3.1.1

Key differences between ELMo and CoVe are:-

- CoVe is trained on supervised machine translation task (english to German) while ELMo is trained on unsupervised Language modeling task.
- CoVe takes the last layer of LSTM based encoders as the context vectors while ELMo is the weighted combination of all the layers in the language model LSTM.

CoVe:-

CoVe idea comes from the pre-trained CNN in ImageNet classification problem. CoVe are obtained from an encoder of Machine Translation task using a two-layers Bi-directional Long short-term Memory network. To be able to translate, the model learns to get the complex relations between the words, so that it gets an advanced representation of the full sentence. So, we can consider that the encoder of the Neural MT task will preserve much more complex semantic relations between words. After training on translation task we can use the encoder as a pre trained layer in other task. IT use a standard, two-layer, bidirectional LSTM network as the encoder (MT-LSTM), initialized with GloVe vectors and a two-layer, unidirectional LSTM with attention as the decoder. The outputs of the pre-trained encoder, MT-LSTM, are treated as the context vectors.

$$\mathrm{CoVe}(w) = \text{MT-LSTM}\ (\mathrm{GloVe}(w))$$

Here , w is sequence of english words and $\mathrm{GloVe}(w)$ is GloVe embedding of the words.

The aim is to use this contextualized word vectors for downstream task. For downstream task each GloVe vector is concatinated with its CoVe vectors.

$$W = [\mathrm{GloVe}(w);\mathrm{CoVe}(w)]$$

ELMo:- It's aim is to learn representations that model the syntax, semantics and polysemy (existence of many possible meanings for a word). ELMo vectors are derived by training a bidirectional LSTM model on language modeling task. ELMo vectors are a function of all the internal layers of the language model and hence deeper than the CoVe. In the forward pass, the history contains words before the target token and in the backward pass, the history contains words after the target token.The predictions in both directions are modeled by multi-layer LSTMs . The final layer's hidden state is used to output the probabilities over tokens after softmax normalization. They share the embedding layer and the softmax layer, parameterized by and respectively. The model is trained to minimize the negative log likelihood in both directions.

3.2.2

Word embedding can only handle seen words.Having the character embedding, every single word's vector can be formed even it is out-of-vocabulary words**.** We first define a list of characters ex. 70characters which including 26English letters, 10 digits, 33 special characters and new line character. Transfer characters as 1-hot encoding and got a sequence for vectors. For unknown characters and blank characters, use all-zero vector to replace it. Using CNN layers to learn the sequence.

Advantage of using a character convolution layer for static representation of the input is to overcome the problem of unknown or out of vocabulary words which most common when we use word level static representation like Glove or Word2Vec which uses fixed vocabulary. Another benefit is that it good fits for misspelling words, emoticons, new words.

Alternative of characters embeddings are using subwords tokenizers. Most of the recent Pre-trained Language models uses this subword tokenizer. It advantages are:-

- It mostly avoids the out-of-vocabulary (OOV) word problem.Word vocabularies cannot handle words that are not in the training data. Subword vocabularies allow representing these words. By having subword tokens (and ensuring the individual characters are part of the subword vocabulary), makes it possible to encode words that were not even in the training data.

- It gives manageable vocabulary sizes. Current neural networks need a pre-defined closed discrete token vocabulary. The vocabulary size that a neural network can handle is far smaller than the number of different words

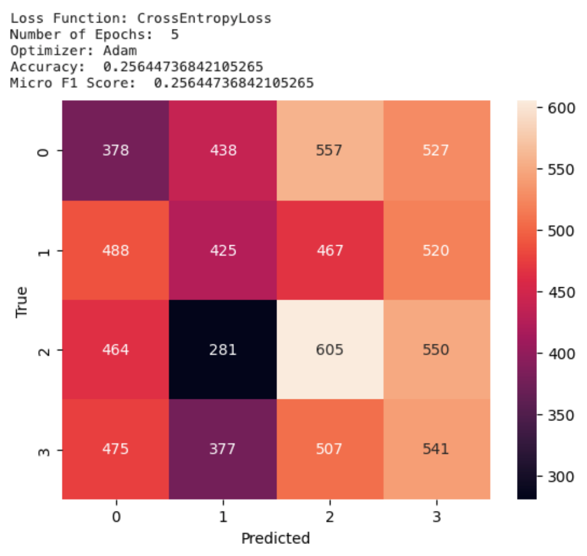
- It mitigates data sparsity. In a word-based vocabulary, low-frequency words may appear very few times in the training data. This is especially troublesome for agglutinative languages, where a surface form may be the result of concatenating multiple affixes. Using subword tokenization allows token reusing, and increases the frequency of their appearance.

pre-training Elmo:-

- We use Glove Emeddings of 200 dimension.
- learing rate = 0.01
- Loss = negative log likelihood (nn.NLLLOSS())
- Optimizer = Adam
- Epoch = 5

Classification Task:-

- learing rate = 0.01
- Optimizer = Adam
- Loss = Cross entropy loss (nn.CrossEntropyLoss())
- Epoch = 5
- Accuracy on test-data = 25.64 %



```
Loss Function: CrossEntropyLoss
Number of Epochs:  5
Optimizer: Adam
Accuracy:  0.25644736842105265
Micro F1 Score:  0.25644736842105265
```