Assignment 4: ELMO

Course: Introduction to Natural Language Processing

Deadline: 29 March — 23:59

1 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. A single .zip file needs to be uploaded to the Moodle Course Portal.
4. Please start early since no extension to the announced deadline would be possible.

2 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, GloVe 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 different downstream task. ELMo looks at building a layered representation of a word through stacked Bi-LSTM layers, separately weighing in syntactic and semantic representations.

3 Implementation and Training

3.1 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. **[5 Marks]**

3.2 Model Pre-training

Train the ELMO using Brown Corpus given in the previous assignment. This involves training on the word prediction task in forward and backward directions for training the Bi-LSTMs in the network. **[14 Marks]**

3.3 Downstream Task

For the downstream task, you will be using AG News Classification dataset. Freeze the weights of pretrained ELMO model and use the embeddings from this freezed pretrained ELMO model for the downstream task. You are free to use any kind of RNN for the downstream task. **[10 Marks]**

4 Corpus

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

Note that you have to use the `Description` column of the train.csv as input and `label/index` column as target for the classification task.

5 Hyperparameter Tuning

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

$$\hat{E} = \sum_{i=0}^{2} (e_i \cdot \lambda_i) \tag{1}$$

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

5.1 Trainable λs

In this setting, you have to train and find the best λs. **[1 Mark]**

5.2 Frozen λs

In this setting, you have to randomly initialize and freeze the λs. **[1 Mark]**

5.3 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.

$$\hat{E} = f(e_0, e_1, e_2) \tag{2}$$

[4 Marks]

6 Analysis

Compare the performance of ELMo embeddings with the SVD, Skip-gram, and CBOW embeddings implemented in the last assignment. To do this, train the downstream classification model separately using each type of embedding. Ensure that the hyperparameters of the downstream model remain the same across all embeddings for a fair comparison. Evaluate performance using metrics such as accuracy, F1 score, precision, recall, and confusion matrices on both the train and test sets. Provide a detailed analysis of which embedding method performs best, rank them based on performance, and explain why one technique outperforms the others. Additionally, report the best hyperparameter settings (Section 5) and justify why they work well. Include confusion matrices for all hyperparameter settings and discuss their impact on performance.
[15 Marks]

7 Submission Format

Zip the following files into one archive and submit it through the Moodle course portal. The filename should be `<roll number>_assignment4.zip`. For example, `2021101051_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 and save the classifier model.
  - `inference.py`: Test the saved classifier model for the downstream task using the word representations from the biLSTM. Given a news article description, it should predict probabilities of all the classes.

Command to run:
python inference.py <saved_model_path> <description>

Output Format:
class-1 0.6
class-2 0.2
class-3 0.1
class-4 0.1

- **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.**

8 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: **5 marks**
- ELMO Pre-training: **14 marks**
- Downstream Task: **10 marks**
- Hyperparameter Tuning: **6 marks**
- Analysis: **15 marks**
- Viva during Evaluation: **50 marks**

9 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!