SMAI
Final- Evaluation Presentation

Duplicate Question Detection in the Quora dataset

By (Team: DNA)

Dhruv - 2020201065

Nayanika - 2020201056

Adrija - 2020201063

Overview

In this presentation, we present all our experiments performed on given problem and report the accuracies obtained by different models.

The Problem Statement

- Quora is a question answering website where users ask questions and other users respond.
- Duplicate questions on this site are not uncommon, particularly as the number of questions asked grows.
- Duplicate questions may prevent a user from seeing a high quality response that already exists.
- It may also prevent responders from answering the same question twice.

Dataset Description

Exploration

- The Quora Question Pairs (https://www.kaggle.com/c/quora-question-pairs/data) dataset contains 404,290 question pairs for training and 2,345,795 question pairs for testing.

- We will take only the training set as our entire dataset and shall perform splitting into training, validation, and test sets.

- Each sample point has the following fields:
  - **id**: unique ID of each pair
  - **qid1**: ID of first question
  - **qid2**: ID of second question
  - **question1**: text of first question
  - **question2**: text of second question
  - **is duplicate**: 0 indicates not duplicate, 1 indicates duplicate

Solution Approach (Generic)-

- The Duplicate Question Detection is a binary classification problem on various length strings.
- The challenging part of the problem is to represent sentences as numerical inputs such that the learning algorithms can work on it.
- We apply hand engineered feature generation on the raw textual data.
- We split our transformed dataset into training, cross-validation and test sets.
- After the feature extraction, we plan to set up a baseline model and train our data on it.
- Then we train other intermediate models and compare their performances.
- After the comparison and tuning the hyperparameters, we will select a final model as our duplicate question detection classifier.

Data Cleaning

- Stop Word Removal- removing words which have no semantic value
- Stemming- converting word into its dictionary version
- Removing punctuations and brackets

Feature extraction

We extracted the following features for our dataset:

- Basic Features
    - Length of question1
    - Length of question 2
    - Number of words in question1
    - Number of words in question2
    - Difference between length of both questions
    - Common Words

Feature extraction (extended)

We extracted following more features for our dataset:

- Fuzzy features:
  - Fuzzy ratio
  - Fuzzy partial ratio
  - Fuzzy token set ratio
  - Fuzzy token sort ratio
  - Fuzzy partial token set ratio
  - Fuzzy partial token sort ratio

Feature extraction (extended)

We extracted following more features for our dataset:

- Applied word2vec and found:
  - Cosine similarity
  - Manhattan distance
  - Euclidean distance
  - Kurtosis
  - Skewness

We have used the pre trained ‘Google news Word Vectors’ to get the word2vec form for each word in the question and then summed the vectors for the all the words to get the vector for whole question.

Feature extraction (extended)

We extracted following more features for our dataset:

- N-gram features
  - Unigram features
  - Bigram features
  - Trigram features
  - Unigram + bigram + trigram features

Training Baseline Model

- We chose Logistic Regression as our baseline model.
- Then we split it into train, validation and test sets.
- We used sklearn’s logistic regression to train our data.
- We experimented with different set of features for the baseline model and observed the accuracies.

Features	**Training Accuracies**	**Testing Accuracies**	**F1 scores**
Unigram	63.73	63.72	0.537
Bigram	65.37	65.57	0.564
Trigram	63.95	63.99	0.507
Unigram+Bigram+Trigram	65.24	65.36	0.591
Basic features	65.68	65.90	0.646
Fuzzy features	66.07	66.39	0.658
Word To vec features	67.05	67.27	0.671
Word to vec +bigram	68.63	68.80	0.685
Fuzzy +word to vec +bigram	68.68	68.89	0.687

Observations on baseline model

- The baseline model is giving accuracies in the range of 63-68 .
- We are getting highest training and testing accuracies when we are using Fuzzy features and Word to vec features and bigram combined i.e 68.68% and 68.89 % respectively.
- We tried to tune the hyper parameters but the variations in accuracy were not much significant .
- We used the following hyper parameters for conducting the experiments
  - Tolerance=0.0001, C=1.2, max iterations=1000, solver='liblinear'

Training Tree Based Model

- For Tree based models we chose Random Forest Classifier and Decision Tree Classifier
- Then we split it into train and test sets.
- We used sklearn’s random forest classifier and decision tree classifier to train our data.
- For decision tree we used entropy as classification criterion.
- The accuracies observed are tabulated in next slide

Random Forest

Number of estimators	Training Accuracy	Validation Accuracy
5	96.37	68.35
10	98.25	69.45
20	99.33	70.14

Decision Tree

Max Depth	Training Accuracy	Validation Accuracy
10	71.24	70.92
15	74.30	70.54
20	79.34	69.24

Observations on tree based Models

- Random forest classifiers are giving very high training accuracies
- Best Random classifier is with number of estimators = 20 which is giving validation accuracy of about 70 percent and f1 score of 0.697
- The Decision tree classifier with max_depth of 10 i.e about 71% and f1 score of 0.712
- So the tree based models are working slightly better than the baseline model

Support Vector Machines

- The entire dataset that contains about 4 lakh rows was consuming a large amount of time to get executed ,Thus we sampled the data and used about 1 lakh rows for training and 80K for testing.
- We used linear kernel .
- For running SVM we used the feature set containing fuzzy features ,word2vec and bigram features.
- We got accuracy of about 67% on both training and testing data
- Also we got f1 score about 0.67

Neural Network Models

We have used the following models:

- Continuous Bag of Words(CBOW)
- Long Short Term Memory(LSTM)

Both the models are built to produce a single vector representation for each question, and use both representations to compute the label prediction.Then we concatenated both representations and element-wise products and passed this concatenated result to a single tanh layer

For both models, to get word embeddings, we used GloVe vectors

CBOW Model

- Using Stanford's GloVe 'glove.840B.300d', calculated an 'embedding_vector' matrix in the form of a dictionary.
- Pre-processing i.e. cleaning of questions was performed by removing stop words and using regular expressions.
- Feature matrix for both questions was made using the above pre-processed questions, using 'embedding_vector' for each question in the question array.
- Created difference and hadamard feature matrices as per research paper.
- Neural Network Model creation used sequential model with three blocks of dense layers and dropout layers of 10%, along with a final dense layer with activation 'softmax'. Optimizer 'adam', gives the desired results.
- Highest accuracy of 79.85% is achieved on 50 epochs.
- F1 score of 79.897 (out of 100) is achieved.

LSTM Model

- Vanilla RNNs are incapable of handling long range dependencies. Hence, LSTM is used to eliminate the Vanishing Gradients Problem
- Used Stanford’s GloVe ‘glove.6B.200d’, to calculate an ‘embedding_vector’ matrix in the form of a dictionary.
- Pre-processing was done using keras’ built-in tokenizer
- Two LSTM models were created for question1 and question2 and their output was multiplied element-wise.
- The models had two recurrent network layers followed by two dense layers.
- The merged output was flattened and passed through three dense layers
- Model was tuned by experimenting with different epochs and adam and SGD optimizer.
- Adam optimizer gave slightly better accuracy than SGD optimizer.
- Highest accuracy of 72.84% was achieved on 20 epochs. (Due to time constraint, 20 was highest number of epochs tried)

Adam Optimizer

Number of epochs	Training Accuracy	Validation Accuracy
5	65.16	67.83
10	70.12	69.16
20	75	72.84

SGD Optimizer

Number of epochs	Training Accuracy	Validation Accuracy
2	63.08	63.05
5	68.02	68.21
10	68.34	68.22

Contributions

Initially We worked together on the following aspects with equal contribution:

- Analyzing data set
- Feature extraction
- Training baseline model

Further we divided work as follows :

- Nayanika - Worked on Further feature extraction and experimenting with baseline model and tree based models
- Adrija - Worked on LSTM and tree based models
- Dhruv - Worked on SVM and CBOW models.

Then we worked together on documentation

Discussion and Future Work

We hoped that the neural network models would outperform the linear and tree-based models since there are many ways to express a question such that it has the same meaning but uses different words. Neural network models, which make use of word embeddings, are better at capturing these semantics, and so we expected this family of models to perform better than the linear models and tree-based models, which do not represent words in this way. Though our CBOW model gave the highest accuracy, surprisingly LSTM's accuracy was same as the tree based models. We are hoping to perform more data cleaning and experiment with different layers to see if the performance of LSTM improves.

Checkpoints

Week 1

Understanding Problem Statement
In-depth study of research papers related to the problem. Gaining a grasp on the problem statement and the requirements to convert it into an ML trainable model

Week 2

Feature extraction and baseline training
Converting the given textual raw dataset into a transformed numerical dataset fit to apply algorithms on. Extracting various features by hand engineering and training it with Logistic Regression model.

Week 3

Hypertuning parameters, training on other intermediate models
Training our dataset on other models to see the performance, tune hyper-parameters, compare performance of the models and finalize a model.

Week 4

Finish testing on final model and documentation
Finish testing on the final selected model, document the various performances, and complete project documentation for final submission

Thank you.