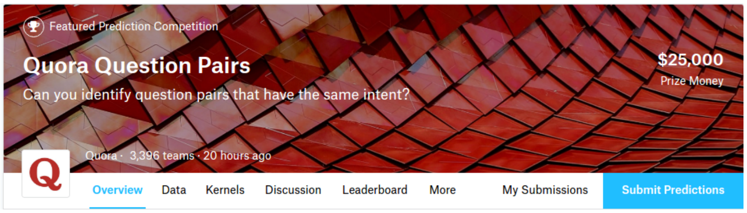
Identifying duplicate questions on Quora | Top 12% on Kaggle!

Shubhankar Srivastava Follow
Jun 8, 2017 · 10 min read



Featured Prediction Competition

Quora Question Pairs

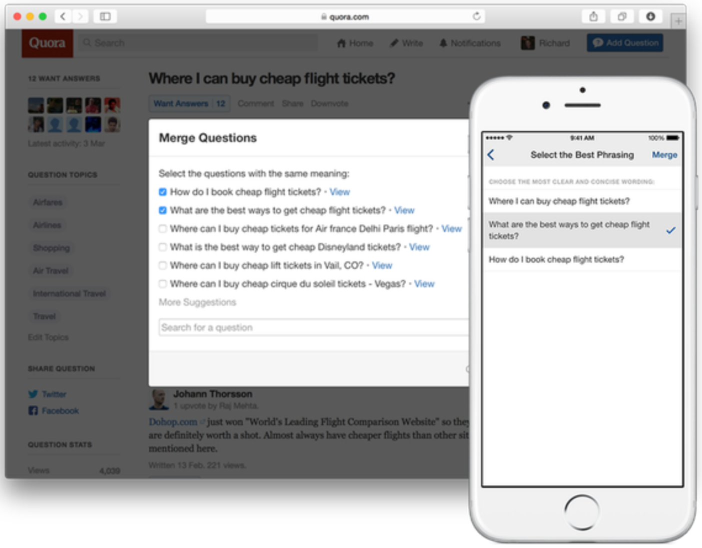
Can you identify question pairs that have the same intent?

$25,000 Prize Money

Quora · 3,396 teams · 20 hours ago

> *If you are a regular Quoran like me, you have most likely stumbled on duplicate questions asking the same essential question.*

This is a bad user experience for both writers and seekers, as the answers get fragmented across different versions of the same question. In fact, this is a problem which is very evident on other social Q&A platforms like StackOverflow. A real-world problem out there, begging AI to solve it :)



Where I can buy cheap flight tickets?

Merge Questions

Select the questions with the same meaning:

☑ How do I book cheap flight tickets? · View
☑ What are the best ways to get cheap flight tickets? · View
□ Where can I buy cheap tickets for Air france Delhi Paris flight? · View
□ What is the best way to get cheap Disneyland tickets? · View
□ Where can I buy cheap lift tickets in Vail, CO? · View
□ Where can I buy cheap cirque du soleil tickets - Vegas? · View

More Suggestions

Johann Thorsson
1 upvote by Raj Mehta.
Dohop.com just won "World's Leading Flight Comparison Website" so they are definitely worth a shot. Almost always have cheaper flights than other sit mentioned here.
Written 13 Feb. 221 views.

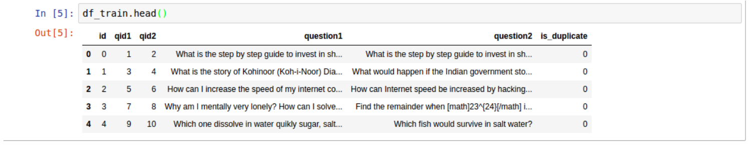
Select the Best Phrasing

CHOOSE THE MOST CLEAR AND CONCISE WORDING:

Where I can buy cheap flight tickets?
What are the best ways to get cheap flight tickets? ✓
How do I book cheap flight tickets?

For the past month or so i've spent my nights cracking my skull open at this problem, along with three thousand other kagglers! It was my second serious competition in a row, it was stressful at times, but i learnt a lot overall—Landed a Top 12% position, a discussion gold medal and a couple of kernel bronze medals :)

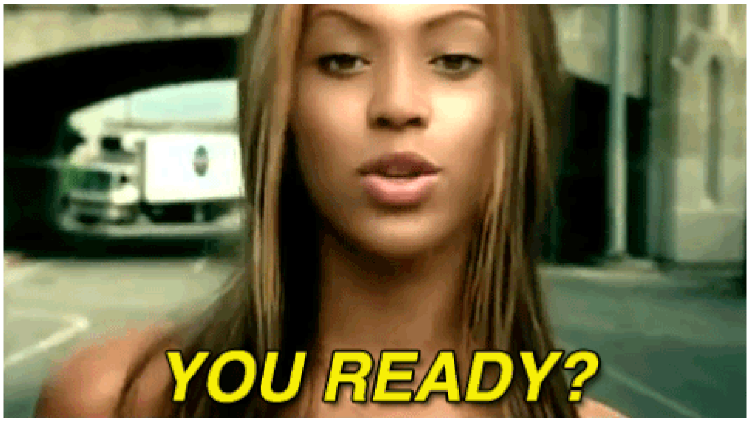
About the problem—Quora has given an (almost) real-world dataset of question pairs, with the label of is_duplicate along with every question pair. The objective was to minimize the logloss of predictions on duplicacy in the testing dataset. There were around 400K question pairs in the training set while the testing set contained around 2.5 million pairs. Yeah, 2.5 million! A large majority of those pairs were computer-generated questions to prevent cheating, but 2 and a half million, god! I was maxing out my poor 8GB machine every other hour :(



```
In [5]: df_train.head()
Out[5]:
```

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

My approach—I started off with the xgboost starter by @anokas, and built upon it gradually. My feature-set involved around 70 features which is on a rather lower range, when compared to the Top Kagglers' approaches. My features could be broadly classified into NLP-based features, word `embedding based distances and graph-based features. Let me elaborate:

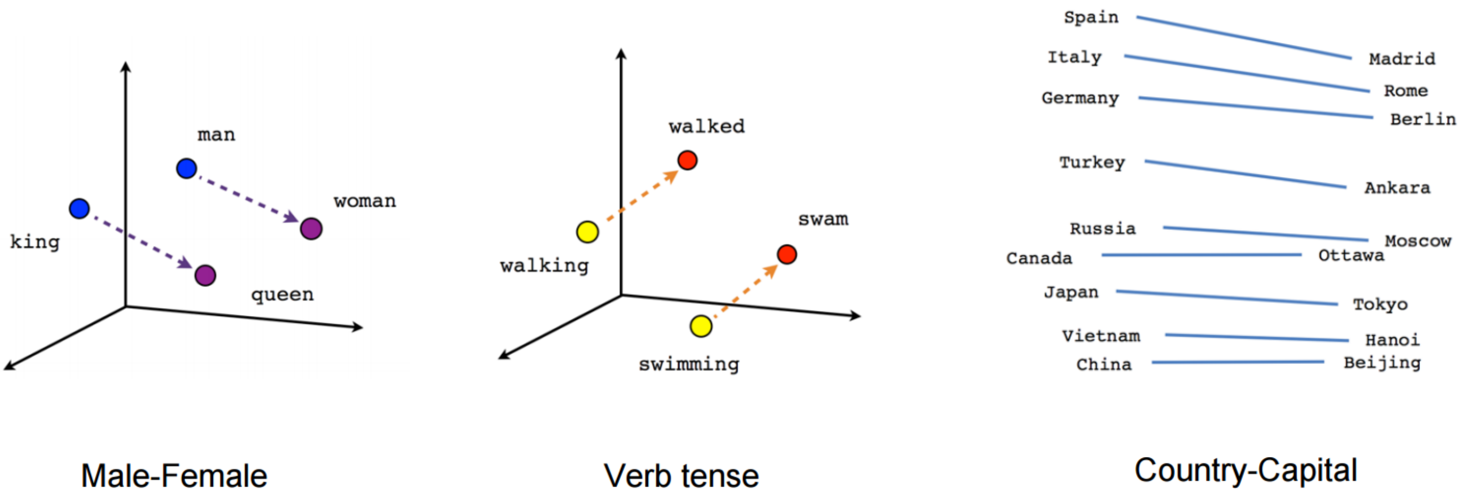


NLP Features: Some very obvious text-based features are percentage of words matching between the two questions, length of the two questions, number of words, number of sentences, number of stopwords, the usual natural language stuff! I tried going ahead with tfidf scores, but that was both not very useful and computationally expensive. Instead, one of the most important features turned out to be weighted word match share, where each word's weight is the inverse frequency of the word in the corpus(basically idf)—if i have a rare word in both the questions, they might be discussing similar topics. I also had features like cosine distance, jaccard distance, jarowinkler distance, hamming distance, and n-gram matches(shingling).

A library i used extensively for NLP tasks was Spacy, which has been developing some great functionalities lately—spacy similarity turned out to be a good feature too. Some creative ones i thought of were related to the kind of question—whether it is a "How" question or "Why" question—dependent on the first word of the sentence. Strangely, when modelling this i should've thought of building "last word similarity" as well, completely missed that! Named entities are a crucial key to understand the context of a question—thus common_named_entity score and common_noun_chunk score were an obvious choice. I pushed out a kernel about calculating similarity via the wordnet corpus, however wordnet fell short of spacy in terms of ease of use, speed and vocabulary size.

Word Embeddings based Distances:

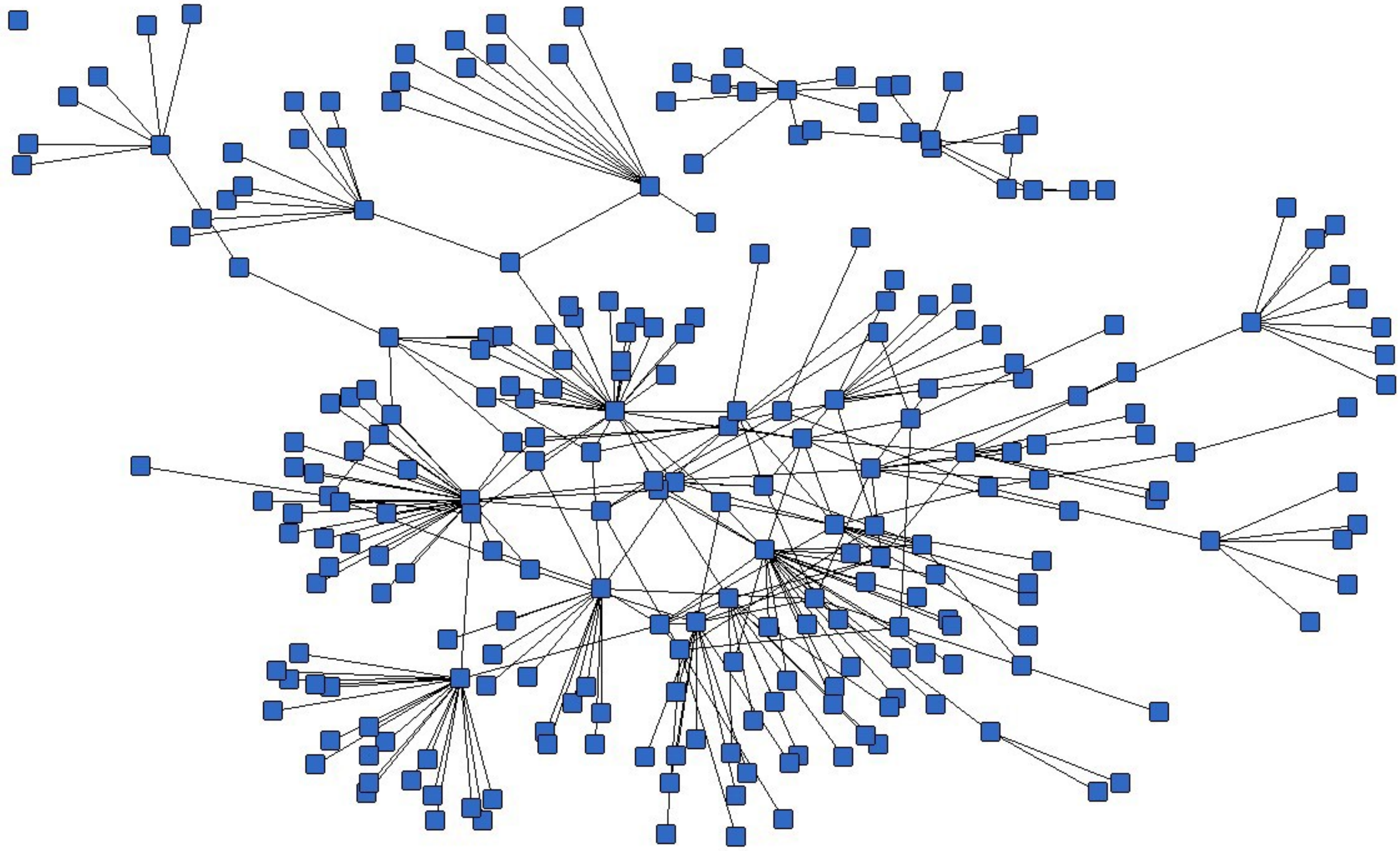
When doing a NLP competition, can Word2Vec be left behind! I feel like word2vec might be the coolest computer science concept i've read, i always get blown away by its effectiveness. Every.Single.Time.



Word2Vec Magic!

Anyway, i had the questions mapped to 300-dimensional vectors in a Sent2Vec format, resulting in a vector for every question. Naturally, distance-based features between vectors were built—cosine, cityblock, jacard, canberra, euclidean and braycurtis. Must mention @abhishek's scripts for the inspiration for these features. Unfortunately, i couldn't build the real embedding layers which i could pass in to the lstm layers in keras—my RAM just wouldn't let me. I'm getting some serious hardware, soon!

Graph Features: In an NLP competition, these graph features played quite the spoilsport! However, it was a nice reminder of how the theories of social networks might apply in datasets like Quora's question pairs.



A typical graph structure in social media

Here, every question is a node in the graph and a question pair in the dataset indicates an edge between the two nodes. We can use the graph structure of the test data as well, since we're not considering the is_duplicate label anywhere—Those 2 million edges contributed a whole lot to the graph! There were heated discussions between kagglers on whether the graph-based features are supposedly "magic features", which should be released to level the playing field. Anyway, all of these features gave significant boosts to most models:

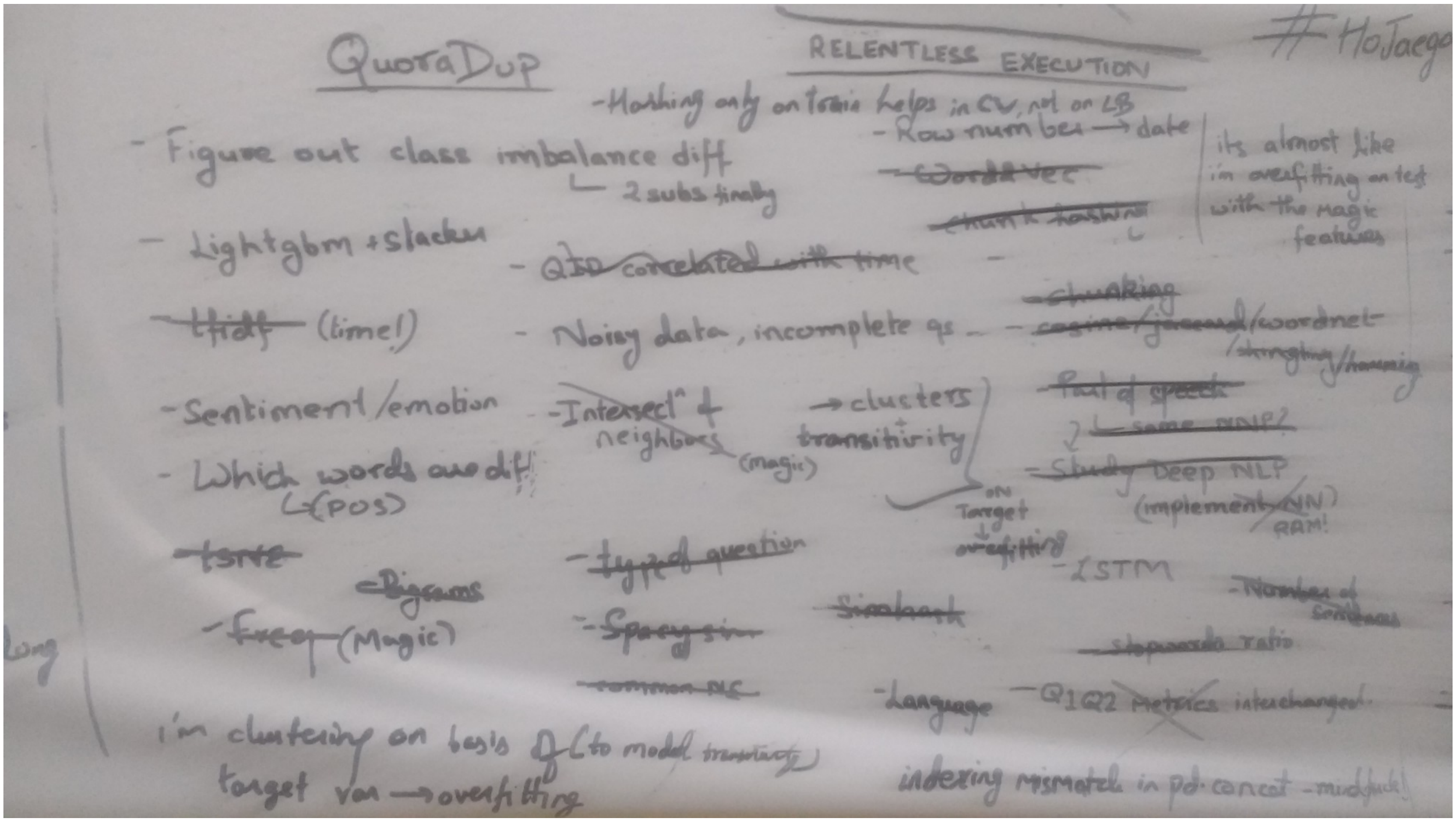
- Degree of a node: Essentially, frequency of the question, there'll be as many edges as many times that question has occurred in the dataset. This feature gave a huge gain, because the questions sampling(upsampling the duplicates) done by Quora was most likely dependent on this frequency.
- Intersection of neighbors: Percentage of first-degree neighbors of the question pair, for eg. Q1 has neighbors Q2,Q3,Q4 and Q2 has

neighbors Q1,Q3. The common neighbors for (Q1,Q2) are Q3, one half of all first degree neighbors.

- Degree of Separation: This was a feature i thought of and implemented via breadth-first-search, didn't give a lot of improvement though.
- PageRank: I implemented this feature, even put out a kernel on kaggle—Higher pageranked questions are linked to important(higher pageranked) questions, while spammy questions are linked to spammy ones.
- kcore/kclique: K-core is basically the largest subgraph where every node is connected to at least "k" nodes, didn't give me a lot of gain though. I had thought of kclique, but didn't implement it because of lack of time :(Turned out it was a pretty important one!
- Weighted Graph: Later in the competition, fellow kagglers shared the idea of having a weighted graph, where weight of every node is the weighted_word_share(we'd discussed this earlier). Neighbor intersection in this weighted graph was useful.

Transitivity Magic—Continuing on the graph structure, modelling transitivity between the questions was an obvious approach. For eg. if Q1 is similar to Q2 and Q2 is similar to Q3, it implies Q1 is similar to Q3 more often than not(as per our dataset). This was one of the features i had begun building but left it midway, a costly mistake. Many top solutions used this feature in some form or another, a simpler version was to average probabilities of duplicacy between every question and its counterpart's neighbors.

My Whiteboarding Sessions



QuoraDup

RELENTLESS EXECUTION

#HoJaega

-Hashing only on train helps in CV, not on LB

- Figure out class imbalance diff
  └ 2 subs finally
- Lightgbm + stacker
- ~~Q1D correlated with time~~
- ~~tfidf~~ (time!)
- Noisy data, incomplete qs
- Sentiment/emotion
- ~~Intersect^n of neighbors~~ (magic) → clusters + ~~transitivity~~ on Target ↓ overfitting
- Which words are diff!!
  └(POS)
- ~~tsne~~
- ~~Bigrams~~
- ~~Freq~~ (Magic)
- ~~type of question~~
- ~~Spacy sim~~
- ~~common NE~~
- ~~Siamese~~
- ~~Language~~
- Row number → date
- ~~Word2Vec~~
- ~~chunk hashing~~
- ~~chunking~~
- ~~cosine/jaccard~~/wordnet /~~stringdist~~/~~hamming~~
- ~~Part of speech~~
  ~~└ same NNP?~~
- ~~Study~~ Deep NLP (implement ~~NN~~) RAM!
- LSTM
- ~~Number of sentences~~
- ~~stopwords ratio~~
- Q1Q2 ~~metrics~~ interchanged.

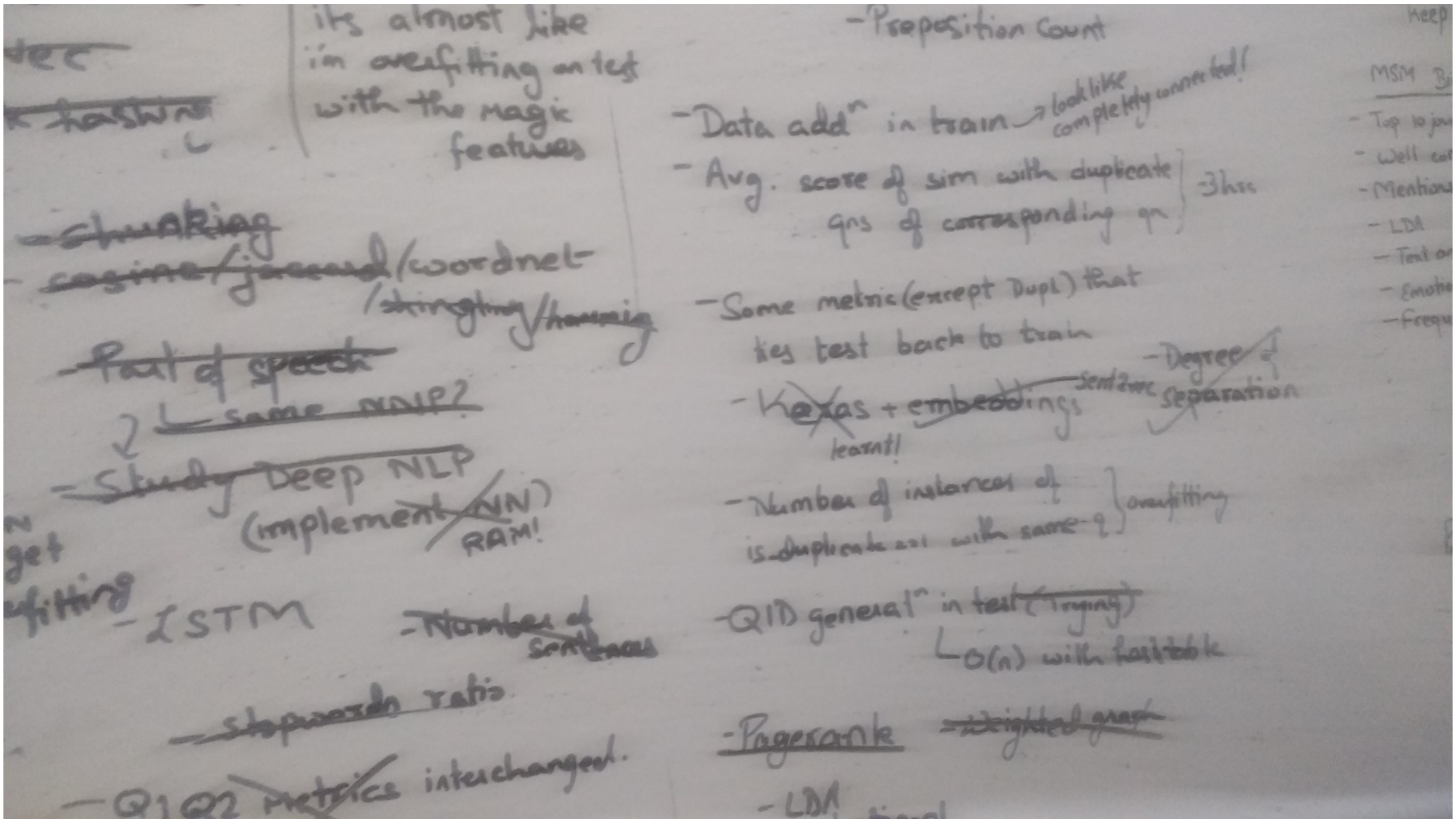
its almost like i'm overfitting on test with the magic features

i'm clustering on basis of target var → overfitting (to model transitivity)

indexing mismatch in pd.concat - mindfuck!

long

Part 1 of 2



its almost like i'm overfitting on test with the magic features

~~vec~~
~~k hashing~~
- ~~chunking~~
- ~~cosine/jaccard~~/wordnet /~~stringdist~~/~~hamming~~
- ~~Part of speech~~
  ~~└ same NNP?~~
- ~~Study~~ Deep NLP (implement ~~NN~~) RAM!
- LSTM
- ~~Number of sentences~~
- ~~stopwords ratio~~
- Q1Q2 ~~metrics~~ interchanged.

-Preposition Count

-Data add^n in train → look like completely connected!

- Avg. score of sim with duplicate qns of corresponding qn } 3hrs

-Some metric (except Dupl) that ties test back to train

- ~~Keras~~ + ~~embeddings~~ Sent2Vec learnt!

-Degree of separation ✓

-Number of instances of is_duplicate ==1 with same q } overfitting

-QID generat^n in ~~test (Trying)~~
  └ O(n) with hashtable

-Pagerank ~~weighted graph~~

- LDA

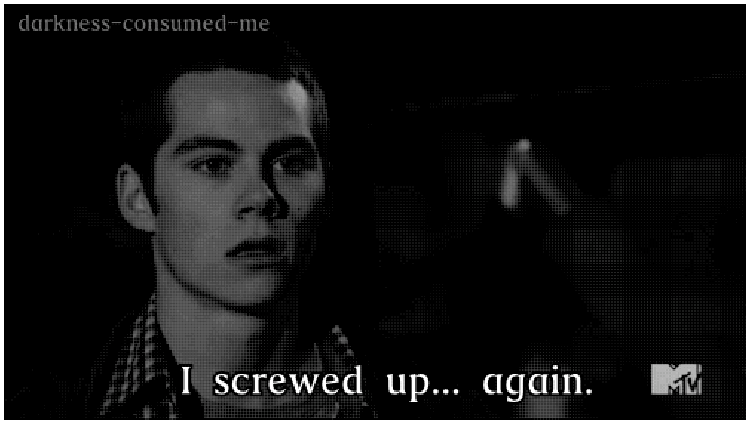
keep

MSM B

- Top 10 jou
- Well co
- Mention
- LDA
- Text o
- Emotio
- Frequ

Part 2 of 2

Some fuckups



When a noob like me dives head-first into kaggling big-time, there are bound to be fuckups. Even though i make a conscious effort to keep my pipeline modular and version-controlled, i lost almost a weekend's worth of work because of a nasty bug in my pipelining. Learning it the hard way, i git reverted and remodelled the pipeline on an ipython notebook. A major challenge that i faced when building features was writing memory-efficient code and not rebuilding previous features, modularity was key. This led to another fuckup—since i couldn't handle the whole of my testing dataset in my RAM, i was breaking it into six subsets and building features iteratively. This meant i was doing a pandas concat between the older dataframe with the new feature, little did i concentrate on indexing :(Spent more than a couple days scratching my head at all NaN features. Noob mistake.

The QID Conundrum:

The training dataset had an ID for every question, thus a QID1 and QID2 in every row. However, this wasn't the case with the testing dataset which meant "QID" couldn't be used as a feature straightaway. A fellow kaggler released an incredibly creative observation of the decreasing average duplicate ratio(rolling mean) with increasing QID —Mostprobably an indication of Quora's improving algorithm with time, thus reducing the number of duplicate questions with increasing ID. This inference is based on the assumption that the QID values aren't masked and are truly representative of time of posting the question. To model QIDs in our testing dataset, i had a hashtable which mapped question texts to QIDs. Now iterating over all questions in test_df—If i encountered an existent question, the corresponding QID was assigned to it. Else we assumed that this was a new question in order of time of posting, incrementing QID by 1. This led to various features like QID difference, mean QID, min QID with the hope of modelling the decrease of duplicacy over time.

Class Imbalance:

A major part of discussions were centered around guesstimating the class split in the testing dataframe, which wasn't clearly similar to the testing split. The mathy folks figured a narrow range of the split via a couple of constant valued submissions, here and here. There were around 34% positive duplicates in the training set, while the testing set had an estimated 16%-17% positive duplicates—Probably a result of the improved Quora algorithm or a result of the computer-generated question pairs. Anyway, a misrepresentative training dataset wasn't going to help matters—people came up with oversampling solutions(duplicating the negative rows in training), or rescaled their predictions by an appropriate factor.

> ***My models—XGBoost is love, XGBoost is life***

Very embarassed to admit, but my submission was just a single model solution, a 2000-round xgboost. To be fair, i didn't spend a lot of time on parameter tuning or building diverse models, left it for too late. I tried the default random forest and GBM, not to a better effect than XGB. Stacking and ensembling were in the roadmap too, but that just stayed there :(

Instead, i spent a very long weekend at learning Keras and building dense neural nets—'twas my first time! Went through understanding various hyperparameters and ideal architectures, the theory behind activation functions, dropout layers and optimizers! Very interesting stuff! I played around with my a two-layer sigmoid neural net with my 70 features tuning it to perfection, it never came close to the xgboost though. As i mentioned earlier, i couldn't build embedding layers or LSTMs with my hardware, that would've definitely helped. Soon.

Proud of my modelling pipeline though, didn't have to fret over making a function for every model– will be helpful when i build hundreds of models to stack. Soon.

Top Solutions:

Post-competition writeups of the gold medalists just makes me feel like a complete noob! I have to really up my game, and work harder to get up there, the bronze isn't far away! Some of the major learnings from winners' solutions:

- Most teams rescaled their final predictions based on the weighted graph's edges or the frequency of nodes
- Every top team had built a stacker with hundreds of models, with a reusable pipeline for model building.
- State-of-the-art neural net architectures(Siamese/Attention NNs), LightGBMs were used, but even lower performing models like ExtraTrees or Random Forests help!
- Graph features were the core of many solutions, as teams used various techniques to derive value out of it. Some removed spurious(less frequent) nodes, some used mean/median of neighbors' weights, while most modelled transitivity.
- Apparently, being Question1 or Question2 mattered too. Surprising!
- NLP: process the text in many different ways—lowercase and unchanged, punctuation replaced in different ways, stop words included and excluded, stemmed and not stemmed, etc.
- Stacking is important, but people achieved 0.14x with a single xgb model too.
- First place post. Truly humbling.

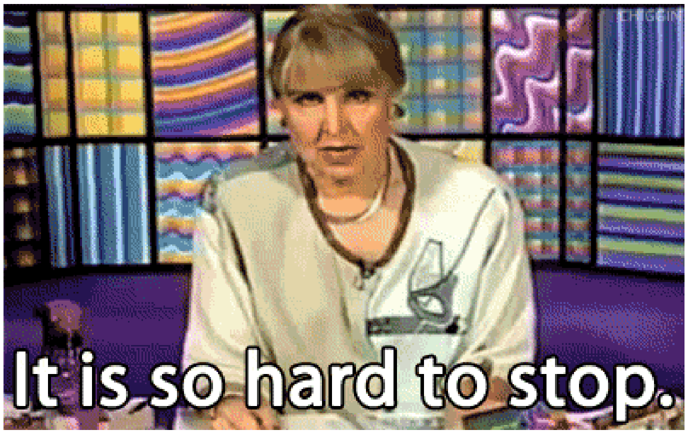
Conclusions:

I should manage my time better, assigning appropriate time for exploration, feature engineering, model building and stacking. Really felt the crunch in the last week as i ran short of submissions.

- My modelling pipeline was good for testing, but i need to scale up operations on building more models on different hyperparams/subsets of data.
- Don't give up on building features midway, some of them turned out to be costly misses.

- Don't be too bent up on a feature just because you spent time on it. This is kinda funny lol!
- Spend a lot of time on Kaggle Kernels and Discussions. They're damn cool :)

So yeah, that's that. One competition ends, another begins. Kaggle is addictive!



It's true!

This was originally posted on my blog. One Blog Every Week