis (LSA) and distributional models, rely on large text corpora for similarity calculations.

However, recent advancements have seen a shift towards Deep Neural Network-Based Methods, including Convolutional Neural Networks (CNNs), Long Short Term Memory (LSTM), Bidirectional LSTM, and Transformer models, which excel in capturing complex semantic relationships. These models, exemplified by works such as those by Wang et al. (2016)[3], Shao et al. (2017)[5], Tai et al. (2015)[2], and He et al. (2016)[2], have demonstrated superior performance on benchmark datasets like SICK and STS. Additionally, transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) and its variations, including ALBERT, have shown significant advancements in capturing deep semantic meanings and context. Evaluation metrics such as Cosine Similarity, BERTScore, Word Mover's Distance, and Massive Text Embedding Benchmark serve as tools to assess the effectiveness of these approaches.

Our goal is to evaluate the effectiveness of fine-tuning large language models (LLMs) as competitive models for semantic similarity classification. This includes exploring different combinations of datasets, particularly considering the role of overlapping words, and assessing their performance in the binary classification of sentences as similar or non-similar.

2 Methods and Experiments

2.1 Datasets

For our experiments, we chose the three most popular datasets that are used in the Semantic Similarity task: Semantic Text Similarity (STS), Microsoft Research Paraphrase Corpus (MSR) and Quora Question pair (QQP) datasets. While the STS dataset has been popularised in various semantic similarity detection tasks, we found MSR dataset worth including as it contains various examples of sentences which are semantic similar and don't necessarily have overlapping words. Some of the examples are paraphased quite cleverly and very differently as well. Lastly, we decided to also include QQP, inspired by the paper [7], where the authors compare performace of pre-trained models across all three STS, MSR and QQP datasets for binary semantic similarity classification.

Semantic Text Similarity (STS)

The dataset includes a similarity score in the range $[0, 5]$, which we have to discretize in order to do binary classification, for this we need to choose a threshold from which to consider sentence pairs as similar. Motivated by the SemEval annotation rubric, which states that for value 3.0 sentences are "...roughly equivalent, but some important information differs..." [4], we ultimately decided to use this as our threshold for binarization.

Microsoft Research Paraphrase Corpus (MSR)

Released in 2005, MRPC dataset [1] (here referred as MSR) contains 5800 pairs of sentences which have been extracted from news sources on the web, along with binary human judgement indicating whether each pair captures a paraphrase/semantic equivalence relationship. The data is imbalanced such that 67% all labels are semantically similar. The train and test split is 70-30.

Quora Question pair (QQP) Released by Quora in 2017[8], QQP is a dataset which contains two questions and a boolean attribute determining whether they are similar or not. The dataset is a compilation of question pairs posted in the site Quora that have been classified both by humans and algorithms as duplicates or not, with fairly high accuracy.

The dataset contains more than 400,000 pairs of questions. However, we take a balanced sample of 5,000 pairs for train and 1,000 for test, particularly because of computational constraints and fairness in comparison to other smaller datasets that we have.

This dataset has the advantage of containing a wide array of questions, from people of all ages, all cultures, making it challenging, but also a good training for the language model.

2.1.1 Dataset Comparison

How many overlapping words does each Dataset have?

Intuitively, it can be hypothesized that the more the overlapping words, the easier it should be to classify if the sentences are similar or not. If they're not similar, how could the classification be impacted? Below, we compare the distribution of % of overlapping words in sentence pairs of each dataset and report it for the entire dataset. This information could be useful for later inference with the classification performance.

We see that MSR has the most overlapping words, followed by STS. However, it is worth noting that MSR has almost twice the average words per sentence than the other two datasets.

Table 1: Datasets - overlapping words distribution

Dataset	**% of overlapping words (OW)**	**Average words per Sentence**
STS	39.104	9.905
MSR	50.551	18.927
QQP	30.314	11.110

How indirectly similar are the sentences in the datasets? – Comparsion using Synonymity Metric

Sentences can be semantically similar without any overlapping words, particularly because they're related synonymically.

Inspired by the idea of authors i.e. Theron et. al. in [7] we realized that a good metric for the datasets is the number of overlapping synonyms. The way we measured this was through the NLTK module of Wordnet, this package has been used by a lot of papers to find the synonyms of a certain word. The idea is simple, to calculate the *synonymity_score*, for each word in each sentence obtain the synonyms of that word and if any word of the next sentence is in the synonyms then augment the counter and at the end divide that counter by the sum of the lengths of those sentences.

Lastly, *Norm_OW* i.e. Normalised overlapping words is calculated by normalising overlapping words by the number of sentences. This is supposed to indicate that if you choose any pair of sentences in a dataset, on average it should have that reported value.

STS dataset has the maximum synonymity score which could help in better results of similarity.

Table 2: Datasets - synonymity analysis

Dataset	Norm_OW	synonymity_score
STS	36.5	35.9
MSR	44.1	32.3
QQP	34.1	26.1

2.2 Methodology

We first want to define baseline models that can serve as a starting point for the more complex models to come, in this case we decided to go with TF-IDF and Sentence level embedding using the SBERT[6] model, which will provide vectors that we can then be compared using cosine similarity. We also consider LLM that we will fine-tune as another baseline, compare the performances of fine-tuned LLM with the two baselines, over the same test set.

2.2.1 Baseline Model Selection

Baseline with vector representations

For the task of finding similarity between two texts, it is widely observed in literature to first find a suitable vector representation for the texts and then find how close or far apart they are, in that vector space. We choose two popular methods for the vector representation: Term Frequency - Inverse Document Frequency (TF-IDF) and Sentence-BERT (SBERT). Cosine similarity is used as for the similarity metric between these representations.

We fit the TF-IDF and SBERT on the test data and observe how well they are classified as similar or not similar.

Baseline with LLM model

With the selection of Llama2 as our LLM baseline model, we wanted to investigate that even popular LLM's can perform poorly without fine-tuning towards a specific goal. The training of an LLM is limited to the GPU's and computation power that the team has. Hence, in order to overcome this issue we chose to work with a cloud computing service that could provide the necessary GPU's for the training, and our possible choices were Replicate, OpenAI's API for chat-gpt, and using Llama on AWS BedRock. After a quick analysis we selected Llama on Replicate. Replicate provides a great variety of models to fine-tune and from those models we observed that the best one and most popular was Llama, nevertheless, due to financial constraints, we chose Llama with 7 billion parameters as our baseline model, further experiments should be done on a model with more parameters.

2.2.2 Prompt Engineering

One of the challenges of fine-tuning an LLM is coming up with a prompt that will have higher chances of triggering the right response from the language model. In our case, with Llama 2, we can use internal tokens "[INST]" and "<<SYS>>" to delimit system instructions, this allows to give an instruction to the model, followed by the input sentences and the expected response outside of the instruction block.

In this particular case, we used the following prompt:

> [INST] <<SYS>> For the given two sentences, classify them as semantically similar with 'yes' or 'no' <</SYS>>
>
> Sentence 1:{Sentence1}

Sentence 2:{Sentence2}

Are they semantically similar?:[/INST]

Response:{yes, no}

This prompt has the property of clearly stating what is the desired output, then giving the two input sentences followed by the question that we want an answer to, plus a placeholder for the answer to be completed by the model.

This makes it easy for the model to understand what is the required output, preventing it from generating output that is not 'yes' or 'no'.

2.2.3 Fine-tuning Strategies

The strategy for fine-tuning Llama started with uploading our data to Replicate's cloud, so that the model could read the data. The next step was to make an API call to Replicate setting the fine-tune parameters with 3 epochs, train batch size of 4, learning rate of 1e-4, and a Low-Rank Adaptation of Large Language Models (LoRa) dropout of 0.05 as the most significant parameters for the fine-tuning process. After the model was fine-tuned we relied on the API to make our invoice calls in order to use and evaluate the LLM.

2.2.4 Evaluation metrics

For the evaluating the goodness of the models, we decided to settle on the simple accuracy metric and also F1-score. In this type of task, we are not really interested in differentiating between false positives or false negatives, though it could still be interesting for a outside reader, that's why we decided to still include F1-score as a summary of how well the model did with recall and precision.

3 Results

For the first baseline model, the TF-IDF, we simply vectorize the test data as a whole and then compute cosine similarity between each pair of sentences of which we want to know the similarity score, this one gives a fairly good score that can be visualized in Table 3. Similarly, for the second baseline model, the SBERT embedding, we simply embed every sentence in the test dataset and compute the cosine similarity, the results can be observed in Table 4.

Table 3: Accuracy and F1-score of Baseline TF-IDF.

Dataset	**Accuracy**	**F1-Score**
STS	75.5	74.2
MSR	71.1	80.3
QQP	50.0	66.7

Table 4: Accuracy and F1-score of Baseline SBERT.

Dataset	**Accuracy**	**F1-Score**
STS	76.6	80.0
MSR	68.9	80.7
QQP	50.0	66.7

For the last baseline model, LLM without any fine-tuning, we run into issues because even with our quite specific prompt, it can still decide to generate more text than needed, and start explaining, deviating from the binary answer we need. We had a proportion of wrong outputs of around 32-45%, depending on the dataset. These outputs would yeild unexpected tokens, which are not "yes", or "no", such as [INST]. However, ignoring these, we still obtain a significant accuracy measure for this baseline as can be seen in Table 5. The accuracy is quite low, getting close to coin-flipping, reinforcing the need for fine-tuning of the model.

Table 5: Accuracy of Baseline LLM (Llama 2) without fine-tuning.

Dataset	**Accuracy w/o errors**	**% of errors**
STS	51.5	37.5
MSR	56.1	32.0
QQP	52.3	44.8

For the fine-tuning model, our goal is to surpass the baselines established, to accomplish this, we follow the procedure specified in previous sections for all three datasets, obtaining the accuracies in Table 6.

Table 6: Accuracy and F1-score of fine-tuned Llama 2.

Dataset	**Accuracy**	**F1-Score**
STS	79.0	81.0
MSR	66.0	77.0
QQP	69.6	73.5

As a side note, in the case of fine-tuning with the MSR dataset, we encounter the same issue as with the baseline LLM, the model generates outputs that are not 'yes' or 'no', neither can be easily converted into the result we need. This could probably be attributed to the length of the sentences in this dataset to be almost twice as long as the other datasets, making it more difficult for the LLM to produce text as instructed. Also, another issue was that in the prompt when fine-tuning this model we including a space character separating the string "Response:" from the actual response 'yes' or 'no', this probably could've forced the model to generate an extra space token before giving the answer making it less consistent, but we're not sure about this theory.

To explore this theory, we decided to fine-tune the model again but removing that extra space that we first inadvertently left, and turns out that reduces the error rate to less than 0.2%, only 2 of the test examples produce a wrong output, where before it was of 5.8%.

Finally, to close this section, we decided to show a comparison of the accuracy scores for every model in one single table, for ease of reading. The summary can be seen in Table 7.

Table 7: Accuracy comparison of baseline models and fine-tuned LLM

Dataset	**No fine-tuning**	**TF-IDF**	**SBERT**	**Fine-tuning**
STS	51.5	75.5	76.6	79.0
MSR	56.1	71.1	68.9	66.0
QQP	52.3	50.0	50.0	69.6

4 Conclusions / Discussion

While finding out the topic for this project, a particular example which intrigued us towards the topic of semantic similarity, it was the comparison between the following two sentences. "She is doing great!" and "She's killing it!". While there are no obvious overlapping words, or comparable

synonyms (given the word killing can be interpreted in a very different context), they still meant the same thing. Could fine-tuning an LLM be able to capture it? *Spoiler Alert: It could! :)*

This led us to inspect two particular factors, the % of overlapping words in each data and the synonymity_score.

The STS dataset has the maximum synonymity_score and the second highest overlapping words (10% less than MSR), the baselines and fine-tuned LLMs were able to classify the semantic similarity well on this dataset. While various factors such as the size of a dataset, label balance, complexity of the context in the sentences, play a role, we believe that synonymity_score also plays an important role. This could be further reinforced by seeing that QQP dataset has the least synonymity_score and overlapping words (while the number of training examples were comparable to other two datasets).

Additionally, while we saw that MSR has the highest overlapping words on average, it also had the worst performance. However, it also had twice the average number of words than the other two datasets. While it is not as simple to deduce what other factors played a role in affecting it's performance, capturing the semantic context for much longer sentences increases in difficulty, even for human judges. Sometimes, sentence length may not even play a role but the classification of semantic similarity could be debatable. Would the following two sentences be considered semantically similar? "A man is playing a football." and "A man is manoeuvring a soccer ball with his feet."

During this experiments we explored the variety of datasets and the complexity of classifying two sentences as similar. From classifying two sentences as similar due to the number of overlapping words, all the way to having human annotators assessing whether a pair of sentences are similar or not.

What we've proposed with this experiment is an alternative to focusing on effective representation learning and exploring semantic similarity with them. With a lot of open-source models, fine-tuning an LLM for a difficult task as semantic similarity can be simplified, given that there is an availability of compute power. The fine-tune of the LLM not only helped us to overcome the challenges of the task, but also proved to be generally better than the baseline models. We learnt how little changes in training of a model, for example, just adding a space character in the prompt can severely affect its results and therefore its performance.

5 Division of labor

- **Everyone**
  - Development of final report.
- **Carlos Rivera**
  - Fine-tuning of Llama 2 on MSR dataset.
  - Development of synonymity metric.
  - Evaluation of baseline with Llama, no fine-tuning.
  - Development of training and testing code for LLM fine-tuning.
- **Eduardo Fernandes**
  - Fine-tuning of Llama 2 on QQP dataset.
  - Pre-processing of datasets
- **Priyanshi Pal**
  - Fine-tuning of Llama 2 on STS dataset.
  - Development of overlapping words metric.
  - Development and evaluation of baseline model with TF-IDF and SBERT.

References

[1] "Microsoft research paraphrase corpus," 2005. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=52398.

[2] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds., Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1556–1566. DOI: 10.3115/v1/P15-1150. [Online]. Available: https://aclanthology.org/P15-1150 (visited on 03/19/2024).

[3] Z. Wang, H. Mi, and A. Ittycheriah, "Sentence similarity learning by lexical decomposition and composition," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Y. Matsumoto and R. Prasad, Eds., Osaka, Japan: The COLING 2016 Organizing Committee, Dec. 2016, pp. 1340–1349. [Online]. Available: https://aclanthology.org/C16-1127 (visited on 03/19/2024).

[4] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, S. Bethard, M. Carpuat, M. Apidianaki, S. M. Mohammad, D. Cer, and D. Jurgens, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. DOI: 10.18653/v1/S17-2001. [Online]. Available: https://aclanthology.org/S17-2001 (visited on 04/19/2024).

[5] Y. Shao, "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)*, S. Bethard, M. Carpuat, M. Apidianaki, S. M. Mohammad, D. Cer, and D. Jurgens, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 130–133. DOI: 10.18653/v1/S17-2016. [Online]. Available: https://aclanthology.org/S17-2016 (visited on 03/19/2024).

[6] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1908.10084.

[7] D. Theron, "Contextualizing the limits of model & evaluation dataset curation on semantic similarity classification tasks," *arXiv preprint arXiv:2311.04927*, 2023.

[8] Quora. "First quora dataset release: Question pairs," Quora. (), [Online]. Available: https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs (visited on 04/19/2024).

A Code

The code for our project is hosted in the following GitHub repository: https://github.com/Carlos-Elias-Riv/SemanticSimilarity.

The train and test data, before and after pre-processing, is hosted in the following: OneDrive folder.