Com Sci 263

Lecture 4

Objective function: Using parameters, you need to maximize the log likelihood, and minimize the negative log likelihood. We use E as the word embedding matrix.

Recurrent Neural Networks(RNNs): You share in certain past states, which is similar to short term memory. The most recent term has the largest weight out of all other previous terms, and this is cumulative. These states then give the neural network context, however not the best due to vanishing and exploding gradients when the amount of sequential information is large. In this case, the context for the neural network can be considered as the hidden states that are passed from one unrolled state to another.

LTSM :

Special type of Recurrent Neural Network.  
The extra thing you do: instead of just using a tanh activation function, you use a more complicated form for it. Below is a LSTM cell. The mathematical calculations are depicted below

A diagram of a diagram of a layer

Description automatically generated

Above is a single unrolled unit for the LSTM. Instead of using multiplications(Vanilla RNNs), it turns them into additions which solves the vanishing gradient problem

Input gate: What information will be taken from the current input and stored in the cell stat/

Consists of two parts:

Sigmoid function(Used as the activation(how much information we should retain)

And the tanh activation function.

Forget gate: Decides what information will be thrown away

A math equations on a white background

Description automatically generated

How do you calculate the new state:

These are the formulas that capture this(Mentioned above)

Dimensions for Wi and Wc are the same. ft is used for the forget gate, we do a dot product with Ct-1 to decide what context goes in. Wo is the output parameters.

RNN/LSTM Language models:

Training the RNN language model:

Steps:

1. Have a big corpus of text-> sequence of words
2. Feed into RNN LM-> compute output distribution for every step t-> probability distribution for every word given all previous words.
3. Loss Function: Cross entropy loss on step t-> on predicted probability distribution and the “true” next word. Average this to get overall loss for the entire training set (summation and then divide to average). Note: During training time, you only base the loss function on true previous word inputs to predict the next word.

Decoding: Selecting the word to generate at each time step:

Greedy step: We always select the word with the highest probability at each step

Problems: No diversity(will produce the same output every time)

Early mistakes will amplify

Beam Search: Explore several different hypothesis instead of just a single one. Keep track of k most probable partial translations, and capture those outputs.

How do we navigate pruning: Pruning is done by (anding the scores of the previous outputs, and chose the highest two only).

There is no guarantee that this would produce the highest and most probable sequence, as it might prune out branches that may have higher probabilities afterwards.

Effect of changing beam size k: Small k increases the greediness of the model-> unnatural, and incorrect due to errors earlier in the decoding stage. Larger k means you consider more hypotheses, however, is more computationally expensive and can introduce other problems.

Top-k sampling -> you sample based on the top k choices.

Top-p sampling- > based on a threshold to truncate the probability distribution.

Top-P sampling: At each step t, instead of randomly sampling from p(xt|x1:t-1)