Deep learning for text classification with entailment in language pragmatics: a comparison between BERT and LSTM

Rossi Lorenzo (Student ID 982595)

University of Milan, MSc in Data science and Economics

Project for the course of Information Retrieval

27 January 2023

Abstract

Natural language processing (NLP) is a crucial task in many applications, and text classification is a key subproblem. In this paper a comparison between BERT and BiLSTM models, which are the state of the art for many text classification tasks. BERT, a transformer-based model, is effective at capturing long-range dependencies in text, while BiLSTM, a recurrent neural network, is well-suited to handling sequential data. The models are trained on the Stanford Natural Language Inference dataset and evaluate their performances. The results show that BERT outperforms the BiLSTM in both accuracy and loss, despite requiring more computational time. However, both BERT and LSTM represent valuable tools in language models.

Introduction

Text classification is a central task in natural language processing (NLP) that involves assigning a predefined label or category to a piece of text. It has numerous real-world applications, including email filtering, sentiment analysis, and topic modeling. There are various approaches to text classification, including rule-based systems, machine learning algorithms, and deep learning models.

In this paper, a comparison on the effectiveness of two of the most effective NLP techniques for text classification, BERT and Long-Short Term Memory networks, is presented. The aim of the project is to determine which of the two models is the most effective approach for a text classification task. More specifically, pairs of sentences will be fed to the algorithms, which will determine if the two entail, contradict or are neutral. For this purpose, the models have been trained and validated on the Stanford Natural Language Inference dataset. Other factors that may impact the performance of text classification methods will be explained, including the size and quality of the training dataset, the choice of preprocessing techniques, and the hyperparameter settings of the models.

Initially the paper will describe the SNLI dataset, which was used for performing the comparison. The following part will explore the theoretical framework of LSTM, BiLSTM and BERT and the mathematical derivation of the models. Consequently, methodology, data preprocessing (such as tokeninzation and word embedding) and the specific architectures of the models will be explained. Finally, the results will be presented.

Dataset

The Stanford Natural Language Inference (SNLI) corpus is a collection of 570,000 human-written English sentence pairs (named Sentence 1 and Sentence 2) manually labeled for balanced classification with the labels 'entailment', 'contradiction', and 'neutral'. Around 550,000 are reserved for the training purpose, while 10,000 for both validation and testing. In this project training and validation will be used. Preliminary analysis show that the three aforementioned labels are equally distributed in the dataset, being 33.3% each [Fig.1]. Examples of Sentence 1 present an average number of

words superior than Sentence 2 (15 ca. against 5 ca.)[Fig.2]. Moreover, the most common words in both the sentences are similar [Fig.3 and 4]. However, due to computational issues and time, the training set was down-sampled to 30% of the original size.

Theoretical Framework

In this section the two models chosen for the comparison will be presented. The first is the Bidirectional Long-Short Term Memory network (BiLSTM), a variation of the original LSTM. This type of architecture has many advantages in real-world problems, especially in NLP with respect to LSTM. The main reason is that every component of an input sequence has information from both the past and present. For this reason, BiLSTM can produce a more meaningful output, combining LSTM layers from both directions.

The second architecture is the BERT model, which stands for "Bidirectional Encoder Representations from Transformers". It is a pre-trained transformer-based neural network model developed by Google. The model is trained on a large corpus of text data and can then be fine-tuned for a variety of natural language processing tasks, such as question answering and sentiment analysis.

LSTM and BiLSTM

LSTM units are firstly proposed by Hochreiter and Schmidhuber (1997) to overcome gradient vanishing problem. The main idea is to introduce an adaptive gating mechanism, which decides the degree to which LSTM units keep the previous state and memorize the extracted features of the current data input. Then lots of LSTM variants have been proposed. Typically, four components composite the LSTM-based recurrent neural networks:

1. One input gate i_t with corresponding weight matrix W_{xi}, W_{hi}, W_{ci}, b_i ;
2. One forget gate f_t with corresponding weight matrix W_{xf} , W_{hf} , W_{cf} , b_f ;
3. One output gate o_t with corresponding weight matrix W_{xo}, W_{ho}, W_{co}, b_o.

Using current input x_i, the state h_{i-1} that previous step generated, and current state of this cell c_{i-1}, these gates help the network in evaluating whether to take the inputs, forget the memory stored before, and output the state generated later. To conclude, the forget gate determines which relevant information from the prior steps is needed, the input gate decides what relevant information can be added from the current step, and the output gates finalize the next hidden state.

A bidirectional LSTM (BiLSTM) is a type of LSTM that processes input sequences in both forward and backward directions. The forward LSTM layer processes the input sequence from the first element to the last element, while the backward LSTM layer processes the input sequence from the last element to the first element. The components of the memory BiLSTM can be represented using the following equations:

$$i_t = \sigma(W_x i x_t + W_h i h_t - 1 + W_c i c_t - 1 + b_i) \tag{1}$$

$$f_t = \sigma(W_x f x_t + W_h f h t - 1 + W_c f c_t - 1 + b_f) \tag{2}$$

$$c_t = f_t \odot c t - 1 + i_t \circ \tanh(W_x c X_t + W_h c h_t - 1 + b_c) \tag{3}$$

$$o_t = \sigma(W_x o x_t + W_h o h_t - 1 + W_c o c_t + b_o) \tag{4}$$

$$h_t = o_t \odot \tanh(c_t) \tag{5}$$

Where x_t is the input vector at time step t, h_{t-1} is the hidden state at time step $t-1$, c_{t-1} is the cell state at time step $t-1$, i_t, f_t, c_t, and o_t are the input, forget, cell, and output gates respectively,

W and b are the weight matrices and bias vectors, and σ and $\odot$ are the sigmoid and element-wise multiplication (the Hadamard product) functions, respectively. The backward LSTM follows the same equations but with different parameters, e.g. W'_{xi}, W'_{hi}, W'_{ci} and so on.

The final output of the BLSTM is the concatenation of the hidden states from both the forward and backward LSTMs:

$$[\overrightarrow{\mathrm{h_t}}; \overleftarrow{\mathrm{h_t}}] \tag{6}$$

Where [;] is the concatenation operator. Hence, current cell state c_t will be generated by calculating the weighted sum using both previous cell state and current information generated by the cell.

For many sequence modelling tasks, it is beneficial to have access to future as well as past context. Bidirectional LSTM networks extend the unidirectional LSTM networks by introducing a second layer, where the hidden to hidden connections flow in opposite temporal order. The model is therefore able to exploit information both from the past and the future.

BERT

The derivation of the Bert model begins with the transformer architecture, which was introduced in a 2017 paper by Vaswani et al. called "Attention Is All You Need." The transformer architecture uses self-attention mechanisms to weigh the importance of different parts of the input sequence when generating the output.

In traditional sequential models such as Recurrent Neural Networks (RNNs), the representation of a word is dependent on the previous words in the sequence. However, in BERT model, the representations of the words are derived using self-attention mechanism, where each word is dependent on the entire sequence. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training.

BERT is trained using a masked language model objective, where the model is trained to predict a percentage of the input tokens, with the objective of predicting the original token given the context. In addition, Bert is also trained with a next sentence prediction objective: the model is trained to predict if two sentences are consecutive or not, helping the model to understand the relationship between sentences. BERT's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models.

The formulation for a Bert model can be broken down into several parts: the input representation, the transformer layers, and the output layers.

1. Input representation: the input to the model is a sequence of tokens, represented as a matrix of embeddings. Each token is mapped to a high-dimensional vector, known as an embedding.

2. Transformer layers: the transformer layers are the core of the Bert model, and they use self-attention mechanisms to weigh the importance of different parts of the input sequence when generating the output. The BERT base model is composed by 12 layers each one with a hidden size of 768, 12 self-attention heads and it has around 110 million trainable parameters. The self-attention mechanism can be represented mathematically using the following formulas:

   Attention score: the attention score for each token i with respect to all other tokens in the sequence is computed as:

   $$Attention(i, j) = softmax(Q_i * K_j^T / \sqrt{d}) \tag{7}$$

   where Q, K and V are the query, key and value matrices respectively, d is the dimension of the key and value matrices, and the dot product is the matrix multiplication.

   Attention output: the attention output for each token i is computed as a weighted sum of the value matrices for all tokens in the sequence:

$$Output(i) = \sum_{j=1}^{n} (Attention(i,j) * V_j) \tag{8}$$

3. Output layers: the output layers take the output from the transformer layers and use it to predict the final output for the specific task.

$$y = softmax(W_{out} * Output(i) + b_{out}) \tag{9}$$

Where y is the prediction, W_{out} and b_{out} are the weight and bias of the output layer respectively.

In case of classification, the input text is passed through the pre-trained BERT layers, which encode the input text into a set of hidden states. These hidden states are then passed through the fine-tuned output layer, which is a fully connected (dense) with a softmax activation function (as in Eq. 11) that produces a probability for each class. The final prediction is obtained by taking the argmax of the output. The network uses backpropagation algorithm to calculate the gradients of the loss function with respect to the weights in the output layer, and then update the weights using an optimizer. During the training process, a cross-entropy loss function is typically used to measure the difference between the predicted probabilities and the true labels.

Methodology

BiLSTM

Data preprocessing: first, all the words in the sentences are tokenized and inserted in a vocabulary. Then, embedding is performed by pre-trained GloVe word vectors. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. For this task, pre-trained word vectors downloaded from the GloVe website were applied to the corpus. More specifically, the chosen package corresponds to 6 billion tokens, a vocabulary of 400,000 words and 300-dimensional vectors.

Model tuning: the BiLSTM is composed as following: an input layer with size 300 (which refers to the dimensionality of the word vectors), two hidden LSTM layers (one forward and one backward) each containing 128 cell memory blocks, followed by two unidirectional layers of 64 and 32 memory blocks respectively. The last layer is the output layer, with the size equal to the labels. Each hidden layer is activated with the ReLU function, while for the output layer, the softmax activation function is applied, as it is common for multilabel classification tasks. Moreover, each hidden layers presents a dropout rate of 0.2, which has the effect of reducing overfitting and improving model performance. Dropout is a regularization method where input and recurrent connections are probabilistically excluded from activation and weight updates while training the network.

In this case, the network will use the memory cell to elaborate the sequences of the sentence pairs and determine whether the two entail, contradict or are neutral. The classifier takes the hidden state $h*$ of the network as input:

$$\hat{p}(y|S) = softmax(W(S)h * +b(S)) \tag{10}$$

$$\hat{y} = argmax(y\hat{p}(y|S)) \tag{11}$$

The network is trained with 10 epochs and a batch size of 32, using an Adam optimizer with learning rate equal to 0.001.

BERT

Data preprocessing: tokenization is performed with the base-uncased BERT tokenizer. Sentence pairs are packed together into a single sequence, differentiated with a special token ([SEP]) used for

separation. The first and last token of every sequence is a special classification token ([CLS]) and the final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. The combined length of the sequence of sentences is set to a maximum of 512 tokens. A Segment ID is assigned to every token indicating whether it belongs to Sentence 1 or Sentence 2. Finally embedding is done by transforming tokens into vector representations (tensors). These vector representations are then passed through the BERT model's transformer layers, which use self-attention mechanisms to encode the input text into a set of hidden states.

Model tuning: the original BERT paper recommends a batch size of 16-32, 3 to 4 epochs, a learning rate from 3e-4 to 2e-05, even if it depends on the task. For this specific project, the pre-trained BertForSequenceClassification model was used. It is composed by 12 layers with 768 features each, like the base BERT model, but it is designed specifically for text classification tasks. The training is done with batch size of 32 sequences (32 sequences * 512 tokens = 16,384 tokens/batch) for 125,000 steps, which corresponds to 5 epochs, even if the BERT base model was trained on just 3 epochs. Adam optimizer is used, with learning rate of 2e-5, $\beta 1 = 0.9$ and $\beta 2 = 0.999$ parameters.

Results

With 10 epochs, the BiLSTM reaches an accuracy of 82,2% in training data and 81,8% in the validation data [Tab.1]. Each epoch took more than 4 minutes to be computed, with a total computation time of more than 40 minutes. It is important to notice that the accuracy kept growing during training and further tests could be done by incrementing the number of epochs in order to evaluate if higher values of accuracy could be reached.

On the other hand, BERT reaches the best results after just two epochs [Tab.2], scoring a 96% accuracy on the training set and 88,21% on the validation set, before overfitting. Being a heavier model, computation time for each epoch is significantly higher than in the BiLSTM (almost 12 minutes each, with a total computation time of one hour for 5 epochs), but the highest accuracy was met after 24 minutes, i.e. at the second epoch. However, BERT models are usually trained with a very low number of epochs (3-4 as literature recommends) compared to LSTMs which, depending on the task, may require a much higher number to converge to the highest accuracy.

A part from computation time, which depends on the weight of the model and the number of epochs selected by the user, it is safe to assume that for this specific task BERT outperforms BiLSTM in both accuracy and loss. However, it is important to take into consideration that performances of BiLSTM may increment by changing the value of the hyperparameters, such as the epoch number, the batch size, or increasing the number or the size of the hidden layers and changing the learning rate. In this case, an higher number of epochs could have lead to a higher accuracy. Hyperparameter tuning can be applied to BERT too, and its performances may vary as well. In the end, both models manage to perform well in handling text classification tasks but BERT has a 10% higher accuracy score and lower error rates.

Conclusion

In this paper the performances of BERT and BiLSTM, two of the models which represent the state of the art for many NLP studies, have been compared for a text classification task applied to the SNLI dataset. The aim was to find which one of the two models could achieve better results in classifying correctly the sentence pairs according to their actual label and determine whether the pairs contained in the dataset entail or contradict. Results have showed that BERT outperforms BiLSTM in terms of accuracy and loss, despite being a computationally heavier and more time consuming algorithm. BiLSTM could have achieved better results but it would have required more computation time where BERT was able to reach the highest accuracy with few epochs. Further works could provide deeper insights on how to fine tune and optimize both models even more in order to achieve better results without increasing too much computation costs.

Bibliography

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* In Proceedings of the 2019

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. *A large annotated corpus for learning natural language inference.* In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Sepp Hochreiter and Jürgen Schmidhuber, *Long short-term memory*, Neural Comput. 9 (1997) 1735–80

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. 2017. *Attention is All you Need.* Advances in Neural Information Processing Systems 30 (NIPS 2017)

Appendix A

Tables

Table 1: Performances of BiLSTM			
Epoch	Training (Loss, Acc)	Validation (Loss, Acc)	Time
1	0.6957, 0.7065	0.5744, 0.7650	00:04:13.41
2	0.5877, 0.7635	0.5236, 0.7906	00:04:11.17
3	0.5478, 0.7830	0.5000, 0.8029	00:04:10.86
4	0.5231, 0.7944	0.4888, 0.8125	00:04:11.08
5	0.5065, 0.8024	0.4784, 0.8153	00:04:11.35
6	0.4938, 0.8079	0.4709, 0.8155	00:04:11.82
7	0.4839, 0.8124	0.4730, 0.8148	00:04:11.67
8	0.4751, 0.8166	0.4730, 0.8177	00:04:12.64
9	0.4667, 0.8206	0.4710, 0.8180	00:04:11.38
10	0.4604, 0.8234	0.4666, 0.8181	00:04:11.50

Table 2: Performances of BERT			
Epoch	Training (Loss, Acc)	Validation (Loss, Acc)	Time
1	0.4529, 0.8244	0.3341, 0.8794	00:11:55.76
2	0.2974, 0.8930	0.3113, 0.8871	00:11:52.97
3	0.2133, 0.9260	0.3551, 0.8842	00:11:52.95
4	0.1541, 0.9483	0.3829, 0.8852	00:11:52.95
5	0.1195, 0.9599	0.4148, 0.8827	00:11:53.17

Images

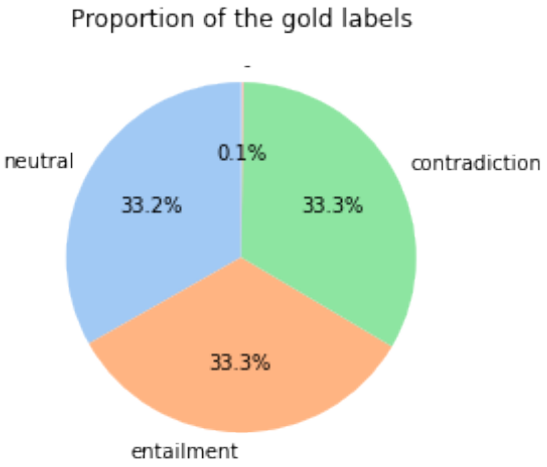

Figure 1: Fig.1 Distribution of labels

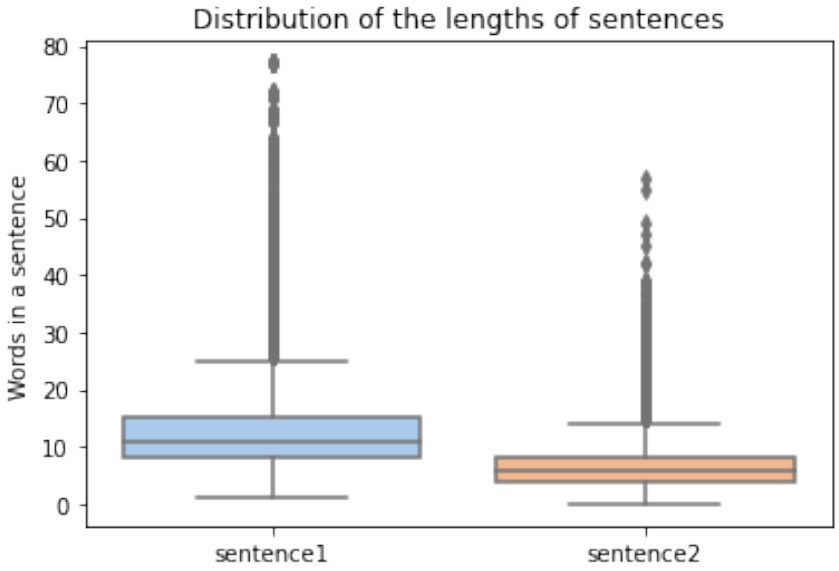

Figure 2: Fig.2 Average length of sentences

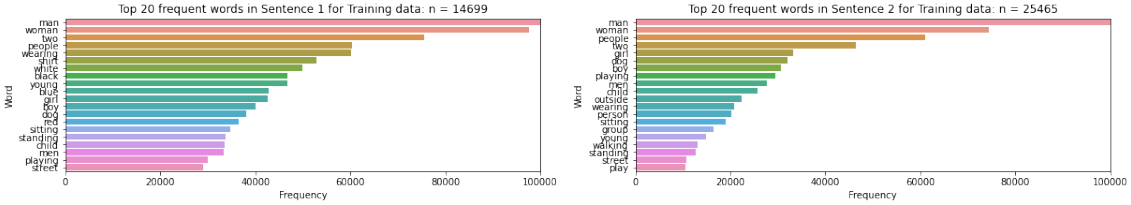

Figure 3: Fig.3 Count of the 20 most common words in training set

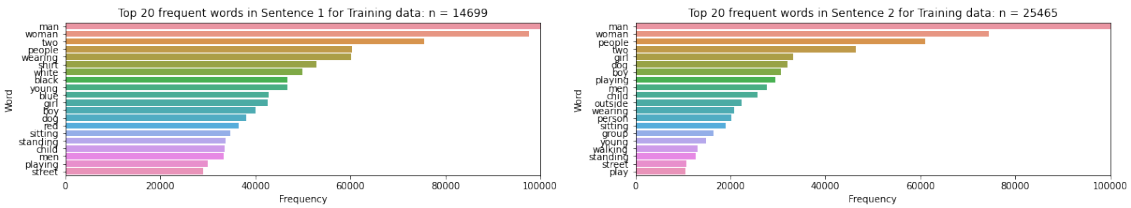

Figure 4: Fig.4 Count of the 20 most common words in validation set