Machine Learning Engineer Nanodegree

Capstone Proposal

Kevin Casado March 14th, 2017

Proposal

Domain Background

In machine learning there are many buzzwords being thrown around lately. One of the big ones is deep learning. This is because "deep learning" has pushed state of the art in many fields such as computer vision and natural language processing. A large part of work that I have done at school as well as what mostly seems to be covered when trying to see what machine learning can do is computer vision. However natural language processing doesn't seem to get as much attention.

There have been many advancements in natural language processing over the past year with the high rise of Recurrent Neural Networks(https://en.wikipedia.org/wiki/Recurrent_neural_network) the state of the art has really been pushed far over the past two years. NLP presents different problems as the data is represented sparsely and not nearly as dense as when looking at computer vision or audio. With the rise specifically of Long short term memory networks (https://arxiv.org/pdf/1701.03441.pdf) people have found success in applying this to semantic processing. (https://arxiv.org/pdf/1502.06922.pdf)

Problem Statement

With words being similar yet represented differently it is hard to tell when one question is the same as the other, there are many approaches like stemming/lemmatizing, removing stop words, part of speech tagging etc. But all of these turn out to not really come close to anything production worthy. Ideally if we could find a way to use machine learning (specifically deep learning) to try and tell if two statements are the same this would be beneficial for so many question answer forums such as stackoverflow or quora as they could narrow down the repetitive nature of experts having to answer many of the same questions.

The input the model will receive is two questions and the output will be a binary classification representing whether the two questions are asking the same thing.

Datasets and Inputs

Quora recently released a dataset that gives question pairs and a label of whether they are considered to be duplicates or not (https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs). They also released a kaggle competition (https://www.kaggle.com/c/quora-question-pairs) to try and help out. After doing some investigation on the data it contains 40,4288 question pairs with a total of 149263 considered to be duplicates. For the approaches I will be using both questions as inputs and the output will be a 1 or

0 depending on if they are considered to be duplicates or not.

I will be doing preprocessing on the questions, first cleaning to make sure they are all valid strings and special characters are taken out, then I will try out different ways of cleaning up the questions such as removing stop words, lemmatization, and using n-gram phrases. This will depend on which approaches give me better results.

Solution Statement

Ideally the solution would be able to detect whether a question is considered a duplicate with another question. So if we give two inputs the machine learning model would be able to detect whether they are asking the same thing. As mentioned above since LSTM networks have shown to give good results on semantic analysis. I will also be using an approach known as word2vec or embedding (https://arxiv.org/pdf/1301.3781.pdf) this approach helps to deal with the sparsity of words and how it is very hard to represent the relation that words have. So I will be using a neural network where it first embeds the words using the google common crawl vectors (https://nlp.stanford.edu/projects/glove/) and then feed this to a LSTM layer and then I will try to use various ways of concatenating the representation for each question. I will explain below more in depth as to the various ways I will try to optimize this solution.

Benchmark Model

Since this is a binary classification the baseline prediction would be 50%. This is because for every chance there are two options. However the goal would be to match the results quora has posted. Quora does not show their existing results of using a random forest but they do share the best scores they get using the RNN approach. Currently the best they get is a f1 score of 88%(https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning). This will be the benchmark result that I shoot for and try to improve upon.

Evaluation Metrics

In this section, propose at least one evaluation metric that can be used to quantify the performance of both the benchmark model and the solution model. The evaluation metric(s) you propose should be appropriate given the context of the data, the problem statement, and the intended solution. Describe how the evaluation metric(s) are derived and provide an example of their mathematical representations (if applicable). Complex evaluation metrics should be clearly defined and quantifiable (can be expressed in mathematical or logical terms). The most common metric used for simple binary classification is f1 score which combines precision, and recall to consider the accuracy of the existing model. For a better explanation of f1 score it can be explained here: (https://en.wikipedia.org/wiki/F1_score) I will be using the base approach which is 2*((precision* recall)/(precision+recall))

Project Design

In this final section, summarize a theoretical workflow for approaching a solution given the problem. Provide thorough discussion for what strategies you may consider employing, what analysis of the

data might be required before being used, or which algorithms will be considered for your implementation. The workflow and discussion that you provide should align with the qualities of the previous sections. Additionally, you are encouraged to include small visualizations, pseudocode, or diagrams to aid in describing the project design, but it is not required. The discussion should clearly outline your intended workflow of the capstone project.

Currently quora uses a random forest with feature engineering to give them results. However with the advancements in deep learning it seems like an Recurrent Neural Net would be the best route to go. Quora has released some approaches they use with the accompanied results here: (https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning). In short they use LSTM's for each question and try different way of representing the resulting representation. One uses simple concatenation where as the other uses the cosine distance and angle. The best result they get is a f1 score of 88%.

For the cleaning the data as I stated above I will try a few different ways, first I will try lowering the vocabulary as well as the sentence length as much as possible. This can be done by removing all stop words and lemmatizing all words. If those do not give satisfiable results another approach will be variations of removing stop words and not lemmatizing words or vice versa. The main idea behind removing stop words is that they do are words that occur very often and thus do not provide unique information about a given statement (https://en.wikipedia.org/wiki/Stop_words). Lemmatizing or stemming is used to preserve the base of the word. For example lemmatizing the word talking returns talk. The idea behind doing this is that you will reduce the vocabulary size while still keeping the meaning of the base word.

After the data is cleaned I will input these into an embedding layer an embedding layer is used because of the sparsity that words are represented in. Word2Vec is an essential technique used for nlp(https://www.tensorflow.org/tutorials/word2vec), there is the iconig king is to queen as man is to woman example that can be shown when using word vectors. I will be using the Glove 300d word vectors from (https://nlp.stanford.edu/projects/glove/) I will try both approaches of training on top of these vectors and using them as the initial values as well as keeping them constant and not trainable.

After the embedding layer I will use variations of LSTM's to try and see which produce the best results, the approaches I will use are the following: 1. Using a vanilla LSTM that feeds similar to the first sequence part of the seq2seq model (https://www.tensorflow.org/tutorials/seq2seq) 2. Use a similar approach as above except use a bi directional LSTM (https://en.wikipedia.org/wiki/Bidirectional_recurrent_neural_networks) 3. Use a similar approach to what currently is state of the art for semantic processing (https://nlp.stanford.edu/pubs/tai-socher-manning-acl2015.pdf) Which uses dependency tree structured representations from the given lstm cells.

From the LSTM portion I will try using either concatenation or using cosing similarity as well as the angle between the two representations. All approaches will then be fed to a 1-2 dense layers which will include dropout and will have a final softmax for the classification.