A Comparison of N-gram Model and RNNLM

Snehal Somalraju

Introduction

In natural language processing, one of the open challenges is to create an accurate language model, which is a probability distribution of word sequences. A language model can be used to predict the next word based on a sequence. For instance, take the phrase: "*Please turn your homework…*" An accurate language model would predict that the next word would be "in" or "over," but not "dog" or "eat."

A real-world application of language models is speech recognition, which involves tasks where words need to be identified from noisy, ambiguous input. Suppose a speech recognition system needs to determine whether an audio signal represents "*I will be back soonish*" and "*I will be bassoon dish.*" It would be helpful to know that the former phrase is much more likely than the latter phrase.

Several techniques have been devised to create a language model. We will compare two widely used approaches: n-gram models and RNNLM.

N-gram Model

The n-gram model is a conventional technique used to create language models. N-gram models are built by counting how often word sequences occur in a corpus text and then estimating the probabilities.

Suppose there are two sentences: "*There was heavy rain*" and "*There was heavy flood.*" Suppose we used two word sequences to calculate the probabilities, which would be called a bigram. "*Heavy rain*" would be more likely to occur than "*heavy flood*," so "*There was heavy rain*" would more likely be selected.

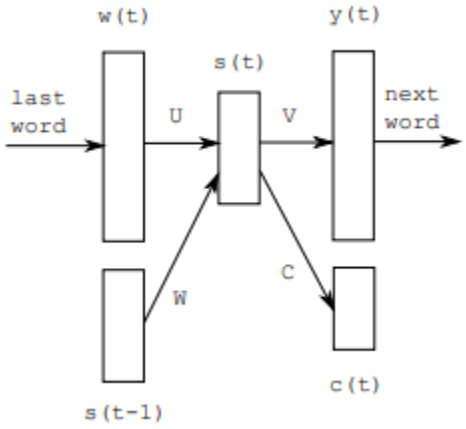
However, n-grams come with their disadvantages. N-grams model may have a sparsity issues, as certain word sequence may not appear in a corpus of text, and smoothing is required to assign probability to it.

Recurrent Neural Networks

An alternate approach is to use recurrent neural networks to create a language model. These language models are called **recurrent neural net language model** (RNLLM). Recurrent neural networks are neural networks that allow previous outputs to be used as inputs. This property can be referred to as memory.

We can see how this would be applicable to creating language models, as it involves computing the probability of a world based on a previous sequence of words. The previous sequence of words can be stored as memory in the RNN.

To explain how recurrent neural networks based language models work, we can examine a toolkit built by Microsoft. Here is the following diagram:



The input layer uses the probability of the previous word (w(t)) and the previous state of the hidden layer (s(t-1)). The neurons in the hidden layer uses a sigmoid activation functions (s(t)) and outputs the probability distribution of the next word given the previous word (y(t)). The class layers can be optionally used in the toolkit to reduce the computational complexity of the model, though accuracy may be slightly compromised.

Studies have shown that language models built with recurrent neural networks have higher performance than those built with n-grams. Another advantage of using RNN is that it is possible to effectively sequences and patterns with arbitrary length. However, RNNs can be computationally expensive.

Conclusion

Due to the advantages of using RNNLM, they may soon become the norm. However, RNNLMs do have their challenges as they are computationally expensive. Fortunately, several improvements have been proposed, such as CUED-RNLLM, which contains GPU-based training for RNNs. Thus, we will reach closer to our dreams of creating machines that can emulate sentences from human languages.

References

https://web.stanford.edu/~jurafsky/slp3/3.pdf

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture06-rnnlm.pdf

https://www.microsoft.com/en-us/research/wp-content/uploads/2011/12/ASRU-Demo-2011.pdf