Lecture-6

The Language modelling task aims at predicting the word which comes next. To perform this task, we first make a simplifying assumption that the word only depends on the previous n words, also called the window size.

We then count the frequency of the n-gram divided by the n-1 gram to get a probability for each word. We might want to assign a small constant term to each frequency count so that we don't get a zero probability of for any word. Also, we backoff if the denominator is zero.

There is an important tradeoff here at play. We want to use a higher-order n-gram, which means we are utilising more context information. But increasing the n amplifies the sparsity problem.

We next try the fixed window neural-based approach. We assume a fixed window size, just as before, and build the neural network configuration just like we did in named entity recognition. The nice thing is that we always have some probability for every word even if it hadn't occurred even once in the context. The problem is that we can't have a big enough window size. Another more subtle problem is that the weight multiplication is local, leading to the construction of a lot of similar functions without completely utilising the context.

Next is the mighty RNN. It has a single hidden matrix which remains the same for any length context. The word embeddings are multiplied with the W matrix, which is summed with the multiplication of the hidden state and the W' matrix(plus the bias ofcourse). The result is a new hidden state. The benefit is that it can handle any length sentence while keeping the model size as constant. It can also theoretically capture any length context. The main disadvantage is that the computation is linear and hence slow.

Training an RNN is simple. We try to predict the probability of the next word each step and calculate the cross-entropy loss. We can compute total loss on a bunch of sentences and take a step. Perplexity is used as the evaluation metric, which is just the exponentiation of the cross-entropy loss.

LM is a benchmark task for measuring progress in the underlying language. It is also a subcomponent of many downstream tasks.

RNN is not LM. It can be used in various tasks, even where the input is fixed and not sequential.