Machine Learning Engineer Nanodegree

Capstone Proposal

Adam Malpass, April 21st, 2017

Domain Background

This capstone project is part of the field of Natural Language Processing (NLP) (http://sentic.net/jumping-nlp-curves.pdf). NLP exists to take in words and try to convey meaning from them. This could take the form of, for example, a chat bot trying to understand the written instructions of a user, or a home assistant such as Amazon Alexa or Google Home attempting to process spoken commands. 'Natural' is really the key here - it is important that the user can communicate in a colloquial way like they could could with another human, not in a strict robotic like manner that is required, for example, by traditional computer input systems such as the command line.

NLP is a rapidly-growing sub-field within machine learning and AI, so it will be exciting to be a part of it. Furthermore, the inspiration and data for this problem comes from a website I visit almost daily, Quora, so it gives extra motivation for tackling this challenge.

Problem Statement

The problem I intend to solve is finding pairs of similar questions on Quora (https://www.quora.com/), a Q&A website. This is currently an active challenge on Kaggle (https://www.kaggle.com/c/quora-question-pairs), a website that organises machine learning competitions. Quora is a very popular site (100+ million uniques / month) and so often the same or very similar questions get posted multiple times, making it harder for people to find exactly the right information they are looking for. To help combat this issue Quora is searching for a solution to determine whether any given pair of questions can be considered duplicates of each other, making it possible to combine the answers to both questions into one set.

Specifically, I will design a model that takes in two questions (of written text), analyses them and outputs a continuous probability as to how similar they are believed to be, whereby 0 would indicate no similarity and 1 would indicate an identical question.

Datasets and Inputs

The datasets are provided by Kaggle and consist of 404290 pairs of questions with a human-generated label to show whether or not that pair is considered to be a duplicate. Each question is also given a unique id (which may, or may not, be useful). This data is ideal for addressing the problem at hand as we have many examples that fit exactly the input and output format that the final model is required to generate, and hence can be addressed as a supervised learning problem.

As this is a real, live, competition, Kaggle also provides another dataset, this time of unlabelled question pairs, which will be used to test the models created and judge the competition. By generating predictions as to how likely it is the items in the testing set are duplicates and submitting to Kaggle's site it is possible to get an unbiased score as to how well the model worked. As we learned during the Machine Learning Nanodegree it is vital to test the model on separate data than was used to train it to avoid overfitting.

However this submission is a manual process and it is only possible to submit predictions up to five times a day. Therefore to have better control over the testing/training process I will split the original labelled training dataset into training and testing subsets. As for the original testing dataset, it will still be interesting/fun to enter the competition as well, so this can be considered as 'competition testing dataset' and I will just use this as an input to my model once regular training and testing is starting to show good results to generate results for the competition.

The original training dataset (to be split into training/testing sets) can be downloaded from Kaggle here:

https://www.kaggle.com/c/quora-question-pairs/download/train.csv.zip

The original testing data set (to be used as competition testing dataset) can be downloaded from Kaggle here:

https://www.kaggle.com/c/quora-question-pairs/download/test.csv.zip

Solution Statement

The data provided is the ideal format for training a supervised machine learning algorithm. Since the final output is expected to be a prediction as to how likely the questions pairs are to be duplicates, it is necessary to use a regression technique to predict a continuous output. I plan to use regressors from the Scikit-learn Python machine learning library (http://scikit-learn.org/) to build the model. Possible candidate algorithms include Linear Regression, Support Vector Machine Regression, Decision Tree Regression and Random Forest Regression.

However these regressors can not directly process the textual information. I must first perform feature engineering to find useful characteristics about it to feed to the learner. I expect this to be the most important part, and biggest challenge of the project. The solution is quantifiable and can be evaluated based on the evaluation metric discussed later. For more detailed overview of how I plan to approach solving this problem, please see the Project Design section.

Benchmark Model

Kaggle has a public leaderboard showing the best log loss scores of everybody who is competing based on their predictions on ~35% of the test data (https://www.kaggle.com/c/quora-question-pairs/leaderboard). At the time of writing (21/4/17) I calculated the following statistics about the submissions so far:

- Unique submissions (i.e. total number of competitors): 1731
- Total submissions (i.e. total number of submissions by all competitors): 6971
- Min (best) score: 0.13627
- Mean score: 1.831
- Median score: 0.364
- Max score: 28.521

The min and max score should give approximate bounds as to where the final result should lie within. The distribution of results is very skewed right so the mean doesn't make much sense here; the median gives a much better idea of the typical score. I will set a target to achieve a lower score than the median (0.364) and come in the top half of the competition!

Evaluation Metrics

In terms of quantifying the solution, Quora have requested to use the log loss between the values predicted by the model and the actual, ground truth values. Since correct labelled values are available (after splitting the training dataset into training and testing datasets) I can calculate the log loss of the predictions and see how well the model is working. Log loss is a commonly used metric when evaluating a model that assigns a probability to an output condition, rather than simply picking the mostly likely output (in our case, whether or not the questions are duplicate). It is the entropy between the actual and predicted values. In mathematical terms for a single sample (http://scikit-learn.org/stable/modules/model_evaluation.html#log-loss):

$$L_{\log}(y, p) = -\log \Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p))$$

The final log loss score is then the average of the above calculation across all samples.

Project Design

First it will be necessary to preprocess the training data to, for example, find a way to deal with NaN or other unexpected values. Next it will be necessary to generate some features from the raw text inputs, since the machine learning algorithms can't work with them directly. Some initial ideas for features I have are:

- Difference in total number of characters of both strings
- Difference in number of words of both strings
- Number of common words between both strings
- % of common words compared to number of words in longest/shortest string
- TF-IDF

I will make visualisations of how these features relate to the likelihood of being a duplicate question to get an intuitive feel for how the model should work. Some additional pre-processing of the text may be required to get better results, for example by removing 'stop-words' (common words that carry little-to-no meaning) or by performing stemming so only the root of words are considered.

Once I am happy the features can produce useful insight into the data it will be time to create the training, validation and testing sub datasets. Only then I can start to feed the data to the machine learning algorithm. As previously stated since a continuous output is required I need to use regression. The most fundamental type is linear regression, so this might be good algorithm to start with. Then I can try ramp up the complexity a bit with a Support Vector Machine based regressor. Alternatively it would be possible to use a Decision Tree Regressor. If this looks promising this could be expanded further using a Random Forest Regressor. Quora said they currently use Random Forests to tackle this issue, so it could be a promising algorithm to experiment with.

Once the model has been developed using the training data and hyperparameters optimised with the validation data, I can then use the model to generate predictions on the testing data. I can then compare these against the ground truth values and calculate the log loss score as the evaluation metric to see how well the model works. At this point it would also be possible to generate predictions on the competition test data and submit results to the Kaggle competition to get further feedback on how well the model is working.

Once this whole pipeline is in place it will then be required to iterate on the data processing, feature engineering and learning algorithm optimisation stages to improve the final score further. Let the challenge begin!