

LAKNATH'S COLUMN

Ramblings of a wanderer

[WHO AM I?](#) [HOME](#)

SEP 12 2017

2 COMMENTS

FUN, MACHINE LEARNING, NLP,
PYTHON

NOTES FROM QUORA DUPLICATE QUESTION PAIRS FINDING KAGGLE COMPETITION

[Quora duplicate question pairs Kaggle competition](#) ended a few months ago, and it was a great opportunity for all NLP enthusiasts to try out all sorts of nerdy tools in their arsenals. This is just jotting down notes from that experience.

Introduction

Quora has over 100 million users visiting every month, and needs to identify duplicate questions submitted — an incident that should be very common with such a large user base. One interesting characteristic that differentiates it from other NLP tasks is the limited amount of context available in the title; in most cases this would amount to a few words.

Exploratory data analysis

The dataset is simple as it can get: both training and test sets consist of two questions in consideration. Additionally, in the training set there are few extra columns: one denoting whether it's a duplicate, and two more for unique IDs of each question.

qid1, qid2 – unique ids of each question (only available in the training set)
question1, question2 – the full text of each question
is_duplicate – the target variable; set to 1 if question1 and question2 essentially have the same meaning; 0 otherwise.

Some quick stats:

- Training set size – 404,290
- Test set size – 2,345,796
- Total training vocabulary – 8,944,593
- Avg. word count per question – 11.06

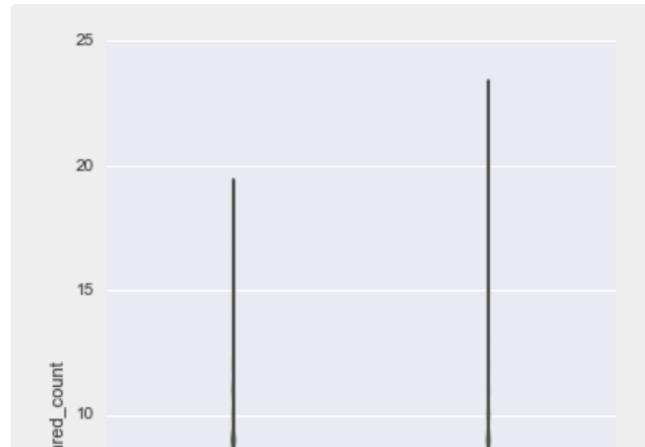
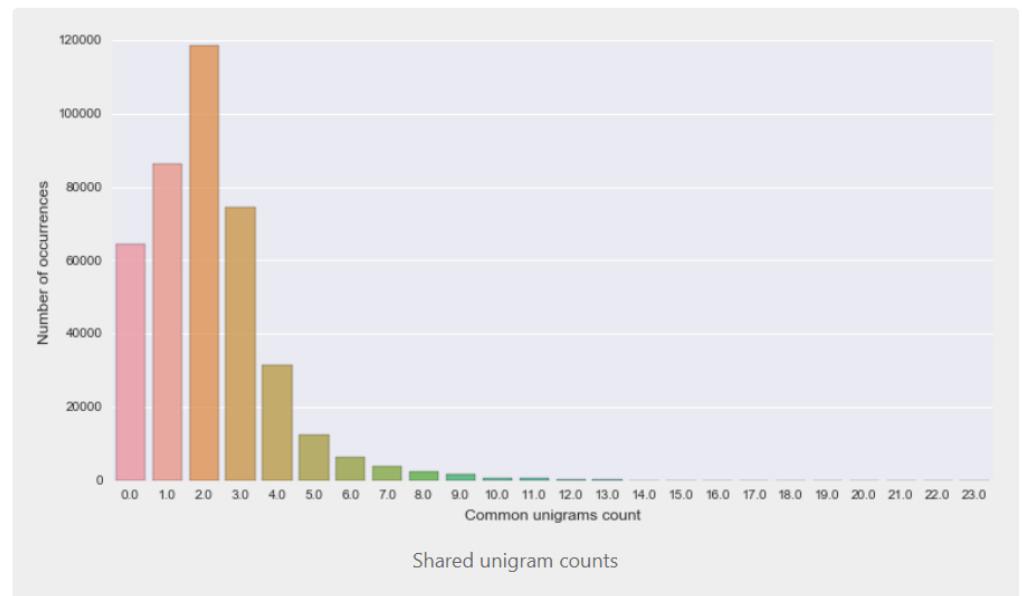
A quick EDA reveals some interesting insight to the dataset.

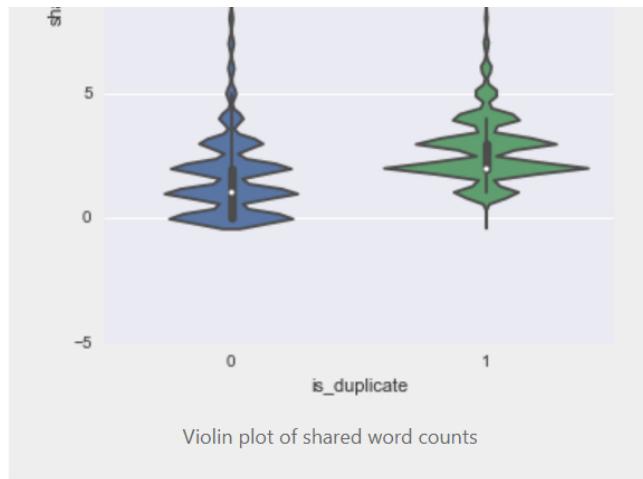
- Classes are not balanced.



In the training/validation set, the duplicate percentage (label 1) is ~36.9%. Since the class balance can influence some classifiers, this fact becomes useful when training models later.

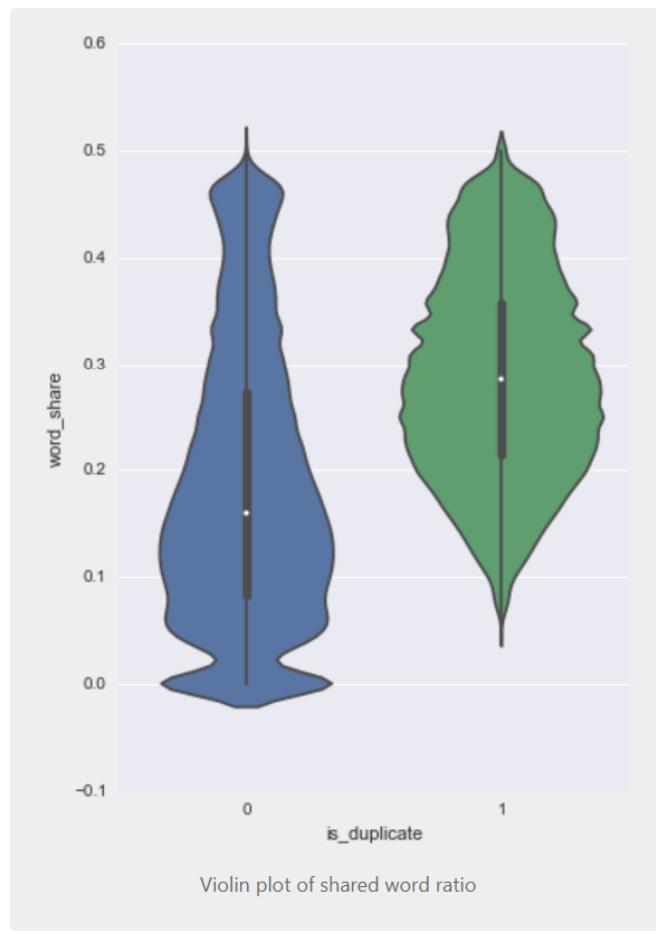
- Normalized unigram word shared counts can be a good feature





When the shared word ratio (Jaccard similarity) is considered, this becomes even more prominent:

$$\frac{\text{question1 words} \cap \text{question2 words}}{\text{question1 words} \cup \text{question2 words}} \quad (1)$$



The correlation of shared unigram counts towards the class further indicates that other n-grams can also perhaps participate as features in our model.

Arguably the best perk of being part of a Kaggle competition is the incredible community. Here are some in-depth EDAs carried out by some of its members:

- <https://www.kaggle.com/philschmidt/quora-eda-model-selection-roc-pr-plots>
- <https://www.kaggle.com/sowbarani/quora-question-pairs-eda>
- https://www.kaggle.com/cudalairikumar/simple_looky_exploration_notebook_quora

Statistical modelling

XGBoost is a gradient boosting framework that has become massively popular, especially in the Kaggle community. The popularity is not underserving as it has won many competitions in the past and known for its versatility. So as the primary model, XGBoost was used with following parameters, selected based on the performance of the validation set.

```
objective = 'binary:logistic'  
eval_metric = 'logloss'  
eta = 0.11  
max_depth = 5
```

Before discussing features used, there's one neat trick that I believe everyone who did well in the competition used. After the first few submissions of prediction results, it became apparent that there's something wrong when you compare the results obtained against the validation set with the Kaggle leaderboard (LB). No matter how many folds were used for the validation set, the results obtained against the validation set didn't reflect on the LB. This is due to the fact that the class balance between the training set and the test set was considerably different, and the cost function (logloss) being sensitive to the imbalance. Specifically, in the training set around 37% were positive labels while in the test set it was approximated to be around 16.5%. So some oversampling of the negatives in the training set was required to get a comparable result on the LB. More on oversampling can be found [here](#) and [here](#).

Features

From a bird's eye view, features used can be categorised into three groups.

1. Classical text mining features
2. Embedded features
3. Structural features

Following features can be categorised under classical text mining features.

- Unigram word match count
- Ratio of the shared count (against the total words in 2 questions)
- Shared 2gram count
- Ratio of sum of shared tf-idf score against the total weighted word score
- Cosine distance
- Jaccard similarity coefficient
- Hamming distance
- Word counts of q1, q2 and the difference ($\text{len}(q1), \text{len}(q2), \text{len}(q1) - \text{len}(q2)$)
- Caps count of q1, q2 and the difference
- Character count of q1, q2 and difference
- Average length of a word in q1 and q2
- Q1 stopword ratio, Q2 stopword ratio, and the difference of ratios
- Exactly same question?

Since a large portion of sentence pairs are questions, many duplicate questions are starting with the same question word (which, what, how .etc). So few more features were used to indicate whether this clause applies.

- Q1 starts with 'how', Q2 starts with 'how' and both questions have 'how' (3 separate features)
- same for words 'what', 'which', 'who', 'where', 'when', 'why'

Some fuzzy features generated from the [script here](#), which in turn used [fuzzywuzzy package](#),

- Fuzzy WRatio
- Fuzzy partial ratio
- Fuzzy partial token set ratio
- Fuzzy partial token sort ratio
- Fuzzy qratio
- Fuzzy token set ratio
- Fuzzy token sort ratio

As for the embedded features, [Abhishek Thakur's script](#) did everything needed: it generates a word2vec representation of each word using a pre-trained word2vec model on Google News corpus using [gensim package](#). It then generates a sentence representation by normalizing each word vector.

```
def sent2vec(s):
    words = str(s).lower().decode('utf-8')
    words = word_tokenize(words)
    words = [w for w in words if not w in stop_words]
    words = [w for w in words if w.isalpha()]
    M = []
    for w in words:
        try:
            M.append(model[w])
        except:
            continue

    M = np.array(M)
    v = M.sum(axis=0)

    return v / np.sqrt((v ** 2).sum())
```

Based on the vector representations of the sentences, following distance features were generated by the same script.

- [Word mover distance](#)
- Euclidean distance
- [Braycurtis dissimilarity](#)
- [Canberra distance](#)
- [Manhattan distance](#)
- [Minkowski distance \(with \$p = 3\$ \)](#)

Combined with these calculated features, full 300 dimension word2vec representations of each sentence were used for the final model. The raw vector addition required a large expansion of the AWS server I was using, but in hindsight brought little improvement.

Structural features have caused much argument within the community. These features aren't meaningful NLP features, but because of the way how the dataset was formed, it had given rise to some patterns within the dataset. It's doubtful if these features will be much use in a real-word scenario, but within the context of the competition, they gave a clear boost. so I guess everyone used them disregarding whatever moral compunctions one might have had.

These features include,

- Counting the number of questions shared between two sets formed by the two questions

```
from collections import defaultdict

q_dict = defaultdict(set)

def build_intersects(row):
    q_dict[row['question1']].add(row['question2'])
    q_dict[row['question2']].add(row['question1'])

def count_intersect(row):
    return len(q_dict[row['question1']].intersection(q_dict[row['quest']])
```

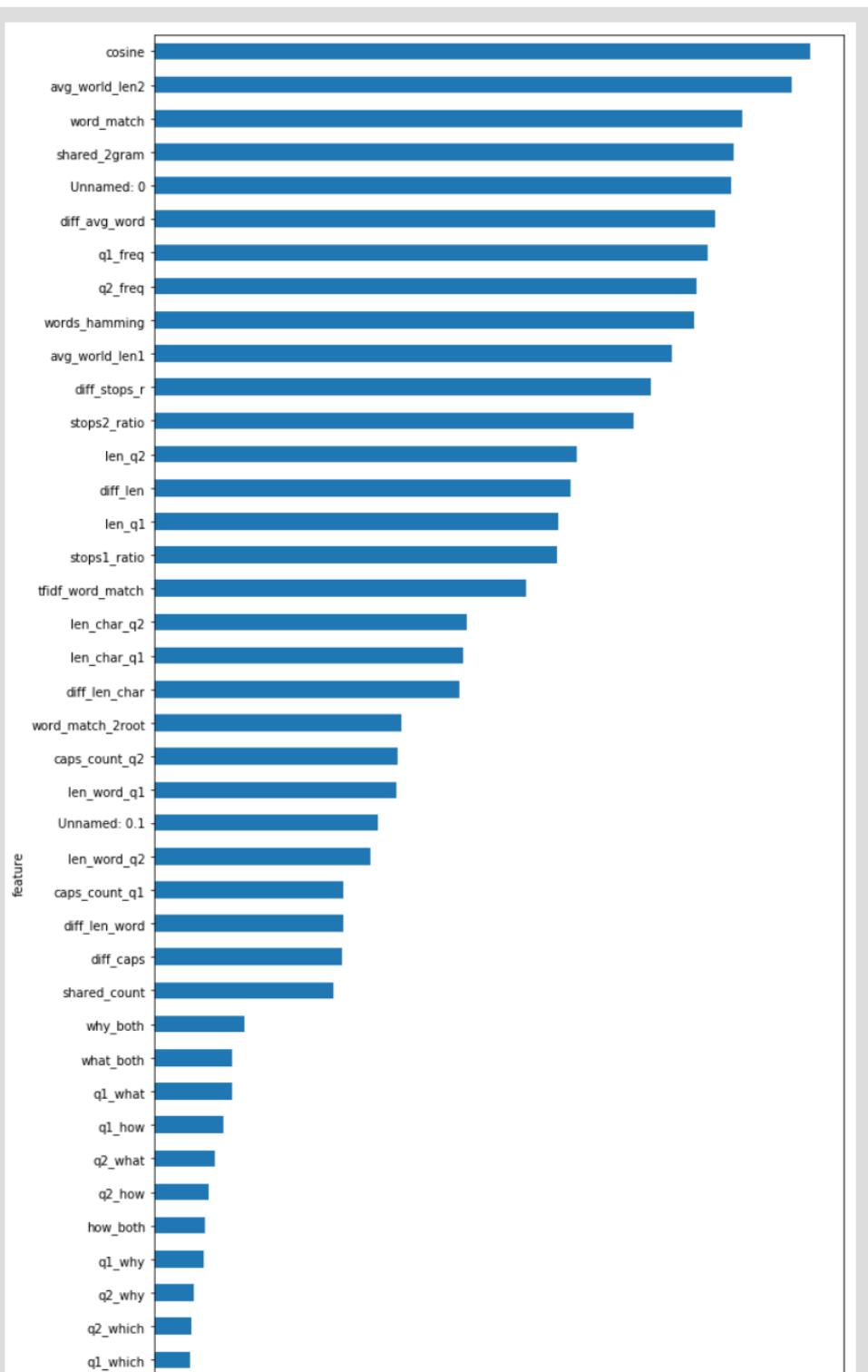
```
df_train.apply(build_intersects, axis=1, raw=True)
df_train.apply(count_intersect, axis=1, raw=True)
```

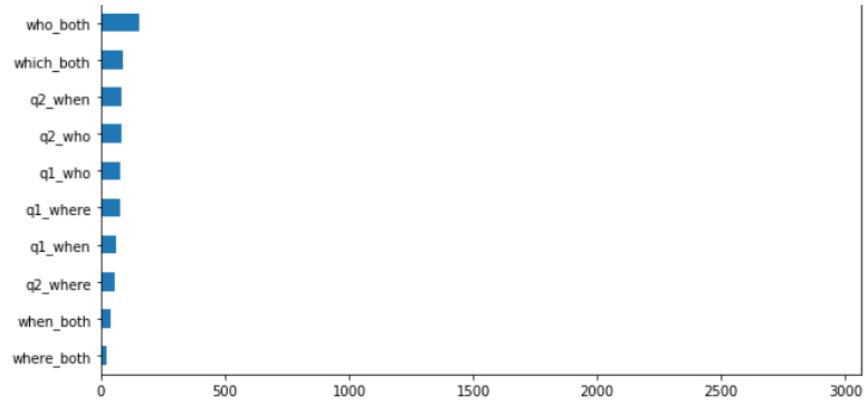
- Shard word counts, tf-idf and cosine scores within these sentence clusters
- The page rank of each question (within the graph induced by questions as nodes and shared questions as edges)
- Max k-cores of the above graph

Results

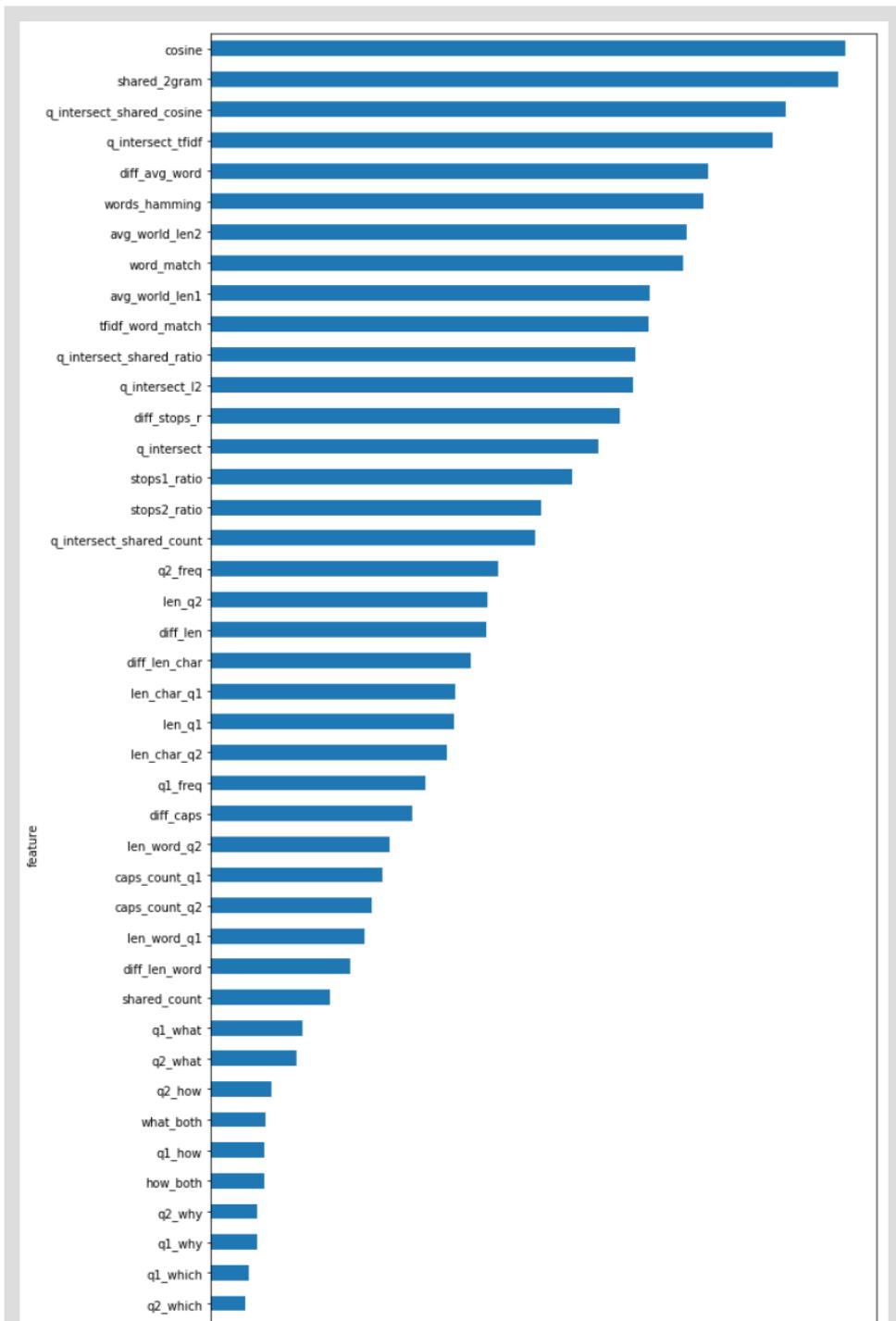
The effect of features on the final result can be summarized by following few graphs.

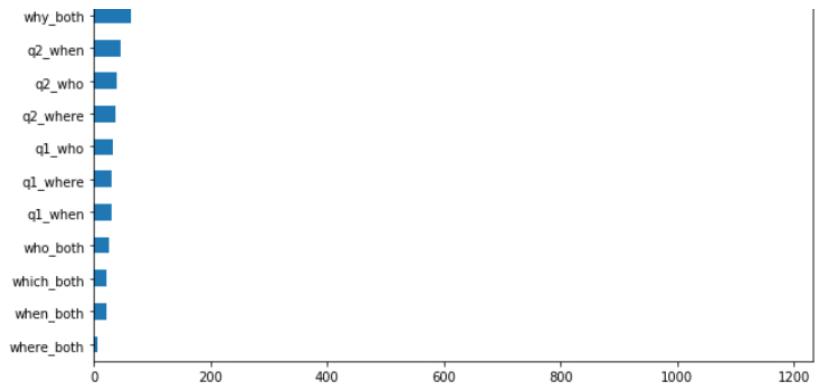
- With only classic NLP features (at XGBoost iteration 800):
 - Train-logloss:0.223248, eval-logloss:0.237988 (**0.21861 on LB**)



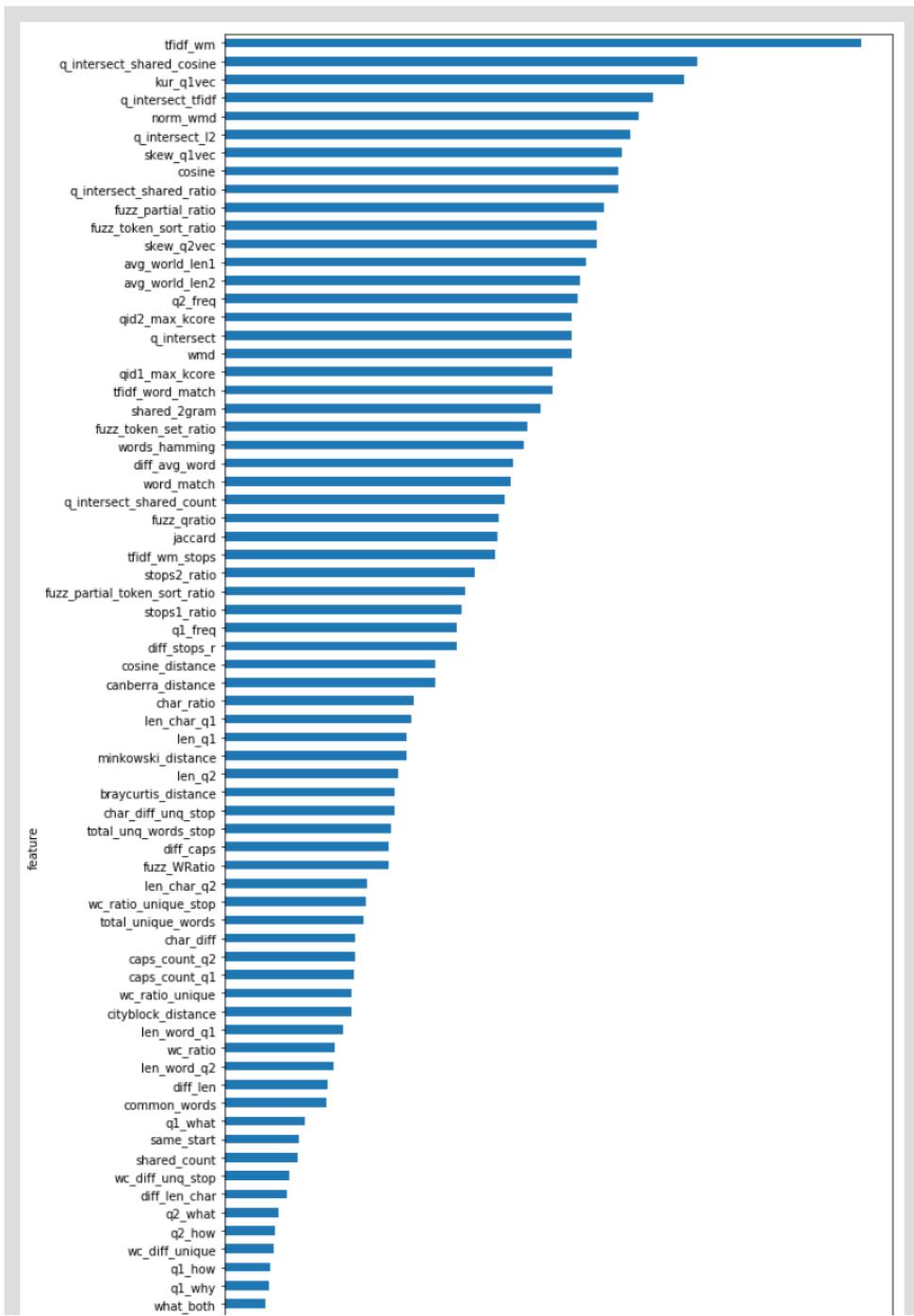


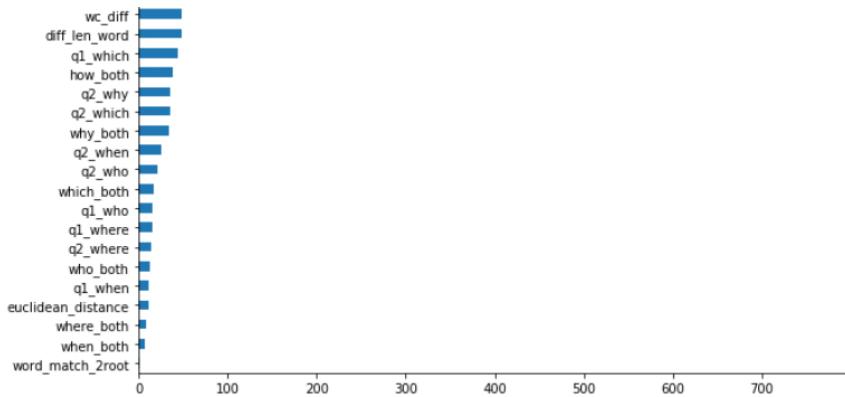
- With both classic NLP + structural features (at XGBoost iteration 800):
 - Training accuracy: 0.929, Validation accuracy: 0.923
 - Train-logloss:0.17021, eval-logloss:0.185971 (**LB 0.16562**)





- With classic NLP + structural + embedded features (at XGBoost iteration 700):
 - Training accuracy: 0.938, Validation accuracy: 0.931
 - Train-logloss: 0.149663, eval-logloss: 0.1654 (**LB 0.14754**)





Rank wise this feature set and the model achieved a max 3% at one point, though it came down to 7% by the end due to my lethargic finish. But considering it was an individual effort against mostly other team works consisting several ensemble models, I guess it wasn't bad. More than anything, it was great fun and a good opportunity to play with some of the [best ML competitors in the Kaggle community](#)/world and collaboratively learn from that community.

I've shared the repository of Jupyter notebooks used and can be found from [here](#).

PS: Some of the stuff that I wanted to try out, but didn't get to:

- [LSTM based deep model](#)
- Distance measures using a lexical database such as [Wordnet](#)
- Embedded features with [GloVe](#), [fastText](#)
- [CNNs](#)
- [Stacking and ensambling](#)

Tagged [kaggle](#), [machine learning](#)

2 thoughts on “Notes from Quora duplicate question pairs finding Kaggle competition”



Jose says:

[February 1, 2018 at 1:35 pm](#)

Thank you for your work. Just a note: there is an indentation error in Abhishek Thakur's code snippet.

[Reply](#)



lknath says:

[February 26, 2018 at 6:00 pm](#)

Thank you Jose! Fixed it.

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

[Comment](#)

Save my name, email, and website in this browser for the next time I comment.

Post Comment

[← Previous post](#) [Next post →](#)

ARCHIVES

- [April 2018](#)
- [September 2017](#)
- [January 2017](#)
- [March 2012](#)
- [December 2011](#)
- [October 2011](#)
- [September 2011](#)
- [December 2009](#)
- [May 2009](#)
- [January 2009](#)
- [December 2008](#)
- [October 2008](#)
- [July 2008](#)
- [June 2008](#)
- [May 2008](#)
- [February 2008](#)
- [January 2008](#)
- [December 2007](#)
- [November 2007](#)
- [August 2007](#)
- [July 2007](#)
- [June 2007](#)
- [May 2007](#)
- [April 2007](#)
- [March 2007](#)
- [February 2007](#)
- [January 2007](#)
- [December 2006](#)

CATEGORIES

- [.Net](#)
- [Backups](#)
- [Cricket](#)
- [Databases](#)
- [Drupal](#)
- [Eclipse](#)
- [education](#)
- [FIT](#)
- [Flash](#)
- [FOSS](#)
- [Fun](#)
- [Gnome](#)
- [GSOC](#)
- [inspiration](#)
- [Java](#)
- [javascript](#)
- [machine learning](#)
- [Maya](#)
- [Movies](#)
- [My Activities](#)
- [NLP](#)
- [Novels](#)
- [PHP](#)
- [politics](#)
- [Python](#)
- [saas](#)
- [servers](#)
- [Sri Lanka](#)
- [ubuntu](#)
- [Uncategorized](#)

LAKNATH'S COLUMN

- [Is this a toxic comment?](#)
- [Notes from Quora duplicate question pairs finding Kaggle competition](#)
- [The necessity of lifelong learning](#)
- [Machine Learning in SaaS paradigm](#)
- [Avoiding AWS potholes](#)

- [Uncategorized](#)
 - [web](#)
 - [Web Designing](#)
-

Proudly powered by [WordPress](#) | Theme: [Chunk](#) by [WordPress.com](#).