Introduction to NLP
Assignment 4

REPORT

April 25, 2024

Ashna Dua
2021101072
ashna.dua@students.iiit.ac.in

1 ELMo

ELMo, which stands for Embeddings from Language Models, is a technique for generating **contextual word embeddings**. Unlike older methods like word2vec or GloVe that provide a single, static representation for each word, ELMo takes into account the context in which a word appears to create a more nuanced and accurate representation. This is achieved through a stacked Bi-directional LSTM network that analyzes the entire sentence, capturing both the preceding and following words to understand the meaning of each word within its specific context.

ELMo's key features are:

- **Contextualized Embeddings:** It generates word representations that consider the surrounding words in a sentence, leading to a more precise understanding of meaning.
- **Deep Bi-LSTM architecture:** ELMo uses stacked Bi-directional LSTMs to analyze the sentence in both forward and backward directions, capturing complex relationships between words.
- **Layered Representations:** The network produces word embeddings at different layers, each capturing different aspects of meaning (e.g., syntax and semantics). These are then combined using trainable weights to create the final word representation.

1.1 Preparing Text for ELMo

The **CustomDataset** class is responsible for processing and preparing textual data for training the ELMo model. It takes raw text data and performs several key steps:

- **Tokenization:** It breaks down each text document into individual words using word_tokenize.
- **Vocabulary Building:** It creates a vocabulary of unique words present in the data, assigning a unique index to each word. An "unknown" token (¡unk¿) is also added for words not present in the vocabulary.
- **Word to Index Conversion:** Each word in the text is replaced with its corresponding index from the vocabulary.
- **Padding:** To enable batch processing, all text sequences are padded to the same length using the padding_value.
- **Data Access:** It provides methods to access individual data samples, returning both the padded token sequence and the corresponding class label.

1.2 Architecture and Pre-Training

This section details the implementation of the ELMo architecture using PyTorch. The core components and functionalities are outlined below:

- **Model Architecture (ELMo Class):**
  - **Initialization:** The constructor defines key parameters like vocabulary size, embedding dimension, and hidden dimension. It also initializes the embedding layer for the input tokens and two layers of Bi-directional LSTMs (forward and backward) for processing the input sequence.
  - **Forward and Backward LSTM Layers:** Two separate Bi-LSTM layers are used to capture information from both directions of the sentence. Each layer outputs hidden states for each word, incorporating context from both preceding and following words.
  - **Weighted Sum of Layer Outputs:** Trainable weights are used to combine the outputs from different Bi-LSTM layers and the initial embedding layer. This allows the model to learn the optimal contribution of each layer to the final word representation.
  - **Padding and Concatenation:** Padding is applied to handle sentences of varying lengths and ensure consistent tensor dimensions. The outputs from forward and backward LSTMs are then concatenated to create a comprehensive representation of each word.
  - **Linear Layer and Output:** A final linear layer is applied to the combined representations, producing a probability distribution over the vocabulary for predicting the next word in the sequence.
- **Pre-Training:**
  - **Loss Function and Optimizer:** Cross-Entropy Loss is used to measure the difference between predicted and actual next words, while the Adam optimizer updates model parameters to minimize the loss.
  - **Training Loop:** The model iterates over the training data in batches, calculating loss, backpropagating gradients, and updating parameters for a specified number of epochs.
- **Evaluation:**
  - **Accuracy Calculation:** The model's performance is evaluated on a test dataset by comparing predicted next words with actual words (not padding) and calculating the accuracy.

2 Downstream Task - Classification

This section explores the application of the pre-trained ELMo model to a downstream task: 4-way text classification using the AG News dataset.

- **Classification:**
  - **Initialization:** The LSTMClassifier class takes the pre-trained ELMo model as input and adds a classification head on top. It includes an LSTM layer to process the contextual word embeddings generated by ELMo and a linear output layer for predicting the class probabilities.
  - **Combining Embeddings:** The model offers various ways to combine the different layers of ELMo embeddings:
    - **Trainable:** Learn weights for each layer during training to determine their relative importance.
    - **Frozen:** Use randomly initialized, fixed weights for each layer.
    - **Function:** Learn a function to combine the embeddings, providing more flexibility than fixed weights.
  - **Classification:** The model can use either the last hidden state or the mean of all hidden states from the LSTM as the input to the final classification layer, allowing for different approaches to capturing the overall meaning of the text.
  - **Training and Evaluation:** Similar to the ELMo pretraining, the model uses cross-entropy loss and the Adam optimizer for training. It provides functions for training and evaluating the model's performance on the classification task.
- **Hyperparameter Tuning:**
  - **Trainable λ:** This approach learns the optimal weights for combining the ELMo layers, allowing the model to focus on the most relevant aspects of the embeddings for the classification task.
  - **Frozen λ:** Here, fixed weights are used, offering a simpler approach but limiting the model's ability to adapt to the specific characteristics of the downstream task.
  - **Learnable λ:** This method learns a function to combine the embeddings, providing more flexibility and potentially capturing complex relationships between the different layers of representation.

3 Results

The accuracy of ELMo model in the pre-training phase is as follows (using 20k sentences only):

	Accuracy
ELmo	41.21

3.1 Train Data

3.1.1 Metrics

The accuracy of different classification hyper-parameters are as follows:

	Accuracy	F1 Macro	F1 Micro	Precision	Recall
Trainable	97.78	97.78	97.78	97.78	97.78
Frozen	84.42	84.42	84.30	84.39	84.42
Function	91.92	91.92	91.93	92.80	91.92

3.1.2 Confusion Matrices

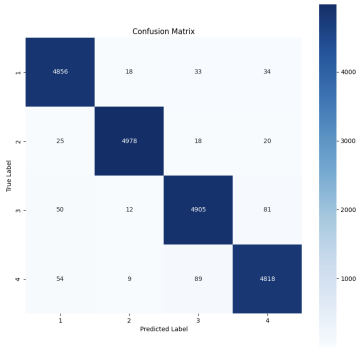

Figure 1: Trainable

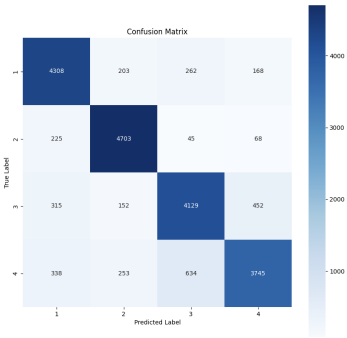

Figure 2: Frozen

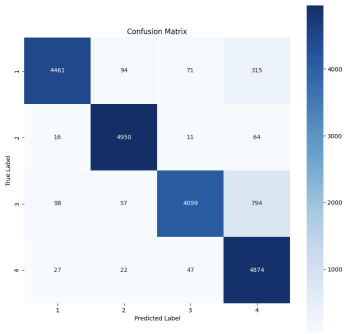

Figure 3: Function

3.2 Test Data

3.2.1 Metrics

The accuracy of different classification hyper-parameters are as follows:

	Accuracy	F1 Macro	F1 Micro	Precision	Recall
Trainable	81.11	81.11	81.12	81.60	81.11
Frozen	77.25	77.25	77.26	77.41	77.25
Function	81.39	81.39	81.51	83.03	81.39

3.2.2 Confusion Matrices

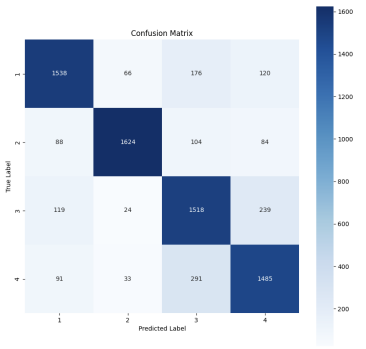

Figure 4: Trainable

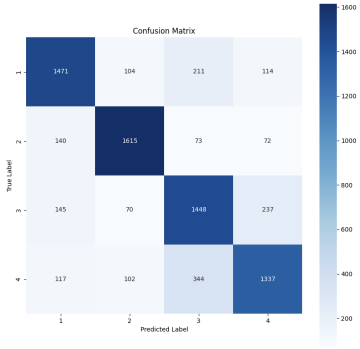

Figure 5: Frozen

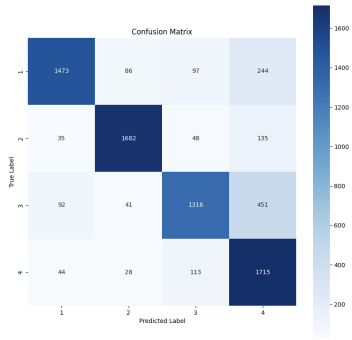

Figure 6: Function

3.3 Comparison with SVD and Word2Vec

3.3.1 Metrics

Model	**Accuracy**	**Precision**	**Recall**	**F1 Micro**	**F1 Macro**
SVD (Test)	81.11	81.23	81.11	81.11	81.12
SVD (Train)	82.48	82.39	82.38	82.48	82.48
Word2Vec (Test)	83.60	83.68	83.60	83.60	83.60
Word2Vec (Train)	84.59	84.51	84.62	84.59	84.59
ELMo (Test)	81.39	83.03	81.39	81.51	81.39
ELMo (Train)	91.92	92.80	91.92	91.93	91.92

Table 1: Metrics for SVD, Word2Vec, and ELMo

3.3.2 Confusion Matrices

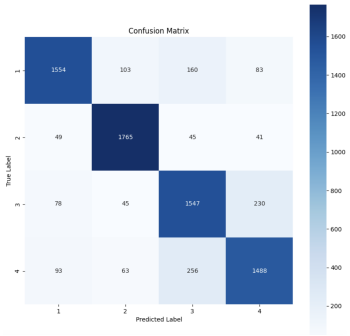

Figure 7: Word2Vec (Test)

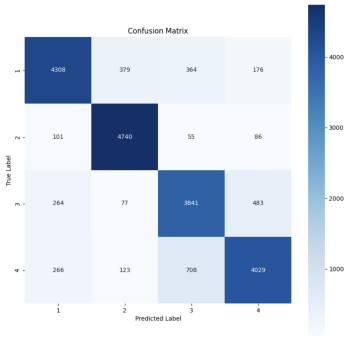

Figure 8: Word2Vec (Train)

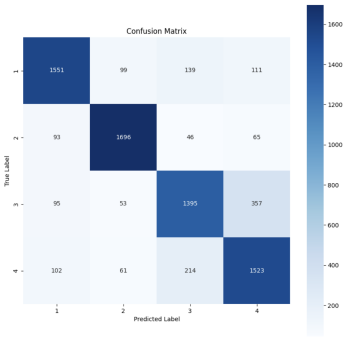

Figure 9: SVD (Test)

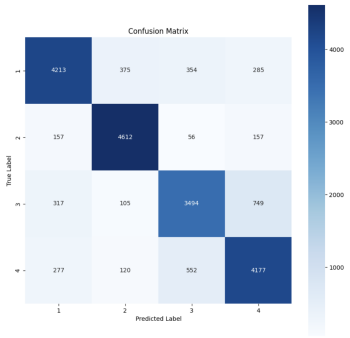

Figure 10: SVD (Train)

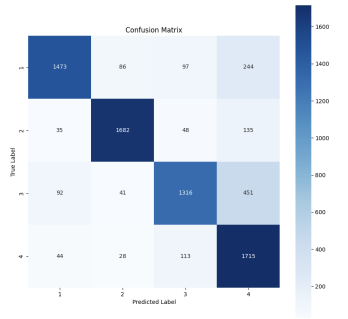

Figure 11: ELMo (Test)

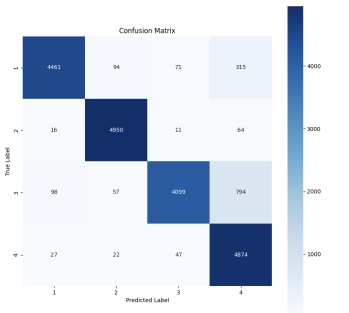

Figure 12: ELMo (Train)

3.4 Observations

3.4.1 Word2Vec, SVD and ELMo

- Word2Vec outperforms both SVD and ELMo in terms of accuracy, F1-score, precision, and recall. This suggests that Word2Vec is better at capturing the semantic relationships between words relevant to the classification task.

- Word2Vec outperforms both SVD and ELMo slightly in terms of accuracy and F1 scores. This could be attributed to the window size parameter being set to 5 for Word2Vec, which might capture more contextual information compared to the default window size of ELMo and SVD. ELMo performs competitively with SVD but slightly lags behind Word2Vec. However, ELMo demonstrates better performance in terms of precision and recall on the test set compared to SVD.

- Word2Vec's superior performance could be partly explained by its context window size of 5. A larger window size allows Word2Vec to capture more contextual information, which might lead to better word embeddings for downstream tasks.
- ELMo embeddings are pre-trained using deep contextualized representations, allowing them to capture intricate semantic and syntactic information. This results in high precision and recall scores on the test set, indicating its ability to accurately classify text data.

3.4.2 ELMo Hyperparameters

Among the explored hyperparameter settings for ELMo embeddings, both the Trainable and Function settings demonstrate nearly equal performance, with Function edging slightly ahead. This suggests that both settings offer effective strategies for combining word representations across layers, leading to optimal performance in downstream tasks. While Trainable allows for adaptability to task data, Function strikes a balance between flexibility and computational simplicity. The Frozen setting exhibits slightly lower performance due to the fixed nature of λ.