**Encoder-Decoder Model for Sequence-to-Sequence Prediction**

Jack Farah

**Option #2: Encoder-Decoder Model for Sequence-to-Sequence Prediction**

Encoder-Decoder models for sequence-to-sequence predictions are used in many applications. Most of these models use long short-term memory (LSTM) to assist the encoder-decoder model to conduct accurate predictions. The encoder’s role is to process the input and summarize the information into a fixed-length vector. Within this vector is holds information from the entire input sequence. Then the decoder is responsible for generating the output sequence using the context of the vector produced by the encoder. Applications such as speech translation, autonomous vehicles, summarization of documents, and question- answer chatbots can benefit from this styled model.

LSTM networks are designed to address the limitations of a traditional Recuring Neural Network (RNN). RNNs are a type of neural network that feeds the activation function results back as an input. The benefit of this is in sequential data. Given that the sequence of data is crucial for comprehending its complexity, the feedback loop enables us to incorporate multiple data inputs, essentially the sequence, in relation to the preceding information. This makes RNN models a primary choice for sequential data. However, every time the feedback loop unrolls, the complexity of the gradient decent increases. These issues are called the exploding gradient and the vanishing gradient (Cui, Wu, Liu, Zhong & Wang, 2018). During each feedback loop, the weight of that loop is carried on as well. The number of unrolling gets amplified to the final iteration by the weight powered by the number of unrolling. For example, if we had a RNN unroll 5 times with a weight of 2, then the final iteration will have an activation that receives weight, or 32. Five iterations of unrolling is not an issue, but this becomes an issue when the number of sequential data is a larger number. This large number will then be used to find the gradient decent, but it is so large that gradient decent will make huge leaps and never find the optimal result. This results in the exploding gradient. On the other hand, if the weight was very small, such as 0.1, combined with a large number of iterations, the gradient decent will be making insignificant progression that it will practically disappear, resulting in the vanishing gradient. In this case, it would be weight, or . This would take an extremely long time to find the optimal result, making it not worth pursuing.

A diagram of a graph

Description automatically generated LSTM networks successfully address and overcome the limitations of traditional Recurrent Neural Networks (RNNs), revolutionizing how we handle sequential data.

Figure 1- LSTM (Starmer, 2022)

Figure 1 gives us an example of a single iteration of an LSTM model. The bias numbers present are objective, and will be different in every case. The green line on top runs through the full LSTM called the long-term memory, or the cell state. This has no weights and biases, assisting in avoiding the exploding gradient and vanishing gradient. It also helps keep track of previous data throughout the LSTM. The pink line on the bottom is the short- term memory, or the hidden state. The hidden state has weights that can be modified. In the start the hidden state and the cell state both receive the respective inputs of the previous LSTM. These are different than the input in the blue square at the bottom of the model. The blue square input is the actual data that the LSTM will be investigating. There are three main gates in the LSTM model: forget gate, input gate, and output gate. The large blue box with the sigmoid activation function is the forget gate. This gate calculates what percentage of information from the previous cell state should be discarded. The second stage is the input gate, represented with the green and orange boxes. The orange box used the input to create a short-term memory input to create a potential long-term memory, and the green box acts as the forget gate that determines what percentage of the potential long-term memory should be remembered. Once the input gate potential long-term memory is determined, it is added to the long-term memory (green line on top). The final gate, called the output gate, will then receive the potential long-term memory from the input gate and pass it to the pink box to determine the long-term memory to pass on to the next LSTM iteration. The purple box is also used to determine the percentage of the newly found long-term memory should proceed, but additionally, it also determines the final short-term memory will be the official output of the LSTM iteration that can be used as the final answer, or to pass on to the next rollout of the LSTM model. The cell state is designed to carry information across multiple time steps with very little linear interaction. This mechanism ensures that the gradient used to optimize the model is still possible avoiding the gradient explosion and vanishing issues.

LSTM is the perfect use case for long sequences of data. It increases efficiency and avoids potential harm from a traditional RNN model. Applications such as speech translation, autonomous vehicles, document summarization, and chatbots thrive on encoding and decoding LSTM models.

Speech translation is used to convert a sentence from one language to another with the input sentence being transmitted through speech. It is a sequence of data that can be gathered and converted into computable information. This will then be tokenized and can be used as input for encode. The decoding will then generate the response in the language the model is trained in (Jia, et al., 2019).

Similarly, autonomous vehicles may use LSTM to make predictions of the vehicle’s trajectory. It does this by training on data that are a sequence of observations. The general concept is that the short-term memory can judge the current state of the vehicle, and the long-term memory will make the predictions (Park, Kim, Kang, Chung, and Choi, 2018).

Summarization of documents and chatbots also benefits from encoding and decoding LSTM models. Sentences, whether spoken or written, are sequence of data. It is important to understand each current word and the words that came before it to create context. I can say the word “water”, however, there is no value to the word without the full sentence, “I want water.” LSTM will help with creating context to sentences. The encoding portion will find patterns of the given data, and the decoder will respond with relevant information based on the dataset. In summarizations, we can have multiple examples of documents and their corresponding responses as training data. Similarly, chatbots can benefit from question-and-answer pairs.

LSTM is a steppingstone for Transformers, an even more advanced RNN model. Regardless, it is important to understand that there is always a trade-off between implementation, cost, and accuracy. It was explained that speech translation, autonomous vehicles, summarizing documents, and chatbots can benefit immensely from LSTM, but may find some advantages with Transformers depending on the project complexity.

**References:**

Jia, Y., Weiss, R. J., Biadsy, F., Macherey, W., Johnson, M., Chen, Z., & Wu, Y. (2019). Direct speech-to-speech translation with a sequence-to-sequence model. *arXiv preprint arXiv:1904.06037*.

Cui, Q., Wu, S., Liu, Q., Zhong, W., & Wang, L. (2018). MV-RNN: A multi-view recurrent neural network for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, *32*(2), 317-331.

Starmer, J. (2022, November 7). *Long Short-Term Memory (LSTM), clearly explained*. YouTube. https://www.youtube.com/watch?v=YCzL96nL7j0

Park, S. H., Kim, B., Kang, C. M., Chung, C. C., & Choi, J. W. (2018, June). Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In *2018 IEEE intelligent vehicles symposium (IV)* (pp. 1672-1678). IEEE.