Quora Duplicate Question Pairs
Project Report

Shalin Anilkumar Amin
Graduate Student, M.S. in Computer Science
University of Texas at Dallas

I. ABSTRACT

This project aims to come up with a system to identify semantically equivalent pairs of questions. The project uses various NLP techniques to extract features from each questions pair and then uses different Machine Learning algorithms as classifiers and compares their performance on the feature set.

II. INTRODUCTION

Where else but Quora can a physicist help a chef with a math problem and get cooking tips in return. Quora is a place to gain and share knowledge—about anything. It's a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world.

This project aims to identify questions, asked on Quora, which have similar intent. Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. 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 of these groups in the long term.

This could be useful, for example, to instantly provide answers to questions that have already been answered. We are tasked with predicting whether a pair of questions are duplicates or not, and submitting a binary prediction against the logarithmic loss metric.

III. METHODOLOGY

a. Flow of Proposed System

This implementation makes use of Natural Language Toolkit(NLTK) and spaCy libraries in Python to perform various relations extractions in order to measure similarity between the questions as obtained from the dataset. Fig. 1 shows a high-level architecture of the proposed approach.

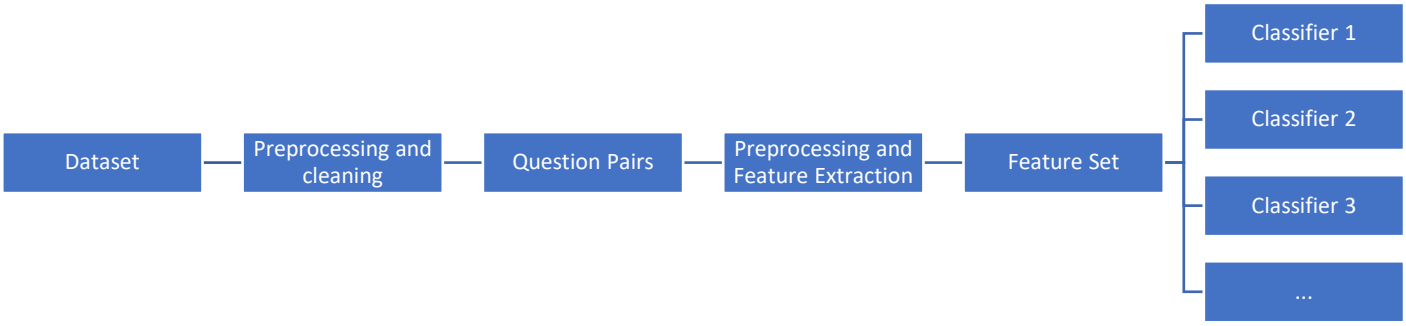


Fig. 1. High Level Architecture

From a high-level perspective, the system needs to perform the tasks in the order as shown in the architecture. The dataset is first cleaned and preprocessed and then for each question pair, certain features are extracted. The features for all the questions are stored in a feature set, which is used to train the classifiers.

b. Data Set

The data for this project has been extracted from Kaggle's website (https://www.kaggle.com/c/quora-question-pairs/data). The data, as is, was provided by Quora for their Kaggle competition, Quora Question Pairs. Each instance of the dataset provided has two

questions and a human annotation of whether the pair is a duplicate pair or not. The labelling of the sentences being duplicate or not may not be 100% accurate, but it has been taken to be informed for the purpose of the project.

c. Text Cleaning and Pre-processing

The dataset, when analyzed displayed high occurrence counts of prepositions, conjunctions, question words (what, how, why etc.), and some other high frequency words which do not change the semantic representation of the text. Hence, these *Stop Words* were removed from being processed into further representations of the text.

d. Feature Extraction

The features extracted for this project are as follows:

1. Lexical similarity of the sentences (using spaCy)
2. Fraction of Nouns in each question
3. Fraction of Verbs in each question
4. Number of different Nouns in the 2 questions
5. Number of different Verbs in the 2 questions
6. Subject-Verb-Object similarity of the 2 questions

IV. Experiments and Results

The feature set containing the above-mentioned features for each pair of questions was fed to machine learning algorithms. For each algorithm, a 4-fold cross-validation was performed and the means of the resulting accuracies have been documented. Python's sklearn library was used to model all of the implemented algorithms, except the neural net, which as implemented using Keras (which is also based on sklearn). Following is a brief introduction and results of each of the algorithms that were tested:

1. SVM:
   - In machine learning, support vector machines (SVMs, also support vector networks[1]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

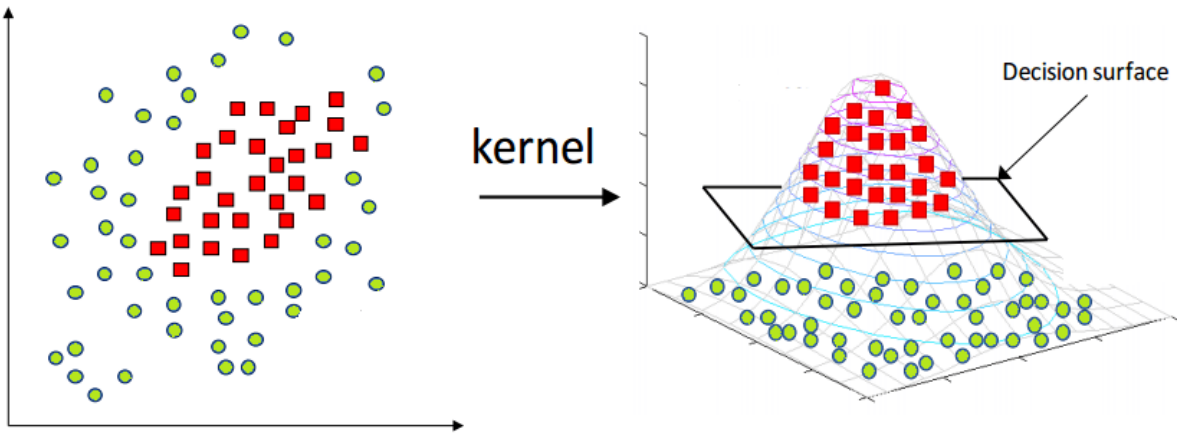


Figure showing how linearly inseparable features are separated in a SVM

   - In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

   - The advantages of support vector machines are:
     - Effective in high dimensional spaces.

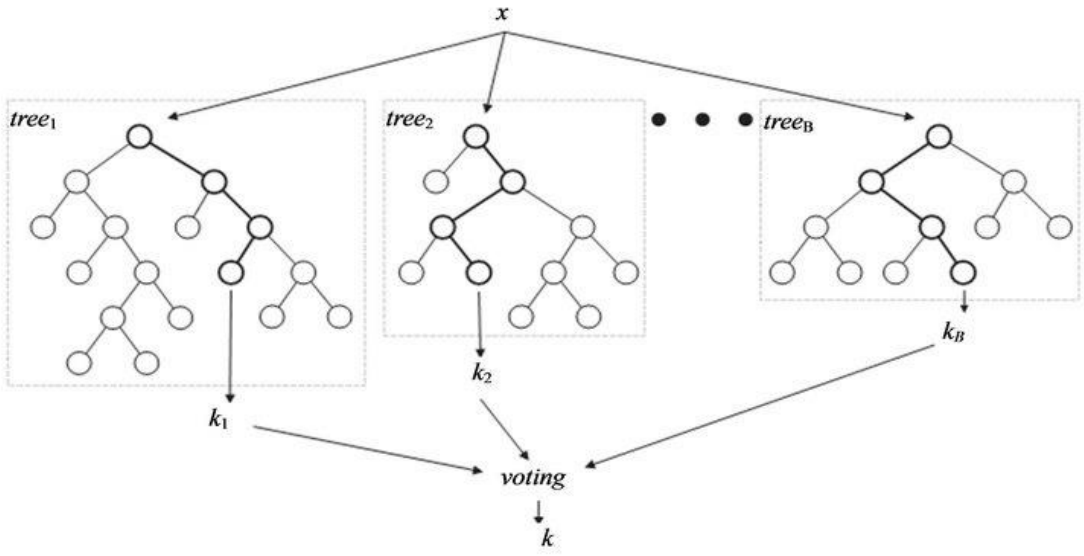
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

- The disadvantages of support vector machines include:
  - If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
  - SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

- SVM can be trained using one of 3 available kernels in sklearn: linear, rbf and polynomial (of any degree)

- Results:
  - SVM Linear Kernel – 67.3%
  - SVM RBF Kernel – 67.5%
  - SVM Polynomial Kernel, degree 2, degree 3 – 68.3%, 68.1%

2. Random Forest:
   - Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

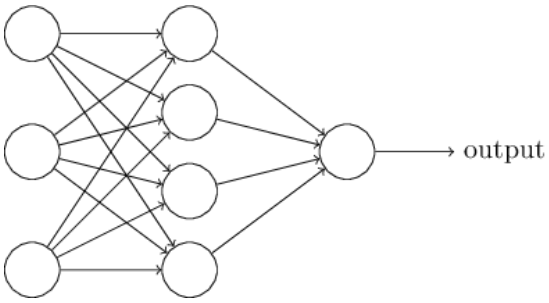


A visual representation of the working of a Random forest

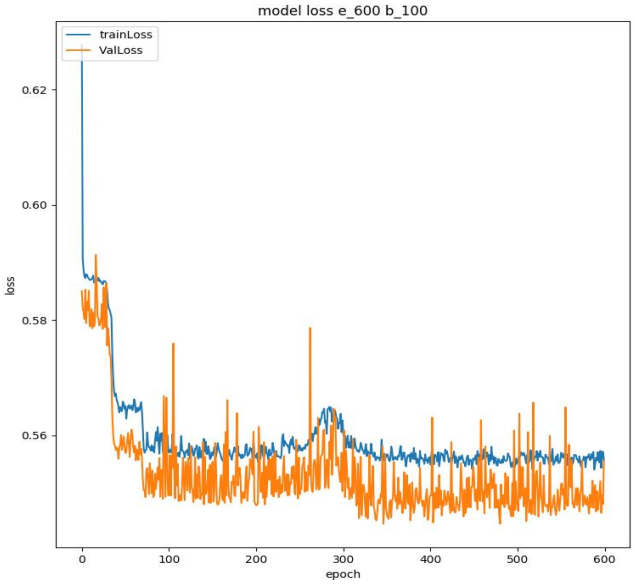
   - Trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.
   - The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners.
   - Results:

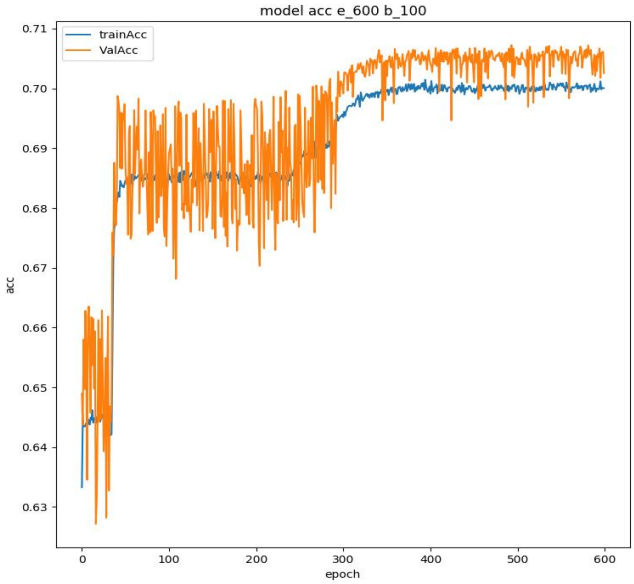
- no_trees = 50 75 100 125 150 175 200
  Accuracy = 72.85% 73.03% 73.12% 73.18% 73.23% 73.25% 73.27%

3. Neural Networks:
   - Artificial neural networks (ANNs) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming.
   - Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.



A visual representation of neural nets





A loss/cross_val loss vs epochs and accuracy/cross_val accuracy vs epochs plot of one of the trial runs

   - Extensive experimentation was done in the case of neural nets using different activation and loss functions, but not all of them have been documented as they would take up too much space and the results were not good either.

   - Results:

     - No. Layers = 1, Hidden Layer size = 8, activation = relu, loss = binary crossentropy, metrics = accuracy, 500 epochs, 50 batch size – 70.12%
     - No. Layers = 1, Hidden Layer size = 8, activation = sigmoid, loss = binary crossentropy, metrics = accuracy, 500 epochs, 50 batch size – 68.53%
     - No. Layers = 1, Hidden Layer size = 8, activation = relu, loss = binary crossentropy, metrics = accuracy, 600 epochs, 50 batch size – 67.72%
     - 300 epochs, 25 batch size – 71.3%
     - No. Layers = 1, Hidden Layer size = 8, activation = relu, loss = binary crossentropy, metrics = accuracy, 600 epochs, 100 batch size – 69.43%

V. Conclusion

The project has compared various machine learning algorithms for an extracted feature set. Since the results are not very good for all classifiers, we can conclude that the extracted feature set is not good enough to make an accurate classification. The feature set needs to be re-thought or increased with features relevant to the semantics of the sentence. Since a not-so-popular library was used to extract Subject-Verb-Object relations, which also failed to give the relations for about 50% of the questions (which were not taken for the feature set), there is a possibility of it giving incorrect SVO some questions. As the feature set significantly depends on SVO, this could significantly contribute to the low accuracy as well.

VI. Future Scope

The results can be greatly improved if more thought is given to feature selection and more reliable methods are used to extract features. Some features that could have been added but were not added because of time constraints include TFIDF, parse trees and entity recognition.