DISTANCE DILEMMA: SIAMESE VS. TRIPLET SBERT NETWORKS FOR INFORMATION RETRIEVAL

Andreas S. H. Rygaard (s220886)[2], *Anna Bramsløw (s194656)*[1], *Nikolai Beck Jensen (s194639)*[1], *Rasmus Bryld Bagger (s194668)*[1]

Technical University of Denmark, [1]DTU Management, [2]DTU Bioengineering

ABSTRACT

In today's digital society, the increasing amounts of data calls for efficient information retrieval methods. While BERT models excel in NLP tasks, its direct use for information retrieval faces limitations due to scalability issues of pairwise comparisons and poor embedding abilities. To harness BERT's NLP capabilities for better sentence embedding, we employ a Siamese and Triplet network structure, yielding a Sentence-BERT (SBERT) model. This is trained to produce sentence level embeddings enabling information retrieval using cheap distance/similarity measures. Our study compares Siamese and Triplet SBERT models for information retrieval using cosine and euclidean similarity/distance measures. Notably, the Siamese model prefers training with cosine similarity, while Triplet strongly prefers training with euclidean distance. Furthermore, a qualitative visual inspection of embedding space was performed using PCA. This revealed a grouping effect in the Triplet cosine model, possibly explaining the poor performance of this model. Overall, the Siamese cosine model has the shortest training time of the two best performing models.

Index Terms— Language models, Transformer networks, Information retrieval, Sentence embeddings, SBERT, BERT, Siamese Network, Triplet Network, MS MARCO

1. INTRODUCTION

Information retrieval is the act of retrieving relevant text passages or documents based on a query. The quality of information retrieval has become increasingly important over the past decades as a result of its usage in various search engines, and the ever growing amount of information data.

Since its introduction in 2019, BERT models have shown top tier results in various NLP tasks [1]. However, BERT is unsuited for similarity search in a large corpus due to combinatorial explosion of pairwise comparisons. In order to use the excellent capabilities of BERT, the model can be finetuned to produce semantically meaningful embeddings by implementing it in a Siamese or Triplet network structure. This approach has yielded models with excellent sentence embedding capabilities, enabling similarity search using cheap distance metrics [2]. BERT models finetuned to make sentence embeddings are known as Sentence BERT models (SBERT).

In this paper, we investigate which of these network architectures are most suitable for information retrieval tasks across different similarity measures. The project aims to understand the similarities and differences between the two model structures when using cosine similarity and euclidean distance to compare embedding vectors. Specifically, we train four different SBERT models using the publicly available MS Marco data set [3]. These four models are evaluated on their performance in information retrieval across four different metrics; precision, accuracy, Mean Reciprocal Rank (MRR) and Normalised Discounted Cumulative Gain (NDCG).

In *Results*, we show what network structures perform best across cosine similarity and euclidean distance in addition to what training time is needed to obtain similar results for the two types of network structures.

2. MODELLING APPROACH

2.1. Data Processing

For this project, we use a subset of the public MS MARCO data set [3]. The MS Marco data is comprised of approximately 1 million anonymized queries, sampled from Bing's search query logs, as well as a corpus, comprising 8.8 million short text passages with information retrieved from Bing webdocuments. We use a subset of this data, which contains 125,000 queries while keeping the full size of the corresponding passage corpus.

With this data set follows a list of CE-scores. For each query, a pre-trained cross encoder (CE) has identified the similarity between all combinations of query and text passages in the corpus [4]. These similarity scores are logit values, ranging from negative infinity to positive infinity. To make them comparable to the range of cosine similarity values as well as euclidean distances, these values are transformed using the sigmoid function.

For all queries, the data set contains a positive passage, which holds the answer to the query, as well as a list of 'hard' negatives for each query, which are the text passages that are related to the query but does not directly contain the answer. Thus, ’hard’ negatives are the text passages, which are difficult to classify, as they semantically resemble the queries without containing the sought after information. The models are fed ’hard’ negatives to ensure that it learns more nuanced and subtle differences between similar sentences. Some of the ’hardest’ negatives are filtered away, by introducing a minimum difference between the CE-score of the positive passage and the negative passages. In our case 3, in the logit range, is decided to be the minimum CE-score difference. The filtering is done such that the hard negatives remain similar to the positive passage while ensuring that they do not contain the answer to the query.

2.2. Model Architecture

Two sentence BERT models are implemented; one with a Siamese network structure and another with a Triplet network structure (Fig. 1). In both network architectures, a BERT model is used to create token level embeddings. Specifically, a distilled BERT model is used, which is a BERT model that has been reduced in size by 40% but is 60% faster than the original BERT model [5]. The model distillation happens through a student-teacher framework, where the distilled BERT model is trained to replicate the behavior of the original BERT model. Consequently, both BERT models are language models based on a transformer network, which is a network made of transformer blocks. Each transformer block is a is a multilayer network, consisting of simple linear layers, feedforward networks, and most importantly self-attention layers [6]. In short, the self-attention layers allow the model to weigh the context of an input, i.e. for each input the model will have access to all previous inputs and will consider these when producing an output.

The Siamese network takes a query and a text passage as an input. The query and text passage are passed through a BERT model in parallel, which results in two simultaneous outputs of token embeddings. For each input, their sentence embedding is found by mean pooling the vectors of token embeddings from the BERT model, to yield a fixed length sentence embedding vector. The similarity between these two embeddings are computed using either cosine similarity or euclidean distance. For cosine similarity, the similarity range is limited to the range between 0 and 1, as this is the scale of the CE-scores. Furthermore, we are only interested in whether the text passages are related to the query or not - not whether the text passages are opposite of the query, i.e. having a cosine similarity of -1. A cosine similarity of 0 indicates that the two embedding vectors are orthogonal, which makes them linearly independent. This means that the two embeddings are unrelated, which is sufficient for our use case. In the Siamese network, the model is trained to minimize the mean squared error between the similarity between the two sentence embeddings and the corresponding CE-score of the pairing.

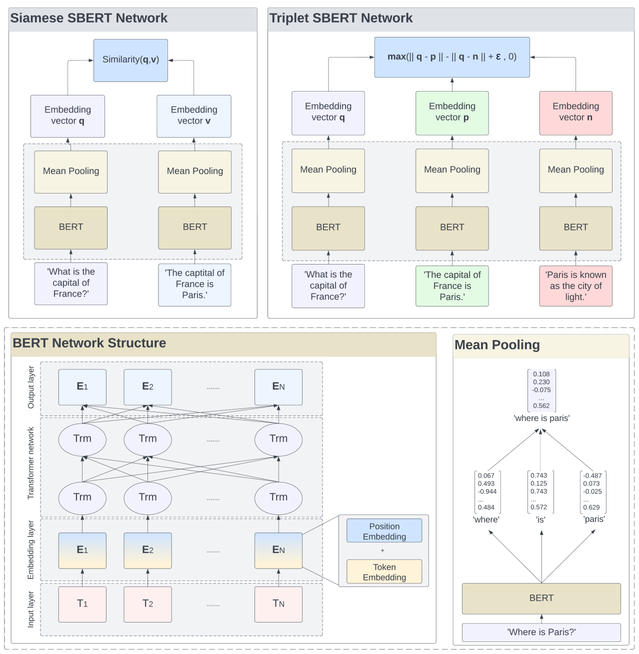

Fig. 1. The network architecture of a Siamese SBERT and Triplet SBERT Network. Each sentence is passed through the same BERT model, and the token level outputs are mean pooled to get sentence level embeddings, which are used in different loss functions to train the models. The BERT model architecture can also be seen, which is based on the structure of a Transformer network.

In the Triplet network, the input is a combination of a query $\mathbf{q}$, also known as the anchor, its most related (positive) text passage $\mathbf{p}$, and one of its ’hard’ negative text passages, $\mathbf{n}$. These inputs are passed through three instances of the same BERT model. As in the Siamese network, the sentence embedding for each input is found by mean pooling the returned token embeddings from the BERT model. The model is trained to place the sentence embedding of the query and the positive text passage close together in the embedding space and the negative text passage further away.

In both network architectures, one could replace the mean pooling of token embeddings with a max pooling. In the literature regarding SBERT [2], mean pooling showed better results compared to max pooling. This is why we chose to use mean pooling in this paper, although max pooling was a suitable alternative.

2.3. Loss Functions

Although the only difference in the structures of the Siamese and Triplet networks is the number of inputs being passed through the BERT-pooling layers, the loss functions and general training approaches are rather different. This section serves to introduce the loss functions for the two architectures and accordingly the subtleties of using different measures to evaluate how similar (or distant) the resulting sentence embeddings are. In this paper, the cosine similarity and euclidean distance is used to measure this similarity, which is a key element in the training objectives. For two n-dimensional vectors **a** and **b**, the cosine similarity is given by $\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||||\mathbf{b}||}$. To calculate the euclidean distance, the equation is $||\mathbf{x}||_2 = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$, where $\mathbf{x} = \mathbf{a} - \mathbf{b}$. One of the key differences between these is that a distance measure deems **a** and **b** relevant to each other with a low distance score whereas the cosine similarity will indicate similarity with a high score. The loss function for each model architecture thus has to be tailored to the corresponding measure.

2.3.1. *Siamese Loss*

For the Siamese network, a supervised learning approach is utilized, where the network is trained to create embeddings which minimize the mean squared error between the given cross encoder label and the similarity/distance of query-passage pair. Thus, given the BERT-embedded and mean-pooled anchor q and passage v as well as the according cross-encoder score y, we compute the loss in the cosine case as:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \frac{\mathbf{q}_i \cdot \mathbf{v}_i}{||\mathbf{q}_i||||\mathbf{v}_i||} \right)^2 \tag{1}$$

Note that in equation (1) and (4) the 0-threshold of the cosine similarity as described in 2.2 is not written explicitly for the sake of readability. In practice, this is implemented by utilizing a Rectified Linear Unit (ReLU).

In the case of euclidean distance, we have to modify the label to $1 - y$. As a result it will yield 0 for a query-passage pair which is highly relevant to each other and subsequently will result in a near-zero distance. The loss function modified for euclidean distance can be written as:

$$L = \frac{1}{N} \sum_{i=1}^{N} (1 - y_i - ||\mathbf{q}_i - \mathbf{v}_i||_2)^2 \tag{2}$$

2.3.2. *Triplet Loss*

For the Triplet model, a contrastive learning approach is utilized, where the network is trained to minimize the distance between the query-positive pair while maximizing the distance between the query-negative pair. Thus, given the BERT-embedded and mean-pooled triplet consisting of anchor **a**, positive **p** and negative **n**, we compute the loss in the euclidean case as:

$$L = \frac{1}{N} \sum_{i=1}^{N} \max\left(||\mathbf{q}_i - \mathbf{p}_i||_2 - ||\mathbf{q}_i - \mathbf{n}_i||_2 + \epsilon, 0\right) \tag{3}$$

The ϵ is a hyperparameter, controlling the distance between the query-positive and query-negative pairs. The implementation uses $\epsilon = 5$ for both the cosine and euclidean implementation. An extensive investigation on the impact of changing this hyperparameter has been left as further work.

In the case of cosine similarity, we use the computation $1 - \cos(\theta)$ to get the cosine "distance", which is 0 when two embeddings are identical and 1 when they are orthogonal. The loss function modified for the cosine similarity measure can be written as:

$$L = \frac{1}{N} \sum_{i=1}^{N} \max\left((1 - \frac{\mathbf{q}_i \cdot \mathbf{p}_i}{||\mathbf{q}_i||||\mathbf{p}_i||}) - (1 - \frac{\mathbf{q}_i \cdot \mathbf{n}_i}{||\mathbf{q}_i||||\mathbf{n}_i||}) + \epsilon, 0 \right) \tag{4}$$

The max function employed in both (3) and (4) is implemented by using the Rectified Linear Unit (ReLU) in PyTorch. All loss functions presented in this section also include a regularization term in the form of a L2 penalty on the weights.

2.4. Training Setup

All models are trained on a NVIDIA V100 GPU on the DTU Compute Cluster [7]. The models are run for 3 epochs with a batch size of 64, thus ensuring comparability in training duration across models. Two measures are taken to increase stability in training and help convergence. Firstly, a learning rate scheduler is employed, such that the learning rate is linearly decreasing throughout the epochs. Secondly, gradient clipping is implemented to prevent exploding gradients that can lead to numerical instability and hinder convergence.

3. EVALUATION

3.1. Information Retrieval Metrics

The four models are evaluated on information retrieval. For this task the models are given 200 new test queries and retrieve the most relevant passages in the corpus of 8.8 million text passages. For each query, the models retrieve the top k passages ($k = 1, 5, 10$). We use the range of k to understand if there are performance differences across varying retrieval sizes. The similarity measure used to determine the most relevant passages for each model are the same as the similarity measure used for the training of each model. The retrievals are compared to the true mappings to calculate the performance metrics. The selected performance metrics are precision, accuracy, mean reciprocal rank, and normalized discounted cumulative gain.

For all the performance metrics presented below an average is calculated across the 200 queries. We utilize a relevance function $rel(k)$, which computes the number of relevant passages retrieved for a lookup size of k passages.

$$rel(k) = \sum_{i=1}^{k} x_i \tag{5}$$

where x_i is an indicator variable equal to 1 if the ith retrieved passage is relevant and 0 otherwise. We can thus compute the accuracy and precision of an information retrieval as such:

$$\text{Accuracy@}k = \begin{cases} 1 & \text{if } rel(k) > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

$$\text{Precision@}k = \frac{rel(k)}{k} \tag{7}$$

These are measures which do not take the order of the retrieved passages into account, but rather scores relevant passages retrieved equally. To evaluate whether the models succeed in ordering the relevance of the passages correctly we utilize the Mean Reciprocal Rank (MRR) and the Normalized Discounted Cumulative Gain (NDCG). MRR measures the average reciprocal rank of the first relevant passage. This is thus a measure that places high emphasis on the ranking and position of the first relevant passage:

$$MRR@k = \frac{1}{\text{rank of first relevant passage}} \tag{8}$$

When a broader perspective on the ranking or all retrieved passages is desired, NDCG can be used. This is a measure that compares the ranking of all retrieved passages to an ideal ranking. The ideal ranking in this case would be relevant passages at all positions, because there are at least 10 relevant passages for each query.

$$NDCG@k = \frac{DCG}{IDCG} \tag{9}$$

where DCG :

$$DCG@k = \sum_{i=1}^{k} \frac{x_i}{\log_2(i+1)}$$

and $IDCG$:

$$IDCG@k = \sum_{i=1}^{k} \frac{x_i^{\text{ideal}}}{\log_2(i+1)}$$

The different performance measures are selected in order to get a comprehensive understanding of their performance in practical implementation. If the models were to be implemented in a search engine, it would be meaningful to know how often the models retrieve one relevant passage or how many of the retrieved passages are relevant - therefore the use of accuracy and precision. In addition to this, it would be important to know in what order the relevant passages are presented to the user, which MRR and NDCG are good measures for. Additionally, these metrics are often used for evaluation of information retrieval in the literature.

Due to the nature of the evaluation metrics they will all yield the same results for $k = 1$, however as k increases the differences in the results becomes more apparent.

4. RESULTS

4.1. Performance on Information Retrieval

It can be seen in Fig 2**A** and Table **??** that the Siamese network model performs best when trained with cosine similarity and that the Triplet network model performs best when trained with euclidean distance. These are the two best performing models and they perform on par.

The two best performing models score highest on all the performance metrics. This means that these two models are able to correctly retrieve passages and in the correct order relative to the other models. It should be noticed that accuracy and MRR increases as k increases whereas the opposite is seen for precision and NDCG. For precision, the information retrieval problem becomes harder as k increases, however the two best performing models manages to have a precision of 0.6 and 0.56 when $k = 10$. Additionally, they both have an accuracy of 0.9 when $k = 10$, meaning that they will in 90% of the information retrieval tasks fetch at least one relevant passage. It is noticed that the Triplet model trained with cosine similarity performs notably worse than the other models in all performance measures. In an effort to troubleshoot whether this was a cause of the margin $\epsilon = 5$ being too large, a Triplet cosine model was also trained using $\epsilon = 1$. However, the results showed similar performance to that of $\epsilon = 5$.

4.2. Convergence and Training Time

Inspecting the convergence plots in Fig. 3, it can be seen that the training loss converge throughout the 3 epochs for all models. In Fig. 3**A** the Siamese model using euclidean distance has a sharp drop in loss initially. Naturally, this is a result of the model having to adjust its weights initially such that the distance between the embeddings is in the right scale compared to the CE-scores. This is not seen on the Siamese model with cosine similarity as this measure is constricted to be between 0 and 1 and thus is already in the appropriate scale of the CE-scores. In Fig. 3**B** the Triplet model with euclidean distance reaches a loss very close to zero, however, there is no clear indication of overfitting, as this is one of the best performing models. However, the fact that the model is able to push the loss to near-zero and thus utilize the full margin $\epsilon = 5$, does raise the question whether there could be a performance increase by tuning this hyperparameter. In terms of which model is computationally preferred, the Siamese model is preferred, taking roughly 11 hours to complete 3 epochs compared to 15 hours for the Triplet network. This is a natural effect of the extra input which has to be forward passed

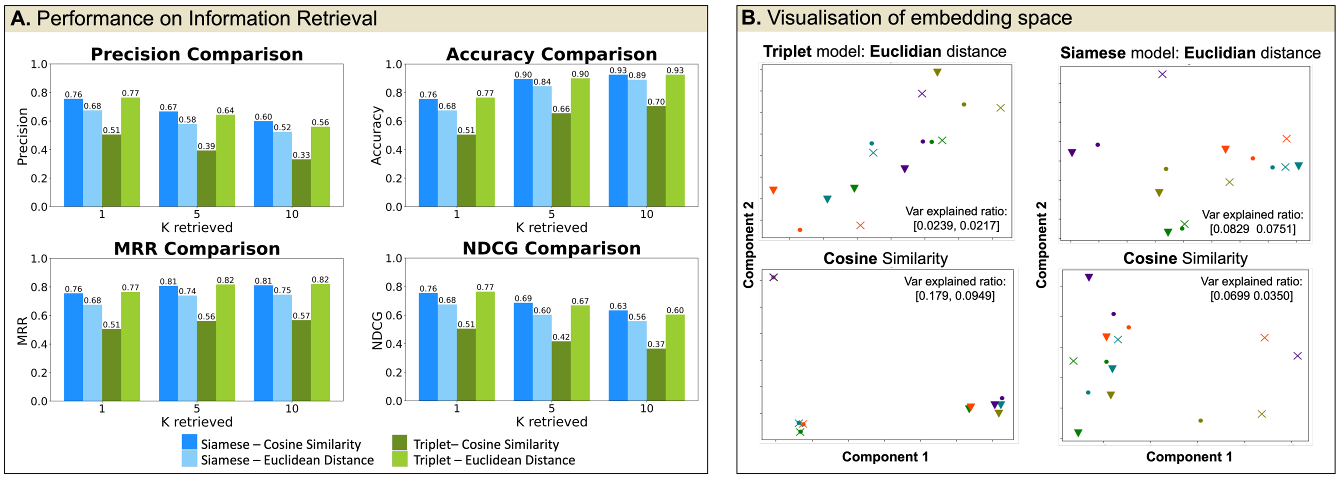

Fig. 2. **A.** Comparison of Siamese and Triplet ($\epsilon = 5$) SBERT architectures, trained with cosine similarity and euclidean distance. The performance is repported for information retrieval with 200 new queries in the corpus of 8.8 mio passages. Refer to the text for description of metrics. **B.** Visualisation of embedding space using PCA. 70k normalised triplet embeddings were used to fit a PCA, and 5 randomly selected triplets were plotted to visualise the embedding space.

through the BERT layers in the Triplet network due to the data structure.

4.3. Visualisation of Embedding Space

A qualitative inspection of the embedding space from the four different models trained in this project is performed using a Principal Component Analysis and can be seen in Fig. 2**B**. 70k normalized triplet embeddings were used to fit a PCA, and of these, 5 randomly selected triplets were plotted. With few exceptions, all models place the positive passage closer to the query, compared to the negative.

5. DISCUSSION

5.1. PCA analysis

The two best performing models, Triplet euclidean and Siamese cosine, seem to be poorly explained in the PCA having the lowest variance explained across models and placing the triplets without any clear pattern. This could indicate a greater complexity of the embedding space of these models, which might explain their superior performance. The Siamese euclidean model places each triplet separated from the other triplets, which makes sense as the hard negatives are still closer to the query than a random passage. Notably, the Triplet cosine model places all the embeddings in three groups, placing most negative in one, and the query and positives identically in another group.

5.2. Information Retrieval Performance

It is observed that the Siamese and Triplet model architectures prefer different measures of similarity (or distance) in our setup.

5.2.1. Siamese models

While the Cosine similarity Siamese model performs slightly better than the Euclidean distance, the Siamese model seems to have little preference for one similarity measure. With a continuous labeling of data given in the CE-scores, the cosine similarity seems to work well for embedding sentences on the contrary to what is seen for the triplet models. The Siamese models receive more nuanced information, as it gets continuous labels. Despite this, the Siamese models do not outperform the Triplet models, but they do train faster to reach the same performance.

5.2.2. Triplet models

The triplet model trained with cosine distance is significantly worse than all other models. We speculated if this was a consequence of using epsilon of 5 in the loss function, as the cosine distance allows a max difference of -1 of the anchor-positive and anchor-negative distances. However, changing the epsilon to 1 did not produce a better model. Looking at the PCA in Fig. 2**B**, the Triplet cosine appears to embed sentences in three distinct groups, which might be a consequence of categorical data labels in contrastive learning, compared with the continuous CE-score in the Siamese network. While the grouping satisfies the loss criterion well, it leaves less flexibility for dealing with the complexity of the data set, which could explain the poor results. In the Triplet euclidean, the problem of grouping is not observed, which might be because of the $\max$ operation in the loss function, whereby the score does not improve if the embeddings are moved further apart, beyond a certain point. With an epsilon above 1, the $\max$ operation is never used in the triplet cosine, however it could be interesting to see if the grouping would be avoided by choosing an epsilon between 0 and 1. In addition, we note that compared to euclidean distance, it seems intuitively harder to embed the triplets with cosine distance, as this depends only on the angle between the vectors, while the euclidean distance depends on the magnitude as well.

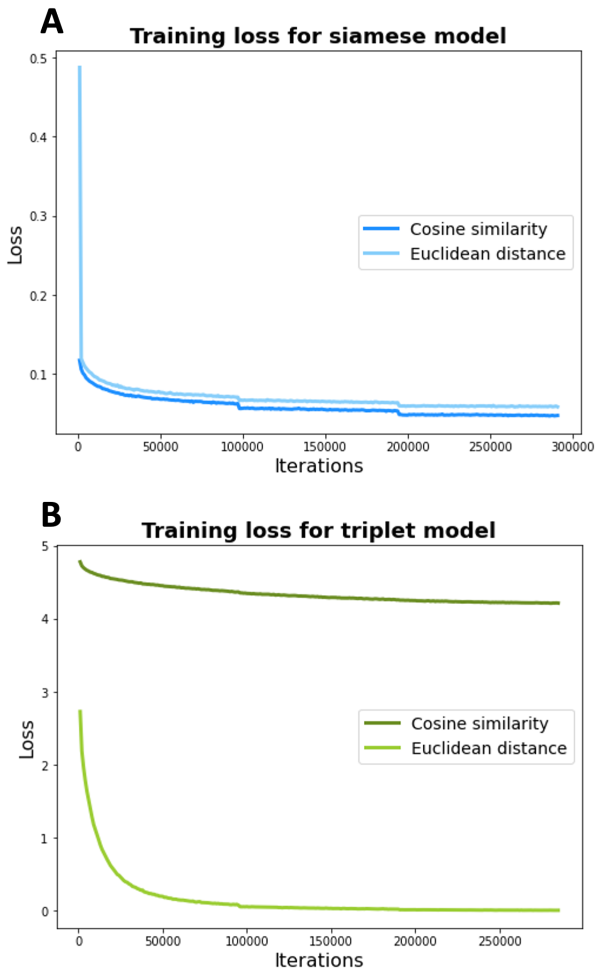

Fig. 3. Training loss convergence of the 4 models: **A** Siamese models **B** Triplet models. Trained with Cosine similarity and Euclidean distance respectively.

6. CONCLUSIONS

In this project, four SBERT models have been implemented and their information retrieval performance has been evaluated. It is investigated which configuration of two network architectures, Siamese and Triplet, and two similarity/distance measures, cosine and euclidean, yield the best performance in information retrieval tasks. The results show that the model configurations have similar performance, however, the Siamese and Triplet network architectures perform best with different measures. In addition, it is found that the Triplet network using cosine similarity show substantially poorer results compared to the rest of the models. Furthermore, it is found that the Siamese network has a shorter training duration than the Triplet model, completing 3 epochs in roughly 11 hours compared to 15 for the Triplet network. As a result, the Siamese network structure in combination with cosine similarity is to be preferred when implementing SBERT models for information retrieval. However, it should be noted that this may not be the case when dealing with different data sets.

7. CODE REPOSITORY

The code which has been created for this project may be found at `https://github.com/annabramsloew/DL-SBert`

8. REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[2] Nils Reimers and Iryna Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang, "Ms marco: A human generated machine reading comprehension dataset," 2018.

[4] Nils Reimers, "Ms marco hard negatives," 2021, `https://huggingface.co/datasets/sentence-transformers/msmarco-hard-negatives`.

[5] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2020.

[6] Dan Jurafsky and James H. Martin, *Speech and Language Processing*, chapter 10, Stanford, 2023, `https://web.stanford.edu/~jurafsky/slp3/10.pdf`.

[7] DTU DCC Computing Center, "Gpu nodes," 2023, `https://www.hpc.dtu.dk/?page_id=2129`.