Assignment 4: ELMO

Course: Introduction to Natural Language Processing

Roll number : 2022201060

Name : Mohit Sharma

1. ELMo: Deep Contextualized Word Representations :

Earlier architectures such as word2vec and GloVe simplified the process of obtaining meaningful, low-dimensional representations of words for semantic tasks. These methods provided a single representation for a word, aggregated from all possible contexts in the training corpus. However, given the complexity and ambiguity of language, more sophisticated representations are needed that consider the context of a word within a sentence. This is where contextual embeddings like ELMo and CoVe come into play. In the previous assignment, I utilized simple, non-contextual word representations for a downstream task. Now, I aim to enhance the quality of embeddings by constructing an ELMo architecture and assessing its efficacy on the same task. ELMo achieves this by building a layered representation of words using stacked Bi-LSTM layers, which separately weigh syntactic and semantic representations.

2. Implementation and Training

Architecture : The ELMo architecture consists of several components designed to capture contextualized word representations:

Embedding Layer:

- An embedding layer converts input words into dense vector representations.
- If pre-trained embeddings are provided, they are loaded and frozen; otherwise, embeddings are learned during training.

Bidirectional LSTM Layers:

- Two sets of bidirectional Long Short-Term Memory (LSTM) layers are used: one for forward processing and one for backward processing.
- The first set of LSTM layers processes the input embeddings in the forward direction, capturing contextual information from preceding words.

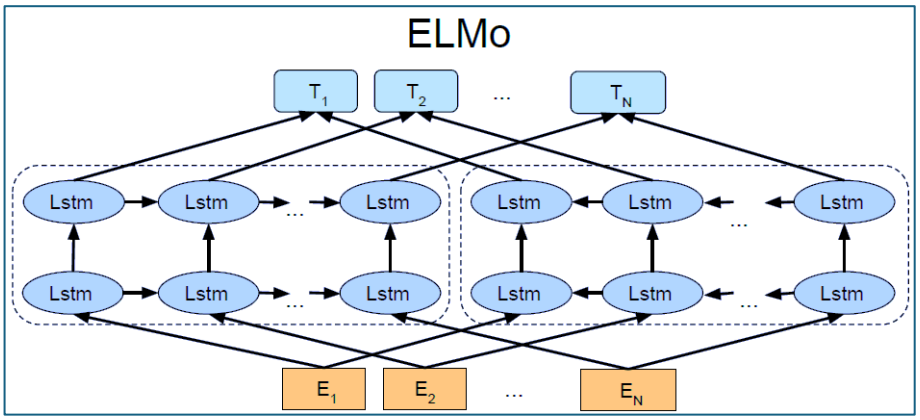
- The second set of LSTM layers processes the input embeddings in the backward direction, capturing contextual information from succeeding words.
- Each LSTM layer consists of a specified number of hidden units, determining the complexity of the learned representations.

Linear Layer:

- A linear layer combines the output representations from both forward and backward LSTM layers.
- The combined representations are passed through a linear transformation to produce the final output logits. from there we can predict the next word.

Dropout Layer:

- Dropout is applied to the output of each LSTM layer to prevent overfitting and improve generalization.



Model Pre-training :

1. In the pre-training phase, the ELMo model undergoes training on the dataset of text to grasp meaningful contextualized word representations.
2. Firstly, the model is switched to training mode, allowing it to update its parameters during the training process. Then, a loop is set up to handle batches of sentences from the training dataset.
3. Each sentence is divided into two parts: the input sequence and the target sequence. The input sequence contains all words except the last one, while the target sequence includes all words except the first one.

   For example, in the sentence "The cat sat on the mat," the input sequence is "The cat sat on the," and the target sequence is "cat sat on the mat."

4. These input and target sequences are converted into tensors and transferred to the appropriate device for computation, such as a GPU.

5. Next, the input sequences are fed into the ELMo model to predict the next word in each sequence. The model processes these input sequences through its layers and generates output logits for each word position in the sequence.
6. The output logits are then compared against the target sequences to calculate the loss. Typically, a loss function like cross-entropy loss is employed to measure the difference between predicted and actual word distributions.
7. After computing the loss, gradients of the loss with respect to the model parameters are computed using backpropagation. The optimizer adjusts the model parameters based on these gradients to minimize the loss.
8. This entire process repeats for each batch of sentences in the training dataset, gradually improving the model's understanding of contextualized word representations.

Downstream Task :

The model defines several layers including the bidirectional LSTM layer (lstm_layer), the fully connected layer (fc_layer), and a linear transformation (linear) for downstream task.

Input sequences are converted into embeddings, capturing contextual information. These embeddings are then fed through two sets of bidirectional LSTM layers (forward_lstm1, backward_lstm1, forward_lstm2, backward_lstm2) to further learn the contextual understanding. The outputs from these LSTM layers are concatenated and adjusted using trainable or non-trainable lambdas (lambda1, lambda2, lambda3), which determine the relative contribution of each component. The final contextualized embeddings are obtained and passed through the fully connected layer as mentioned in the previous, to generate logits, providing raw scores for classification.

For prediction, the predict method takes the logits as input and returns the index of the predicted class for each input sequence. This is accomplished by identifying the index of the maximum value within the logits, indicating the predicted class label for each input sequence.

3. **Corpus :** The model is trained on the provided CSV files, accessible via the provided link to the News Classification Dataset. Specifically, it utilizes the Description column from the train.csv file to train word vectors. For the downstream classification task, the label/index column from the same dataset is used. It's important to note that only the Description column is utilized for training word vectors, while the label/index column is used for classification.

4. Hyperparameter tuning

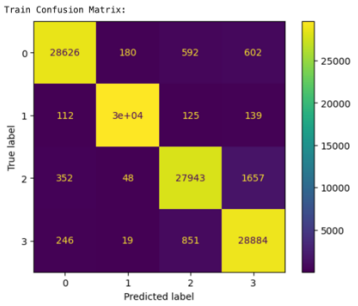
4.1. Trainable λs :

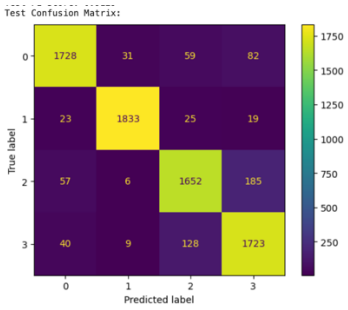
Evaluation on Train Set with Trainable lambda's:

Metric	Value
Accuracy	95.90%
Precision	95.94%
Recall	95.90%
F1 Score	95.91%

Evaluation on Test Set with Trainable lambda's:

Metric	Value
Accuracy	91.26%
Precision	91.36%
Recall	91.26%
F1 Score	91.29%





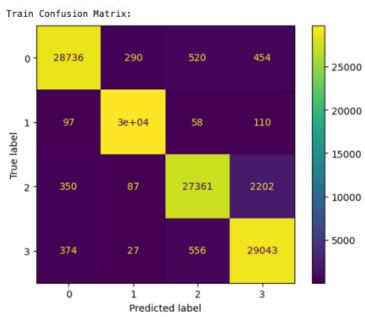
4.2. Frozen λs : λ1 : 0.2 λ2 : 0.4 λ3 : 0.4

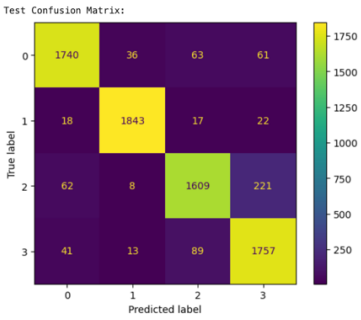
Evaluation on Train Set with Frozen lambda's:

Metric	Value
Accuracy	95.73%
Precision	95.80%
Recall	95.73%
F1 Score	95.73%

Evaluation on Test Set with Frozen lambda's:

Metric	Value
Accuracy	91.43%
Precision	91.56%
Recall	91.43%
F1 Score	91.43%





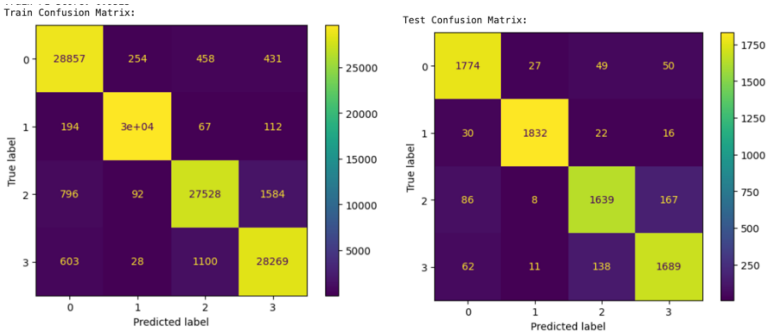
4.3. Learnable Function :

Evaluation on Train Set with Trainable lambda's:

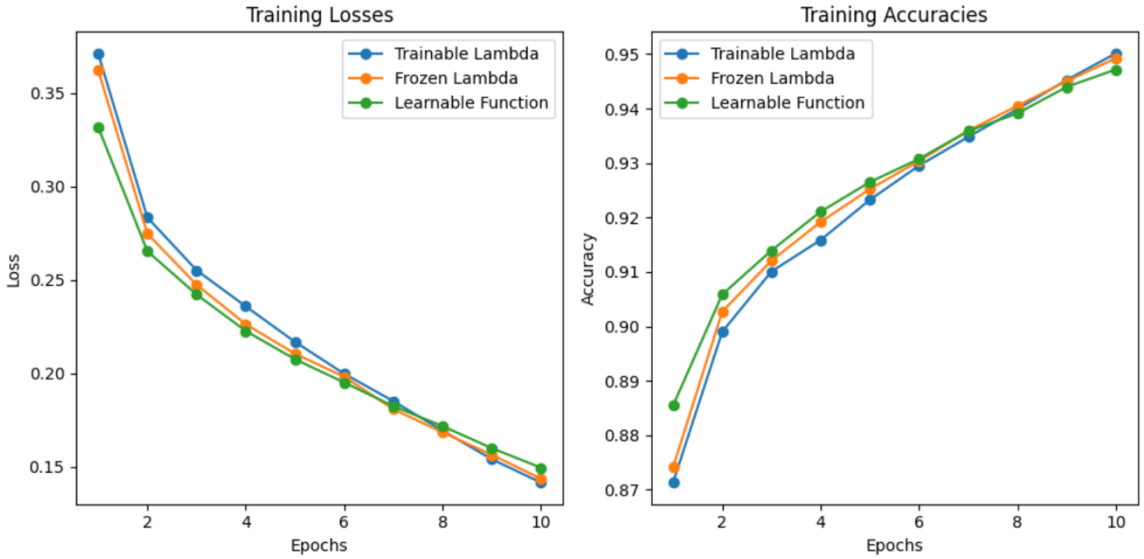
Metric	Value
Accuracy	95.23%
Precision	95.24%
Recall	95.23%
F1 Score	95.23%

Evaluation on Test Set with Trainable lambda's:

Metric	Value
Accuracy	91.24%
Precision	91.25%
Recall	91.24%
F1 Score	91.23%



5. Analysis :



Overall Performance Analysis of ELMo Model :

As per the report we can see that the ELMo model performance is storng in the downstream classification task across different configurations. After 10 epochs of training, it consistently achieves high accuracy and F1 scores on both the training and test datasets.

In the pretraining for 5 epoch losses were 3.27, 2.06, 1.64, 1.37, 1.19 which show s a great improvement over the epochs. In the downstream classification task, t he ELMo model maintains high accuracy and F1 scores across different configur ations. Notably, while the Trainable λs configuration shows promising improvem ents during training, the Frozen λs configuration emerges as the best-performing one after 10 epochs, achieving the highest accuracy and F1 score on the test dat aset. Specifically, the test accuracy for the Frozen λs configuration is approximat ely 91.43%, with an F1 score of 91.43%.

These results highlight the effectiveness of the ELMo model in capturing contextual information and generating high-quality word embeddings for downstream classification tasks. While the Trainable λs configuration may show initial improvements during training, the Frozen λs configuration ultimately

demonstrates superior performance on the test dataset. The model's consistent ability to perform well, especially with the Frozen λs configuration, indicates its reliability and suitability for various natural language processing applications, particularly in classification scenarios.

Train Set Performance Comparison:

Model	Accuracy	Precision	Recall	F1 Score
ELMo (Trainable λs)	95.90%	95.94%	95.90%	95.91%
ELMo (Frozen λs)	95.73%	95.80%	95.73%	95.73%
Skip Gram with Negative Sampling	96.43%	96.44%	96.43%	96.43%
SVD	90.22%	90.25%	90.22%	90.23%

Test Set Performance Comparison:

Model	Accuracy	Precision	Recall	F1 Score
ELMo (Trainable λs)	91.26%	91.36%	91.26%	91.29%
ELMo (Frozen λs)	91.43%	91.56%	91.43%	91.43%
Skip Gram with Negative Sampling	90.24%	90.24%	90.24%	90.24%
SVD	86.87%	86.87%	86.87%	86.86%

In this comprehensive evaluation of word vectorization methods for a downstream classification task, **ELMo embeddings** have emerged as the **top-performing approach**, showcasing superior performance across various evaluation metrics compared to Skip Gram with Negative Sampling and Singular Value Decomposition (SVD).

Train Set Performance:

- **Accuracy**: ELMo with Trainable λs achieves the highest accuracy, closely followed by Skip Gram with Negative Sampling. Both ELMo configurations demonstrate strong performance, surpassing SVD by a notable margin.
- **Precision and Recall**: ELMo consistently exhibits high precision and recall values, indicating its ability to correctly identify positive instances while minimizing false positives and false negatives. Skip Gram with

Negative Sampling also performs well but slightly lags behind ELMo in precision.

- **F1 Score**: ELMo with Trainable λs achieves the highest F1 score on the training set, showcasing its effectiveness in achieving a balance between precision and recall.

Test Set Performance:

- **Accuracy**: ELMo with Frozen λs achieves the highest accuracy on the test set, closely followed by ELMo with Trainable λs. Both configurations outperform Skip Gram with Negative Sampling and SVD.
- **Precision and Recall:** ELMo consistently demonstrates superior precision and recall values on the test set, highlighting its robustness in correctly classifying instances from various classes.
- **F1 Score**: ELMo with Frozen λs attains the highest F1 score on the test set, indicating its effectiveness in achieving a balance between precision and recall, crucial for classification tasks.

Overall:
ELMo embeddings, particularly when fine-tuned with trainable λs, offer the most effective representation of words for downstream classification tasks. They excel in capturing contextual information, leading to better classification performance compared to traditional methods like Skip Gram with Negative Sampling and SVD. ELMo's ability to adapt to different contexts and tasks, as we can see by its performance across different configurations.

Comparison of Word Vectorization Techniques
In the evaluation of word vectorization techniques, such as Skip Gram with Nega tive Sampling, Singular Value Decomposition (SVD), and ELMo embeddings with different hyperparameter settings is performing better by understanding semanti c information also.

Contextual vs. Static Embeddings:
ELMo embeddings, unlike Skip Gram and SVD, offer contextual representations of words. While Skip Gram and SVD provide fixed embeddings for each word regardless of context, ELMo considers the surrounding words in a sentence, resulting in more nuanced and adaptable representations.

Hyperparameter Settings:
The performance variation within ELMo embeddings can be attributed to differen t hyperparameter settings, particularly the use of trainable λs versus frozen λs or learnable function. Trainable λs allow the model to learn the optimal combinatio n of weighted layers during training, leading to embeddings fine-tuned to the spe cific task. In our evaluation, we used ELMo embeddings with 100-size embeddin gs and Adam optimizer, along with trainable λs, to achieve the best results with c ross entropy loss.

Performance Comparison:
ELMo embeddings consistently outperform Skip Gram with Negative Sampling a nd SVD across various metrics. This superiority is due to ELMo's ability to captur e contextual information and adapt to dataset characteristics. Among ELMo con figurations, those with trainable λs perform best, as they allow the model to lear n optimal representations for the task at hand.

Conclusion:
ELMo embeddings, particularly with trainable λs, demonstrate the importance of contextual embeddings in capturing natural language complexities. By consideri ng context, ELMo offers more nuanced representations, leading to superior perfo rmance in downstream tasks. This success underscores the significance of cont extual embeddings in modern natural language processing applications. The use of Adam optimizer and cross-entropy loss further enhances the effectiveness of ELMo embeddings, making them a valuable tool in NLP task