Assignment 4: ELMO

Course: Introduction to Natural Language Processing

Deadline: 20 April | 23:59

General Instructions

1. The assignment must be implemented in Python.
2. The assignment must be done using PyTorch. Usage of other frameworks will not be accepted.
3. Submitted assignment must be your original work. Please do not copy any part from any source including your friends, seniors, and/or the internet. If any such attempt is caught, then serious actions including an F grade in the course is possible.
4. A single .zip file needs to be uploaded to the Moodle Course Portal.
5. Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.
6. Please start early since no extension to the announced deadline would be possible.

1 ELMo: Deep Contextualized Word Representations

Early architectures such as word2vec, GloVe for obtaining distributional representation of text made it easier to obtain a meaningful, low dimensional

representation of a word for use in semantic tasks. These architectures provide a single representation for a word, which can be seen as an aggregated representation based on all possible contexts that word has appeared in the corpus used for training. But considering the complexity and ambiguity in language, we need more sophisticated representations of text that take in the context of a word in a given sentence. The representation of the same word would thus wary depending on what it means in its context. This is where some of the earlier works in developing contextual embeddings like ELMo, CoVe come in. In the previous assignment, you built simple, non-contextual word representation to attempt the downstream task. Now, you'll advance on the quality of embeddings used by building an ELMo architecture by yourselves, and testing its efficacy on the same downstream task. ELMo looks at building a layered representation of a word through stacked Bi-LSTM layers, separately weighing in syntactic and semantic representations.

2 Implementation and Training

2.1 Architecture

Build an ELMo architecture from scratch using PyTorch. Do not worry about the character convolutional layer here. You may use an existing pretrained non-contextual word representation like word2vec for the input embedding similar or you may add your own embedding layer. The core implementation involves adding a stacked Bi-LSTM - each of which gives the embedding for a word in a sentence, and a trainable parameter for weighing the word embeddings obtained at each layer of the ELMo network. You are expected to use only 2 layers of Bi-LSTM in the stack. [15 Marks]

2.2 Model Pre-training

Learn the ELMo embeddings on the bidirectional language modeling objective by making use of the train split of the given dataset as an unlabeled corpus. This involves training on the word prediction task in forward and backward directions for training the Bi-LSTMs in the network. [20 Marks]

2.3 Downstream Task

You will be training the ELMo architecture on a 4-way classification task using the AG News Classification Dataset. Make use of this dataset to build a pipeline for training and evaluating your model on the classification task. [15 Marks]

3 Corpus

Please train your model on the given csv files link here: Link to the corpus(News Classification Dataset)

Note that you have to only use the Description column of the train.csv for training your word vectors. You have to use the label/index column for the downstream classification task. This is the same dataset as the previous assignment.

4 Hyperparameter tuning

For combining word representations across the different layers of the biLSTM in the downstream task, we make use of λs.

$$\hat{E} = \sum_{i=0}^{2} (e_i \cdot \lambda_i)$$

Where $\hat{E}$ is the final contextual word embedding. You have to try out the following hyperparameter settings.

4.1 Trainable λs

In this setting, you have to train and find the best λs. [5 Marks]

4.2 Frozen λs

In this setting, you have to randomly initialize and freeze the λs. [5 Marks]

4.3 Learnable Function

In this setting, you have to learn a function to combine the word representations across layers to build the final contextual word embedding.
$\hat{E} = f(e_0, e_1, e_2)$ [5 Marks]

5 Analysis

Write a comprehensive analysis of the performance of your ELMo model in the pretraining process and the downstream task in a properly compiled report. In this report, ensure you compare the performance of ELMo embeddings with those of SVD and Word2Vec on the downstream task (the same as the previous assignment). Compare and analyze which of these word vectorization methods (Word2Vec, SVD, ELMo) performs better by using performance metrics such as accuracy, F1 score, precision, recall, and the confusion matrix on both the train and test sets. Write a detailed report on why one technique might perform better than the others. In your report, be sure to include which hyperparameter setting performs the best and why? Report the performance metrics and include the confusion matrices for all the hyperparameter settings. [15 Marks]

Submission Format

Zip the following files into one archive and submit it through the Moodle course portal. The filename should be `<roll number>_assignment3.zip`, for example, `2021114005_assignment4.zip`.

- **Source Code**
  - `ELMO.py`:Train the biLSTM on the language modelling task..
  - `classification.py`: Train the classifier for the downstream task using the word representations from the biLSTM.
- **Pretrained Models**
  - `bilstm.pt`: Saved biLSTM model.

  - `classifier.pt`: Saved classifier model used for the downstream task.

- **Report (PDF)**
  - Hyperparameters used to train the model(s).
  - Corresponding graphs and evaluation metrics
  - Your analysis of the results.

- **README**
  - Instructions on how to execute the file, load the pretrained model, implementation assumptions etc.

Ensure that all necessary files are included in the zip archive. Please upload the saved models to onedrive and include the link in the readme.

Grading

Evaluation will be individual and based on your viva, report, and code review. During your evaluation, you will be expected to walk us through your code and explain your results. You will be graded based on the correctness of your code, accuracy of your results, and the quality of the code.

ELMO Architecture: 15 marks

ELMO Pre-training: 20 marks

Downstream Task: 15 marks

Hyperparameter Tuning: 15 marks

Analysis: 15 marks

Viva during Evaluation: 20 marks

Resources

1. Deep Contextualized Word Representations

2. Exploring the Limits of Language Modelling
3. Bidirectional Language Modelling using LSTMs
4. An Introduction to Bidirectional Language Modelling
5. You can also refer to other resources, including lecture slides!