What is the best network architecture to extract sentiment from text?

Tanish Baranwal

Dublin High School

Abstract

The purpose of the project is to determine the best Deep Learning(DL)/Neural Network(NN) architecture to learn to extract sentiment from human statements. Sentiment Analysis is a crucial task in much of Natural Language Processing(NLP). NLP is a subset of the Field of Machine Learning(ML), which involves using large amounts of data to learn patterns and predict. Each major model type was used to obtain the best accuracy while trying to stay below a runtime of 0.5 seconds. The Fully Connected(FC) layers and Convolutional Neural Networks(CNN) had accuracies of under 65%, which showed that their predictions were nearly random chance. The Recurrent Neural Networks(RNN) and Bidirectional RNNs(BRNN) did well, which accuracies between 76% and 78%, but only the 2 Layer RNN had the best accuracy of 77.5% while running in 0.256 seconds, staying under the threshold of 0.5 seconds. This data shows that RNNs and BRNNs can best capture the long range dependencies of the english language. Findings may be useful for further NLP research.

What is the best network architecture to extract sentiment from text?

Introduction

ML is a system learning system where data is utilized to learn patterns make out meaningful results, and predict from multiple applications. DL/NN are specialized techniques to achieve ML goal more efficiently.

The field of ML- especially DL has been growing rapidly in the past several years. Over the years, the conventional amount of enterprise data have increased from about 10,000 - 100,000 to 1,000,000 - 1,000,000,000. This caused a unique issue that researchers hadn't encountered before; the limitations of compute power to train models. To create deeper neural networks, they needed a level of computational power that the time just couldn't provide. This is why the field of ML plateaued for years. Then the rise of computational power allowed researchers to implement deeper, higher dimensional neural networks. New technological advances brought Graphical Processing Units (GPUs) , one of the most efficient hardware for the high dimensional matrix multiplication required for deep learning. Again, the only limitation became the algorithms and architectures of the models, while we could constantly collect more data to improve accuracy.

Now, the field of DL has split into multiple major groups: Computer Vision, Audio understanding and NLP to name some. My interest leaned toward NLP, so I looked further into it. There are a few major industries that NLP can completely redesign. These are customer service, virtual assistants, information retrieval.

In customer assistance, chatbots can streamline customer service, taking care of simple tasks and questions and leaving complex queries to their human counterparts. In the future, DL models could analyze a call and rate the customer satisfaction through sentiment analysis. Virtual assistants use natural language understanding techniques to extract commands from your speech. Information retrieval extract valuable information from unstructured text by using sentiment analysis, and abstractive summarization.

For my project, I chose sentiment analysis, because it is very versatile and is used in almost everything in NLP Sentiment Analysis is used commercially as a way to extract information from consumers. For example, it can be used to analyze customer reviews to find out the general opinion. It can also be applied to live conversations in hand with speech to text systems. It is also very easy to evaluate on it's accuracy. Like any DL problem, I started by sourcing my data.

Method

Data

For this, I am using the twitter data set, which has 1.05 million pieces of data, each labeled positive or negative.

Model

Since words cannot be mathematically manipulated, I used both GloVe word embeddings that represent the words as a 50 dimensional vector, and Fasttext word embeddings that represent the words as a 300 dimensional vector. To build my model, I used a few different technologies. I used a CNN, that applies multiple learnable filters, RNN and BRNN along with Long Short Term Memory (LSTM) units to allow the neural network to "remember" information, and a vanilla neural network, with fully connected layers of neurons. Conventionally, CNNs are used in images to map thousands of pixel values to a classifier. RNNs/BRNNs are used in sequence to sequence applications, where a sequence like a sentence or audio is the input and another sequence is the output. Fully Connected layers are now used as a compliment to CNNs and RNNs, but rarely on their own. By using each type of model, the best type for this problem can be determined.

Accuracy validation Matrix:

(Correctly Predicted)/(Total Predicted)

Goals

The goal of this model is to have the optimal prediction of the sentiment in form of positive and negative over the whole dataset. It also must run in under 0.5 seconds. Accuracy is defined as the number of correct predictions over total predictions. My hypothesis is that if the model must run in under 1 second, and has the optimal accuracy, the best model will be the RNN with the GloVe word embeddings, because the RNN will have the best chance at learning the dependencies in the english language. While it is true that the BRNN should have a better accuracy than the RNN, and the Fasttext word embeddings will have a higher overall accuracy, both of these are much more computationally expensive than the RNN with GloVe embeddings and will not meet the requirement of under 0.5 second.

Procedure:

Each model was made differently, using each of the different architectures. The GUI used was Jupyter Notebook, since it has the option to save the notebook along with the outputs. Tensorflow and Numpy packages were using along with python builtins. First, the data, labels, and word vectors where loaded. Then, the models were created, 2 with each architecture. The 2 RNN and BRNN used 128 LSTM units. One of each had 1 layer(RNN or BRNN) and the other had 2 layers. The CNN has either 1 or 2 layers of convolution with a 3x3 filter followed by a max pooling layer. The simplest architecture, just 1 or 2 fully connected layers. Each model had dropout applied to each layer, and a final softmax predictor at the end. Each model was then trained using the AdamOptimizer, using a learning rate of 0.001 and 10000 iterations.

Results:

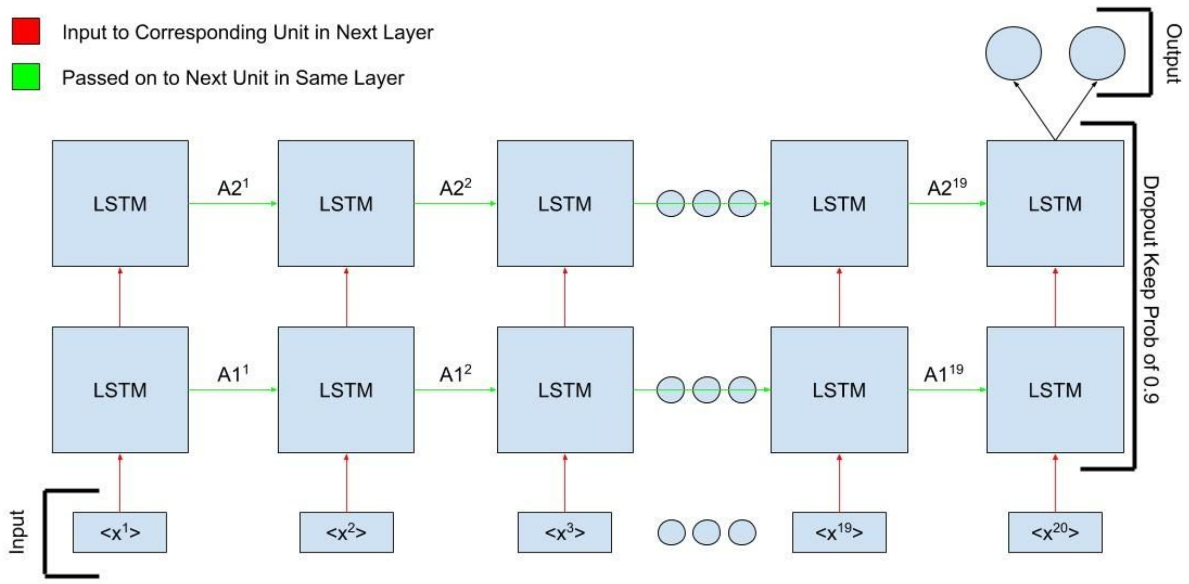
The results of each models were as follows:

Model	FC	FC 2 Layer	CNN	CNN 2 Layer	RNN	RNN 2 Layer	BRNN	BRNN 2 Layer
Accuracy (%)	0.54439	0.54388	0.6329	0.62315	0.76806	0.77486	0.77203	0.77661
Runtime (s)	0.01024	0.0128	0.3072	0.4608	0.1536	0.256	0.3072	0.6656

Analysis

No models could be trained using the FastText Word Embeddings because they were way too large, and my machines did not have sufficient RAM. The accuracy matches what was expected, with the FC layers not doing well at all, while the RNN and BRNN had reasonable accuracy. Because the RNN and BRNN were able to capture the long term dependencies of english, they could extract the sentiment from the input text. The CNN layer took a very long time to train, because it is not designed to be used on words, and shrinks and discards data between layers, instead keeping it, which ultimately made the difference between the CNNs and FCs which did very poorly to the RNNs and the BRNNs which did reasonable on the test data. Based on the accuracy alone, the 2 layer BRNN had the best accuracy, but after weighing in the runtimes, the 2 Layer RNN was the most optimal. The runtimes are so different because the FC and CNN layers did not have as much complexity as the RNN and BRNN layers had. Between the RNN and BRNN, the BRNN has almost double the amount of computations, because the BRNN does both a forward and a backward pass on the sequence, so has double the parameters. The 2 Layer RNN is the most optimal, because it only performs 0.175% worse on the test data than the 2 Layer BRNN. It also has less than half the runtime, which means that the user will have to wait much less.

Below is the architecture of the chosen model:


🟥 Input to Corresponding Unit in Next Layer
🟩 Passed on to Next Unit in Same Layer
Output
Dropout Keep Prob of 0.9
LSTM
A2¹
A2²
A2¹⁹
A1¹
A1²
A1¹⁹
Input
<x¹>
<x²>
<x³>
<x¹⁹>
<x²⁰>

Discussion

Conclusion

The purpose of this experiment is to determine the optimal model architecture for learning the sentiment of input text. Sentiment Analysis is part of the inception of many NLP problems, and it's data is both simple and easily accessible. For an architecture to be considered the most optimal, it must have the best accuracy on the test data, while running in under 0.5 seconds.

The hypothesis was that the 2 Layer BRNN with GloVe word embeddings would be the best model, as the FastText embeddings were too large to have a reasonable runtime. By using tensorflow, the models were created, each with a input embedding matrix, and an output softmax unit with 2 output classes, for positive and negative. In between, 1 and 2 layers of CNN, FC, RNN, or BRNN was inserted. After training for 10000 iterations using the AdamOptimizer, the accuracy and runtime was obtained.

The FastText word embeddings could not be loaded into tensorflow because they were too complex and large. Therefore, this type of word embedding was rejected. The model with the best accuracy was the 2 layer BRNN, but it ran in over 0.6 seconds. The 2 layer RNN had a slightly worse accuracy, but ran in a quarter of a second, meeting both requirements. The hypothesis was right in the sense that the GloVe word embeddings would be more optimal, but should be rejected because it predicted the 2 Layer BRNN would be most optimal. There could be many improvements done on this, because of my computational constraints. My machines could not handle any more complex networks and did not have sufficient RAM to load the FastText word embeddings, which would have almost certainly improved accuracy.

The data, taken from twitter, could have been precompiled to remove repetitions, clean extra characters, and translate acronyms. However, the results, no matter how improvable, are significant. They show that RNNs and BRNNs are the best way to learn the discrepancies and dependencies of the english language. CNNs and FCs cannot capture the workings of english, because they do not remember information across inputs. The question now is what other technologies can be added to the simple BRNN/RNN to improve accuracy further.