# Sentence embeddings for Quora question similarity

**Lynette Gao**
qgao38@stanford.edu

**Yujing Zhang**
yujingz@stanford.edu

## Abstract

Duplicate questions are a common issue in community question answering forums. In this project, we work on the Quora Question Pairs dataset and aim to build a question similarity classifier to detect duplicate questions, in order to provide better Quora user experience. We explored different deep learning model architectures (LSTM [1], BERT [3]) with Siamese networks. Sentence-BERT[6] achieved 99.9% training accuracy and 89.7% dev/test accuracy. To alleviate the overfitting issue, we proposed to replace the softmax layer with a few fully-connected layers in order to capture more general correlations between sentence embeddings and similarity score. We also added a penalty to the similarity score for question pairs with high similarity scores but different question words. Finally, we achieved 91.56% test accuracy.

## 1 Introduction

Recent years have witnessed a tremendous growth of community-based question answering (cQA) forums, such as Reddit, Quora etc. Taking Quora as an example, over 100 million people visit Quora every month. As the number of questions grows, duplicate questions are everywhere. This poses an issue because multiple questions with the same intent can cause seekers to spend more time finding the best answer to their questions, and increases responders' burden to answer multiple versions of the same question.

In this project, we work on the Quora Question Pairs dataset and try to build a question similarity classifier by applying advanced deep learning techniques. This could contribute to the downstream retrieval task, providing better Quora user experience consequently. The input of the classifier is a pair of sentences in text. The output is a similarity score with a range from 0 to 1. We explored different model architectures (LSTM [1], BERT [3]) with Siamese networks and achieved 99.99% training accuracy and 89.7% test accuracy with Sentence-BERT (SBERT) [6]. On top of SBERT, we replaced the softmax layer with a few fully-connected layers and explored dynamic masking and skip connection on question words to mitigate the overfitting issue. Finally, we achieved 91.56% test accuracy with modified SBERT [6] (SBERT*) and a question word penalty.

## 2 Related work

### 2.1 Manhattan LSTM

Siamese networks are popular among similarity tasks. Manhattan LSTM Model [1] applied Siamese architecture in sentence similarity tasks, with a shared LSTM-based RNN encoder (bi-encoder) to generate embeddings of sentence pairs. The shared encoder weights make the mode less tendency to overfit. [2] explored other cells and found that GRU performs better than LSTM. From [1], multi-layer or bi-directional LSTM cannot improve the performance, which indicates that this model architecture has limitation to capture more complex features.

### 2.2 BERT/Sentence BERT

BERT [3] is a transformer-based [4] language model to learn deep bidirectional representations from unlabeled text data. It uses the pre-training and fine-tuning strategy. During pre-training, it trains two unsupervised tasks. The parameters are fine-tuned on downstream tasks with labeled data. It advanced 11 natural language processing (NLP) tasks (including semantic textual similarity (STS)) by generating more general text embeddings with pre-training and is widely used in solving NLP problems. RoBERTa [5] made some modifications (e.g. remove the next sentence prediction objective, dynamic masking) on the pre-training procedure of BERT and achieved better performance on downstream tasks.

For sentence-pair tasks, BERT/RoBERTa used a cross-encoder which takes the concatenated sentences separated by a special token [SEP] as inputs which causes a massive computational cost. Sentence-BERT (SBERT) [6] addressed the computation cost issue of cross-encoder by using Siamese networks to generate independent sentence embeddings that can be compared using similarity measures (e.g. contrastive learning techniques). SBERT largely reduced the training time while maintaining the state-of-the-art accuracy from BERT. Due to the strong power of SBERT, most of our explorations are made on top of it.

### 2.3 Self-supervised learning

Self-supervised learning aims to learn general representations of input data without human annotated labels. It has been widely used in computer vision [7] and NLP. BERT [3] applied "masked language model" (MLM) and "next sentence prediction" (NSP) as two unsupervised tasks. Albert [8] proposed a "sentence order prediction" (SOP) objective that focuses on inter-sentence coherence. [9] applied "Sentence Augmentation Type Prediction" (SATP) on text classification tasks which randomly inserts/replaces/deletes/swaps tokens and predicts the type of augmentation that is applied. These tasks are effective to generate a more general sentence representation during pre-training and to capture task-specific features during fine-tuning. Inspired by the recent success of self-supervised learning, we explored dynamic masking as a self-supervised auxiliary task.

## 3 Dataset

The Quora Question Pairs dataset consists of 404,290 question pairs, provided by Quora [1]. Each example has 6 fields as shown in Table1:

| id | qid1 | qid2 | question 1 | question 2 | is duplicate |
|---|---|---|---|---|---|
| 11 | 23 | 24 | How do I read and find my YouTube comments? | How can I see all my Youtube comments? | 1 |
| 21 | 43 | 44 | What's causing someone to be jealous? | What can I do to avoid being jealous of someone? | 0 |

Table 1: Quora Question Pairs dataset

The dataset is unbalanced with 37% duplicate question pairs and 63% different pairs. We split the dataset into three portions: training set (95%), validation set (2.5%) and test set (2.5%). Each preserves the same ratio of duplicate pairs as the original dataset.

During data preprocessing, we removes stop words, applies lowercasing and tokenization (e.g. WordPiece), and converts a list of tokens to ids.

## 4 Methods

### 4.1 Manhattan LSTM

Manhattan LSTM [1] uses Siamese networks to generate sentence embeddings with a pre-trained word2vec and a single uni-directional LSTM layer. It computes a scaled Manhattan distance (0 - 1) of two embeddings (u and v) for similarity prediction.

$$y_{pred} = exp(-\|u - v\|_1) \tag{1}$$

[1]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

## 4.2 Sentence-BERT

Sentence-BERT [6] uses a pre-trained BERT model as an encoder and Siamese architecture to generate sentence embeddings. The encoder is followed by a pooling layer. SBERT mentioned three pooling strategies: max, mean, and [CLS] token pooling. Then, it concatenates the embedding of sentence 0 (u), embedding of sentence 1 (v), and |u - v|, and feeds the concatenated embeddings to a softmax layer to compute a similarity probability.

$$y_{pred} = softmax(concat(u, v, |u - v|)) \tag{2}$$

## 4.3 Sentence-BERT*

In our work, we made some modifications on SBERT to capture better similarity features.

**Dense layers with reducing sizes**

We replaced a single softmax layer with a few dense (fully-connected) layers whose sizes are in decreasing order as shown in Figure 1, in order to reduce the complexity of features and capture more general correlations between embeddings and similarity scores.

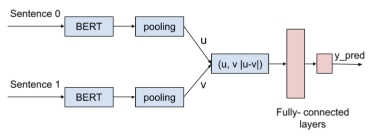
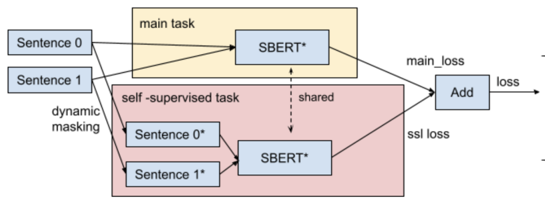


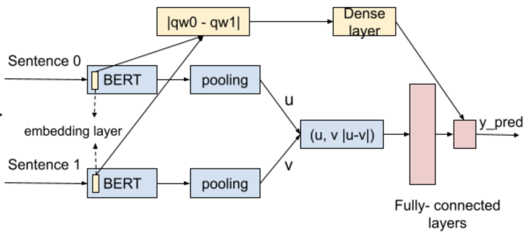
Figure 1: modified Sentence-BERT*

**Auxiliary task: Dynamic masking**

Inspired by the recent success in self-supervised representation, we explored to co-train a two-tower SBERT* model. The two towers share the same SBERT* model but are fed different input data: main task with the original sentence pairs and an auxiliary task with augmented sentence pairs. It is designed to tackle the overfitting issue. Inspired by RoBERTa [5], we augmented the sentence pairs by dynamically masking some tokens in different epochs. Compared to the element-wise dropout applied by BERT models, the dynamic masking is applied on token-level to reduce the impact on a specific token.



(a) SBERT* with dynamic masking



(b) SBERT* with skip connection on question word

**Question word**

Question words (e.g. when, who, how, why, where) play an important role in the semantic of a question sentence. For a pair of sentences with different question words, they are of different meanings even if other tokens are exactly the same. In order to increase the importance of questions words, we explored two different task-specific approaches: 1) add a skip connection from the embedding difference of two question word tokens to the last dense layer for similarity score prediction (skip connection) as shown in Figure 2; 2) add a penalty to the similarity score for evaluation only (question word penalty) which penalizes a high similarity score with different question word tokens ($qw_0$ and $qw_1$).

$$y'_{pred} = y_{pred} - \alpha * y_{pred-true} * |qw_0 - qw_1| \tag{3}$$

## 5 Experiments and discussion

### 5.1 Setup

**Manhattan LSTM** uses a word2vec pretrained on Google News dataset[2] with 300-dimensional vectors for 3 million words and phrases. The hidden size of the LSTM layer is 50. For training, we use Adam optimizer with learning rate 0.01, set mini batch size = 1024 and take mean square error as the loss function.

**Sentence-BERT** loads a pre-trained BERT base model[3] (number of layers = 12, hidden size = 768, intermediate size = 3072, vocabulary size = 30522) as an encoder and fine-tunes it on the training dataset. For training, we use Adam optimizer with learning rate 3e-5, set mini batch size = 32 (64 cannot fit) and take binary cross entropy as the loss function.

We also tried freezing the first few layers and only fine-tuning later layers or freezing the pre-trained encoder and adding additional layers, but we didn't get any performance gain.

### 5.2 Results and discussion

Accuracy, precision, recall and F1 score are some metrics we use for evaluation. Table 2 shows that SBERT largely outperforms Manhattan LSTM in both training and test accuracy even with the same similarity function. The Manhattan LSTM paper mentioned that more LSTM layers and bi-directional LSTM cannot improve the model quality. It indicates that the attention mechanism and pre-training strategy contribute a lot to the model complexity and generality in our task, resulting in much lower bias and variance.

| model | train | validation | test |
|---|---|---|---|
| LSTM (Manhattan distance) | 81.35% | 76.3% | 66.8% |
| SBERT (softmax) | 99.9% | 89.71% | 89.76% |
| SBERT + Manhattan distance | 99.0% | 85.4% | 85.3% |
| SBERT* (2 dense layers) | 99.9% | **91.5%** | **91.3%** |
| SBERT* + dynamic masking | 99.9% | 91.3% | - |
| SBERT* + question word penalty | 99.9% | **91.51%** | **91.43%** |
| SBERT* + skip connection | 99.9% | 91.42% | 91.27% |

Table 2: Accuracy on different models

| Pooling + dense layers | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| Mean + [32, 1] | 91% | 86.5% | 89.5% | 0.88 |
| CLS + [64, 1] | 89.93% | 84.5% | 89.1% | 0.867 |
| Max + [64, 1] | 91.15% | 86.6% | 90% | 0.882 |
| Mean + [64, 1] | **91.5%** | 87.5% | 89.8% | **0.886** |
| Strided mean + [64, 1] | 90.6% | 86.5% | 89.1% | 0.878 |
| Mean + [128, 1] | 91% | 86.6% | 89.5% | 0.880 |
| Mean + [128, 64, 1] | **91.5%** | 87.2% | 90.3% | **0.886** |

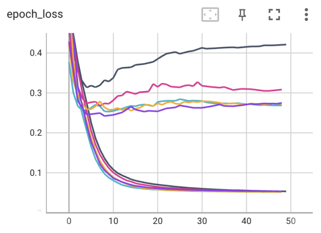
Table 3: Validation metrics with different pooling strategies and dense layers

We experimented on different pooling strategies. In addition to the approaches in the SBERT paper, we also tried strided pooling. It turns out that mean pooling performs the best in our task. We also found that the validation loss starts increasing after 5 epochs if not concatenating |u-v| (Figure 3). So |u-v| also offers additional information to improve the model generality. In addition, We got better validation metrics by replacing the softmax layer with a few dense layers (2 layers with neuron size [64, 1]) for similarity computation as shown in Table 3. Compared to a single softmax layer, the dense layers gradually reduce the feature dimensions from $3 * hidden\ size$ (2304) to much lower dimensions (e.g. 64) which helps to drop useless information and generalize similarity features.

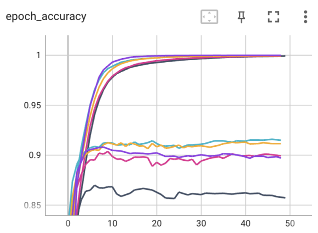
In our experiment, the auxiliary dynamic masking task didn't improve the dev/test accuracy as shown in Table 2. The reason might be that the questions in the dev/test set are completely different from the questions in the training set. However, the token-level masking only helps token-level variations (e.g.

[2]available at https://code.google.com/archive/p/word2vec/

[3]available at https://github.com/google-research/bert



(a) Loss



(b) Accuracy

Figure 3: training/validation loss and accuracy
(*red: CLS + [64, 1]; blue: mean + [64, 1]; yellow: max + [64, 1]; purple: SBERT (mean + softmax); black: mean + [64, 1] without |u-v|)

remove/change/add a token). In reality, there are such duplicate questions. So we believe that the dynamic masking task is still a potential approach to improve the model generality in the real world.

During evaluation, we also tuned the threshold for predictions. If the similarity score exceeds the threshold, we consider the two sentences as duplicate. We got a better accuracy with threshold = 0.7. It makes sense since not all similar questions are considered as duplicate (e.g. *What is the difference between blocking and muting on whatsapp?* and *What is the difference between blocking and muting?*, similarity score 0.56).

We explored two approaches to increase the importance of question words: 1) skip connection for training; 2) question word penalty for evaluation. We got a similar accuracy with 1) and a better accuracy with 2). For 1), the skip connection is added during training, so it tends to fit on the training set as well. And the original model without the skip connection might be the able to capture the same question word feature as well. 2) is a more general approach since the penalty is relatively independent to the training data. From Table 4, it reduces positive predictions especially false positives and meanwhile increases negative predictions. It is reasonable for our task since we care more about false positives than false negatives. With a proper penalty weight, we can even get more true predictions and a higher precision metric though sacrificing a bit on recall. For example, the predicated similarity score of different questions *Why do you downvote answers?* and *How often do you downvote an answer?* reduced from 0.83 to 0.48.

| prediction threshold t, question word penalty weight $\alpha$ | True Positive | True Negative | **False Positive** | False Negative | Precision | Recall | **Accuracy** | **F1** |
|---|---|---|---|---|---|---|---|---|
| t = 0.5, $\alpha = 0.0$ | 3304 | 5826 | 484 | 386 | 87.22% | 89.54% | 91.30% | 0.883 |
| t = 0.5, $\alpha = 0.2$ | 3293 | 5841 | 469 | 397 | 87.53% | 89.24% | 91.34% | 0.884 |
| t = 0.5, $\alpha = 0.35$ | 3278 | 5865 | 445 | 412 | 88.5% | 88.83% | **91.43%** | **0.885** |
| t = 0.5, $\alpha = 0.5$ | 2209 | 5990 | 320 | 1481 | 87.35% | 59.86% | 81.99% | 0.710 |
| t = 0.7, $\alpha = 0.0$ | 3276 | 5871 | 439 | 414 | 88.2% | 88.8% | 91.47% | 0.884 |
| t = 0.7, $\alpha = 0.2$ | 3250 | 5906 | 404 | 440 | 88.9% | 88.1% | **91.56%** | **0.885** |

Table 4: Confusion matrix and evaluation metrics for SBERT* w/o question word penalty

## 6 Conclusion/Future Work

From our experiments, SBERT-based model architecture largely outperforms the Manhattan LSTM model architecture, thanks to the attention mechanism and the pre-training strategy which improve the model complexity and generality. Mean pooling is an effective way to reduce the sequence length dimension, generating a good sentence-level embeddings. The difference of the sentence embeddings can provide additional information to capture similarity features. The newly proposed dense layers with reducing sizes could capture more general similarity features from a very high-dimensional feature vector, which effectively alleviates the overfitting issue. In addition, question words play an important role in the semantic. It could reduce the false positives by adding a small penalty to question pairs with different question words but high similarity scores. With SBERT, 2 dense layer and question word penalty, we achieved the best accuracy 91.56% and F1 score 0.885 on the test set.

The overfitting problem is not completely solved. We would like to explore more data augmentation techniques (e.g. switch sentence segment ordering task, augmented SBERT [10]). It's also worth trying our proposed approaches on other datasets to examine if the same conclusion applies. In addition, we could try a larger model size and other model architectures. BERT maintains a full-length token-level representation at every layer, which contains redundant information for sequence-level representation. We could try Funnel-Transformer [11] to gradually reduce the feature dimensions and learn better sentence-level embeddings.

## 7 Contributions

In this project, Lynette worked on the implementation and experimentation of Manhattan LSTM model. Yujing worked on the implementation and experimentation of SBERT-based model, and explored dynamic masking and question words. We discussed the results and wrote the report together.

Github repo: https://github.com/zhangyujing/cs230-project

## References

[1] Mueller, J., Thyagarajan, A. (2016). Siamese Recurrent Architectures for Learning Sentence Similarity. AAAI.

[2] Ranasinghe, T., Orasan, C., Mitkov, R. (2019). Semantic Textual Similarity with Siamese Neural Networks. RANLP.

[3] Devlin, J., Chang, M., Lee, K., Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL.

[4] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. ArXiv, abs/1706.03762.

[5] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. ArXiv, abs/1907.11692.

[6] Reimers, N., Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. ArXiv, abs/1908.10084.

[7] Gidaris, S., Singh, P., Komodakis, N. (2018). Unsupervised Representation Learning by Predicting Image Rotations. ArXiv, abs/1803.07728.

[8] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. ArXiv, abs/1909.11942.

[9] Zhou, Meng et al. “Self-supervised Regularization for Text Classification.” Transactions of the Association for Computational Linguistics 9 (2021): 641-656.

[10] Thakur, Nandan et al. “Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks.” NAACL (2021).

[11] Dai, Zihang et al. “Funnel-Transformer: Filtering out Sequential Redundancy for Efficient Language Processing.” ArXiv abs/2006.03236 (2020): n. pag.