

**Quora Question Pairs Deduction**

PROJECT REPORT- GROUP 1

DATA MINING AND BUSINESS INTELLIGENCE [OPIM 5671]

**Team members:**

Kokila Dular

Michael Kölbl

Sen Lyu

Jay Mehta

Kaushik Vijayakumar

**Contents**

[1. Executive Summary 3](#_Toc481678380)

[2. Problem Statement 3](#_Toc481678381)

[3. Background 4](#_Toc481678382)

[4. Methodology 4](#_Toc481678383)

[5. Results 7](#_Toc481678384)

[6. Conclusion and Recommendation 8](#_Toc481678385)

[7. References 9](#_Toc481678386)

[8. Appendix 9](#_Toc481678387)

[8.1 Tools Comparison 9](#_Toc481678388)

[8.2 Approaches and accuracy of tools 11](#_Toc481678389)

[8.3 Classification models results 11](#_Toc481678390)

[8.4 Important features 13](#_Toc481678391)

# **Executive Summary**

Quora is an online platform that generates an enormous traffic and is widely used to ask questions about any regard. In an online competition hosted by Kaggle, Quora asks users to develop predictive models to determine whether two given questions are similar, and therefore redundant. Using this model, Quora can alert the users who attempts to post questions instantly when a similar question already exists, thereby enhancing the user experience.

We applied a variety of state-of-the art text mining algorithms to extract features and insights from given dataset for our prediction. Since the business problem of the project is straight forward, we challenged ourselves by building models using different tools. We initially used R and Python for this process and later expanded to SAS JMP Pro 13 and SAS Enterprise Miner, and the approaches are elaborated in the appendix. We find that complexity of a text mining technique is not necessary a representation for predictive performance. Simple measures can also yield satisfying results without demanding large computing capacities. For future projects, we recommend a combined use of different tools where only the best aspects of each solution’s capabilities are used to achieve greater model accuracy.

# **Problem Statement**

Quora is a popular question and answer forum with millions of users. Given the large audience, several questions can be duplicated or paraphrased, which can potentially reduce the quality of the platform and its user experience. Hence, the goal of this project is to explore several tools available in the market and to develop predictive models that uses state-of-art text mining techniques to identify duplicate question pairs.

# **Background**

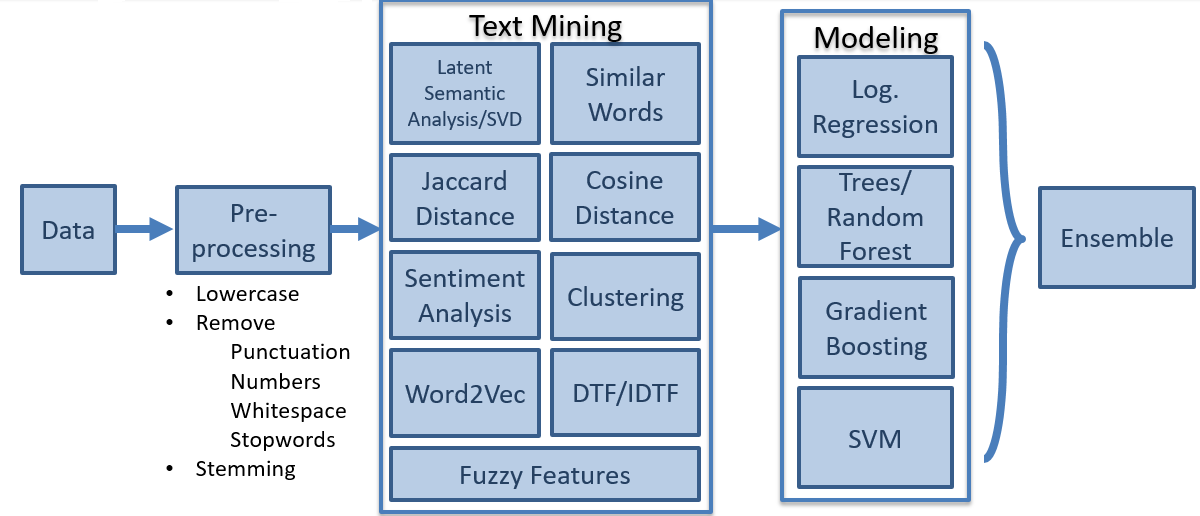
The idea of Quora is that people contribute by asking questions and replying to each other, providing unique insights. One could also call it “crowd intelligence”. This empowers people to learn from each other. Over 372 million people visited the page in March 2017 and made it to the 91st most popular page worldwide [Sim17]. It is therefore no surprise that many ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer and make writers feel the need to answer multiple versions of the same question. This reduces the overall user experience and the quality of the blogging platform.

# **Methodology**

This problem statement has been picked from the Kaggle competition “Quora Question Pairs” [Kag17]. The training dataset consists of six variables with over 404,290 observations. Every row has a pair of questions and a binary variable “is\_duplicate” that specifies whether both questions are redundant to each other. The test dataset consists of three variables with 2,345,796 observations.

After an initial exploration of the data, we concluded that there is no other way than applying text mining techniques to the two questions and append the results as new columns for each record. Those findings shall then be used to build a model to predict the redundancy. An overview of our activities is given in figure 1. Those steps were initially done in R and Python and then tested in SAS JMP and SAS Enterprise Miner.

*Figure 1: Approach*



After creating the subset[[1]](#footnote-1) of data, we performed syntactic operations like removing numbers, punctuations, whitespaces, stop words and building word stem. Furthermore, the data was handled in the “UTF-8” format, since some techniques had problems with reading the text file. Depending on the required input, we either used a document term matrix (dtm) or the preprocessed questions.

Similar Words and Match Ratio

This approach neglects any underlying semantics. Here we simply compute the actual number of words between the preprocessed questions. The Match Ratio is the number of similar words divided by the average length of both questions.

Latent Semantic Analysis/ Single Value Decomposition

Latent Semantic Analysis (LSA) calculates a semantic space from a dtm. It reduces the dimensionality by finding common concepts behind different words. This measure does not simply compare the words’ similarity, but considers underlying semantics as well. Additionally, the LSA\_Correlation uses the *spearman coefficient* to determine the similarity on a scale from -1 to 1.

Jaccard and Cosine Distance

This approach neglects any underlying semantics. Here we simply compute the actual number of words between the preprocessed questions.

Sentiment Analysis

The Sentiment Analysis tries to extract emotions from the questions. The possible for a text are positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust. We calculate the number for each of them per question and set a binary variable to 1, if both texts have the same number of each sentiment. Additionally, we introduce another measure that compares the number of identical to non-identical words. Match\_Count is a ratio of common words to distinct words for both questions.

Fuzzy Features extraction:

Fuzzy features use Levenshtein Distance to calculate the differences between sequences in a simple-to-use package available in python called *fuzzywuzzy*. Levenshtein distance between two words, on a high level, is the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one word into the other.

Word2vec:

Word2vec is a group of related models used to produce word embeddings. It takes a large corpus of text as input and produces a vector space with hundreds of dimensions with every word being assigned a corresponding vector in the space. The word2vec considers the text semantics, hence similar words will have the same directions and the smaller distance between them [WMD17]. For of creating the Word2vec model in python, we used *Genism*, which is a pre-trained model using Google text with millions of words with Deep learning technique.

Clustering

The idea behind clustering is that the more similar, and therefore duplicate, two questions are, the higher the likelihood that a cluster algorithm puts them into the same box. We merge both questions into one vector and run a k-means algorithm with 25 or 50 different clusters. We decided against hierarchical clustering, because this approach may degenerate and either lead to a very small or very big number of clusters, due to the heterogeneity of the text input. Finally, we set another Boolean variable that is 1, if both questions are within the same cluster.

# **Results**

After the initial data preprocessing and applying text mining techniques, we use the extracted features to build predictive models. The goal is to accurately predict whether two questions are duplicate or not. As shown in figure 1, we tested logistic regression, decision trees, random forest, gradient boosting as well as support vector machines for classification. Using a training and testing set with 5,000 observations each, we achieve an accuracy of approx. 73% respectively.

A decision tree map shows *match\_count, jaccard* and *CommonWords* as the three top branches, which have the highest predictive value. A variable importance plot for the random forest model confirms this observation, having *match\_count, match.ratio, cosine* and *jaccard* as most important variables. Interestingly, the results of the cluster analysis seem never to play any important role. Using *gradient boosting, cosine, LSA\_Cor* and *match\_count* have the highest relative influence.

To conclude, due to their nature, various algorithms may use input variables differently and therefore have deviating results. However, among all the inspected models, *match\_count, jaccard, CommonWords* and *match ratio* occurred several times and were therefore valid predictors.

# **Conclusion and Recommendation**

We used a variety of different text mining techniques that range in terms of complexity from low (counter similar words) to highly advanced (Word2Vec, Fuzzy features). Interestingly, we found that complexity is not essentially attribute to better predictive accuracy and that simpler measures attributed to substantial predictive accuracy. The tools we used are code-based (R, Python) or have a user-friendly interface (SAS JMP, SAS Enterprise Miner). The very topic of predicting duplicate questions demands a more sophisticated approach. Therefore, both SAS tools got quickly outdated since they could not offer any of those features. For the future, it is therefore imperative to have knowledge in software tools and in coding. We recommend having rather a wide knowledge of different tools and coding languages than being an expert within one. You can combine several best practices and simply use a tool’s strength for such topics.

Despite the reasonable accuracy achieved in our project, there is still a long way to go. The data preprocessing and feature extraction plays a crucial role in the model accuracy. We can adapt several more feature engineering, normalization, and variable reduction techniques to further enhance the model performance. The hardware limitations forced us to sample the dataset to a few thousand rows. So, with better hardware we will be able to accommodate more training data, which will reasonability improve the accuracy of the model. Human language with all its semantics can hardly be captured by simple text mining techniques like a correlation between two strings. Applying cognitive solutions and deep learning techniques are the next step to overcome those issues, which we have not explored in the current scope of the project.

# **References**

[SAS17] SAS. New in JMP Pro 13. Internet, URL:

<https://www.jmp.com/en_us/software/predictive-analytics-software/new-in-jmp-pro.html>, retrieved on 04/26/17.

[Sim17] SimilarWeb. Quora.com. Internet, URL:

<https://www.similarweb.com/website/quora.com>, retrieved on 04/24/17.

[Kag17] Kaggle. Quora Question Pairs. Internet, URL:

<https://www.kaggle.com/c/quora-question-pairs>, retrieved on 04/24/17.

[WMD17] Word Movers Distance. Internet, URL:

<http://vene.ro/blog/word-movers-distance-in-python.html>, retrieved on 04/20/17.

# **Appendix**

## 8.1 Tools Comparison

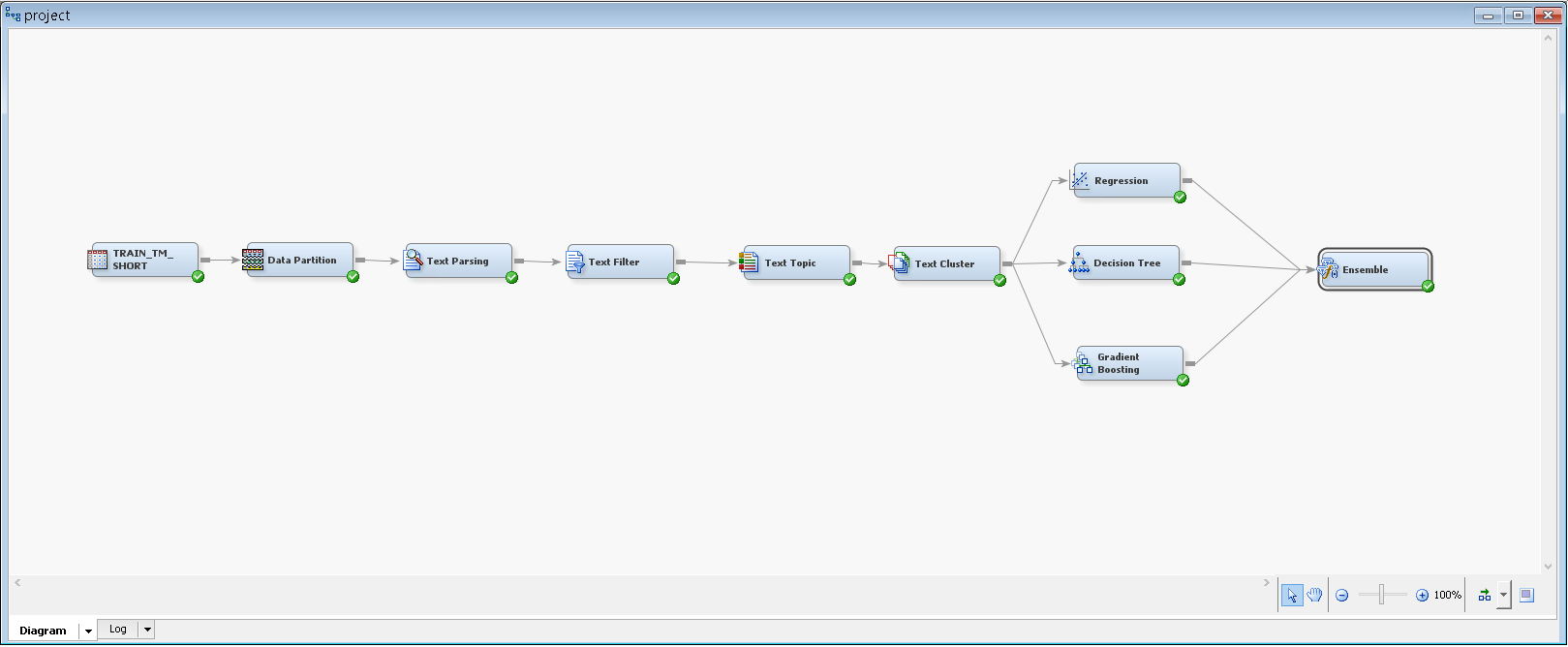
After we built and validated the models that are shown in figure 1, we tried to expand this approach to SAS JMP and SAS Enterprise Miner.

The new version of SAS JMP Pro 13 was not available to us, while we conducted this project. We used the normal JMP 13 version, which was available as a trial, instead. JMP Pro 13 included a text explorer as new feature [SAS17]. We had no practical chance to validate on how the text mining applications from figure 1 could be done in JMP. We used the basic JMP version with its limited text explorer and read the user manual to decide if those applications were possible. We concluded that everything except the Sentiment Analysis, Similar Words, Word2Vec and Fuzzy Features were doable without applying coding.

The following steps were followed in Enterprise Miner for this project:

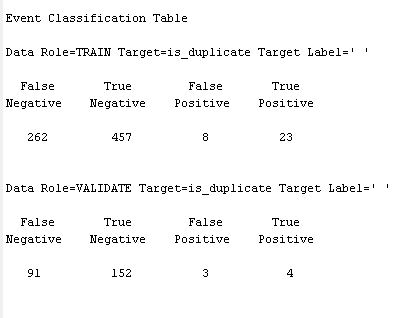
A data partition node was used to partition the data in training/validation in the ratio of 75/25. A text parsing node was attached to the data partition node using all the default properties and SASHELP.ENGSTOP as the stop list. The Text Parsing node enables us to parse the questions in order to quantify information about the terms that are contained therein. A text filter node was attached to the text parsing node and the term weight property was changed to Inverse Document Frequency. The text filter node can be used to reduce the total number of parsed terms that are analyzed. Therefore, helps us to eliminate extraneous information so that only the most valuable and relevant information is considered. After text filter node, a text topic node was attached to the text filter node. The text topic node enables you to explore the questions by automatically associating terms and documents according to both discovered and user-defined topics. In the next step, a text cluster node was attached to the text topic node for clustering the questions into disjoint sets of questions and reports on the descriptive terms for those clusters. After the text cluster node, linear regression, decision tree and gradient boosting models were developed for prediction. And finally, an ensemble node was attached which provided the better results than individual models.

*Figure 2: Project Diagram in SAS Enterprise Miner*



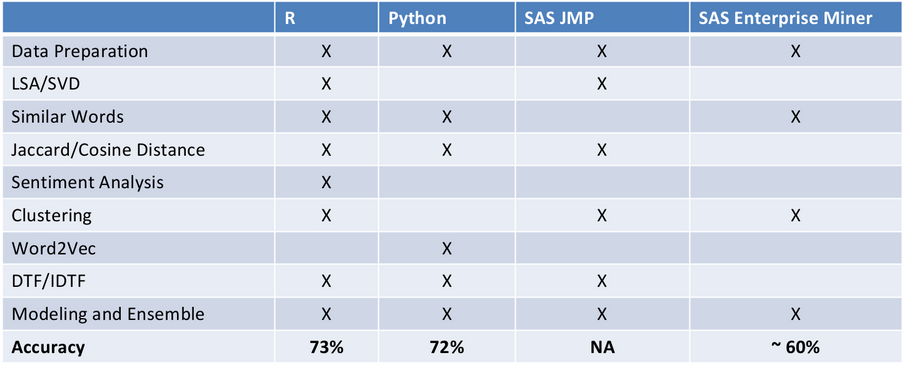
After we implemented the Below is the accuracy of the model we achieved in Enterprise miner.

*Figure 6: Project Diagram in SAS Enterprise Miner*



## 8.2 Approaches and accuracy of tools

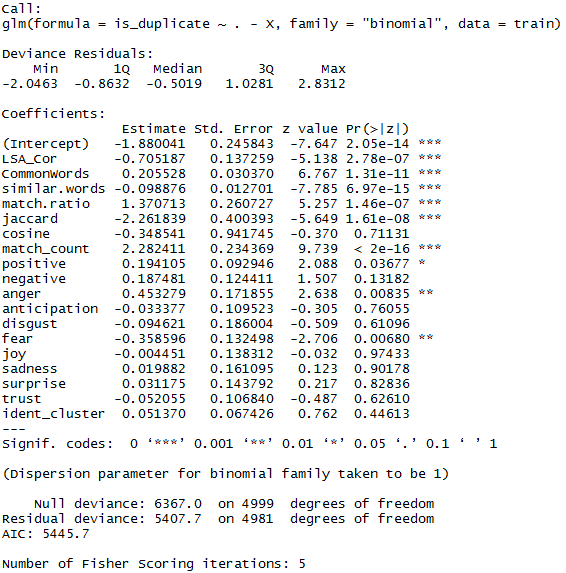
*Figure 3: Overview of capabilities between different tools*



## 8.3 Classification models results

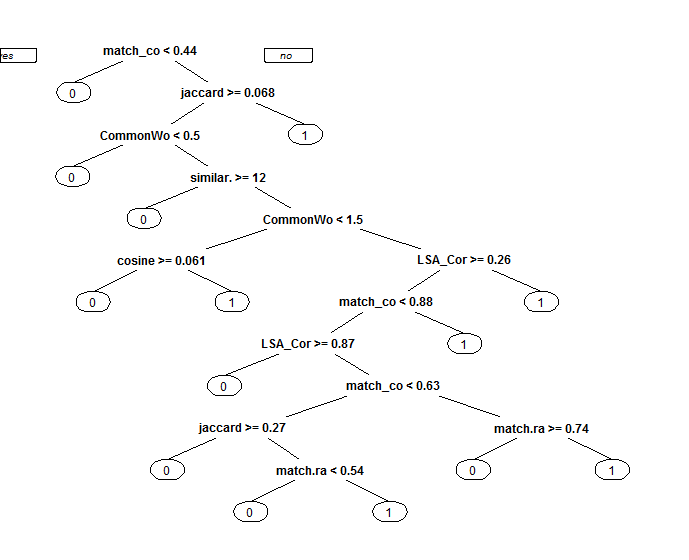
We have used several classification models for our project. Random forest and Gradient boosting techniques gave us the best results. Below is a screenshot of the logistic regression model’s results.

*Figure 4: Logistic Regression Results*



The decision tree model performs the split based on entropy and information gain. The top variables in the decision tree are the ones which contributes the most to the model accuracy. Below is the screenshot of the decision tree we built.

*Figure 5: Decision Tree Map*



## 8.4 Important features

We were also able to assess the contribution of the features extracted towards the accuracy of the mode. As explained earlier, simpler features such as match count, match ration seems to contribute towards the model accuracy more than the advanced features. This is observed in the below screenshot for the Random forest and Gradient boosting models.

|  |  |
| --- | --- |
| *Figure 7: Random Forest Importance Chart* | *Figure 8: Gradient Boost Importance Table* |
|  |  |

1. Working on the complete data set lead to some performance issues [↑](#footnote-ref-1)