**Name**: Chia Bing Xuan

**Matriculation Number**: A0259419R

**Title**: DSA4213 Assignment 2 – Small Language Models Exploration

1. **Introduction**

In this assignment, I conducted experiments on two types of small language models:

* Vanilla Recurrent Neural Networks (RNNs)
* Long Short-Term Memory RNNs (LSTMs)

With the use of a selected corpus, RNNs and LSTMs were built and trained for language modelling. Both qualitative and quantitative metrics were used to evaluate and compare their performances.

1. **Explanation of Models**
   1. **RNNs**

RNNs belong to a family of neural architectures designed to process sequential data like text. In the context of language modelling, they can therefore be trained to sequentially predict the next word, given the preceding sequence of tokens.

Note that RNNs can process input sequences of any length. Suppose we have an input sequence of textual tokens, where is the sequence length. At each sequential time step , the corresponding token is fed into the RNN. It is first converted into a word embedding with the transformation matrix

Note that the RNN possesses a hidden state that will be updated during each time step. During time step , the hidden state from the previous step, , will be updated using the word embedding to yield the new hidden state

where is the sigmoid function and , ,are learnable parameters in the RNN.

Finally, the hidden state of the current time step is used to output a probability distribution over all distinct tokens in the vocabulary :

where andare learnable parameters in the RNN. To predict the next token, it can be chosen by selecting the token corresponding to the highest probability. Alternatively, the next token can be sampled from the output probability distribution.

Note that the same weights , will be repeatedly applied on every time step. They will only be updated through gradient descent after all the input tokens in the sequence have been processed. Hence, there is symmetry in how these input tokens are being processed.

* 1. **LSTMs**

LSTMs are RNNs that aim to alleviate the vanishing gradient problem – an issue which makes it difficult for vanilla RNNs to learn long-range dependencies and preserve information over many timesteps.

Suppose we are at the end of the previous time step . The LSTM has a hidden state and a cell state , the latter of which can store long-term information. To update the hidden state and cell state for the current time step , we first determine three gates – forget gate , input gate and output gate . We also determine the new cell content that can be written to the cell in this time step, . These vectors are dynamically obtained based on the current word embedding :

where the ’s, ’s and ’s are learnable parameters.

To determine the new cell state , the forget gate selectively removes some content from the previous cell state , whilst the input gate selectively writes some new cell content from :

where refers to element-wise multiplication.

Finally, the output gate selectively reads some content from the updated cell state , so as to obtain the updated hidden state for the current time step:

With the presence of forget, input and output gates, LSTMs are better able to preserve information over many time steps within its cell state, thereby serving as a solution to the vanishing gradient problem.

1. **Methodology**
   1. **Data Processing**
   2. **Model Training**
2. **Evaluation**
3. **Conclusion**
4. **Appendix**