Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Large Language Model Augmented Contrastive Sentence Representation Learning

Bachelor's Thesis

Owen Du
owendu@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Zhao Meng
Prof. Dr. Roger Wattenhofer

August 13, 2023

Acknowledgements

I thank Zhao Meng for supervising this thesis, providing me inspiration and guidance for difficulties during my research. I would also like to thank Xiaochen Zheng and Zeyu Yang who also did not hesitate to help me regarding questions about NLP or programming and provided valuable insights to our discussions about the project.

Furthermore, I thank Prof. Dr. Roger Wattenhofer and the DISCO group for providing me the opportunity to write the thesis and allowing me to use the technical infrastructure to conduct my experiments.

Abstract

In this work, we propose an approach which uses large language model augmentation to boost contrastive sentence representation learning. This thesis builds upon the unsupervised SimCSE approach by using paraphrases generated by large language models such as ChatGPT and Vicuna. First, we generate a new sentence dataset and then empirically determine a suitable prompt for paraphrase generation, using metrics that measure sentence similarity and lexical and syntactical structure. After generating paraphrases for each sentence in the dataset, we use them as positive examples during contrastive learning, whereas unsupervised SimCSE only relies on dropout. Our models manage to outperform the unsupervised SimCSE models on semantic textual similarity tasks, by employing our paraphrase training method or only using the new sentence dataset in the unsupervised SimCSE training framework. We also perform an analysis on the STS datasets and compare alignment and uniformity of our models to the SimCSE models.

Contents

Chapter 1

Introduction

In Computer Science, more specifically in the field of Natural Language Processing (NLP), learning sentence embeddings is a fundamental problem which has been studied thoroughly in past papers. Learning deep representations of sentences allows us to perform various downstream tasks such as text classification, sentiment analysis and machine translation, as the learned embeddings contain information about semantic, syntactic and lexical structures of the sentence.

One method to obtain meaningful sentence representations is through training machine learning models by employing self-supervised contrastive learning. Contrastive learning works by pulling embeddings of semantically similar sentences closer together and pushing embeddings of sentences with different meanings further apart. In the SimCSE [1] paper, the authors demonstrate the effectiveness of contrastive learning combined with pre-trained language models such as BERT [2] or RoBERTa [3].

In this work, we build upon the SimCSE model by using data augmentation methods provided by ChatGPT and other large language models (LLM). Due to the self-supervised contrastive objective, data augmentation plays a big part for this type of training. However, this has been a difficult challenge in NLP because of the discrete characteristics of the human language. As large language models have risen in popularity and have become commercially available, users have discovered that they are useful tools for data augmentation methods such as paraphrasing. This thesis showcases the effectiveness of using paraphrases generated by LLMs for the contrastive learning objective.

Chapter 2

Background

2.1 Contrastive Learning

The aim of contrastive learning is to generate effective embeddings of sentences by pulling representations of semantically similar neighbors together and pushing dissimilar neighbors apart. This self-supervised learning method thus requires a set of sentence pairs $\mathcal{D} = \{(x_i, x_i^+)\})_{i=0}^m$ where x_i and x_i^+ are semantically similar sentences and x_i^+ is called the positive example for x_i. As our work is based on the SimCSE [1] framework, we also follow the contrastive objective proposed in SimCLR [4], which uses the cross-entropy loss function with in-batch negatives.

Let h_i and h_i^+ be the embeddings of x_i and x_i^+, obtained by feeding the inputs through a pre-trained embedding model like BERT [2] or RoBERTa [3]. Then the training objective for (x_i, x_i^+) in a batch of N pairs of sentences is:

$$l(x_i, x_i^+) = -\log \frac{\exp(sim(h_i, h_i^+)/\tau)}{\sum_{j=1}^N \exp(sim(h_i, h_j^+)/\tau)}, \tag{2.1}$$

where τ is the temperature hyperparameter and $sim(h_i, h_i^+)$ measures the similarity, in this case the cosine similarity $\frac{h_i^\intercal h_i^+}{||h_i|| \cdot ||h_i^+||}$.

One challenge in contrastive learning is the prerequisite of having semantically related example pairs. Methods such as word deletion, reordering or substitution have been tried, however, just using dropout as we will see in 2.3 has proven to work better than the previously mentioned approaches, as shown by the SimCSE paper.

To characterize the effect of contrastive learning, Wang and Isola [5] propose metrics called alignment and uniformity. Alignment describes the expected distance between embeddings of positive sentence pairs. Conversely, uniformity calculates how uniformly distributed the embeddings are in the latent space. Thus, contrastive learning aims to increase alignment of positive pairs by reducing the distance between positive pairs while seeking to increase the uniformity so embeddings of random sentences are further apart.

2.2 Pre-trained Encoder Models

Pre-trained encoder models are language models which are trained on large amounts of unsupervised text data using self-supervised learning to learn a distribution on the given vocabulary of words. This allows the model to understand context of words and sentences and capture semantic relationships within the language. The following subsections will introduce two pre-trained encoders.

2.2.1 BERT

Bert, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers, is a machine learning model based on transformers [6], which uses the attention mechanism to allow the model to focus on context and enables parallel computation. It utilizes the encoder part of transformers by stacking blocks of attention layers and feed-forward neural networks on top of each other.

Pre-training the BERT model works by employing the two following methods in parallel: Masked Language Modeling (MLM) uses a large corpus of text, which is first turned into input tokens. Then, some percentage of tokens are randomly masked and the model is tasked with predicting the original masked tokens based on the surrounding context. This process enables the model to understand deep bidirectional relationships between words in a sentence. On the other hand, Next Sentence Prediction (NSP) allows the model to learn sentence-level relationships and coherence by giving the model two sentences and letting the model predict whether or not they appear consecutively in the original text. Here, the second sentence is actually the next sentence in the original sequence 50% of the time, else a random sentence from the corpus is taken.

After pre-training, BERT can be fine-tuned to downstream tasks such as text classification, machine translation and sentiment analysis, by using the pre-trained parameters and tuning them with data from the respective task.

2.2.2 RoBERTa

RoBERTa, which stands for **R**obustly **O**ptimized **BERT A**pproach, is an encoder model which shares many similarities in architecture and training strategy with BERT. The key differences are the following: RoBERTa is trained on a larger corpus of text than BERT. While BERT uses MLM and NSP for pre-training, RoBERTa removes the NSP task and instead focuses only on MLM. In addition, the sentences used for pre-training are longer and a larger batch size is used, along with other small tweaks to the training procedure. Furthermore, RoBERTa also uses the transformer architecture, however, with more parameters, which allows it to capture more complex language representations.

2.3 SimCSE

SimCSE is a simple contrastive learning framework for sentence embeddings. It takes the pre-trained BERT and RoBERTa models and then continues to train the parameters using contrastive learning, making use of the contrastive objective presented in section 2.1. The authors propose both an unsupervised and a supervised method and evaluate the model on standard semantic textual similarity (STS) [7, 8, 9, 10, 11, 12] tasks.

2.3.1 Unsupervised SimCSE

Unsupervised SimCSE utilizes a simple key idea: given a set of sentences $\{x_i\}_{i=1}^{m}$, take $x_i^+ = x_i$ in the contrastive loss formula given in 2.1. In order for the contrastive objective to work, the framework uses independently sampled dropout masks for the two inputs. These are the same type of masks described in the original transformers paper, which are placed on the fully-connected layers and attention vectors. During training, each input is fed to the encoder with the respective dropout mask. Thus, the model generates two different embeddings, which are then used in the contrastive loss function. In the original implementation, unsupervised SimCSE is trained with one million randomly sampled sentences from English Wikipedia.

By using dropout as a form of data augmentation, the model achieves better scores for various STS tasks than methods such as word deletion, word replacement or using the next sentence as a positive example. The default value of $p = 0.1$ is used for dropout as determined by the authors of SimCSE through experiments.

Our work builds upon unsupervised SimCSE by taking $x_i^+ = p_{ij}$ where p_{ij} represents the j-th paraphrase of x_i with $j \in \{1, .., 5\}$. The exact training setup is showcased in section 4.1.

2.3.2 Supervised SimCSE

Supervised SimCSE leverages smaller, supervised datasets to perform contrastive learning. Natural language inference (NLI) datasets [13, 14] contain supervised data about the relationship between two sentences, namely entailment, neutral or contradiction. For each given sentence, human subjects are tasked to write another sentence which is definitely true (entailment), one which might be true (neutral) and one which is absolutely false (contradiction).

Due to the fact that for every sentence x_i there is a positive example x_i^+ and a negative example x_i^-, the contrastive objective can be extended to incorporate the negative example as follows:

$$l(x_i, x_i^+, x_i^-) = -\log \frac{\exp(sim(h_i, h_i^+)/\tau)}{\sum_{j=1}^{N} \left(\exp(sim(h_i, h_j^+)/\tau) + exp(sim(h_i, h_j^-)/\tau) \right)}, \quad (2.2)$$

where h_i represents the respective embedding after feeding it into the encoder during training. This new objective in combination with the supervised training data leads to a significant performance increase compared to unsupervised SimCSE.

2.4 ChatGPT

On November 30, 2022 OpenAI announced ChatGPT[1], an artifical intelligence model trained to generate a detailed response given an instruction in a prompt. The model architecture is based on GPT-3.5 from OpenAI's series of GPT (Generative Pre-Training) models [15], however the exact structure of the original model is undisclosed.

ChatGPT is trained using Reinforcement Learning from Human Feedback (RLHF) [16] using human AI trainers. They provide conversations in which they imitate both the user and the AI assistant. Furthermore, random model-generated examples are selected and additional responses are sampled. Then, AI trainers rank the generated responses by quality to create a reward model. Finally, Proximal Policy Optimization (PPO) [17] is used to fine-tune ChatGPT with the help of the gathered supervised data.[2]

ChatGPT gained massive popularity among the general public after its release due to its question answering abilities and versatility. It allows users to customize and guide a conversation towards their preferred length, style, level of detail and language in virtually any domain.

2.5 Vicuna

Vicuna-13B[3] is an open-source AI chatbot, published on March 30, 2023. The model is trained by fine-tuning LLaMA [18] using data from ShareGPT[4], where users can share conversations they had with ChatGPT. The Vicuna team claims over 90% quality of its model compared to ChatGPT by letting GPT-4 judge the models' responses in terms of helpfulness, relevance, accuracy and detail. One

[1]https://openai.com/blog/chatgpt
[2]https://openai.com/blog/chatgpt
[3]https://lmsys.org/blog/2023-03-30-vicuna/
[4]https://sharegpt.com/

special aspect of Vicuna is the low training cost of only $300, using 8 A100 GPUs for one day.

Vicuna is of particular interest for us, as it is open source and thus can be used in our own cluster for paraphrase generation instead of paying for ChatGPT API. Additionally, the proclaimed 90% quality compared to ChatGPT is promising. Later in section 4.2.1 we showcase the quality of the paraphrases generated by ChatGPT compared to the examples generated by Vicuna.

Chapter 3

Data Generation

This chapter showcases the methods we use to generate the data needed to conduct the experiment. This includes the training dataset with the original sentences and the paraphrase dataset, containing 5 unique paraphrases for each original sentence. Additionally, we examine the difference in quality of the paraphrases generated by ChatGPT and Vicuna.

3.1 Training Dataset

The dataset for unsupervised SimCSE [1] contains 10^6 randomly sampled sentences from English Wikipedia according to their published paper. After inspecting the first few hundred sentences of the dataset, it can be concluded that the authors sampled random Wikipedia articles and then split the text into sentences using a sentence tokenization tool. As the raw article text contains titles for sections and tokenization tools are not fully reliable, the sentence dataset contains numerous examples which are short phrases or even single words. These instances pose a challenge for paraphrases generated by LLMs, as the answer is either that is not possible to paraphrase the input or the response contains 5 paraphrases with fictitious information generated by the LLM.

Consequently, we look for a way to generate a new sentence dataset with 10^6 sentences, which are more suitable for paraphrases. We decide to take sentences from English Wikipedia like in the unsupervised SimCSE experiment, using the Wikipedia dataset provided by Hugging Face[1]. This dataset contains over 6 million cleaned articles where markdown syntax and unwanted sections like references are removed. To parse the articles into sentences and words, we employ the `sent_tokenize` and `word_tokenize` tools from NLTK[2]. However, as mentioned, these tools are not fully reliable and struggle with the line breaks contained in the raw article data, which is stored in a single string. Thus, we explore 3 ways to sample 10^6 sentences from the Hugging Face Wikipedia dataset.

[1]https://huggingface.co/datasets/wikipedia
[2]https://www.nltk.org/

One method is to sample a random article first and remove all line breaks, characterized in Python strings as `\n`. Then, we use `sent_tokenize` and get an array of strings which should contain a sentence each. If the result of the sentence tokenization gives less than 10 sentences, we resample the article, as very short articles tend to contain few suitable sentences for paraphrasing. Finally, we pick a random sentence from the first 60% of the article, because often times the articles in the dataset still contain references and external links at the end, although the articles are cleaned. Additionally, we use `word_tokenize` to split the chosen sentence into words and finally take the sentence if it contains between 10 to 50 words. This measure is taken to filter out examples, which are only a short phrase or several sentences concatenated into one string. We repeat this procedure 10^6 times to generate the whole sentence dataset.

The second method differs in the order of operations. First, a random sentence is sampled from a random article without filtering out line breaks. Then, only sentences which do not contain line breaks are chosen for the dataset, with the same conditions on word length as in the first method.

The third approach tries to generate sentences similarly to the unsupervised SimCSE dataset. We sample a random article, this time containing more than 20 sentences, without removing line breaks from the article. Afterwards, we take the article's first 4 sentences which do not contain line breaks and consist of between 10 and 50 words.

We generate 3 datasets for each of the methods, resulting in 9 total sentence datasets. To evaluate the quality of the sampled sentences we train the unsupervised SimCSE model with BERT-base [2] as pretrained model with each of the 9 datasets and compare the results to the unsupervised SimCSE-BERT-base model from the SimCSE paper. We use the same evaluation setup as SimCSE which conducts 7 semantic texual similarity (STS) tasks, namely STS 2012-2016 [7, 8, 9, 10, 11], STS Benchmark (STS-B) [12], which contains a collection of sentences from previous STS datasets and lastly SICK-relatedness (SICK-R) [19]. Additionally, we use the same setting as in the SimCSE paper for comparing the results, which means no additional regressor for STS-B and SICK-R datasets, Spearman's correlation as evaluation score and aggregating the results in the "all" setting. This means for each STS task, which contains several subtasks, we concatenate all examples and take the overall Spearman's correlation instead of reporting the average correlation of the subtasks. We also use the same training setup as the SimCSE code, with 1 training epoch, learning rate of $3e-5$ and batch size 64.

In table 3.1 the results of the models trained with the generated datasets can be seen, compared to the scores achieved by unsupervised SimCSE with BERT-base as pretrained encoder. Surprisingly, the first dataset generated with method 1 already scores a higher average than the SimCSE-BERT$_{base}$. In general, we can see that method 1 performs better on average than the other procedures. Thus,

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised SimCSE-BERT$_{base}$	68.40	**82.41**	74.38	80.91	**78.56**	76.85	**72.23**	76.25
Method 1-1	68.78	81.59	**75.19**	**81.44**	77.41	**78.50**	72.22	**76.45**
Method 1-2	67.45	79.38	72.55	81.34	77.89	76.87	71.25	75.25
Method 1-3	67.73	78.35	69.51	79.18	77.70	75.77	69.83	74.01
Method 2-1	64.93	78.56	69.67	79.26	75.57	74.31	69.11	73.06
Method 2-2	64.84	78.01	69.67	77.38	77.47	75.89	71.17	73.49
Method 2-3	**68.80**	80.22	71.40	78.17	76.42	76.11	69.06	74.31
Method 3-1	66.88	80.48	71.03	80.47	76.87	76.23	70.38	74.62
Method 3-2	63.05	74.20	67.24	76.63	76.65	75.23	69.37	71.77
Method 3-3	65.69	77.32	70.83	79.59	76.14	75.65	69.30	73.50

Table 3.1: Sentence embedding performance (Spearman's correlation) of unsupervised SimCSE models trained with our generated datasets compared to the model from the SimCSE paper. Method *x*-*y* denotes the *y*-th dataset generated using method *x*. The highest score among all models for each evaluation task is highlighted.

we decide to take the dataset from Method 1-1 as the final dataset and use its sentences to generate paraphrases.

3.2 Prompt

First, we need to define the word paraphrase in order to understand how we can create a suitable prompt for our needs. In an online dictionary[3], paraphrase is defined as: "a restatement of a text or passage giving the meaning in another form, as for clearness; rewording". To ensure that two sentences give the same meaning in another form in our case, we can look at the contrastive objective in 2.1 and observe the term $sim(h_i, h_i^+)$. This measures the semantic similarity between 2 embeddings, which we should keep high for the generated paraphrases. Another aspect of the definition concerns rewording. We can measure this property by looking at the lexical and syntactical structure of the sentences.

In order to generate 5 paraphrases for each sentence x_i in the sentence dataset, we create the following prompts for ChatGPT and Vicuna:

1. Generate 5 paraphrases of the following: x_i
2. Generate 5 new sentences, which are semantically similar but lexically and syntactically divergent from the following: x_i

[3]https://www.dictionary.com/browse/paraphrase

3. I want you to act as a paraphrasing tool. I will provide you a sentence and your task is to generate 5 paraphrases. These will act as augmented data that I will use to train a sentence embedding model evaluated on a semantic text similarity task. The sentence is: x_i
4. On a scale of 1 to 5, where 1 is the most semantically similar but least lexically divergent and 5 is the least semantically similar but most lexically divergent, generate a paraphrase for each scale of the following: x_i
5. Generate 5 paraphrases, where the first paraphrase has the highest semantic similarity but the lowest lexical divergence and the last paraphrase has the lowest semantic similarity and the highest lexical divergence, of the following: x_i

The first prompt is simple and general and is used to test the LLMs understanding of the word paraphrase. The second prompt focuses more on the definition of paraphrasing by separating the orthogonal metrics of semantic similarity and lexical and syntactic structure of the generated paraphrase. Prompt 3 is inspired by Awesome ChatGPT Prompts[4] and describes the role the LLM should play and the use of the generated paraphrases. The last 2 prompts aim to generate paraphrases of different quality in terms of semantic similarity and rewording.

In order to test the quality of the prompts, we sample the first 100 sentences of the sentence dataset and generate 5 paraphrases for each sentence with ChatGPT. We repeat this for every prompt and obtain 5 paraphrase datasets. To empirically evaluate the quality of the prompts, we measure the metrics semantic, lexical and syntactical similarity using the methods described in the following subsections.

3.2.1 Parascore

Parascore [20] is a metric for evaluating the quality of paraphrases. It calculates semantic similarity using the BERT-base model and uses normalized edit distance to measure lexical and syntactical divergence. The final score is a trade-off between the two values and performs better than previously proposed paraphrase evaluation metrics. Ultimately, we decide against using parascore as BERT-base is a weaker model than SimCSE, which we want to improve on. Thus, we focus on a simpler metric described in the next section.

3.2.2 Semantic Similarity and BLEU

To evaluate the quality of our generated paraphrases we use a simpler method by first calculating the semantic similarity *sim*. This is done using the cosine similarity of the original sentence embeddings and paraphrase embeddings generated

[4]https://github.com/f/awesome-chatgpt-prompts

by the supervised RoBERTa$_{large}$ SimCSE model. Then, we normalize the values so they lie between 0 and 1. To measure the differences in lexical and syntactical structure of the original sentence compared to the paraphrases we use the BLEU score [21]. It counts how many unigrams, bigrams, trigrams and four-grams occur in the hypothesis (original sentence), as well as in the reference (paraphrases). Thus, for our generated paraphrases we aim to reach a high semantic similarity score while keeping the BLEU score low. This leads us to the following metric:

$$S(x_i, p_i) = \frac{sim + 1 - BLEU}{2}, \tag{3.1}$$

where x_i is the original sentence and p_i is the corresponding set of 5 paraphrases. The score ranges from 0 to 1 and the higher the value the better paraphrase quality we get. We understand that this metric is not fully reliable, as sentences which are not related to each other at all would have a high score, because the BLEU score would be close to 0. However, for our purposes we ensure that all paraphrases have high enough semantic similarity in addition to comparing the total scores.

The following table shows the results of averaging the semantic similarity, BLEU and total scores across 100 sentences and their respective paraphrases:

	Sim	**BLEU**	**Total**
Prompt 1	0.947	0.403	0.772
Prompt 2	0.926	0.353	0.786
Prompt 3	0.947	0.452	0.746
Prompt 4	0.911	0.331	0.790
Prompt 5	0.906	0.313	0.797

Table 3.2: Average scores of paraphrases generated by each prompt. For each of the first 100 sentences of the sentence dataset, we generate 5 paraphrases and then calculate similarity using cosine similarity and RoBERTa$_{large}$ supervised SimCSE. Note that for prompt 4 and 5 only 10 sentences were paraphrased to test the effectiveness of the prompt. BLEU is calculated with `sentence_bleu` from NLTK. Total score is calculated with equation 3.1.

We can observe that prompt 1 and 3 perform similarly, while prompt 2 has a lower BLEU score and only slightly lower similarity score, resulting in a higher total score. For the last 2 prompts it has to be noted that only 10 sentences were paraphrased to test whether the generated paraphrases actually display a gradient in terms of similarity and BLEU values. In this case, we do not observe any gradual change in similarity values across all generated sets of paraphrases. Additionally, the length of these 2 prompts lead to longer response times and higher token usage, making them less feasable. Due to these results, we decide to take the second prompt to generate all paraphrases for our sentence dataset.

3.3 Paraphrase Generation

3.3.1 Paraphrase Generation with ChatGPT

On March 1, 2023 OpenAI announced the ChatGPT API, which gives developers access to query ChatGPT on a large scale, instead of through the web browser only. For every API request, the prompt, represented in the form of unstructured text, is turned into tokens. These tokens can span from one to several characters inside a word and the pricing for the API is set to $0.002 per 1k tokens.

In our case, we need to prompt the model 1 million times and parse each response into 5 paraphrases. We first generate paraphrases for 150k sentences and conduct the experiment to evaluate the effectiveness of the approach before scaling it to 1 million sentences. Each API response takes around 10 seconds and uses 300 tokens with deviations in both directions as the length of the input sentence varies. Thus, we utilize the `multiprocessing` library from Python to speed up the paraphrase generation process. We specifically use the `Pool` class, where each worker is assigned a distinct interval of sentences. Afterwards, each process sequentially prompts the model to generate paraphrases for the assigned sentences. The results are then parsed and in the end we concatenate the results into one JSON file.

One problem encountered while using the API were various errors (timeout, internal server error) when sending requests. To combat this, we set timeouts whenever errors occurred, which remedied the issue. Furthermore, despite using the newly generated sentence dataset, which should only contain full sentences, there were some examples where sentence parsing failed and ChatGPT could not generate 5 different paraphrases. In this case, we skip these sentences and add the index of the sentence to a list. After generating the rest of the paraphrases, we go through the indices and replace the bad sentences by sampling new ones using the first method described in section 3.1.

With this procedure, we are able to generate 5 paraphrases for each of the 1 million sentences in the Wikipedia dataset, costing around $500 in total.

3.3.2 Paraphrase Generation with Vicuna

To set up Vicuna on our own cluster to work like the ChatGPT API, we configure a controller and register multiple model workers. Then, we create an API server which works the same as the one used by OpenAI. Thus, we can reuse the code for paraphrase generation for ChatGPT.

We first generate paraphrases for 150k sentences and conduct the experiment to test the performance of Vicuna-generated paraphrases against ChatGPT-generated paraphrases. We find that prompt 2 from section 3.2 does not work well with Vicuna, as it generates new information not contained in the original sentence. Therefore, we use prompt 1 and find following evaluation results for 100 sentences and their respective paraphrases:

	Sim	**BLEU**	**Total**
Prompt 1 with Vicuna	0.951	0.483	0.734

Table 3.3: Average scores of paraphrases generated by prompt 1 from 3.2. We use the same 100 sentences used in 3.2.2 for evaluating.

The results show that the paraphrases generated by Vicuna are more similar to the original sentence in terms of semantic similarity, however, in lexical and syntactical structure as well. In section 4.2.1 we analyze whether these paraphrases perform better or worse than examples from ChatGPT.

Chapter 4

Experiment

4.1 Setup

With the new sentence dataset and the generated paraphrase dataset we can conduct our experiment which uses paraphrases as positive examples in the contrastive loss function. The training of the model works as follows: Let $j = e \bmod 5$. During epoch e, for each input sentence x_i, we take the j-th paraphrase p_{ij} and feed both of them through the encoder, receiving h_i and k_{ij}. These embeddings are used in the contrastive loss objective, which pulls the representations of x_i and p_{ij} (positive pair) closer together and pushes all other examples within the batch (negative examples) further apart. Thus, the new training objective is:

$$l(x_i, p_{ij}) = -\log \frac{\exp(sim(h_i, k_{ij})/\tau)}{\sum_{l=1}^{N} \exp(sim(h_i, h_l^+)/\tau)}. \tag{4.1}$$

Here $sim(\cdot,\cdot)$ represents the cosine similarity, τ is the temperature hyperparameter and h_l^+ is the embedding of the l-th sentence of the batch generated by the encoder, where l iterates over the indices within the current mini-batch. Due to the chosen value of j we iterate through every paraphrase exactly once per 5 epochs, thus, for 10 epochs we would use each paraphrase twice.

We implement this training setup by modifying the code from SimCSE's [1] codebase. In their implementation of the model training, which is used to train both supervised and unsupervised SimCSE, they use the default value of 0.1 for attention vector and fully connected layer dropout. This means that even for training supervised SimCSE on NLI datasets, a dropout mask is used. We modify the code so we can specify the dropout value ourselves and experiment with using different dropout values during training.

Next, we implement paraphrase mode during training. The original code concatenates 2 identical instances of the original sentence dataset together before tokenizing each sentence. Instead, we concatenate the original sentence dataset with paraphrase dataset P_i, where P_i consists of the i-th paraphrase for each sentence in the original sentence dataset. Thus, we obtain 5 different datasets, which we also pass to the tokenizer.

After tokenization, we turn all the concatenated datasets into `Dataloader` instances from Pytorch, using the `get_train_dataloader` function from Hugging Face. Here, we initialize the `Dataloader` with a random sampler, which ensures that for every epoch the dataset is shuffled into a random order. The result is 5 `Dataloaders`, each containing one set of original sentences and one set of paraphrases. Finally, for epoch e, we use the ($e \mod 5$)-th dataset for training, such that every dataset is used exactly once every 5 epochs.

We evaluate our models on 7 STS tasks, namely STS 2012-2016 [7, 8, 9, 10, 11], STS-Benchmark (STS-B) [12] and SICK-relatedness (SICK-R) [19]. We use the same evaluation setup as SimCSE, where the score for each task represents Spearman's correlation. Additionally, we do not use an additional regressor for STS-B and SICK-R datasets and aggregate results in the "all" setting, explained in section 3.1.

4.2 Results

4.2.1 Experiment with 150k Sentences

First, we showcase the results from training with paraphrases for 150k sentences. In the following, we compare the results from using paraphrases generated by ChatGPT with paraphrases generated by Vicuna. We use different pre-trained encoder models with varying hyperparameters and present the models that achieve the highest average score for each encoder. The scores achieved by the unsupervised SimCSE counterparts are also shown in table 4.1 for comparison. For all of our models in the table, we use a learning rate of $3e-5$ and batch size of 64. The number of epochs and dropout used during training is shown in table 4.2.

From these results, using 150k sentences with 5 paraphrases each, we see that our model using RoBERTa [3] as pre-trained encoders performs better than the SimCSE models when averaging the evaluation results. However, when using BERT [2] as pre-trained model, our model performs worse than SimCSE-BERT$_{base}$ on average. One peculiarity when looking at the evaluation scores of our RoBERTa models is the difference in the SICK-R score. Our models using RoBERTa as pre-trained model perform around 6 points better than the SimCSE counterparts. We give our hypothesis for this phenomenon in section 4.3.

We also observe the difference in performance when using paraphrases generated by ChatGPT versus Vicuna. For BERT, the model trained with Vicuna-generated paraphrases even outperforms the corresponding SimCSE model in some cases, however the average score is still lower than the score achieved by the model using ChatGPT's paraphrases. For RoBERTa, the difference in the average score is even more striking, as the model trained with Vicuna's para-

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised SimCSE reference models								
SimCSE-BERT$_{base}$	68.40	**82.41**	**74.38**	**80.91**	**78.56**	76.85	**72.23**	**76.25**
SimCSE-RoBERTa$_{base}$	70.16	**81.77**	**73.24**	**81.36**	80.65	**80.22**	68.56	76.57
SimCSE-RoBERTa$_{large}$	**72.86**	**83.99**	75.62	**84.77**	81.80	**81.98**	71.26	78.90
ChatGPT-generated paraphrases								
bert-base-uncased	69.5	80.41	72.18	78.76	78.88	76.41	71.77	75.42
roberta-base	**71.89**	81.44	72.43	80.01	80.32	79.66	**74.11**	**77.12**
roberta-large	72.10	82.91	**76.06**	83.70	**82.78**	81.52	**77.64**	**79.53**
Vicuna-generated paraphrases								
bert-base-uncased	**71.12**	78.57	72.78	79.02	**78.97**	**76.92**	70.0	75.34
roberta-base	70.41	80.18	71.63	79.63	78.6	77.94	73.18	75.94

Table 4.1: Sentence embedding performance of models trained with our experiment setup using 150k sentences. Unsupervised reference models refer to the best unsupervised models presented in the SimCSE paper. For the ChatGPT and Vicuna sections, the model denotes the pre-trained encoder used for contrastive learning. The highest number per column among models with the same pre-trained encoder is highlighted.

Model	Paraphrases	Epochs	Dropout
bert-base-uncased	ChatGPT	10	0.0
roberta-base	ChatGPT	5	0.1
roberta-large	ChatGPT	5	0.1
bert-base-uncased	Vicuna	10	0.0
roberta-base	Vicuna	20	0.0

Table 4.2: Number of epochs and dropout value used during training of models in 4.1. Model refers to the pre-trained encoder used in the experiment. Paraphrases denotes which LLM generated the paraphrases used in the respective experiment. The dropout value is used for both fully connected layers and attention vectors.

phrases is worse in every evaluation category than its ChatGPT counterpart. We attribute the difference in performance to the quality of the generated paraphrases. When comparing the scores of prompt 2 in table 3.2 with the values in table 3.3, we see that paraphrases generated by Vicuna have a higher similarity score while having higher BLEU [21] score as well. Thus, the generated paraphrases are more similar in syntactic and lexical structure to the original sentence and the model sees a smaller variety of words and sentence structures during training. For the RoBERTa-base model there is a larger difference in performance, as our paraphrase method seems to perform better with the RoBERTa architecture than BERT, hence the quality of the paraphrases is more crucial.

Due to the above findings, we decide to scale the experiment to 1 million sentences for ChatGPT. We extend our paraphrase dataset and generate 5 paraphrases for each of the remaining 850k sentences in the sentence dataset using ChatGPT.

4.2.2 Experiment with 1 Million Sentences

After generating the rest of the paraphrases, we have the original sentence dataset with 1 million sentences and a paraphrase dataset with 5 million sentences in total. With the same training setup as before, we obtain the results showcased in table 4.3. We again use a learning rate of $3e-5$ and batch size of 64. Other parameters such as epochs and dropout value are displayed in table 4.4.

Model	**STS12**	**STS13**	**STS14**	**STS15**	**STS16**	**STS-B**	**SICK-R**	**Avg.**
				Unsupervised SimCSE reference models				
SimCSE-BERT$_{base}$	68.40	**82.41**	**74.38**	**80.91**	**78.56**	**76.85**	**72.23**	**76.25**
SimCSE-RoBERTa$_{base}$	**70.16**	**81.77**	**73.24**	**81.36**	**80.65**	**80.22**	68.56	**76.57**
SimCSE-RoBERTa$_{large}$	**72.86**	**83.99**	75.62	**84.77**	81.80	**81.98**	71.26	78.90
				Paraphrase method using 1 million paraphrases				
bert-base-uncased	**69.54**	78.96	70.89	77.39	77.73	75.41	71.68	74.51
roberta-base	69.51	80.36	71.99	79.34	80.13	79.67	**74.87**	76.55
roberta-large	72.46	83.78	**76.66**	81.71	**83.21**	80.70	**75.85**	**79.20**

Table 4.3: Sentence embedding performance of models trained with our experiment setup using 1 million sentences. Unsupervised reference models refer to the best unsupervised models presented in the SimCSE paper. For the paraphrase method section, the model refers to the pre-trained encoder used for contrastive learning. The highest number per column among models with the same pre-trained encoder is highlighted.

Model	**Epochs**	**Dropout**
bert-base-uncased	5	0.0
roberta-base	5	0.1
roberta-large	15	0.0

Table 4.4: Number of epochs and dropout value used during training of models in 4.3. Model refers to the pre-trained encoder used in the experiment. Paraphrases denotes which LLM generated the paraphrases used in the respective experiment. The dropout value is used for both fully connected layers and attention vectors.

From table 4.3 we see that when using 1 million sentences, only our model using RoBERTa-large performs better than its SimCSE counterpart. Comparing the average scores, we observe that all 3 models trained on 1 million sentences perform worse than the models presented in table 4.1 using 150k sentences and

ChatGPT-generated paraphrases. Our hypothesis for this phenomenon is overfitting and insufficient number of parameters of the models.

The SimCSE training framework uses the development set of STS-B [12] as validation set during training. For our models trained on 1 million sentences, we observe that the evaluation value reaches its maximum early during training. Thus, we speculate, that our models overfit easily and further data does not contribute to performance as the SimCSE framework chooses the best checkpoint for final evaluation on the test sets.

Additionally, we hypothesize that our models using bert-base-uncased and roberta-base as pre-trained encoders perform worse than its SimCSE counterparts because the models have too few parameters. As the 3 encoders used in the experiment all have different architectures, we can rank them by their number of parameters. We see that bert-base-uncased has the lowest amount and roberta-large uses the most. From the average scores in 4.3 we can also observe that our model using bert-base-uncased performs notably worse than the corresponding SimCSE model, while roberta-base achieves a similar score and roberta-large even outperforms SimCSE.

Finally, we again notice the increase of evaluation scores in the SICK-R category for our RoBERTa models compared to SimCSE. Looking at our roberta-large model, the evaluation scores in all other categories are similar to its SimCSE counterpart. We analyze the evaluation datasets in section 4.3 to investigate this jump in performance.

4.3 Analysis

4.3.1 Evaluation Datasets

In order to understand the results of the experiments better, we take a closer look at the evaluation datasets, namely STS12-16, STS-Benchmark and SICK-relatedness. These consist of sentence pairs, each labeled with a similarity or relatedness score.

We analyze each evaluation dataset by calculating the average amount of words per sentence using NLTK's `word_tokenize`. As each evaluation dataset consists of several subsections we also provide an analysis for each category inside the datasets. Table 4.6 displays the results of our analysis. We observe that many STS tasks contain examples, which are not full sentences but rather short phrases or expressions. The length of the examples also varies between categories within the STS tasks.

To contrast these values, we additionally calculate the average number of words per sentence of our original sentence dataset and compare it with the original unsupervised SimCSE Wikipedia dataset. We show the results in table

4.5, where we see that our sentence dataset contains longer sentences compared to the SimCSE dataset and sentences from the evaluation datasets. However, our inital hypothesis that our model performs well on SICK-R due to bias on the length of the sentences does not appear to be true.

Nevertheless, we see that all SICK-R datasets contain full sentences, while STS12-16 and STS-B contain examples which are phrases or expressions. As our sentence datasets and paraphrase datasets only consist of full sentences as opposed to the SimCSE sentence dataset, our models might perform better when receiving full sentences as input.

Dataset	**Amount**
Unsupervised SimCSE sentence dataset	22.6
Our original sentence dataset	25.9

Table 4.5: Average number of words per sentence per dataset.

Name	Description	Amount
	STS12	
MSRpar	Full sentences from news articles	20.7
MSRvid	Short descriptions of videos	7.6
SMTeuroparl	Sentences from speeches and debates in the European parliament	12.3
OnWN	Expressions and phrases from lexical resources	8.8
SMTnews	Sentences and phrases from news articles	13.5
	STS13	
FNWN	Full sentences from lexical resources	23.3
headlines	News headlines	7.7
OnWN	Expressions and phrases from lexical resources	8.2
	STS14	
deft-forum	Short sentences and phrases from a forum	10.1
deft-news	Full sentences from news articles	17.2
headlines	News headlines	7.8
images	Short descriptions of images	10.2
OnWN	Expressions and phrases from lexical resources	8.8
tweet-news	News titles and tweet comments	12.1
	STS15	
answers-forums	Full sentences from answers posted in a forum	17.6
answers-students	Short sentences and expressions of explanations	10.8
belief	Spoken expressions and phrases	14.9
headlines	News headlines	7.9
images	Short descriptions of images	10.6
	STS16	
answer-answer	Short sentences from answers	11.1
headlines	News headlines	8.2
plagiarism	Full sentences from lexical resources	15.2
postediting	Full sentences from lexical resources	23.5
question-question	Question sentences	11.7
	STS-B	
sts-dev	Collection of examples from STS12-16	13.3
sts-test	Collection of examples from STS12-16	14.3
sts-train	Collection of examples from STS12-16	11.2
	SICK-R	
test_annotated	Descriptions (Full sentences)	9.7
train	Descriptions (Full sentences)	9.7
trial	Descriptions (Full sentences)	9.9

Table 4.6: Analysis of each evaluation dataset and its subsections. Amount refers to the average number of words per sentence of the sentences contained in the datasets.

To investigate the difference in performance between the STS tasks and SICK-R further, we look at the distribution of similarity/relatedness score in the datasets. Specifically, we look at STS-B, as it contains a collection of examples from STS12-16 and SICK-R. For STS-B, the similarity score ranges from 0 to 5 with 5 being labeled to sentence pairs, which are semantically the most similar. For SICK-R, the relatedness score ranges from 1 to 5, where 5 means the sentences are highly related to each other. The following figure shows the distribution chart of the respective datasets:

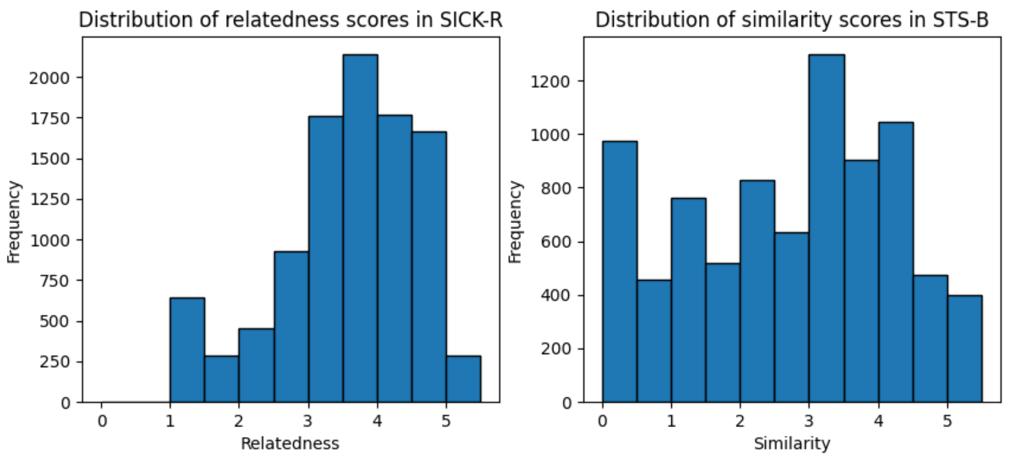

Figure 4.1: Distribution of relatedness/similarity scores of SICK-R and STS-B. Each sentence pair in both datasets is labeled with a score.

We see that SICK-R has more examples with higher scores assigned to them, while STS-B contains many sentence pairs, which have low similarity score. We can verify this by calculating the mean score of both datasets and normalizing the values to the interval $[0, 1]$. We receive the following values:

Dataset	**Score**
SICK-R	0.63
STS-B	0.52

Table 4.7: Mean relatedness/similarity score of SICK-R and STS-B normalized to $[0, 1]$.

From the results, we see that SICK-R indeed has a higher mean relatedness score than STS-B.

4.3.2 Alignment and Uniformity

In section 2.1 we mention the terms alignment and uniformity that measure how well representations are distributed in the latent space. To reiterate, alignment describes the average distance between positive pairs of inputs and uniformity represents the average distance between any pairs of inputs. To characterize these metrics, we use the definitions proposed by Wang and Isola [5]:

$$l_{align} = \mathop{\mathbb{E}}_{(x,x^+)\sim p_{pos}} ||f(x) - f(x^+)||^2 \tag{4.2}$$

$$l_{uniform} = \log \mathop{\mathbb{E}}_{\substack{i.i.d.\\ x,y\sim p_{data}}} e^{-2||f(x)-f(y)||^2} \tag{4.3}$$

Here, we take sentence pairs from STS-Benchmark with a similarity score higher than 4 as p_{pos} and all sentences from STS-Benchmark as p_{data} like in the SimCSE paper. For both metrics, lower numbers mean better performance.

We calculate the alignment and uniformity scores for BERT-base, RoBERTa-base, RoBERTa-large, our best performing models from 4.1 using ChatGPT-generated paraphrases and their SimCSE counterparts. The results can be seen in table 4.8.

We can see that our models greatly improve uniformity of the pre-trained encoders, while the base models retain a very low alignment score. Especially for the RoBERTa base models, we see that both alignment and uniformity are extremely low, meaning most embeddings generated by these encoders occupy a very small space. Compared to the SimCSE models, our models have a better alignment score but worse uniformity. One possible explanation for this occurence is that we perform data augmentation only on positive examples but still take the in-batch sentences as negative examples in the contrastive objective 4.1.

Moreover, we see that the alignment and uniformity metrics are not sole indicators of the model's performance. SimCSE-RoBERTa$_{base}$ and SimCSE-RoBERTa$_{large}$ have very similar values for both alignment and uniformity, with the model using BERT having marginally better scores. However, the average of the RoBERTa based model is still higher. Specifically, we acknowledge that the alignment score for example does not take into account that sentence pairs with a higher similarity score should have a higher similarity value than less similar sentence pairs within p_{pos}.

We visualize the alignment and uniformity scores of the models in figure 4.2. Even though we know that lower scores are better for both metrics, it is not clear how exactly they correlate with performance and what the trade-off between alignment and uniformity is. We can see that our models generally have lower alignment but higher uniformity than SimCSE models, but the difference in performance is not large.

Model	**Alignment**	**Uniformity**
Base pre-trained encoders		
bert-base-uncased	0.0579	-0.2399
roberta-base	0.0008	-0.0079
roberta-large	0.0034	-0.1479
Unsupervised SimCSE models		
SimCSE-BERT$_{base}$	0.2135	-2.6608
SimCSE-RoBERTa$_{base}$	0.2136	-2.6478
SimCSE-RoBERTa$_{large}$	0.2309	-3.1722
Our models using paraphrases		
bert-base-uncased	0.1523	-2.2253
roberta-base	0.1241	-2.0072
roberta-large	0.1358	-2.5677

Table 4.8: Alignment and uniformity scores of the given models calculated using 4.2 and 4.3. The lower the numbers the better.

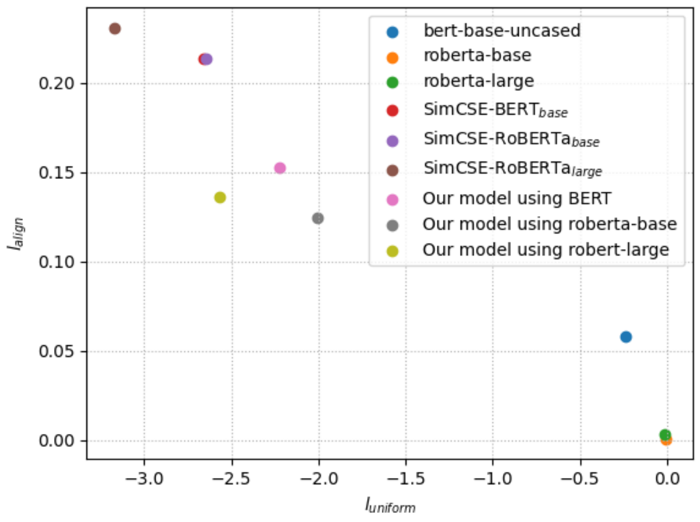

Figure 4.2: Plot visualizing alignment and uniformity for models in table 4.8. For both metrics, lower numbers are better.

Chapter 5

Conclusion

In this work, we utilize commercially available large language models such as ChatGPT and Vicuna for data augmentation of sentences and use the generated data to improve the unsupervised SimCSE approach. First, we empirically determine a suitable prompt to generate paraphrases by using paraphrase evaluation metrics. Then, we use the original sentences and their paraphrases to enhance the simple unsupervised SimCSE model. We accomplish this by using paraphrases during contrastive learning instead of only relying on dropout as data augmentation.

We manage to achieve better performance on STS tasks than all proposed unsupervised SimCSE models by either using a new generated sentence dataset and employing the unsupervised SimCSE training framework or using our paraphrase approach. Finally, we analyze the performance of our models by examining the evaluation datasets and calculating alignment and uniformity values of our models, comparing them to the SimCSE models.

Bibliography

[1] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," 2022.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1597–1607. [Online]. Available: https://proceedings.mlr.press/v119/chen20j.html

[5] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," 2022.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[7] E. Agirre, D. Cer, M. Diab, and A. Gonzalez-Agirre, "SemEval-2012 task 6: A pilot on semantic textual similarity," in **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012).* Montréal, Canada: Association for Computational Linguistics, 7-8 Jun. 2012, pp. 385–393. [Online]. Available: https://aclanthology.org/S12-1051

[8] E. Agirre, D. Cer, M. Diab, A. Gonzalez-Agirre, and W. Guo, "*SEM 2013 shared task: Semantic textual similarity," in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity.* Atlanta, Georgia, USA: Association for Computational Linguistics, Jun. 2013, pp. 32–43. [Online]. Available: https://aclanthology.org/S13-1004

[9] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe, "SemEval-2014 task 10: Multilingual semantic textual similarity," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 81–91. [Online]. Available: https://aclanthology.org/S14-2010

[10] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigau, L. Uria, and J. Wiebe, "SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 252–263. [Online]. Available: https://aclanthology.org/S15-2045

[11] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe, "SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 497–511. [Online]. Available: https://aclanthology.org/S16-1081

[12] 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)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 1–14. [Online]. Available: https://aclanthology.org/S17-2001

[13] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 632–642. [Online]. Available: https://aclanthology.org/D15-1075

[14] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://aclanthology.org/N18-1101

[15] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[16] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” 2023.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.

[18] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, “Llama: Open and efficient foundation language models,” 2023.

[19] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli, “A SICK cure for the evaluation of compositional distributional semantic models,” in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 216–223. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2014/pdf/363_Paper.pdf

[20] L. Shen, L. Liu, H. Jiang, and S. Shi, “On the evaluation metrics for paraphrase generation,” 2022.

[21] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://aclanthology.org/P02-1040