**ADV NLP**

Name: Anantha Lakshmi

Roll.No:2020101103

Q1. How does Elmo differ from Cove? Discuss and differentiate both the strategies used to obtain the contextualized representations with equations and illustrations as necessary.

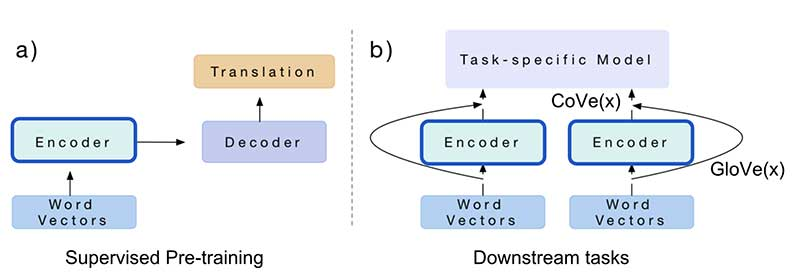
Ans:

* Both Elmo and Cove are contextual embeddings. Cove uses supervised tasks such as Machine Translation and Elmo uses unsupervised tasks such as language modeling.
* We transfer what is learned by the MT-LSTM to downstream tasks by treating the outputs of the MT-LSTM as context vectors. If w is a sequence of words and Glove(w) the corresponding sequence of word vectors produced by the Glove model, then

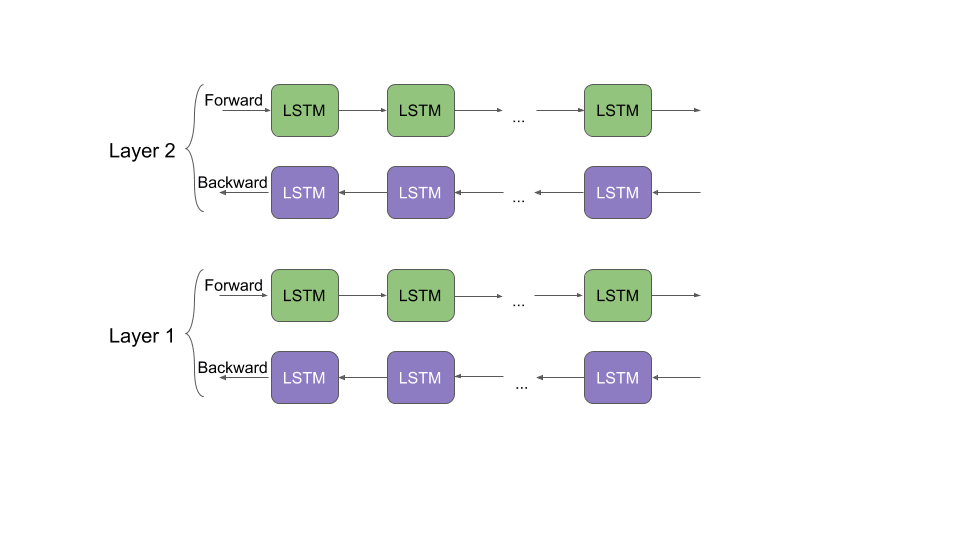
Cove(w) = MT-LSTM(Glove(w))

is the sequence of context vectors produced by the MT-LSTM. For classification and question answering, for an input sequence w, we concatenate each vector in Glove(w) with its corresponding vector in Cove(w)

w˜ = [Glove(w); Cove(w)]



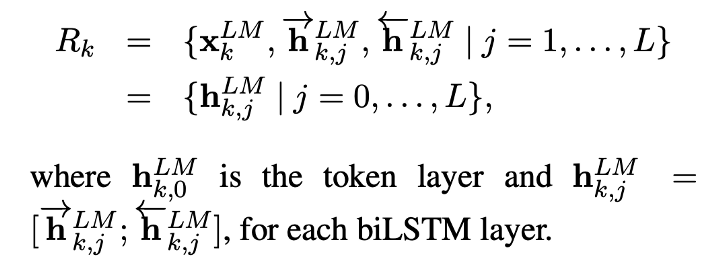
* There is a difference in training. Cove uses machine translated dataset and Elmo embeddings trained on the language model data.
* Elmo word vectors are computed on top of a two-layer bidirectional language model (biLM). This biLM model has two layers stacked together. Each layer has 2 passes — forward pass and backward pass:



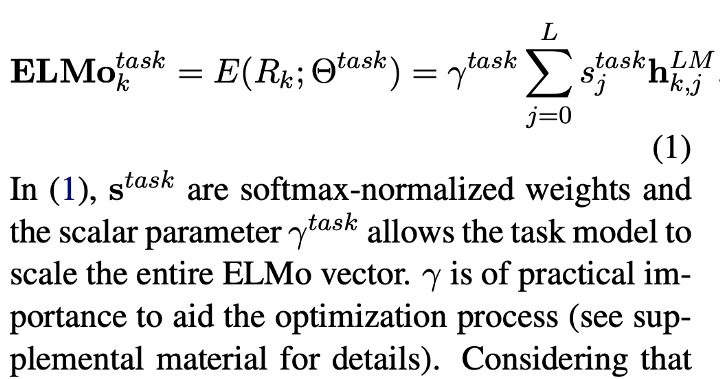
* The architecture above uses a character-level convolutional neural network (CNN) to represent words of a text string into raw word vectors
* These raw word vectors act as inputs to the first layer of biLM
* The forward pass contains information about a certain word and the context (other words) before that word
* The backward pass contains information about the word and the context after it
* This pair of information, from the forward and backward pass, forms the intermediate word vectors
* These intermediate word vectors are fed into the next layer of biLM
* The final representation (Elmo) is the weighted sum of the raw word vectors and the 2 intermediate word vectors

As the input to the biLM is computed from characters rather than words, it captures the inner structure of the word. For example, the biLM will be able to figure out that terms like *beauty* and *beautiful* are related at some level without even looking at the context they often appear in. Sounds incredible!

* The limitation of CoVe is obvious: (1) pre-training is bounded by available datasets on the supervised translation task; (2) the contribution of CoVe to the final performance is constrained by the task-specific model architecture.
* we have token representation for the kth token Rk as follows:



* The ELMo vector for token k is weighted sum of hidden vectors where weights learned during training:



Elmo is based on a language model.

Q2. 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]

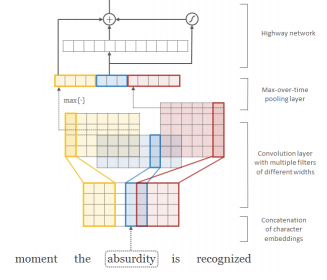
Ans:

There, in the forward method, you can see that the input to the network is a tensor of dimensions (batch\_size, sequence\_length, 50), where 50 is the maximum number of characters per word. Therefore, before passing the text to the network, it is segmented in words, and each character is encoded as an integer value.

This is what happens in the forward method before the highway layers:

* The tensor gets prepended and appended sentence boundary tokens (beginning-of-sentence (BOS), end-of-sentence (EOS)).
* The tensor goes through an embedding layer (this is somewhat similar to one-hot encoding and a matrix multiplication). This gets us a vector for each character.
* The tensor goes through different 1D convolutions of configurable kernel sizes.
* The resulting activation maps are concatenated and passed as input to the highway networks

This architecture was proposed by [kim et al. (2015)](https://arxiv.org/abs/1508.06615), and is summarized well in one of the figures of the paper:



The first word embedding uses a convolutional layer at ELMo's base level. In the training phase, when we need to do some embedding (one-hot encoding being the most basic), the length of the vector can be large enough and we have a very sparse set of words. So when we go into character-level encoding, the length of the input vector is short, since the English alphabet has only 26 characters. Also, ConvNets that use letter folding at their base work with embeddings without needing words. There are many alternatives for this, including word2vec, GloVe, BagOfWords, etc., but I found this to be the best, with the least bugs, of these renderings.

Reference links:

* <https://ted-mei.medium.com/from-static-embedding-to-contextualized-embedding-fe604886b2bc>
* <https://www.topbots.com/generalized-language-models-cove-elmo/>
* <https://ahmedhanibrahim.wordpress.com/2019/07/01/a-study-on-cove-context2vec-elmo-ulmfit-and-bert/>
* <https://arxiv.org/pdf/1708.00107v2.pdf>
* <https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/>