SROBERTA YIELDS SIMILAR PERFORMANCE ON DUPLICATE QUESTION RETRIEVAL USING REDUCED EMBEDDING SIZES

Andreas Bagge s214630, Nikolaj Topsøe s214653, Gustav Rørhauge s214657.

DTU Lyngby

ABSTRACT

When performing queries for information retrieval, simply matching keywords rarely performs optimally. Instead, a language model can be trained to embed queries within a semantic space, where it is then possible to match similar queries based on some distance metric. In this way, queries can be matched with sentences in the same semantic space, without necessarily sharing any words at all. In this paper, we train a sRoBERTa model to find duplicates in a corpus of questions asked on the website Quora, and investigate the effect of changing the dimensionality of the embedding space. Reducing the embedding space comes with a decrease in computation time, which can be of great value. Our findings support the conclusion that for certain applications, the embedding size can be greatly reduced without significant loss in performance.

1. INTRODUCTION

When searching for information on the internet or in other databases, it can be frustrating to miss out on key information just because of differences in wording. In classic lexical search where word-for-word similarity is the key ranking metric, the query and the information searched for need to use the same words in order for the search to be successful. If possible, searching based on the context of the query, i.e the *meaning* of the sentence or question, would make it possible to find answers to other essentially identical queries (Compare 'What are the short-term effects of climate change?', with 'How will global warming affect the near future?') and thereby retrieve relevant information with much higher accuracy. We attempt to solve this problem using large pre-trained models such as RoBERTa which already has a developed semantic space, which is then tailored and finetuned to the specific problem. In particular, we will be using a dataset of Quora questions, consisting of questions paired with another similar question, and a dissimilar one. The original objective of this dataset is to detect duplicate questions with different wordings. Previously, others working on similar Quora datasets have shown duplicate detection rates of up to 89% [1][2]. In this paper, we showcase how using the language model RoBERTa can enable us to match queries that use different wordings to ask essentially the same questions. Specifically, we look at how the size of the embeddings affects the performance of the query-matching dataset. Given that the standard embedding size is 768 (using the RoBERTa-base model), we investigate whether similar results can be achieved using embeddings of different dimensionality. In particular, we aim to compare the following embedding sizes: $\mathbf{D} \in \{8, 64, 128, 512, 768, 1024, 2048\}$. Embeddings of different sizes, especially embeddings in low-dimensional space, are relevant for multiple reasons: the "dimensionality of language" is not determined and it is therefore unknown which specific embedding size yields the best results for our specific task. Secondly, lower-dimensional embeddings can be easier to interpret and give faster computational times when calculating pairwise distances.

Our findings support the conclusion that at least for some applications, severely reducing the embedding sizes will only marginally affect performance, while providing improvement in computation time.

2. DATA

To attack the problem of finding duplicate questions to a given query we have used the 'Quora duplicates triplets' dataset from `https://public.ukp.informatik.tu-darmstadt.de/reimers/sentence-transformers/datasets/paraphrases`. We have $\sim$ 100k 'triplets' of natural language data, collected from the website Quora. A triplet is a collection of three sentences consisting of (Anchor, Positive, Negative). The anchor is our query, being a question or a statement for which we want to find similar queries. Finding similar statements or questions can be extremely useful if, for example, you pose a question on Quora that has already been posed before, but you happen to phrase it differently i.e. with different wording. If we can detect these equivalent queries based on similar meanings or contextual information we can redirect or suggest existing question-answer pairs and thereby reduce redundancy and improve search results. The second part of the triplet is the Positive. This phrase is contextually similar to our Anchor. Lastly, we have the Negative which is a phrase that is semantically further away from the Anchor, than the Positive example. An example of a triplet from the dataset could be:

- **Anchor:** *'How do you know if you are in love?'*
- **Positive:** *'How do you know when it is true love?'*
- **Negative:** *'How do you know when you love yourself?'*

During training the model should learn to position the anchor and the positive near each other in the embedding space, while pushing the negative away. What humans consider meaningful clustering/spacing in the linguistic semantic space is undefined and hard to quantify. By using the Quora dataset, we narrow this problem down to a specific distribution with which we can instruct a model on how to learn these spacings and clusterings.

3. METHOD

Instead of training our own RNN, LSTM, or transformer model from scratch, we used the RoBERTa base model from Facebook's research group found at `https://github.com/facebookresearch/fairseq/tree/main/examples/roberta`. With 125M parameters, this sophisticated transformer model was trained on a number of different tasks such as fill-in-the-blank and continuation prediction, thus giving it broad linguistic expertise which is obviously a helpful starting point for our task. This model takes as input a sentence and gives a vector embedding of each word as output. Individual words in a sentence are called tokens. Let a sentence be of length $\mathbf{k}$, then RoBERTa outputs an embedding vector $\in \Re^{768\times\mathbf{k}}$. An average is taken along the token axis so we end up with a fixed-size sentence-wide embedding $\mathbf{x} \in \Re^{768}$. This is the same mean-pooling technique used in [4]. Max-pooling is also an option but we selected to go with the default setting, i.e. mean-pooling to make our results more comparable with existing literature. Additionally, we add a linear layer such that the model can make embeddings of any size we choose. We will call the size of this output dimension $\mathbf{D}$. We also keep a "base" model that simply outputs the vector obtained from the mean-pooling for reference, i.e. a model that has no dense layer at the end. Defining our data, the triplet, as the Anchor ($\mathbf{A}$), Positive ($\mathbf{P}$), and Negative ($\mathbf{N}$), and defining our model as f : 'text' $\longrightarrow \Re^{\mathbf{D}}$, the Triplet Margin Loss function [3] can then be defined by means of the Euclidean Distance function:

$$\mathcal{L}(\mathbf{A},\mathbf{P},\mathbf{N}) = \max(\| \mathrm{f}(\mathbf{A}) - \mathrm{f}(\mathbf{P})\|_2 - \| \mathrm{f}(\mathbf{A}) - \mathrm{f}\,(\mathbf{N})\,\|_2 + \epsilon, 0) \quad (1)$$

Where we have chosen $\epsilon = 0.05$. ϵ, which should always be positive, is a hyperparameter that can be tuned to the specific task at hand. If ϵ is very high, the negative will be pushed further away from the anchor-positive-pair. ϵ also ensures that the model doesn't simply output zeros, which, by quick inspection, would optimize the loss function in the case where $\epsilon = 0$. It should be obvious that this constraint will force the model to separate disparate sentences and gather equivalent sentences in this $\mathbf{D}$-dimensional space, in order to achieve the lowest loss score. After training the model and extracting embeddings for each question in our corpus, the model finds duplicates by comparing distances between the query vector and the corpus in the $\mathbf{D}$-dimensional vector space and returning the k closest sentences. These sentences should be the sentences that the model deems the most similar in semantic space. During the training and evaluation, we used Euclidean distance to select the most similar questions. One could also have chosen to use Manhattan distance or cosine similarity, as these have been shown to produce roughly the same results[4].
As baseline comparisons, we have included two different 'simple' models: *Baseline 1*, which is a base model (i.e. no dense layer) that is also untrained. Since the RoBERTa model already has a broad understanding of linguistics, we theorize that this model could achieve tolerable results. *Baseline 2* is a classical, non-deep-learning approach using bag-of-words to achieve a binary vector for each sentence, and we then use Jaccard similarity to rank sentences. This model has no idea of context and we therefore theorize that the sRoBERTa-models will outperform it.

3.1. Training

For our model to learn a spatial representation of language semantics we must feed it with large amounts of training samples. We perform a training, test, and validation split of 80%, 15%, and 5% on our data to do this. For each batch of training samples we map them into embeddings $\mathbf{A}, \mathbf{P}, \mathbf{N}$ and compute the Triplet Margin Loss. Then we take a gradient step and repeat until finished. The training algorithm can be seen below. For each epoch, we test the model on validation samples to get an estimate of performance. We used a dropout rate of 0.1 between the RoBERTa and the linear layer. We used no activation function on the output to hinder constraints on the output. Each model was trained for 10 epochs. In reality, our training converged after roughly 2 epochs, which means fine-tuning can be done more swiftly for future applications. We only trained on one seed. To better generalize our results, we could have trained on multiple seeds and averaged the results but this was not possible due to time constraints. The training was performed on Nvidia TESLA v100. The code for this project can be found at our GitHub.

Training algorithm

```
1: Ensure embedding size $\mathbf{D}$
2: Ensure model $f$ : 'text' $\longrightarrow \Re^{\mathbf{D}}$
3: repeat
4:     $\mathbf{A}, \mathbf{P}, \mathbf{N} \sim$ Quora
5:     $L = \mathcal{L}(\mathbf{A}, \mathbf{P}, \mathbf{N})$
6:     Take gradient descent step on $\nabla_\theta L$
7: until convergence
```

3.2. Testing

To evaluate the performance of our model we use Euclidean distance to find the embeddings that are closest in semantic space. Other metrics such as cosine similarity can also be used but we chose to use the same as in`https://arxiv.org/pdf/1908.10084.pdf` for comparability.
Since we are interested in finding texts of equivalent meaning, our test-set consists of duplicates i.e. only Anchors and Positives. In practice, we concatenate all the anchors and positives and label them according to the index of their original pairing. This yields a test set of ~10,000 questions, with each question paired with another, which is the most similar. We then embed all the questions and pick the top-k most similar questions for each question. The accuracy of the model is then the frequency with which the actual original duplicate is contained in the top-k questions, where the "top k questions" are the k questions that are the closest in the semantic space.
We also used more subjective methods of illustrating the overall performance of the models such as handcrafting hard-to-distinguish questions and examining how the models manage to separate them.
Lastly, we also wanted to show how the different embedding sizes affected the computational times required to calculate the pairwise distances between the embeddings to show the trade-off between performance and time efficiency.

4. RESULTS

Figure (1) shows an example of the Euclidean distances between a sample of four questions, as modeled using the 512-dimensional embedding.

In figure (2) we do PCA on the questions embedded by the model with dimension 512, and plot 5 questions in the space defined by the two most significant dimensions. Unfortunately, it is very hard to visualize the structure in a very high-dimensional embedding space, since the first few principal components only explain very little of the variance. Still, the placement of the embeddings seems reasonable.

In figure (3) accuracies for 8 models with different embedding sizes are shown. To the left, we see how the accuracy changes when testing whether the correct question is in the top-k = 1, 2, 3, 4, or 5 closest embeddings. To the right, we see the accuracies for all models using the top-k = 5 retrievals.

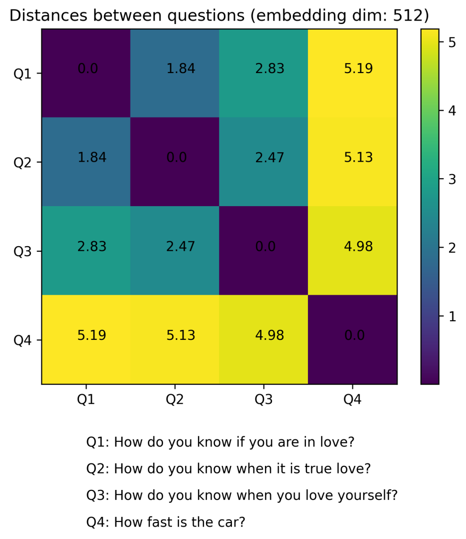

Fig. 1. Distance between four questions using $\mathbf{D} = 512$. Q1 and Q2 are almost identical in semantic meaning and therefore lie very close. Q3 is different to Q1/Q2 but close, and Q4 is very different from all the other questions.

Lastly, figure (4) shows the difference in performance vs. the time efficiency of each model.

5. DISCUSSION

As expected, we see that the larger embeddings give higher accuracies on the test set. Somewhat surprisingly, even the model with an embedding size of only 8 was able to find the correct duplicate question ~69% of the time with top 1 testing, and almost 98% using top 5 testing. This means, that the size of the embeddings can most likely be significantly reduced from the standard 768 without much loss in accuracy. In fact, our results also show that the potential loss in accuracy is balanced out with increased computational performance. The computational time doubles if the embedding size doubles, but the accuracy is only increased by a few percentile points. It should be noted however, that the dataset doesn't provide the true top 5 questions relevant to a given question, so although the smaller embedding sizes show similar performance when tested on whether the most relevant duplicate is in the top 5, the overall quality of the top 5 closest questions may be lower, and not affect the accuracy when measured in this way.
The results also show that the base RoBERTa-model (with no dense layer) performs better than its counterpart with an equally large embedding size that uses an extra dense layer. We will attribute this to the fact that the dense layer has not undergone as much training as the already pre-trained RoBERTa-model, therefore leaving the final dense layer as a bottleneck for the overall performance. We also included an

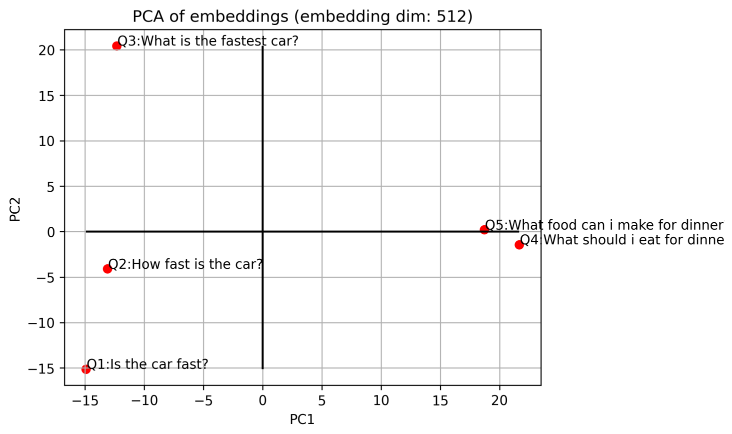

Fig. 2. Principal component analysis on 5 embedded questions. Though it is difficult to examine the behavior in high-dimensional space, the model ($\mathbf{D} = 512$) seems to correctly distinguish the questions. Q4 and Q5 are identical in semantic space. Q1 and Q2 are also close to each other but different (notice Q1 is a yes/no question, while Q2 is a more open question), and Q3 is different from all others but closest to Q1/Q2

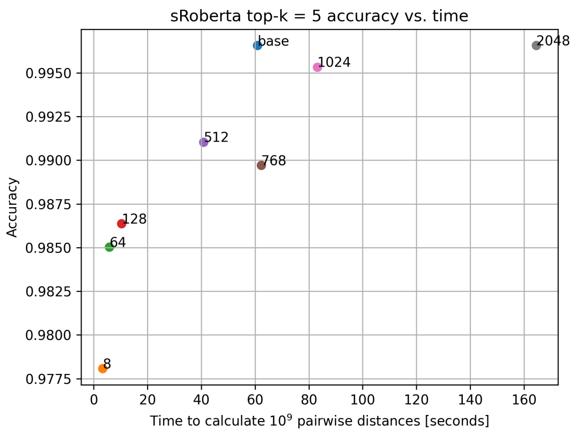

Fig. 4. Difference in accuracy on the top-k = 5 retrieval task vs. time efficiency. ”Time efficiency” is defined as the time used to calculate 10^9 pairwise distances with the given embedding size. Computational times are calculated using an Nvidia T4 GPU.

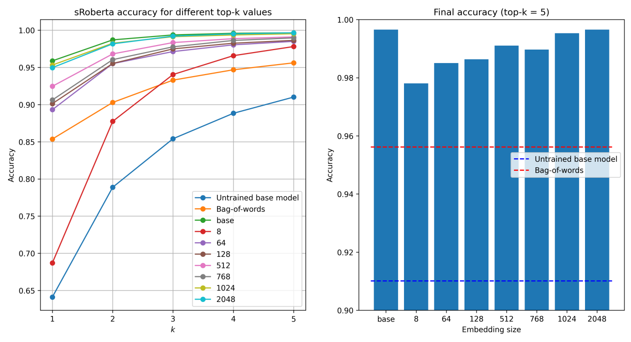

Fig. 3. Accuracies for different embedding sizes, for top 1 to top 5 results. Top k accuracy refers to the frequency with which the correct question is included in the top k nearest questions. Raw results can be seen in table 1

			k		
Model	1	2	3	4	5
baseline 1	0.641	0.789	0.854	0.888	0.911
baseline 2	0.854	0.903	0.933	0.947	0.956
base	**0.959**	**0.987**	**0.994**	**0.996**	**0.997**
8	0.687	0.877	0.94	0.966	0.978
64	0.893	0.955	0.971	0.98	0.985
128	0.901	0.955	0.975	0.983	0.986
512	0.925	0.968	0.983	0.989	0.991
768	0.906	0.96	0.978	0.986	0.99
1024	0.953	0.983	0.991	0.994	0.995
2048	0.95	0.982	0.992	0.995	**0.997**

Table 1. Accuracies for different embedding sizes for top 1 to top 5 results. Top k accuracy refers to the frequency with which the correct question is included in the top k nearest questions. *Baseline 1* is the untrained base model, and *baseline 2* is the bag-of-words model.

”untrained” baseline comparison metric as well, i.e. an untrained base model, which showed significantly worse results compared to the finetuned baseline-sized model. Therefore, even though the RoBERTa model already has a broad understanding of linguistics, a sRoBERTa model should still be fine-tuned on the specific task at hand. Our other baseline, the bag-of-words method, generally produced worse results than the other models.
We see, that with this dataset, the differences in accuracy decrease when going from top 1 to top 5 testing. Already at the top 3 testings, the difference between using e.g. 64 and 512 dimensions seems quite small.
Interestingly, on the top 5 retrieval tasks, the base model and the $\mathbf{D} = 2048$ model obtain the same accuracy on the test set ($\approx$ 0.997% accuracy). This could hint towards the fact that the models obtain near-perfect performance and that the few missrankings could be attributed to noise in the dataset.

6. CONCLUSION & FUTURE WORK

From our results, we see that the sRoBERTa model is very capable of solving the problem of finding matching queries, with the best model finding the exact true duplicate 96% of the time (using top 1 search). We also see, as expected, that with decreasing embedding size the accuracy of the model goes down. However, depending on the application, our results also show that significant gains in computation time may be achieved by reducing embedding size while suffering only a minimal hit to accuracy. Generally, for optimal performance, one should go with the base model producing 768-dimensional embeddings. For faster computational times, one can freely choose any embedding size less than 768-dimensions with little loss in performance.

7. REFERENCES

[1] Quora-question-similarity-using-bert-embeddings-with-siamese-networks-as-feature-extractors. *GitHub*, November 2019. [Online; accessed 20. Nov. 2023].

[2] Bert-for-quora-duplicates. *GitHub*, November 2023. [Online; accessed 20. Nov. 2023].

[3] https://en.wikipedia.org/wiki/Triplet_loss.

[4] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv*, August 2019.