Learning sentence representations from natural language inference data

University of Amsterdam, ATCS: Practical I

April 2023

1 Introduction

The first practical of the Advanced Topics in Computational Semantics course concerns learning general-purpose sentence representations in the *natural language inference* (NLI) task. The goal of this practical is threefold:

- to *implement* four neural models to classify sentence pairs based on their relation;
- to *train* these models using the *Stanford Natural Language Inference* (SNLI) corpus (Bowman et al., 2015);
- and to *evaluate* the trained models using the SentEval framework (Conneau and Kiela, 2018).

NLI is the task of classifying entailment or contradiction relationships between premises and hypotheses, such as the following:

Premise Bob is in his room, but because of the thunder and lightning outside, he cannot sleep.
Hypothesis 1 Bob is awake.
Hypothesis 2 It is sunny outside.
Hypothesis 3 Bob is lying in his bed.

While the first hypothesis follows from the premise, indicated by the alignment of 'cannot sleep' and 'awake', the second hypothesis contradicts the premise, as can be seen from the alignment of 'sunny' and 'thunder and lightning' and recognizing their incompatibility. The third hypothesis is not necessarily entailed by the premise, and neither is contradicted. Therefore, its relation to the premise is considered to be neutral.

For a model to recognize textual entailments, it has to reason about semantic relationships within sentences. Hence, a thorough understanding of natural language is required which can be transferred to other tasks involving natural language. In this assignment, we focus on pretraining a sentence encoder on NLI, and afterwards evaluate its sentence embeddings on a variety of natural language tasks.

2 Assignment task

Your task in this assignment is to replicate some of the results reported by Conneau et al. (2017). In the following, we will first introduce the dataset, the model architectures and evaluation to use.

2.1 SNLI corpus

Throughout this assignment, we will use the SNLI corpus.[1] This corpus (version 1.0) is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of NLI that is also known as recognizing textual entailment.

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

2.2 Model architectures

The general architecture proposed by Conneau et al. (2017) consists of a sentence encoder and a classifier. The premise and hypothesis are embedded by the same sentence encoder into a fixed sized feature vector, u and v respectively. The two feature vectors are combined by concatenating both additionally with their absolute difference $|u - v|$ and element-wise product $u * v$. The resulting features are processed by a small MLP with a final classification layer for distinguishing the entailment relationships. An overview of the architecture is shown in Figure 1.

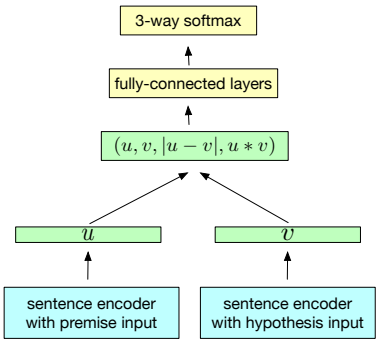
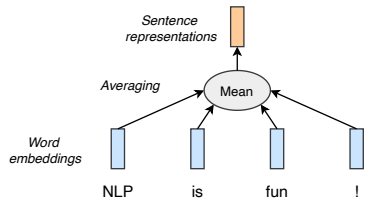


Figure 1: General architecture for sentence classification Conneau et al. (2017)

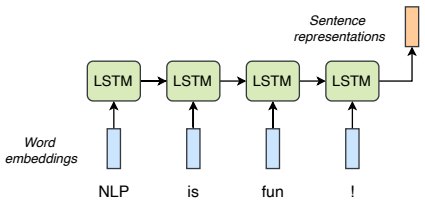
For the sentence encoder blocks, your task is to implement the following four models:

1. Baseline: averaging word embeddings to obtain sentence representations.
2. Unidirectional LSTM applied on the word embeddings, where the last hidden state is considered as sentence representation (see Section 3.2.1 of the paper);
3. Simple bidirectional LSTM (BiLSTM), where the last hidden state of forward and backward layers are concatenated as the sentence representations;
4. BiLSTM with max pooling applied to the concatenation of word-level hidden states from both directions to retrieve sentence representations (see Section 3.2.2).

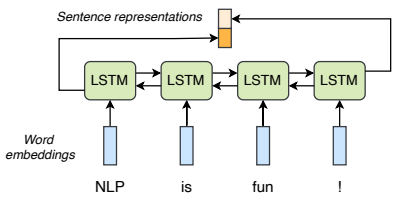
A visualization of the four models can be found below. For details, see Section 3.2 of Conneau et al. (2017).



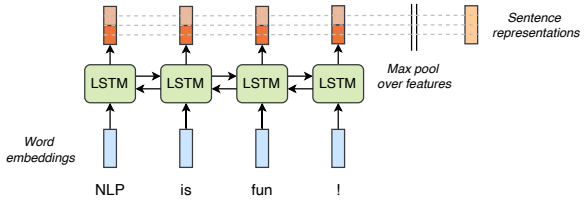
(a) Word embeddings



(b) Unidirectional LSTM



(c) Bidirectional LSTM



(d) Bidirectional LSTM with max pooling

Figure 2: Visualization of the four sentence encoders.

For each of the models, you should use the 300-dimensional GloVe[2] word embeddings trained on Common Crawl 840B. The word embeddings should be fixed during training. To achieve comparable results and avoid extensive hyperparameter search, we advise you to use the hyperparameter settings as described in Section 3.3 of Conneau et al. (2017).

Note
In Section 3.3 of Conneau et al. (2017), a weight decay of 0.99 is reported. However, this should not be interpreted as the classical weight decay for regularization, but is a learning rate decay which is applied after each epoch.

2.3 Evaluation

The SNLI dataset consists of three splits, (1) *train*: which you should use to train your models, (2) *dev*: which you can use for hyperparameter tuning, and (3) *test*: which you should use to evaluate the models. You should evaluate your models on the SNLI task and report the accuracy on the validation and test set.

In addition, you need to use SentEval[3] (Conneau and Kiela, 2018) to evaluate the sentence representations obtained from each of the models. SentEval, the Facebook evaluation toolkit for sentence embeddings, is a library for evaluating the quality of sentence embeddings by applying them on a broad and diverse set of downstream tasks called 'transfer' tasks. Report both the macro and micro accuracy metrics as in Table 3 in Section 5 of the paper. See the first paragraph of Section 5 for an explanation of the two metrics.

3 Practical Matters

3.1 Recommended Reading

Before starting to work on the practical, read these papers thoroughly:

- 'A large annotated corpus for learning natural language inference' of Bowman et al. (2015)
- 'Supervised learning of universal sentence representations from natural language inference data' of Conneau et al. (2017)

You are very much encouraged to have group discussions about these papers to fully understand the task and the models.

3.2 Implementation Tips

In this assignment, we ask you to implement the training framework in Pytorch. Below, we have summarized a few good practices that you can use:

1. Loading the SNLI corpus can be done either from file[4] or from HuggingFace[5]. As preprocessing steps, apply lowercasing and tokenization. You can choose the tokenizer of your choice, but we recommend to use `spaCy` or `nltk.word_tokenize`.
2. Be careful to initialize your `nn.Embedding` module *after* aligning the GloVe vocabulary with the dataset, using the dataset's vocabulary.
3. Use tensorboard to visualize and monitor the training process of your models. Tensorboard support is integrated into PyTorch.[6]

[2]GloVe embeddings: https://nlp.stanford.edu/projects/glove/.
[3]SentEval's Github repository: https://github.com/facebookresearch/SentEval.
[4]SNLI https://nlp.stanford.edu/projects/snli/
[5]HuggingFace datasets https://huggingface.co/datasets/snli
[6]A tutorial can be found at https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html

4. You are allowed to use PyTorch Lightning[7] if you are already familiar with it. Lightning reduces the necessary code for logging, saving/loading a model, and more.

5. We recommend using a small portion of the data for debugging the framework. Ensure that the implementation works using this subset before training on the original dataset.

6. Try to build up your code as modular as possible so that it is easy to be extended to new models and/or tasks. For instance, you should have at least two separate modules for training and inference. The interface for training can look similar to:

   ```
   python train.py <model_type> <checkpoint_path> ...
   ```

   Additional arguments could include hyperparameters, e.g. the learning rate. The evaluation in this assignment includes testing a trained model on the SNLI test dataset and evaluating its sentence encodings with SentEval. A possible interface is as follows:

   ```
   python eval.py <checkpoint_path> ...
   ```

7. Online, we share a SentEval tutorial in which GloVe word vectors are averaged to form sentence representations.[8] There are examples on the SentEval repository as well.[9]

8. Take care of handling padding symbols in *all* models. For LSTM-based encoders, the functions `torch.nn.utils.rnn.pad_packed_sequence` and `torch.nn.utils.rnn.pack_padded_sequence` can be helpful.

9. Use the Lisa cluster to train your model. Try to run your training early, as your jobs might not start instantaneously due to queues, and your accounts have a limited number of jobs that can be run in parallel. However, sufficient hardware support should be provided to train all of the models in a reasonably short time.

10. Some general advice on debugging can be found on the site of the Deep Learning course[10].

Note
The code of Conneau et al. (2017) is available through Github: `https://github.com/ihsgnef/InferSent-1`. While you are allowed to look at it for inspiration and implementation details, we strongly encourage you to implement your own training framework. This is especially important concerning your final project, for which you will use your code as a starting point. Note that the standard plagiarism rules of UvA apply for this assignment.

3.3 Deliverables

Your submission should consist of the following components:

Code Please include all python files in your submission which are necessary to train and evaluate any of the models described in Section 2.

Documentation Provide a github ReadMe file describing the package/installation requirements, a guide to train and evaluate a model, and the structure of your code.

[7] PyTorch Lightning is taught in the Deep Learning course, with a short tutorial here: `https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial5/Inception_ResNet_DenseNet.html#PyTorch-Lightning`

[8] Tutorial: `https://uva-slpl.github.io/ull/resources/practicals/practical3/senteval_example.ipynb`.

[9] `https://github.com/facebookresearch/SentEval/tree/main/examples`

[10] `https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/guide3/Debugging_PyTorch.html`

Pretrained models Upload the final checkpoint file (i.e. checkpoint which was used for testing) and Tensorboard of the training run for each model to a (free) file sharing service such as GoogleDrive, and include the download links in your ReadMe. Please *do not* include the pretrained models in your zip file submitted to Canvas, as this causes issue during upload and download. Additionally, make sure to *not* include the GloVe embedding vectors in the checkpoints, as it significantly increases the file size.

Results and error analysis Prepare a Jupyter notebook, where you load a trained model and demonstrate how it works for different examples. For instance, it should be possible to specify a new premise and hypothesis as strings, and predict their entailment with one of your pretrained models. Besides the demonstration, you should also include an overview of your results in the notebook (both SNLI and SentEval), and do some error analysis. You can look at questions like: Why is model A performing better than model B? Where do the models fail? What information does the sentence embedding represent, and what information might be lost? Additional points will be awarded for further research questions that you identify and answer yourself. In particular, we are looking for a clear motivation of your research questions, novelty and clarity of presentation. You can also include screenshots from tensorboard if suitable. Try to find good ways to visualize and present your findings.

You should do this practical individually and submit a compressed zip of the deliverables with the title **ATCS-Practical1-FullName** to Canvas. The submission deadline for this assignment is **23:59 on Friday, 21 April**.

References

S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

A. Conneau and D. Kiela. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.