

Quora Question Pairs

Business Problem

Description

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.

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.

> Credits: Kaggle __ Problem Statement __ - Identify which questions asked on Quora are duplicates of questions that have already been asked. - This could be useful 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.

Mapping the real world problem to an ML problem

It is a binary classification problem, for a given pair of questions we need to predict if they are duplicate or not.

Performance Metric

Source: <https://www.kaggle.com/c/quora-question-pairs#evaluation> (<https://www.kaggle.com/c/quora-question-pairs#evaluation>)

Metric(s):

- log-loss : <https://www.kaggle.com/wiki/LogarithmicLoss> (<https://www.kaggle.com/wiki/LogarithmicLoss>)
- Binary Confusion Matrix

We build train and test by randomly splitting in the ratio of 70:30 or 80:20 whatever we choose as we have sufficient points to work with.

Our Approach

After doing eda we found out that:

1. There are more 0's than 1's.
2. The 1's are 36.92% whereas the 0's are 63.08%
3. There are 537,933 unique questions of which 111,780 occurred more than once.
4. Only one question appeared 157 times
5. There were no duplicates but we found out that there are 2 questions with null Values. So we replaced them with a space

Basic Feature Extraction (before cleaning)

Let us now construct a few features like:

- **freq_qid1** = Frequency of qid1's
- **freq_qid2** = Frequency of qid2's
- **q1len** = Length of q1
- **q2len** = Length of q2
- **q1_n_words** = Number of words in Question 1
- **q2_n_words** = Number of words in Question 2
- **word_Common** = (Number of common unique words in Question 1 and Question 2)
- **word_Total** = (Total num of words in Question 1 + Total num of words in Question 2)
- **word_share** = (word_common)/(word_Total)
- **freq_q1+freq_q2** = sum total of frequency of qid1 and qid2
- **freq_q1-freq_q2** = absolute difference of frequency of qid1 and qid2

Preprocessing of Text

- Preprocessing:
 - Removing html tags
 - Removing Punctuations
 - Performing stemming
 - Removing Stopwords
 - Expanding contractions etc.(like 1000 to 1k, % to percentage etc)

Advanced Feature Extraction (NLP and Fuzzy Features)

Definition:

- **Token**: You get a token by splitting sentence a space
- **Stop_Word** : stop words as per NLTK.
- **Word** : A token that is not a stop_word

Features:

- **cwc_min** : Ratio of common_word_count to min length of word count of Q1 and Q2

$$\text{cwc_min} = \text{common_word_count} / (\min(\text{len}(q1_words), \text{len}(q2_words)))$$
- **cwc_max** : Ratio of common_word_count to max length of word count of Q1 and Q2

$$\text{cwc_max} = \text{common_word_count} / (\max(\text{len}(q1_words), \text{len}(q2_words)))$$
- **csc_min** : Ratio of common_stop_count to min length of stop count of Q1 and Q2

$$\text{csc_min} = \text{common_stop_count} / (\min(\text{len}(q1_stops), \text{len}(q2_stops)))$$
- **csc_max** : Ratio of common_stop_count to max length of stop count of Q1 and Q2

$$\text{csc_max} = \text{common_stop_count} / (\max(\text{len}(q1_stops), \text{len}(q2_stops)))$$
- **ctc_min** : Ratio of common_token_count to min length of token count of Q1 and Q2

$$\text{ctc_min} = \text{common_token_count} / (\min(\text{len}(q1_tokens), \text{len}(q2_tokens)))$$
- **ctc_max** : Ratio of common_token_count to max length of token count of Q1 and Q2

$$\text{ctc_max} = \text{common_token_count} / (\max(\text{len}(q1_tokens), \text{len}(q2_tokens)))$$
- **last_word_eq** : Check if First word of both questions is equal or not

$$\text{last_word_eq} = \text{int}(q1_tokens[-1] == q2_tokens[-1])$$
- **first_word_eq** : Check if First word of both questions is equal or not

$$\text{first_word_eq} = \text{int}(q1_tokens[0] == q2_tokens[0])$$
- **abs_len_diff** : Abs. length difference

$$\text{abs_len_diff} = \text{abs}(\text{len}(q1_tokens) - \text{len}(q2_tokens))$$
- **mean_len** : Average Token Length of both Questions

$$\text{mean_len} = (\text{len}(q1_tokens) + \text{len}(q2_tokens)) / 2$$
- **fuzz_ratio** : <https://github.com/seatgeek/fuzzywuzzy#usage> (<https://github.com/seatgeek/fuzzywuzzy#usage>)
<http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/> (<http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/>)
- **fuzz_partial_ratio** : <https://github.com/seatgeek/fuzzywuzzy#usage> (<https://github.com/seatgeek/fuzzywuzzy#usage>) <http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/> (<http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/>)
- **token_sort_ratio** : <https://github.com/seatgeek/fuzzywuzzy#usage> (<https://github.com/seatgeek/fuzzywuzzy#usage>)
<http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/> (<http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/>)
- **token set ratio** : <https://github.com/seatgeek/fuzzywuzzy#usage> (<https://github.com/seatgeek/fuzzywuzzy#usage>)

ML MODELS:

1. We did a random model so that we can find out the threshold for the log-loss.
2. After Building it we found out that the log loss was 0.89 wich means that the models would have a log loss of less than 0.89.
3. We did tf idf w2v and tiidf vectorizers on the dataset.
4. We did the following models and following are our results:

tfidf w2v

SR	Model	Log-Loss
1.	Random	0.887
2.	Logistic Regression	0.520
3.	Linear SVM	0.489
4.	XGBOOST	0.357

tfidf

SR	Model	Log-Loss
1.	Random	0.890
2.	Logistic Regression with SGD Classifier	0.492
3.	Logistic Regression	0.488
4.	linear SVM	0.492
5.	XGBOOST	0.429