Project : The Stanford Natural Language Inference (SNLI) Corpus

Antonin Aubry, Astrid Benamou et Pierric Méléard

April 2022

GitHub Repository : https://github.com/AstridBenamou/NLP-Project-SNLI-Corpus

1 Introduction

Natural Language Processing can be divided into two subsets: Natural Language Generation and Natural Language Understanding.

Natural Language Inference (NLI) or Recognizing Textual Entailment (RTE) is an increasingly important task in natural language understanding. Its aim is to determine the inference relation between a hypothesis and a premise, which are usually short texts.

For these tasks we need a dataset large enough to train models using neural networks with different methods. The SNLI corpus was created to achieve this goal. It was developed as a benchmark for NLI or to be used as a resource for developing NLP models of any kind.

2 Presentation of the corpus and the methods used for its creation

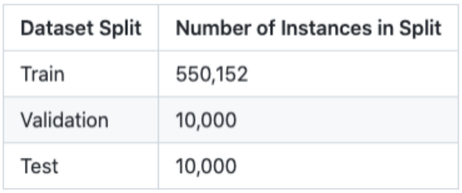
2.1 Presentation of the corpus

The SNLI corpus is composed of 570k human-written English sentence pairs (premise and hypothesis) manually labeled for balanced classification. There have been some high quality datasets before but rarely exceeding the size of a thousand lines or produced and labelled using automatic methods.

Data fields:

- Hypothesis: String that may be true or false
- Premise: String used to determine the truthfulness of the hypothesis
- The three labels correspond to the relation between the 2 sentences, their values are:
  0: Entailment – The hypothesis entails the premise
  1: Neutral – The premise and hypothesis neither entail nor contradict each other
  2: Contradiction – The hypothesis contradicts the premise

Train/Test/Validation split:



Dataset Split	Number of Instances in Split
Train	550,152
Validation	10,000
Test	10,000

Figure 1: Size of our datasets from SNLI corpus.

The corpus was made to adress the issue of size and quality of labels in datasets for NLI, as well as resolve the issues of indeterminacies of events and entity coreference. This last problem occurs when the situation of the premise or hypothesis is not precise enough to determine a single label between the 3.

2.2 The construction of the corpus

To address the three issues of size, quanlity and indeterminacy mentionned, the process of collecting data was adapted when asking the workers to write the pairs of premise and hypothesis.

The 2,500 workers were given a premise scene description to narrow the possibilities to a scenario and then asked to write a hypothesis for each of the three labels: entailment, neutral and contradiction. The premises' scene description were taken from captions in a corpus of 30k images. Nevertheless, the labels must be recovered without needing access to these images.

This step helps to balance out the number of pairs in the corpus for each label.

More specifically, they ensured the quality of the corpus by:

1. Taking examples in specific scenarios and asking the workers to view the situation from a certain perspective to control event and entity coreference.
2. Leaving some freedom to produce entirely new sentences to enrich examples.
3. Selecting a subset of those sentences and running them through a validation task.
4. Providing guidance on the task and precising a maximal length and the desired level of complexity for the pairs of sentences.

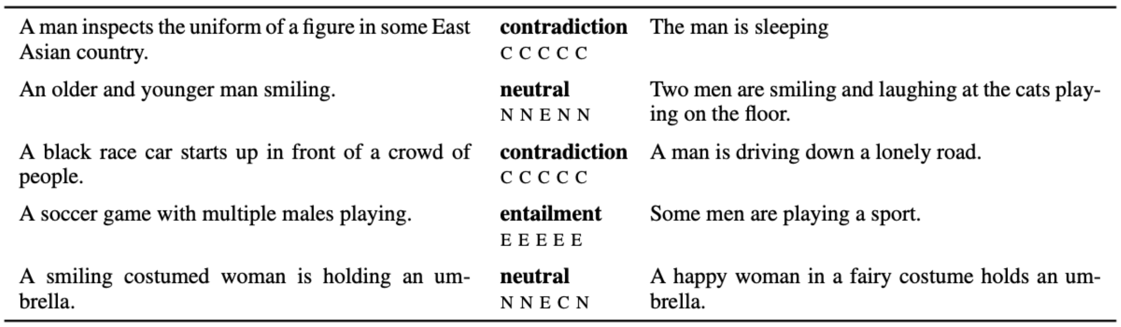
They observed statistics on the collected data and even did an additional validation step :

30 trusted workers were selected for this task. Its aim was to measure the quality of the collected corpus. They ran this task on 10% of the data. Showing 4 workers the pairs of sentences 5 by 5 and asking them to assign a label. Each pair then had 5 assigned labels including the one given by the original author.

A pair was assigned its 'golden label' if at least three of the five annotators had picked it. Otherwise it was assigned a placeholder label '-' and was removed during the NLI tasks performed on the corpus further on.

The rate of agreement in this validation was extremely high, only 2% of the validation data was not assigned a gold label. This indicated the pairs of sentences were pretty clear, providing evidence of the high quality of the overall corpus. The level of english and variety of formulations in the overall corpus is also a likeable.

An improvement path that can still be taken would be on the punctuation and capitalization.



A man inspects the uniform of a figure in some East Asian country.	**contradiction** C C C C C	The man is sleeping
An older and younger man smiling.	**neutral** N N E N N	Two men are smiling and laughing at the cats playing on the floor.
A black race car starts up in front of a crowd of people.	**contradiction** C C C C C	A man is driving down a lonely road.
A soccer game with multiple males playing.	**entailment** E E E E E	Some men are playing a sport.
A smiling costumed woman is holding an umbrella.	**neutral** N N E C N	A happy woman in a fairy costume holds an umbrella.

Figure 2: Examples from the annotated corpus

The corpus is available for free, the aim of the project was to accelerate inovation in NLI models by making it accessible to researchers.

This will allow us to test different models on the SNLI corpus and others, such as the Sentences Involving Compositional Knowledge (SICK) dataset. The SICK corpus is an appreciated corpus but much smaller than the SNLI and encounters some problem. It will be a good tool to compare the performances of our models.

3 Application to models using the SNLI corpus : BI-LSTM model

3.1 The Bidirectional-Long short-term memory model

Recurrent Neural Networks are capable of treating sequences of the data and use data patterns to give the prediction. They are widely used in Natural Language Processing. They use feedback loops which allows the data to be shared to several nodes in order to make a prediction based on the whole shared memory.

A **Long short-term memory (LSTM)** models are a special kind of recurrent neural network. Unlike standard feedforward neural networks, LSTM has feedback connections. It is perfect to use in our NLI task as it is capable not only to process single point data but also entire sequences of data, which is necessary for our application to the SNLI corpus and for most NLP tasks. Today, it is widely used and is even said to be the most cited neural network.

In regular RNN, the problem frequently occurs when connecting previous information to new information. LSTMs were introduced to avoid this long-term dependency problem.

In regular LSTM, information can only flow in one direction (forward or backwards) which is why we introduce **Bidirectional-LSTM** models. They are sequence processing models that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. In this process any neural network has the sequence information in both directions backwards (future to past) or forward(past to future).

This kind of network can be used in text classification, speech recognition and forecasting models. In our project we used it for our NLI task.

3.2 Methodology

- The dataset was obtained from *torchtext.datasets*. This provides a wide range of functionality as the iterators can be built with just one line of code.
- The vocabulary is built using the $build_vocab$ function in torchtext, and this returns the sentence converted into a matrix.
- The model has an embedding layer, followed by a Relu and a dropout layer, which was added to prevent overfitting. This is followed by a Bi-LSTM module with 3 hidden layers. The output from the LSTM was fed into a sequential dense network with 3 hidden layers for the classification task.
- The encoding for the premise and the hypothesis was generated separately and then finally concatenated and fed into the dense network for classification.
- Parameters like batch size of the dataset, the embedding dimension, epochs for training were varied.
- The optimizer used was Adam, but we could have also RMSProp, Adagrad or SGD like in the notebooks of the course.
- Learning rates were also varied and the results have been reported.
- The loss function used was cross entropy, and the learning curves were plotted for every trial and some of them have been shown in the observations below.

- Pre-trained embeddings like GloVE could also have been used in the model, and the $requres_grad$ parameter set to False. But this requires good bandwidth for downloading and uploading the vectors every time for the trials. In this project the initial embeddings were generated by the vocabulary generator in the torchtext library.
- As the training progresses, the embedding weights for the input vectors are learned. The hope is that, say if the premise and the hypothesis are entailed, then the embeddings get close together, and vice-versa for contradiction.
- The twi vectors of the premise and the hypothesis are concatenated and then passed into the classification dense network. Other methods like addition and substraction will lead to skewed and sparse vectors respectively for entailed sentences. Thus, keeping in mind that sparse vectors would not help the network to learn, concatenation was used here.

3.3 Results and observations

3.3.1 Trial 1 : Optimal Epochs

The Bi-LSTM model was trained for 50 epchs, and the learning curves for both training and validation was plotted (cf. Fig. 3). This was done to inspect if the model was overfitting in any manner, and the results were interesting. From this it was found that the validation loss curve starts deviating at around 15 epochs, indicating overfitting (cf. Fig. 4). The curve for the model trained with 15 epochs has also been reported below.

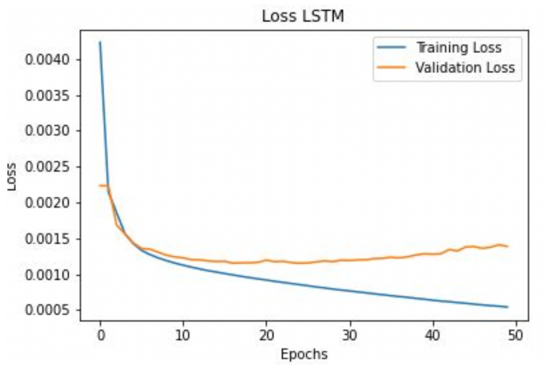


Figure 3: Learning curve of the Bi-LSTM model for 50 epochs

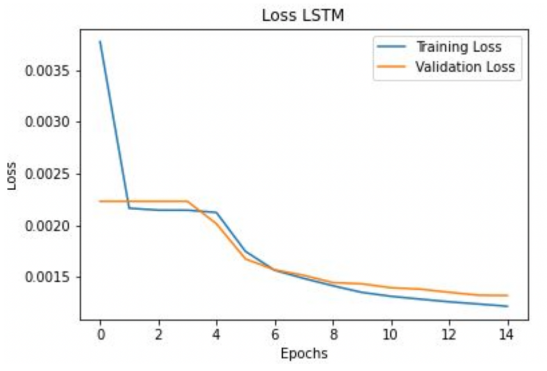


Figure 4: Learning curve of the Bi-LSTM model for 15 epochs

3.3.2 Trial 2 : Optimal learning rate and optimizer

The learning rate varied from 0.001 to 0.005 in steps of 0.001. The results have been plotted bellow (cf. Fig. 5).

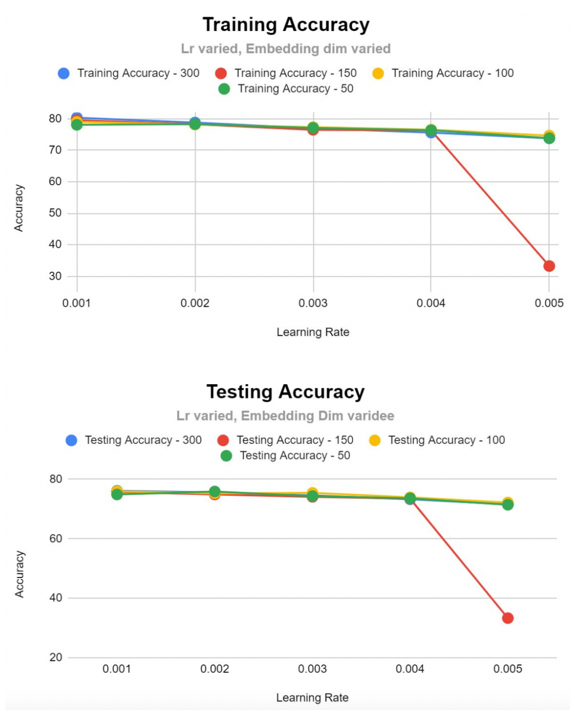


Figure 5: Training and testing accuracy for different number of total epochs and different learning rates

One thing to notice is that, at a learning rate of 0.005 and an embedding dimension of 150, we can notice a dip in the accuracy to a low 33%. Overall, we can see that a rate of 0.001 and an embedding of size 300 works best for the Bi-LSTM.

3.3.3 Trial 3 : Optimal Embedding dimension

The embedding dimensions have been varied as 300, 150, 100 and 50. The results are shown bellow (cf. Fig. 6)

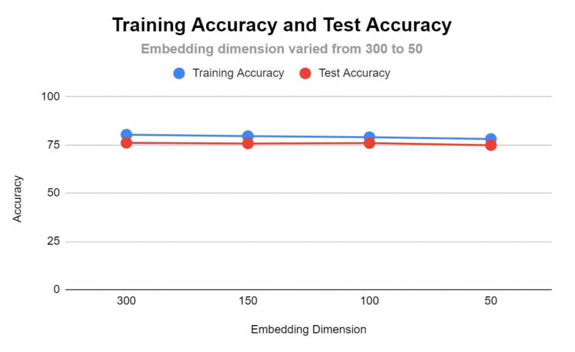


Figure 6: Training and testing accuracy for different embedding dimensions

From the plots, we can see that the embedding size of 300 is the best in terms of training and testing accuracy.

3.3.4 Trial 4 : Optimal Batch size

The following graph shows the variation of the batch size with respect to the training and the testing accuracies (cf. Fig 7), the other parameters being fixed at their optimal value determined by the previous trials.

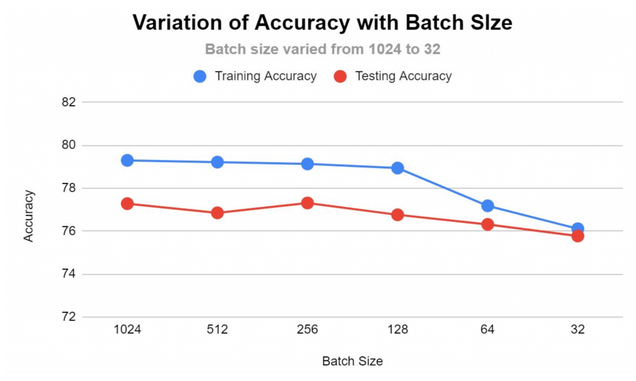


Figure 7: Variation of accuracy with batch size

We can see from the graph that batch size of 1024 produces the best accuracies. It constantly keeps decreasing as the batch size decreases. The line goes more steeper as the batch size decreases.

Bellow we can see the training time for one epoch for different batch sizes (cf Fig. 8). The optimal batch size with regards to training time seems to be 256 or 128. Batch size of 64 and 32 show rise in the training time.

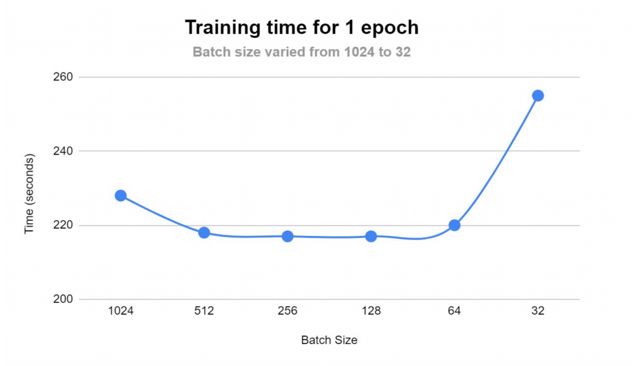


Figure 8: Variation of the training time with batch size for 1 epoch

On the whole, we can see that a batch size of 1024 or 512 is the most appropriate with regards to accuracy and training time (slightly more than 256 and 128).

3.4 Conclusion

Parameters chosen :

- Batch size : 512
- Embedding size : 300
- Dropout ratio : 0.2
- Hidden layer dimension in Bi-LSTM : 256
- Optimizer and learning rate : Adam & 0.001
- Bidirectional : True
- Vector Preparation : Concatenation

The final loss after training was found out to be 0.001253 and the training and validation accuracy came out to be 80.3326 and 76.11257 respectively. The final test accuracy was reported as 75.5904.

References

[1] Website : https://nlp.stanford.edu/projects/snli/

[2] Article : Samuel R. Bowman, A large annotated corpus for learning natural language inference. (2015)

[3] References for BI-LSTM : https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/ https://github.com/pytorch/pytorch/blob/master/torch/nn/modules/rnn.py