Capstone Proposal

Ankur Saikia

21-OCT-2017

Quora Question Pairs

Domain Background

Quora is a place to gain and share knowledge. It is a platform to ask questions which are answered by the community itself. Being such a large community, related questions are often asked by members. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both groups in the long term. It is because of this reason Quora hosted the question pair challenge on Kaggle where they provided a train and test dataset consisting of similar and different question pairs. The expectation from the Kaggle community was to build state of the art models to predict whether the question pair is similar.

Problem Statement

Predicting question pair similarity is a classification problem. It is a well-known problem in the field of Natural Language processing. We already have many state of the art models that attain high accuracy developed by the participants of the challenge on Kaggle. Before the challenge Quora was using Random Forest for classifying the question pairs. This problem can be approached in many ways and there are well known solutions based on boosting, others based on deep neural networks and many more.

Datasets and Inputs

The dataset for the problem can be found in the competition page on kaggle but Quora has also released a updated dataset for the same problem (http://qim.ec.quoracdn.net/quora_duplicate_questions.tsv). This dataset is publicly available and we will use this to tackle the problem. The training dataset consists of question pairs and the target class for that pair (0 for non-duplicate pair and 1 for duplicate pair). It contains 404290 rows of data, of which 63% rows are non-duplicate pairs and 37% are duplicate pairs. Hence, we will divide the training set into three parts: train, cross validation and test. We will use sklearn function train_test_split to do the splitting. We will use the train set to train our model and the cross validation set to tune the hyper parameters. We will measure the performance of our model using the test set.

Solution Statement

We will use a deep neural network architecture for this problem. We will use keras which is a deep learning library in python. It provides us with many predefined functions to preprocess text e.g. Tokenizer() and text_to_sequences() etc. We start by removing the alphanumeric characters and converting short form of words to their full forms. We will use regular expressions for this purpose. Now we will convert our text to vectors using already available keras functions. Then we will feed the vectors to a neural network architecture for the classification task. We will use Embedding layers with pretrained weights, LSTM and Convolution 1D layers as the starting layers. The weights are calculated using embedding matrix created with the help tokenized words in the dataset and with the pretrained word embeddings model: stanfordNLP's GloVe: http://nlp.stanford.edu/data/glove.840B.300d.zip.

These layers extract useful semantic information using the word vectors and their arrangement. On top of these layers we will use Dense Fully connected layers for our predictions. The final layer will give us the predicted probabilities which we will use to predict whether the question pairs are duplicate or not. We will then check the accuracy score to improve our model and parameters. The training, cross validation and testing will be done on different mutually exclusive subsets of the data and those will be prepared using python libraries such as numpy, pandas and scikit-learn

Benchmark Model

There are many state of the art models to tackle this problem. We would not be considering the models that were created using ensembles of many different models. Instead we will consider a single model as the benchmark that has been published on Medium: Implementing MaLSTM on Kaggle's Quora Question Pairs competition (https://medium.com/@eliorcohen/implementing-malstm-on-kaggles-quora-question-pairs-competition-8b31b0b16a07). This model scores around 82.5% accuracy on the validation set.

Evaluation metrics

We will use Accuracy as our evaluation metric for this problem. We already know the validation accuracy of our benchmark model which is 82.5%. Accuracy score would be most suitable for this problem because the dataset is not too skewed (even though it is not 50-50 split for positive and negative) and because precision and recall need equal weightage.

Project Design

We will use Keras and Tensorflow to train our neural network. Keras provides many high level functions to pre-process our data and train our neural network with ease.

First, we pre-process our text data (i.e. the questions) by removing symbols with blank space, other short forms with their full forms etc. e.g. replace can't with cannot and so on. We now convert our sentences to vector sequences of a given max length using keras pre-processing functions. We use padding to make the sequences equal in size. We also create an embedding matrix using all the tokenized words present in our dataset and give weights to it using the pre-trained GloVe model for NLP.

After pre-processing the data, we create mutually exclusive datasets: train, cross validation and test. We take 60% as train, 20% as cross validation and 20% as test. For each set, we check that we have a fair share of positive and negative label data. If the distribution is skewed say (positive : negative ratio less than 0.1) we reshuffle and split again.

We now define our neural network architecture. We start with a simple architecture consisting of one Embedding layer with GloVe initiated weights, a TimeDistributed Dense layer, a Lambda layer calculating the sum of the TimeDistributed layer outputs and finally the fully connected layer consisting of a Dense layer with 300 nodes, a PReLU layer, a Dropout of 20% along with BatchNormalization and a final Dense layer with a single output and softmax activation. We train the network for 100 iterations using our train set. Now we try different architectures by adding LSTM layers, Convolution 1D layers and trying different fully connected layers and train them for 100 iterations. We use Accuracy on the cross validation set to evaluate the performance. Finally, we select the model that gives the highest cross validation accuracy and train the network using Early Stopping (network will stop if there is less than 0.0001 accuracy improvement in 100 iterations). We use BinaryCrossentropy as our loss function and ADAM as our optimizer for our training purpose. We will also use the checkpoint functionality in keras to save the best model weights. The training process will be done in Amazon AWS using p2-instances since those offer high speeds and large amounts of RAM.

We evaluate the final performance of the model using the test set. We do not use the test set performance to retrain the model and improve it. This will only be done as the last step in the process