Evaluating LSTMs on Document Classification

Report by Aashrith Madasu (asm6590)

GitHub Repo

---

1. Problem Statement

Document classification is a special type of sentence classification task where the input is documented as opposed to a sentence or a paragraph. Long context lengths make this more challenging. To achieve good performance a model to accurately infer relations between words far apart in a document. This task needs significantly more computational power and data for training good models.

2. Methodology

2.1 Word Embeddings

For word embeddings, we use **GloVe** pre-trained embeddings to **initialize** our embedding layer. Large scale pre-training on diverse corpora enables rich and accurate representation learning of word tokens. In Particular, we choose the dimension to be **100** and set the word embeddings to be learnable - this way the word embeddings can be tuned and adjusted for the downstream task.

2.2 Sequence Encoding

For encoding the token sequence (document), Long Short-Term Memory (LSTM) is chosen to encode the document into a latent space. LSTM is a special type of Recurrent Neural Network (RNN) with an additional **cell state** to store long-range dependencies. LSTMs are particularly good at modelling long sentences and are a natural choice for document classification.

Equations to update hidden state (h) and cell state (c) inside an LSTM.

$$
\begin{aligned}
\boldsymbol{i}_t &= \boldsymbol{\sigma}_i(\boldsymbol{x}_t \boldsymbol{W}_{xi} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hi} + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \boldsymbol{\sigma}_f(\boldsymbol{x}_t \boldsymbol{W}_{xf} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hf} + \boldsymbol{b}_f) \\
\boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \\
&\quad + \boldsymbol{i}_t \odot \boldsymbol{\sigma}_c(\boldsymbol{x}_t \boldsymbol{W}_{xc} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{hc} + \boldsymbol{b}_c) \\
\boldsymbol{o}_t &= \boldsymbol{\sigma}_o(\boldsymbol{x}_t \boldsymbol{W}_{xo} + \boldsymbol{h}_{t-1} \boldsymbol{W}_{ho} + \boldsymbol{b}_o) \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \boldsymbol{\sigma}_h(\boldsymbol{c}_t)
\end{aligned}
$$

Particularly, a **bi-directional** LSTM is chosen as context from both sides of a word can enrich the word representation. The outputs from the final layer of the stacked LSTM model are aggregated using **mean-pooling** to get the final document encoding which will be used for classification.

2.3 Classification

For classification, we use a **fully-connected** linear layer that takes the document encoding from the previous layer as input. The fully-connected layer outputs a vector of logits as big as the number of classes.

3. Experimentation

3.1 Training

For training the proposed model, we choose **Adam** (Adaptive Moment Estimation) as the optimizer with a **batch size** of 32. No scheduler was used, we use a **fixed learning rate** of 0.001. We train the model for a total of 50 **epochs** and consider the epoch with the **best validation accuracy** to represent the performance of the model.

3.2 Evaluation

We choose the following **datasets** (both provided by **NLTK**) to evaluate the trained model:

1. Reuters: 90 classes
2. Movie Reviews: 2 classes
3. Brown Corpus: 15 classes

To gauge the performance on the validation set **Accuracy** (number of correct / number of total samples) is chosen as the metric.

3.3 Results

	Reuters	Movie Reviews	Brown
1 layers, uni-directional	72.6 %	76 %	26.7 %
1 layers, bi-directional	77.3 %	78 %	14 %
2 layers, uni-directional	70.1 %	71 %	-
2 layers, bi-directional	77.1 %	84 %	-

Observation: From the results on both the Reuters and Movie-Reviews dataset, single-layered bi-directional LSTM outperforms two-layered uni-directional LSTM - this emphasizes the importance of encoding **bi-directional context** for effective representations. Moreover, single-layered uni-directional LSTM outperforms two-layered uni-directional LSTMs - this emphasizes the negative effects of over-parameterization.

NOTE: The performance on “brown” dataset is bad because the **number of data points are less** - hence, the experimentation is limited considering the computational constraints.

4. Conclusion

In conclusion, LSTMs are an effective way to model long-range dependencies and particularly useful for tasks like document classification. Although increasing the number of layers benefits the performance usually, in some cases it might lead to over-parameterization which will negatively impact the performance.

5. Learnings

In this project, I learned the math and the **internal operations of an LSTM** and understood how the extra cell state helps learn long-range dependencies. My learnings also involves implementing **Multi-GPU** usage through pytorch’s **Distributed Data Parallel** API. I also learned to process various datasets available through **NLTK** API.