Flagging Duplicate Questions with Gradient Boosting

1st Kailen Shinmoto
Natural Language Processing
Occidental College
Los Angeles, United States
kshinmoto@oxy.edu

2nd Rowan Fitch
Natural Language Processing
Occidental College
Los Angeles, United States
rfitch@oxy.edu

***Abstract*—Duplicate questions are common among online forums, and often times two duplicate questions go undetected and each one develops a long thread of comments and discussions. Identifying duplicate questions reduces the number of posts related to a certain topic, and reduces the space taken up in the forum's database. Questions can be marked as duplicate by forum users, but this is not a guarantee. Automating the process of flagging duplicate questions can save online forums lots of computing resources by reducing the size of their database, and also direct the users to the answer they are looking for faster.**

I. Introduction

To classify duplicate questions, we used a Gradient Boosting Classifier. A Gradient Boosting Classifier uses a collection of decision trees with only one split, called weak learners, and combines them to develop a strong learning model [1]. Being that the classification type is binary, duplicate or not, the Gradient Boosting Classifier was an excellent model due to the level of complication in the data set.

II. Methodology

A. Data Preparation

Cleaning the data for the Gradient Boosting Classifier was a crucial step in the classification of duplicate questions. The data set contained many different spelling errors, abbreviations, and word contractions that all held the same meaning, but would not be detected as the same word by any Natural Language Processing tool. In order to clean the data, all abbreviations, contractions, proper nouns, and common non alphanumeric symbols were standardized using regular expressions. Many of the suggested abbreviation and contraction standards for cleaning the data set came from a post on Kaggle.com by David Currie [2]. After the data set had been properly cleaned, a $TfidfVectorizer$ object from the Python package $sklearn$ was used to create a TFIDF-matrix for each question.

B. Training the Gradient Boosting Classifier

Our Gradient Boosting Classifier took the TFIDF-matrices of the questions as a parameter, along with the desired output of each pair of questions. We set the maximum number of weak learners in the model to be 80, and the maximum depth of a decision tree to be 50.

III. Results and Analysis

A. Results

Precision	0.79
Recall	0.79
F1 Score	0.79

TABLE I
Model Metrics

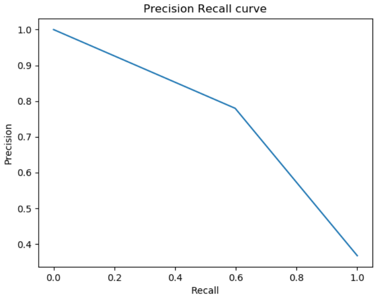

Fig. 1. Precision Recall Curve

Our model achieved a precision, recall, and f1 score all equal to 0.79, proving this model to be moderately effective in identifying duplicate questions.

B. Analysis

Our model achieved precision, recall, and f1 scores all equal to 0.79 (see I), proving it to be decent in classifying duplicate questions. The success of our model can be attributed to the attention to detail in cleaning the data set. Because more words that held the same meaning were changed to the same word, the $TfidfVectorizer$ was able to more accurately represent the underlying meaning of a question through a TFIDF-matrix. For example, the words ”U.S.A” and ”United States” hold the same meaning, but without standardizing them, a $TfidfVectorizer$ will detect them as dfferent words. However, one issue is that the cleaning of our data was slightly tailored to this specific problem, and if a tool like this was going to be deployed onto a real online service, the process of cleaning data would have to be abstracted to account for the vast amount of questions posted.

IV. Threats to Validity

Our project is subject to threats from validity both from the Python package $sklearn$ and from the regular-expression suggestions from David Currie [2]. Because we used the $TfidfVectorizer$ from $sklearn$ as well as a Gradient Boosting Model developed in part by $sklearn$, our project is subject to errors from the package's developers. However, given that $sklearn$ is a popular, commonly used package that is developed by experienced professionals, this is unlikely. Using suggested regular-expressions from David Currie on Kaggle.com subjects our data set to improper or over fitted cleaning. However, given that David Currie is ranked in the top 1% on Kaggle, huis suggestions are most likely valid due to his experience with data science.

V. Conclusion

Our model achieved a precision, recall, and f1 score all equal to 0.79, proving it to be moderately effective. The success of our model can be attributed to the thoroughness of cleaning our data. However, if a model like this were to be deployed onto an actual online service, it would need to be more accurate, and also be less specific in cleaning of data, as this data cleaning schema is subject to overfitting of the problem.

Related Work

Two additional sources we used to help conceptualize how to create this project were ”Identifying Duplicate Questions: A Machine Learning Case Study” [3] and ”Natural Language Understanding with the Quora Question Pairs Dataset” [4]. ”Identifying Duplicate Questions: A Machine Learning Case Study” gave great insight as to how to use the Gradient Boosting Classifier from $sklearn$, and ”Natural Language Understanding with the Quora Question Pairs Dataset” provided the initial idea of using a tree based classifier to predict outcomes from the Quora data set.

References

[1] Sci-Kit Learn, https://scikit-learn.org/stable/modules/generated/sklearn .ensemble.GradientBoostingClassifier.html

[2] David Currie, ”The Importance of Cleaning Text”, https://www.kaggle.com/currie32/the-importance-of-cleaning-text

[3] Surabathula, ”Identifying Duplicate Questions: A Machine Learning Case Study”, https://medium.springboard.com/identifying-duplicate-questions-a-machine-learning-case-study-37117723844

[4] Lakshay, Laura, Nikita, Utku, ”Natural Language Understanding with the Quora Question Pairs Dataset”