CS7.501: Advanced NLP — Assignment 2

Instructor: Manish Shrivastava

Deadline: Sep 24, 2023 — 23:59

1 General Instructions

1. You should implement the assignment in Python.
2. Ensure that the submitted assignment is 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, serious actions, including an F grade in the course, are possible.
3. A single .zip file needs to be uploaded to the Courses Portal.
4. Your grade will depend on the correctness of answers and output. Due consideration will also be given to the clarity and details of your answers and the legibility and structure of your code.
5. Please start early to meet the deadline. Late submissions won't be evaluated.

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, CoVe come in. In the previous assignment, you made use of simple, non-contextual word representation to attempt the next word prediction task. Now, you'll advance on the quality of embeddings used by building an ELMo architecture by yourselves, and testing its efficacy on a downstream task. ELMo looks at building a layered

representation of a word through stacked Bi-LSTM layers, separately weighing in syntactic and semantic representations.

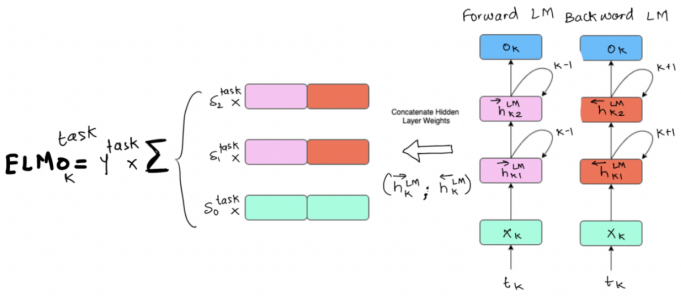

Figure 1: Schematic of the ELMo architecture

As seen in Figure 1, the ELMo architecture contains forward and backward language models trained through its Bi-LSTMs. The stacked Bi-LSTM-based langauge model has a layer of non-contextual word embeddings (like word2vec, or a character convolutional network as in the original ELMo) at the input. While a forward language model can be trained by the next word prediction task, for a backward language model - to put it simply - it's nothing but previous word prediction task. In either of the cases, the LSTM model makes the word prediction based on the context. By context, we mean all the words preceeding (forward LM) or succeeding (backward LM) the word to be predicted in the sentence. By training these Bi-LSTMs using the language modeling objective, we essentially pretrain the weights of the ELMo architecture, making it suitable for application on downstream tasks. To break this down, when we pretrain the model, we make the model learn the nature of the language itself. The model does not really learn any useful task in this process - instead it captures the general dependencies in the language, which makes it smarter to be able to solve any kind of task thrown at it later quickly. This task is what we refer to as the downstream task. On pretraining the model, we expect it to give good performance on a given NLP task with lesser training and data than it would take without pretraining. Specific to ELMo, a stacked Bi-LSTM is used in order to capture the complexities in natural language in a layered manner - with the lower layers focusing on the syntactic aspects and the higher ones on more complex, semantic aspects. Also, using a Bi-LSTM enables us to capture the context for a given word in the sentence based on all the surrounding words in that sentence. At each layer, we get this bidirectional context of increasing complexity by simply concatenating the representations obtained from the forward and backward LSTMs at each layer. Finally, we add up the representations from each layer of the model (including the non-contextual input embeddings) to have the final contextual word representation of our ELMo architecture. In this assignment, you'll code up such an ELMo architecture from scratch, pretrain its weights using the bidirectional language modeling objective and test

its efficacy on a popular 5-way sentiment classification task in NLP.

3 Questions & Grading

[Total marks: 100, Bonus marks: 25]

3.1 Theory [20]

1. How does ELMo differ from CoVe? Discuss and differentiate both the strategies used to obtain the contextualized representations with equations and illustrations as necessary. [10 marks]

2. The architecture described in the ELMo paper includes a character convolutional layer at its base. Find out more on this, and describe this layer. Why is it used? Is there any alternative to this? [Hint: Developments in word tokenization] [10 marks]

3.2 Implementation & Training [60]

1. **Architecture [20 marks]** 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 to the way you did in the previous assignment. 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.

2. **Model Pretraining [20 marks]** 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.

3. **Downstream Task [20 marks]** You will be training the ELMo architecture on a 4-way classification task using the AG News Classification Dataset, referred in the Section 4. Make use of this dataset to build a pipeline for training and evaluating your model on the classification task.

3.3 Analysis [10]

Write up sufficient analysis on the performance of your ELMo model in the pretraining process & the downstream task in a properly compiled report. This includes the hyperparameters you selected in training (like loss functions, # epochs, learning rate, optimizer, etc.), model performance on the loss (for both the tasks) and metrics (for the downstream task). Use accuracy and F1 score

(preferred: micro F1) as the primary metrics for reporting, any other metrics can be optionally added. Also, display a confusion matrix showing the performance on each class as a part of the report.

3.4 Presentation [10]

We would check these points at the time of individual evaluations that would involve code walk-through, explanation of the report analysis, and questions based on the implementation.

- Implementation efficiency & quality
- Report quality
- Inclusion of a readme along with the code
- Crisp & clear explanation of the code during evals

3.5 Bonus [25]

1. In the main task, you had a trainable scalar parameter to weigh in the representation at each layer. Now, try out 4 different hard coded configurations for weighing in the layer-wise representations, and re-run the classification task experiment using these hardcoded parameters. One of these con- figurations should directly use the last layer's Bi-LSTM embeddings only. Document and compare the results obtained in the report. [10 marks]
2. Replace the pretrained non-contextual word embeddings with character convolutional network which you'll be building from scratch. Re-run the experiments on the pretraining & downstream tasks and add the complete analysis using this approach as for the main task. [15 marks]

4 Training corpus

The AG News Classification Dataset dataset should be used for the task. Utilize the first 7600 entries of train as the development set.

5 Submission format

Zip the following into one file and submit it on the Moodle course portal:

1. Source code (Should include .py files only.)
2. Pre-trained model and the generated embeddings
3. Report answering all the questions in PDF format

4. Readme file (should contain instructions on how to execute the code, restore the pre-trained model, and any other information necessary for clarity)

   Note: If the pre-trained models and embeddings cross the file size limits, upload them to your OneDrive folder and share the read-only accessible links in the readme file.

6 FAQs

1. Do I need to use lemmatization as a part of my data preprocessing?
   No, lemmatization is not necessary for this assignment.

2. Can I use tools like spaCy, torchtext, etc. for data cleaning and preparation?
   Yes, you are encouraged to use such tools for ease in the data preparation process. Focus of the assignment is on the core implementation and training using the implemented architecture.

3. Is it okay to include the test data in the pretraining process?
   No, going by the general ethics in machine learning, data used for evaluation should be something completely unknown to the model. You may make use of the train data or any other data not used in evaluation for the pretraining process.

4. How should the analysis be like?
   We have added the primary points you need to add in Section 3.3. You are free to develop the analysis section based on these pointers. Try to make it as descriptive and well documented as possible, giving the reader a clear idea on the experiment(s) you've tried and the results obtained. Kindly note that presentation holds its weight in the grading as well.

5. Can I submit my code in Jupyter Notebooks?
   No, the final submission should be a Python script. You may work using Jupyter Notebooks, but make sure to convert them to .py files before submitting.

7 Resources

7.1 Papers

1. Deep contextualized word representations

2. Learned in Translation: Contextualized Word Vectors

3. Exploring the Limits of Language Modeling

7.2 Explanatory Supplements

1. A quick introduction to the idea of ELMo
2. More explanation on ELMo & using an off-the-shelf model
3. ELMo & CoVe
4. ELMo, CoVe in contrast with BERT
5. On the character convolutional layer in ELMo
6. An introduction to bidirectional language modeling
7. Bidirectional language modeling using Bi-LSTMs