Modeling

Zachary Fitzgerald – 100769868

AIDI-2005 Capstone II

Marcos Bittencourt

February 20, 2020

In this module I will be exploring models that I am looking at using for this project. I will list each model and evaluate them to see how I would be able to use them within my project. Following the first section I will be listing the software tools that I will be using for the project.

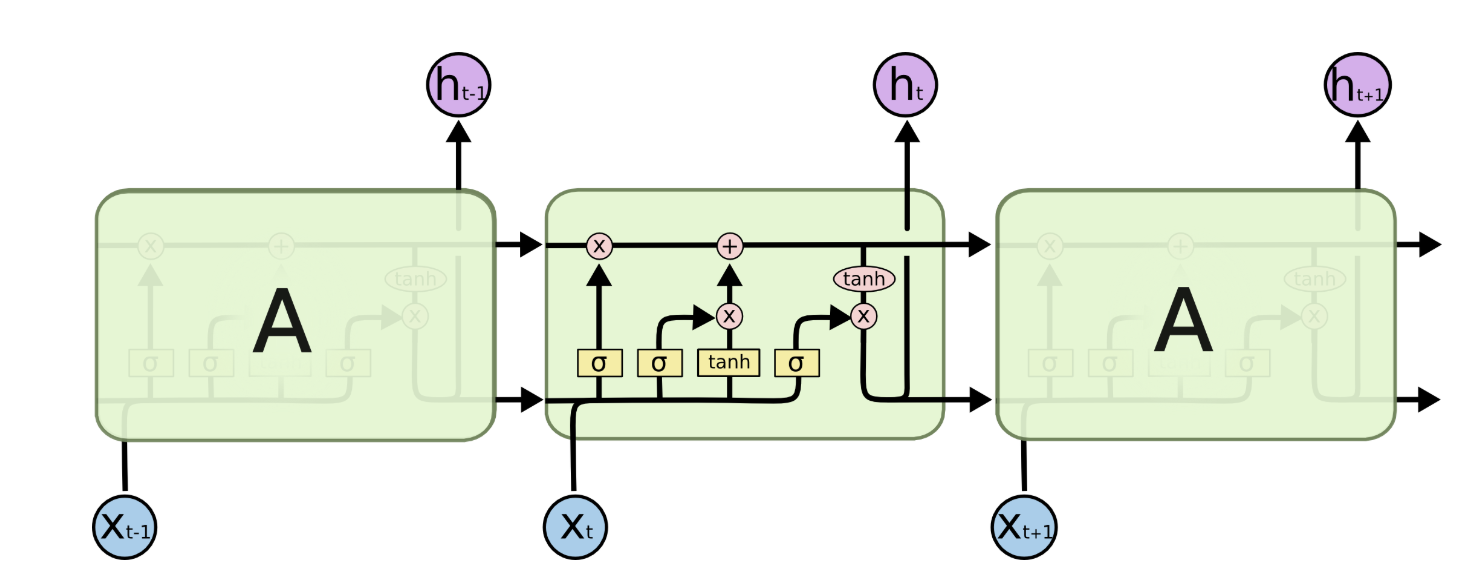
**Model Structures**

**Sentiment Analysis LSTM Network**

I have already built this model and it can be found within my GitHub repository. LSTM stands for long short-term memory. These models are great at working with long term dependencies such as linguistic variables. Since my project is focused on working with text data, a LSTM network is perfect.

LSTM’s are a form of RNN (recurrent neural network). They are designed to combat long term dependency issues. Remembering information for long periods of time is essentially what they do best. This is very important when looking at sentiment analysis as it is crucial to keep word/body of text context within the program’s memory. This helps my project to be able to predict sentiment of articles or simple paragraphs of text with relative ease.

Here is a visualization of a LSTM network.



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

**Recurrent Neural Network for Text Summarization**

This is the main model that I will be working on throughout the project. In particular I will be creating a sequence-to-sequence network which is the standard for text summarization within the natural language processing community. There is a key drawback to using sequence-to-sequence networks for text summarization. I will speak about this after I show a visualization of this model.

Sequence-to-sequence network

A close up of a map

Description automatically generated

This visualization shows a basic sequence-to-sequence network. The source text is a sentence about Germany winning a sports game. I’ll explain this model in the simplest way possible. The text is first put through an encoder so that the system can understand it. Through numerous other tasks the system ends up using context vectors and the attention distribution to decide what words are most important. The final step of this model is to return a partial summary of the text using words from the vocabulary distribution. One major issue with this model is that the text can be either false or it will simply just print out the same words repeated after one another. To solve this, I will need to use a point-generator network.

I will also be prototyping a point-generator network which is a hybrid network. This network can choose to copy words from the source via pointing, while retaining the ability to generate words from the fixed vocabulary. This model will be able to solve the issues that arise from the sequence-to-sequence model. Below you can find a visualization of this network.

A close up of a map

Description automatically generated

This diagram shows the third step of the decoder, when we have so far generated the partial summary Germany beat. As before, we calculate an attention distribution and a vocabulary distribution. However, we also calculate the generation probability, which is a scalar value between 0 and 1. This represents the probability of generating a word from the vocabulary, versus copying a word from the source. This is the most important difference from the above model, being able to choose if we want to copy a word from the source or generating a word from the vocabulary. This also solves our issues that we had with the sequence-to-sequence model.

Point-generator networks are faster to train and more accurate with less training than sequence-to-sequence networks. It is far better suited to take words from out-of-vocabulary text, which means that it can generate unseen words to make the summary better. Lastly, the model makes it very easy to copy words from source text. It is clear that the point-generator network will work the best for my project. That is why I have proposed to work with it.

**Software Tools**

I will be using the following tools to complete this project

* Tensorflow
* Keras
* Numpy
* Pandas
* NLTK (for NLP projects)

All of these tools are python libraries which are already installed on my computer. I am well versed with them and will be able to extract the full power of each of these libraries.

**Modeling Part Two**

In this module I will be discussing my proposed model architectures and software pipeline. I will also speak about the assumptions, limitations, and constraints about my dataset. Finally, I will choose my candidate model and create a model selection scorecard.

**Model Architecture/Software pipeline**

**Sentiment Analysis LSTM Network**

Using the visualization that I presented in part one of the modeling we can see how LSTM’s are set up. My model consists of a cell (the memory part of the LSTM unit) and three "regulators", usually called gates, of the flow of information inside the LSTM unit: an input gate, an output gate and a forget gate. As the neural network iterates, information gets added or removed via gates. The gates are different neural networks that decide which information is allowed. The gates can learn what information is relevant to keep or forget during training.

I mentioned that there are three gates or “regulators” these are the forget gate, input gate, and output gate. The forget gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. The input gate updates the cell state with information that the model deems to be important. Finally, the output gate decides what the next hidden state should be. The hidden state contains information on previous inputs.

All of this information can seem very complicated, it is much easier to understand LSTM networks by following a tutorial online although I have tried my best to explain them here. This architecture is what I have used for my sentiment analysis neural network. This will be what I am using to measure the sentiment of each body of text within the dataset I have chosen.

**Recurrent Neural Network for Text Summarization**

The main model that I have chosen to work on for this project is a point-generator network. A visualization of this model is displayed in part one of the modeling. I have already gone over the architecture of this model, but I will paste that same body of text here so that you do not have to go back and look.

First, we must calculate an attention distribution and a vocabulary distribution. However, we also calculate the generation probability​​, which is a scalar value between 0 and 1. This represents the probability of generating a word from the vocabulary, versus copying a word from the source. The generation probability​ is used to weigh and combine the vocabulary distribution (which we use for generating) and the attention distribution (which we use for pointing to source words​​) into the final distribution. Using the following formula: A screenshot of a cell phone

Description automatically generated

We can find the probability of producing word (w) is equal to the probability of generating it from the vocabulary (multiplied by the generation probability) plus the probability of pointing to it anywhere it appears in the source text (multiplied by the copying probability). This network will be used to summarize the text found in documents submitted into my program.

**Dataset Assumptions, Limitations, and Constraints**

**Limitations:**

* My knowledge of NLP will need to improve in order to ensure that the performance of this model is the best that it can be.
* There are 1.5 million entries within the dataset. This will require a lot of computing power unless I want to use less of the dataset.
  + If I shorten the dataset, then the model might not perform as well.
* I will need to improve the NN that I made to predict sentiment because it was not trained on data like this.
* The data within the dataset is focuses on ‘HowTo’ articles. This could limit the generalizability of the results.

**Assumptions:**

* I assume that the text found within the dataset is complete and has actual sentences.
  + It is hard to manually check every single entry for completeness and coherency because there are over 1.5 million entries.

**Constraints**

* Time: in order to train a model successfully, I will need plenty of time since it will potentially take days of training to complete.

**Model Selection Scorecard**

To validate my models, I will be using cross-validation as it is both the most accurate but also most computationally expensive form of model selection and validation. Since I have already chosen the models that I will be using for this project I feel that a model selection scorecard would be wasted. It is clear that I will be using a LSTM network for sentiment analysis and pairing that with a recurrent neural network called a point-generator network. I have already completed the LSTM model and will be improving it in the coming weeks to work well with my current dataset. I will begin work on the point-generator network which I have chosen due to the simple fact that it is better and more suited for my dataset than the alternative which is a sequence-to-sequence network. All of my thoughts have been outlined in the model structures and the model architecture sections of this paper.