Machine Learning Engineer Nanodegree

Capstone Proposal

Rishab Ghanti

27th April 2018

Proposal

Quora Question Pair (Kaggle Competition)

Domain Background

Quora is a very famous platform (website and Android app) all over the world where questions can asked by anyone in the community which are answered by Quora's 100 million monthly users. There are different platforms like *StackOverlFlow, Yahoo! Answers, WikiAnswers* etc. for different domains but Quora is by far the most popular general question-answer platform.

Due to its huge user base, it is expected that many users ask questions structured differently but having the same meaning. For example, the questions 'What is the population of USA' and 'how many people live in USA' are duplicates. Marking both these as duplicates and showing users one version of the question with the correct answer will allow users to get answers to their questions quickly and giving them more value.

A very useful resource that I found which is solving the problem in a similar way that I intend to. I will use this link for further reference.

Problem Statement

The goal of this project is to identify from the given question pairs, which pair is a duplicate, i.e. which pairs have questions with the same meaning. This is a natural language processing problem which is a popular field in Artificial Intelligence. I plan to use advanced techniques in machine learning that I learnt during the Nanodegree course to solve this problem. Solving this problem will allow users of Quora to have a more valuable experience on the platform.

Datasets and Inputs

The dataset that I plan to use in this project has been provided by Quora as a Kaggle competition.

The input fields are:

1. id – id of the training set question pair.
2. qid1, qid2 – unique ids of each question (available in train.csv).
3. question1, quesion2 – full text of the question
4. is_duplicate – set to 1 if the questions have the same meaning, else 0.

The total number of entries in this data set is 4,04,290 with 1,49,263 being positive entries (is_duplicate = 1) and 2,55,027 of them being negative entries (is_duplicate = 0)

Hence, the training set has 363,861 (90%) entries and the testing set has 40,429 (10%) entries.

Solution Statement

The solution to solve this problem will be prediction if the question pairs are duplicates or not. I plan to extract some features such as word length, similar words between the questions in the pairs etc. and do some visualization to get a better understanding of the data. Then I plan to do feature extraction and perform some function on the features such as max pooling to decrease the computation power and time I will need to train my model. I will then build a deep neural network to train the data and find the accuracy for the model based on the evaluation metric explained below. I will tweak different parameters of the neural network to get the best accuracy model which I will use on the testing dataset. I plan to 90% of the dataset as training data and 10% as testing data. To improve the training of the model I will use dropout layers to better train the neutrons in the neural network.

Benchmark Model

The benchmark model that I have considered is to find the percentage prediction of an event, i.e. x percent prediction that the pair is duplicate and (100 – x) percent prediction that it is not. A baseline model to do this is to look at the bag of words similarity of duplicate and non-duplicate questions using the Jaccard index. This is a naïve method and hence I plan to achieve a better model than this in my capstone project.

Evaluation Metrics

I plan to F-score as the main evaluation metric for my model in addition to *accuracy*.

$$F = 2 * \left(\frac{precision * recall}{precision + recall}\right)$$

precision – number of positive results divided by the number of positive results returned by the classifier

recall – number of correct positive results divided by the number of all relevant samples (all relevant samples that should have been identified as positive)

We know that negative records >> positive records in the dataset (precision > recall), hence as seen from the equation above numerator will be greater than denominator. Hence f-1 score is a good option to use to judge the model.

After exploring the data set I see that the dataset is unbalanced, i.e. it has 36.91% of positive entries whereas it has 63.08% of negative entries. Thus, *accuracy* as a metric will not be appropriate to use in this case since there will be a lot of true negatives compared to true positives.

Project Design

I plan to the do the following to do build a good model:

1. Feature selection – I will first extract features from the data using GloVe (Global Vector) word representation for every word in the dataset for the ‘question1’, ‘question2’ and ‘is_duplicate’ features. This will give me the number of times a context word occurs in

context to the target word, which in turn will give some semantic context too. I the plan to concatenate the max and mean of values of these rows to reduce the dimensionality. I will use the pickling method to save these extracted features on my machine.

```
# Get GloVe vectors for each word
def get_vectors_for_text(nlp, text):
    return np.array([w.vector for w in nlp(text)])

# Take the mean of all the rows, thus getting a single
# row in the end
def mean_pool(text_vectors):
    return np.mean(text_vectors, axis=0)

# Take the max from all rows, thus getting a single
# row in the end
def max_pool(text_vectors):
    return np.max(text_vectors, axis=0)

# Concat of max and mean pool of all vectors
def features_for_text(nlp, text):
    vectors = get_vectors_for_text(nlp, text)

    return np.concatenate((max_pool(vectors), mean_pool(vectors)))
```

2. Build the network – I have planned to use different combinations to build the network as below.

```
# Define all combinations
model_params = {
    # Should we use batch normalization or not?
    'batch_norm': [True, False],

    # No. of weights in each layer. All layers will have the same
    # weights with ReLU activation
    'no_weights': [100, 200, 300],

    # No. of layers
    'no_layers': [2, 4, 6, 8]
}

# Generate all combinations
def generate_combinations(params):
    keys = params.keys()

    # Generate combinations of current key
    current_key = keys[0]
    current_combinations = []
    for value in params[current_key]:
        d = {}
        d[current_key] = value
        current_combinations.append(d)

    # Check if we need recursive processing
    if len(keys) > 1:
        other_combinations = generate_combinations({k:v for k,v in params.iteritems() if k != current_key
})
        all_combinations = []

        for i in current_combinations:
            for j in other_combinations:
                all_combinations.append(dict(i.items() + j.items()))

        return all_combinations
    else:
        return current_combinations

all_combinations = generate_combinations(model_params)
print(all_combinations)
print(len(all_combinations))
```

3. Train the network – 90% of the data will be used as the training data which will give me 355993 questions to train my model on.

```
TEST_SPLIT = 0.9
TEST_INDEX = int(TEST_SPLIT * len(vectors['question1']))

print(TEST_INDEX)
print(len(vectors['question1']))
```

```
355993
395548
```

4. Test the Network – I plan to test the network with the different combination as explained above and then pick the combination having the best score as below:

```
test_scores = {}

t_question1 = vectors['question1'][TEST_INDEX:,]
t_question2 = vectors['question2'][TEST_INDEX:,]
t_is_duplicate = vectors['is_duplicate'][TEST_INDEX:,]

for combination in all_combinations:
    name = get_name_from_params(combination)
    model = train_model(vectors, combination)

    test_scores[name] = model.evaluate([t_question1, t_question2], t_is_duplicate)
```

First I plan to explore the data and see how it is formatted and how I can use each of the columns to improve my model. Then I will extract different features such as number of positive entries and negative entries in the data set, word count of the questions, number of common words in the question pair etc. I plan to use the Jaccard index to see if the number of common words in the questions can help me better predict if the questions are duplicates or not.

I plan to the do the following to do build a good model:

5. Feature selection - Since the Quora dataset does not have a lot of features, this step will be pretty easy.
6. Feature transformation – This step would include tokenizing the questions and finding word vectors for them which would give me a matrix for each question. To reduce the size of the matrix I plan to use a max pooling layer.
7. Neural networks – I then plan to create a neural network probably a deep neural network. I plan to use Siamese Manhattan LSTM deep neural network to solve this problem similar to this article. Siamese neural networks are networks with 2 or more identical networks in them. The 2 networks have the same weights and have been seen to perform well in tasks finding similarity between comparable things. Few applications handwriting recognition, finding sentence similarity etc. Hence I think this will be a good use case for this project since we are trying to find similarity in 2 questions to find out if they are similar.

To better train the model and improve the accuracy of the model, I will use dropout layers. I am not 100% sure about the model and it may change once I start coding and tweaking the parameters to improve the performance of the model

References

- Kaggle
- Quora