Overview of Seq2seq model

1 Introduction

The sequence-to-sequence (seq2seq) model has emerged as a powerful tool in the machine learning realm, particularly in the domain of Natural Language Processing (NLP)[1]. Applications include language translation, image captioning, conversational models, and text summarization. It was first introduced for machine translation, by Google. Before that, the translation worked in a very naive way. Each word that you used to type was converted to its target language giving no regard to its grammar and sentence structure. Seq2seq revolutionized the process of translation by making use of deep learning. It not only takes the current word/input into account while translating but also its neighborhood.

2 Architecture

The model consists of two main components:

- Encode
- Decode

2.1 Encode

The encoder's primary role is to understand and compress the information from the input sequence into a fixed-size context vector.

Input Sequence: The encoder takes a sequence (e.g., a sentence) as its input. This sequence is typically represented as a list of tokens. In NLP tasks, tokens can be words or subwords, and each token is often represented by a unique integer or a vector (word embedding).

Recurrent Cells: The encoder processes the input sequence using a recurrent architecture, which can be a basic RNN, LSTM, or GRU. Each of these has its strengths:

- RNN: Simplest form but can suffer from vanishing gradient issues, making it less effective for long sequences.
- LSTM: A type of RNN that has gating mechanisms (input, output, and forget gates). It can capture long-range dependencies and is less susceptible to the vanishing gradient problem [2].
- GRU: A simpler version of LSTM with fewer gates, offering a balance between complexity and performance.

Context Vector: As the encoder processes each token of the input sequence, it maintains a hidden state. The hidden state from the last token is used as the context vector, which is expected to capture the essence of the entire input sequence. This context vector is passed to the decoder.

2.2 Decode

The decoder's task is to generate the target sequence from the context vector provided by the encoder.

Context Vector: The decoder starts with the context vector received from the encoder. This vector serves as its initial hidden state.

Recurrent Cells: Like the encoder, the decoder also contains a recurrent architecture (RNN, LSTM, or GRU). It uses this to generate the output sequence token by token.

Token Generation: At each step, the decoder generates a probability distribution over the possible output tokens. The token with the highest probability can be chosen as the output for that step. This chosen token is then fed as input to the next step of the decoder.

End Token: The decoder continues generating tokens until it produces an end-of-sequence token or until a maximum sequence length is reached.

3 Extensions

Attention Mechanism: Traditional seq2seq models can struggle with very long input sequences. To address this, the attention mechanism was introduced. Instead of relying solely on the fixed-size context vector, the decoder uses attention weights to focus on different parts of the input sequence at each decoding step. This allows the model to capture more nuanced relationships between the input and output sequences[3].

4 Conclusion

The seq2seq model, with its encoder-decoder architecture, has been foundational in numerous sequence generation tasks. Its modular structure has also made it amenable to various enhancements, ensuring its continued relevance in the rapidly evolving landscape of deep learning and NLP.

References

[1] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[2] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[3] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).