Data Visualization: Final project

RNN Occlusion visualization

Reid McIlroy-Young

June, 2018

Introduction

The existing work on visualization of of recurrent neural networks (RNN) is quite lacking, the best I am aware of is a sequence to sequence and word/character model visualizer that uses a parallel coordinates plot: lstm.seas.harvard.edu/client/index.html. This is a visual of how the Long Short Term Memory (LSTM) layers are working but is busy, complex and unintuitive. There have been a few other pieces of work done on the topic, mostly focusing on sequence to sequence or generative models, but none on classifiers that I could find. The goal of this project is get a better visualizer of of LSTM/RNN classifiers as that is what I am using in my work.

For this project I limited my scope to a single pre-trained RNN, the RNN reads social science papers an assess how computational they are, with positive 1 being very and negative 0 being not at all. It does this by reading the papers abstract and title which have been converted to a sequence of word2vec vectors after standard

tokenization and normalization (i.e. regex based tokenizing). Internally there are 8 LSTM sections four for the title and four for the abstract with two stacked and reading forward and the other two stacked and reading backwards. Then the final outputs are combined by a single layer of neurons.

Methods

The first attempt I made was looking at the activations of the RNN, Figure 1, tries to show the activations at each word that were fed to the single layer before being converted to a binary output. Note, that in main usage and training only the last output is used. This display has numerous problems, first of which is that it is attempting to display 512× the number of words, floating point values and has no good way of relating them spatially as column 1 and 2 could be just as related as columns 1 and 400 so the x-axis is categorical, but the scale is too small for this to be obvious. Additionally the magnitude of the activation is not guaranteed to correlate with significance, so this display is even more fundamentally flawed.

The second attempt I made is at mapping NN with each neuron a node in a graph and each connection an edge. This had many issues, first the number of edges is huge, so the network appears to be a hairball, unless carefully laid out and second once laid out nothing is visible since the layout must be huge. My solution to this was to plot the nodes and edges on a map, but this failed as there is not really a good way of doing it in R. I think this is still worth considering but right now is non-trivial.

The third attempt was inspired by work done in convolutional neural networks (CNNs). CNNs often have a direct visual interpretation since they are based on

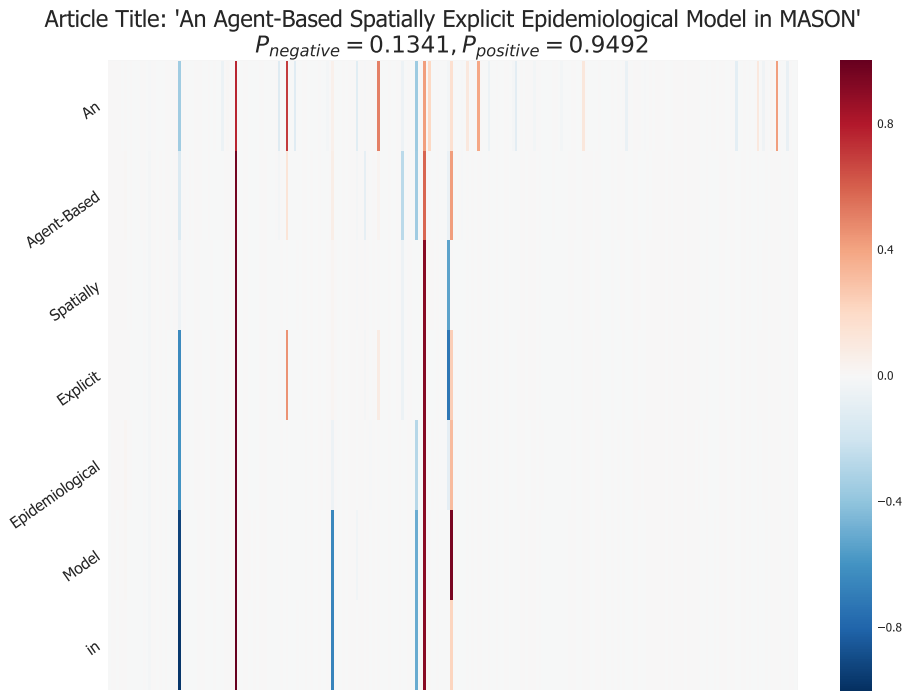

Figure 1: *Example of output activations for a positive example*

naturally occurring neural structures for processing images. A method used to some success in convolutional neural network (CNN) visualization and introspection is the *occlusion experiment*, where an image is broken into subsections and the CNN is asked to make a classification with the subsection removed. This method allows one to measure the significance of each of the subsections by measuring how their removal affects the classification certainty. Figure 2 shows the effects of removing words from the title and abstract on the classification probability for a toy example with the title of *Exploration of humans, society and R* and abstract '*We used methods and techniques to do stuff. Weber Freud and vocabulary acquisition were explored. Then we did more stuff. Our code can be found on Github.*'.

We can see in the bottom left that if the title is *Exploration* and the abstract simply *We used methods and techniques to do stuff.* then the full model gives it a 0.64 probability of being computational, but if you add the sentence *Weber Freud and vocabulary acquisition were explored.* the probability drops to 0.41. Similarly adding the final sentence *Our code can be found on Github.* firmly puts the record as computational with a probability of 0.89 and if *R* is added to the title, the probability goes up to 0.95.

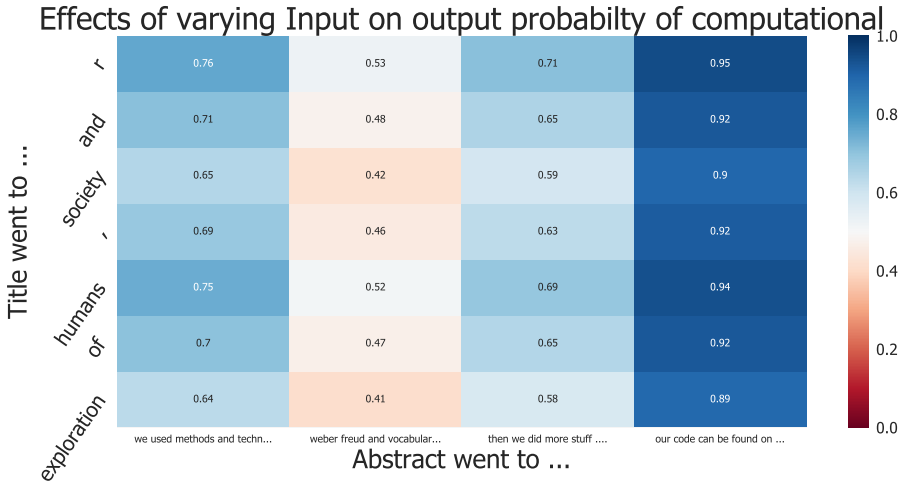

Figure 2: *Model prediction as the title and abstract are added to, colour indicates the probability of the record being computational with blue being high and red low.*

Getting R and Python to interoperate nicely was very challenging and basically fruitless. I used *reticulate* to wrap Python for R which works well for pandas and numpy operations, and just those. If other stuff is brought in the interface becomes unstable. Beyond general instability, the wrapper is inefficient, code that was running fine in pure Python became slower and more memory intensive in *reticulate*. The largest cause of this was the word2vec embeddings which are loaded into memory for faster lookup. To solve this I converted them into a zip file

that contains one file for each word in the vocab with the word as it's title and the vector as its value. This makes lookups a disk operation, which is much slower, but requires almost no memory overhead and can be done with only the standard library. But even with this hack the systems interface would still fail frequently. To solve interfacing issues I resorted to writing the heatmaps to a temporary csv file, then reading that with R. This works sometimes, but on the server would crash R. So I have a process running to create the csvs for each of the records in the collection, but at 800 records an 1-2 minutes each (server is slow) I ran out of time and wasn't able to get a stable flexdashboard app running. You can look in *dash.Rmd* to see the framework of the app.

Results

My final results thus are the heatmaps, these are a novel way of visualizing RNN classifiers and as such I believe are enough for this assignment. The code in the Rmarkdown file creates a series of heatmaps from selected records and has the capacity to do dynamic input, but because of the Word2Vec being in a zip preparation takes much longer than running the model. If I had a week to work on this, and funding for a real server I could get something pretty running, but I don't. Thus you can try running the code, or look at the images generated in the images directory.

The actual visuals produced use a heat map as that's a good way of showing 2D data, they all use the same colour scheme with the same 0-1 scale to make cross comparison work. the colours are a diverging red to blue pallet that nicely goes to white at .5 which is the turning point of the predictions and thus is useful to be

noted. Figure 3 shows an output on real data, notice that the title and abstract's changes are both visible as lines vertically and horizontally. I beilve some kind of blending of cells might work better for longer abstracts as they do tend to get crowded, also the numbers on each cell are optional.

Asking how truthful deep NN visuals are is always tricky, and I would have to do some math to see how significant a value change has to be to be important, but you can clearly see how the RNN is affected by words being added, and get a sense of how they affect the outcome. Hopefully this will lead to better visuals as this is certainly a first pass.

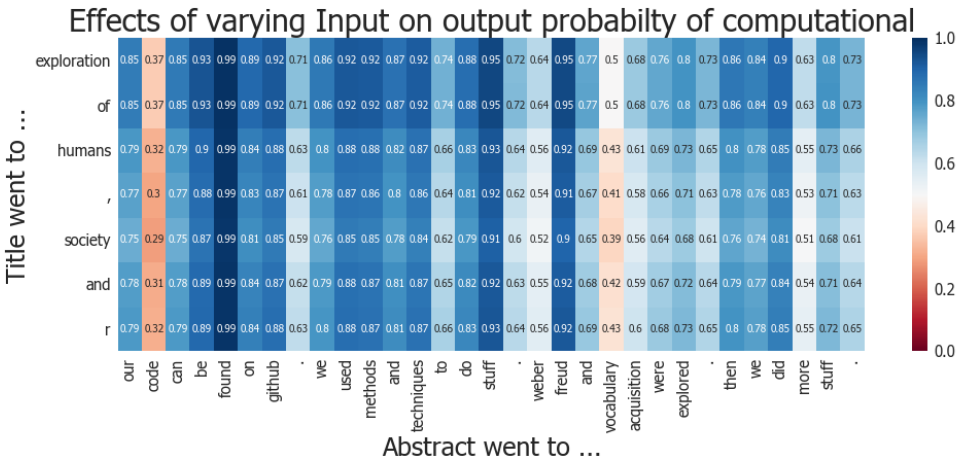

Figure 3: *Model prediction as the title and abstract are added to, colour indicates the probability of the record being computational with blue being high and red low.*