Q1

ELMov

- Is learned by using language modeling task. Which has is more available data compared to machine translational task.
- We use two Deep LSTM which learn the Forward and Backward LM to create embedding.
- We create embedding by concatenating the output of the two LSTM at every layer.
- We take the concatinated output from every layer take a weighted sum.
- The weights are decided/optimized/trained for the downstream task.

CoVe

- It is learned by using machine translation task. Which has is less available data compared to language modeling task.
- We feed GloVe(wx) to a standard, two-layer, bidirectional, long short-term memory network.
- In Clove we use the output of the last layer of the LSTM as the embedding.

Similarity

- ELmov with only one layer trained on Machine translation task will yield to CoVe.

References

- Learned in Translation: Contextualized Word Vectors, https://arxiv.org/abs/1708.00107

Q2

Describe this layer.

- We create one-hot representation for every character.
- Given a word/token for we need to find the embedding we create a matrix where the coloums are the one hot representation of the characters in the word/token.
- Consider this to be grey scale image. We apply a 1D convolutional filter to this image (word).
- The output of the convolutional filter is a vector. This vector is our word/token embedding.

Why is it used?

- This layer is used compute a context-independent word/token representation.
- We can compute token/word representation for tokens/words which are not seen during traning.

- This is similar to word2vec, glove in the sense that all these are methods to compute global word/token embeddings.
- Any contextual embeddings requires global embedding.
- Contextual embedding models combine the global embedding of the words in the context to get a compute a contextual embedding.
- This CharacterCNN servers the purpose of giving global embedding to tokens/words in the ELMov architecture.

Is there any alternative to this?

- The alternative to Char-CNN is Char-LSTM.
- A character embedding matrix initialized randomly. The character embeddings corresponding to every character in a word are given to a birectional LSTM. We use the concatination of the its forward and backward representation as the word/token embedding.
- We can also use any regural global embedding like word2vec, glove along with char-token embedding.
- We can use both Char-CNN and Char-LSTM together.

References

- https://arxiv.org/abs/1509.01626, Character-level Convolutional Networks for Text Classification
- https://www.kaggle.com/code/anubhavchhabra/character-level-word-embeddings-using-1d-cnn
- https://aclanthology.org/N16-1030/, Neural Architectures for Named Entity Recognition