CAP 6640 – Natural Language Processing
Homework 3
February 21, 2019
Due: in 5 days (due by Feb 26, 2019: firm).
Preferred typesetting system LaTeX

Answer all of the following questions. Keep in mind the following instructions: individual work, cite all references, document your code, type in all answers, and in mathematical expression, use standard notations.

1. What is language modeling? Give two possible use case.

   Language Modeling is process of predicting what word comes next in a given corpus. Traditionally this can be seen as $P(x^{(t+1)}|x^{(t)}, ..., x^{(1)})$. This allows a distribution to be formed so based on our training we can then determine the pattern of the language and use this information to predict the next word. Google search uses this technology when users begin to type in the search menu. Google will suggest to the user what they might be Google-ing. Another example of language modeling can be seen by classifying words in the component structure so like noun, verbs, adjective, etc to then use this create another corpus based on a prescribed context possibly.

2. Explain n-grams technique with an example. List two limitations and possible solutions.

   The n-gram technique uses a ”chunk” of sequential words to predict the next words. If our corpus was: ”My pride and joy was sailing that boat.” with n = 2. The chunks would be:

   1. My pride, 2. pride and, 3. and joy, 4. joy was, 5. was sailing, 6. sailing that, 8. that boat

   If n=1 then it would be the indivual words themselves as the chunks, and if n = 3 then the few chunks would be: *1. My pride and, 2. pride and joy, 3. and joy was.*

   Limitations in this technique are sparsity and storage. Sparse data can limit the model because it could be trained on a local population instead of a global population relative to the trained corpus. This could cause problems when we encounter something in the testing data we have not trained for in the training data. Sparsity can also cause issues in that if something never occurs in the training set then w has a probability of 0. A partial solution to this would be to add a small number to w to the frequency count cause it to not be 0. In general having an n less than 5 keeps that sparsity manageable but a balance needs to be made in the intelligence of the model, its sparsity, and the storage needed. If sparsity because too much an issue then a larger corpus can be used to train a new model. Yet keep in mind a larger corpus give for a larger need for storage for the model.

3. Explain fixed window language model mathematically. Explain the limitations, their foundations and how it improves n-gram language model.

Language models take in sequences of words for 1 to t, and will output a probability distribution of the next word. Using a fixed window model we can take the window of words as our input. These words are then concatenated together in the next layer. Passing through these concatenations will form a hidden layer which will be passed through a softmax function to calculate a probability distribution of the next word from the window. Mathematically the inputs for a three word window would be: $x^{(1)}, x^{(2)}, x^{(3)}$. This input X would be then fed to the next layer where the words are concatenated in the to e = $[e^{(1)}, e^{(2)}, e^{(3)}]$. Note the length of vector is based on the window, so if we had a four word fixed window then e would have a size of 4 like x. The concatenation of the words would then be fed into the hidden layer where its function follows: $h = f(We + b_1)$. This hidden layer is passed through a soft max function where $\hat{y} = softmax(Uh + b_2) \in \mathbb{R}^{|V|}$. The limitations in a fixed window model are in the size of the window. Too small, theres no context and its been proven we cant make our window large enough for better representations. Additionally there is no symmetry since our inputs used non identical weights. The improvements over n-grams are in its main issues: sparsity and storage. The fixed window LM solves that sparsity problem and through the window we dont need to store all n-grams.

4. Explain the RNN language model (including training and backpropagation) mathematically and list 2 advantages and 2 disadvantages.

The RNN Language Model allows us to predict a probability distribution of every word, given the words before it. Essentially the corpus is a sequence of words, and each word is the input for the Neural Network. The input words are passed to the word embedding along with the weight matrix W_e. Next we initialize an initial hidden state $h^{(0)}$, where $h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$. W_h is the weights for the hidden layer that are repeatedly used to calculate the next hidden layer. So the model will use the first word and its embedding, and use the initialized hidden state to calculate the next hidden layer. Then the next word is added to the sequence for the word embedding which is passed to the next hidden layer along with the bring hidden layer and its weights. The continues until we pass the output into a softmax function that will give us a distribution of the probabilities, which can be seen as $\hat{y}^{(t)} = softmax(Uh^{(t)} + b_2) \in \mathbb{R}^{|V|}$. The loss function is used primarily to train the model, shows the cross-entropy between the predicted probability distribution $\hat{y}^{(t)}$ and the true next word $y^{(t)}_{x_{t+1}}$. So the loss function is

$$J_t(\theta) = CE(y^{(t)}, \hat{y}^{(t)}) = -log(y^{(t)}_{x_{t+1}})$$

. In terms of the entire data set we average $J(\theta)$ so that

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} -log(y^{(t)}_{x_{t+1}})$$

During the training process we compute $J(\theta)$ for a batch of sentences, compute gradients and update the weights, and repeat this. By using SGD we can save computation. Using back-propagation we can update these new weights and update the model, and this is done by $\frac{\partial J^t}{\partial W_h} = \sum_{i=1}^{t} \frac{\partial J^t}{\partial W_h}|_{(i)}$. This is considered back-propagation through time because we are summing the gradients as we go to the next previous layer.

Advantages to using an RNN language model are they can process any length of data, so this can be really useful for time series data sets. RNN has symmetry in processing its inputs since the weights are applied at every time step and the model size does not increase as we use more data. Disadvantages to using an RNN are that is can be very computationally heavy and in development, an RNN might have problems trying to access previous information in their respective step.

5. Explain vanishing and exploding gradient problem. Explain how LSTM and GRU solve the vanishing gradients problem and compare these two methods.

   The vanishing gradient refers to the phenomenon when a gradient signal from faraway is lost because it is much smaller than gradient signals close by. So in other words, the gradient is only able to take into account local gradient signals and not signals way in the past. This causes issues because we cant tell which words are dependent of each other as there is certainly missing information. Additionally we wont know if we have wrong parameter leaving a bias in our results.

   The exploding gradients refers to the phenomenon when the gradient get extremely large which causes a bad update in the model. Observing exploding gradients we see the steps are far to large, and so we need a slower learning rate. This can be seen by graphing $J(\theta)$ as we walk down the slope the gradient will overshoot in reality. A solution to this is to set a threshold and if the gradient surpasses this gradient, clip it before applying the SGD update. This allows us to walk down this slope and if we surpass this threshold we recalculate the gradient with a smaller learning rate or step size.

   LSTM or Long-Short Term Memory solves these problems by using cells to store long term information. LSTM's can read, write, and erase information in its cells. This selection of information is determined by three different gates: the forget, input, and output gates. Each can be closed (0), open(1), or between two numbers. Each gate is dependent on their current context. The forget gate controls what it stored vs erased from the previous cell state. The input gate controls what sections of the new cell content are stored. The output gate what sections of the cell are output to the hidden state. LSTM has three cells are: New cell content, Cell state, and the Hidden state. The New cell content is stores the new content. The Cell State erases the forget gate from the last cell state and write the input gate for new cell content. The hidden state then reads the output gate from the cell state. So in theory LSTM solve the problems by being able to store information easier for the RNN to preserve over multiple time steps in the future. LSTM in now way guarantees we wont see the gradient problems but it does allow for a more intuitive way to model dependencies that are far in the future.

   Gated Recurrent Units (GRU) was proposed as an alternative to LSTM. For each time step we have an input and a hidden state with no cell states. GRU has fewer parameters and is ultimately quicker to update than LSTM. GRU uses its New hidden state content which is the reset gate, and the Hidden state which is the update gate allow for better control of information as time pass through. Formally the update gate controls which areas of the hidden state are stored vs updated. The second gate, the reset gate determines which of the previous hidden states are to be used for the new content calculation. The New hidden state content uses the reset gate by choosing the portions of the previous hidden state and then use this and current input to calculate the new hidden content. The Hidden state uses the update gate to monitor what is stored from the previous hidden state, and what is updated to the new hidden state. Research have not been able to conclude and major differences between the techniques. It has been mentioned to start off with LSTM and if you want your model more streamlines try the GRU next.

References:

Class Slides