# Detecting Duplicate Quora Questions

## Objective

Predict whether the questions in a pair are a duplicate of each other or not.

## Applicability

Forum Sites:

- Better user experience
  - inquirers  may get their answers right away
  - respondents do not get annoyed by repeat questions
- Better use of the infrastructure for the provider, may result in cost savings

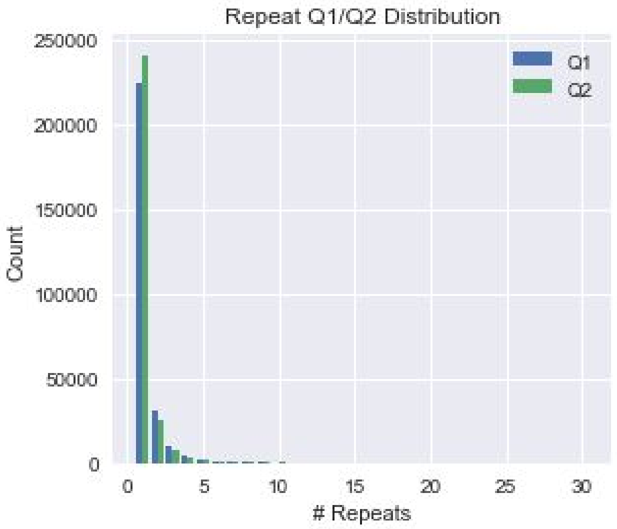
## Input Data

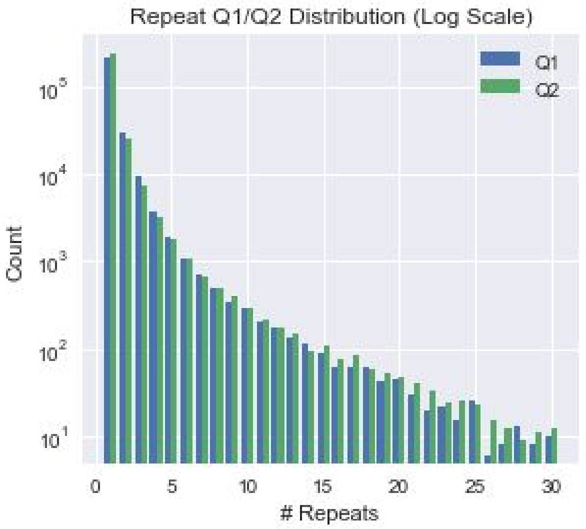
A CSV file provided by Kaggle with the following structure:

```
"id","qid1","qid2","question1","question2","is_duplicate"
"0","1","2","What is the step by step guide to invest in share market in
india?","What is the step by step guide to invest in share market?","0"
"20","41","42","Why do rockets look white?","Why are rockets and boosters
painted white?","1"
```

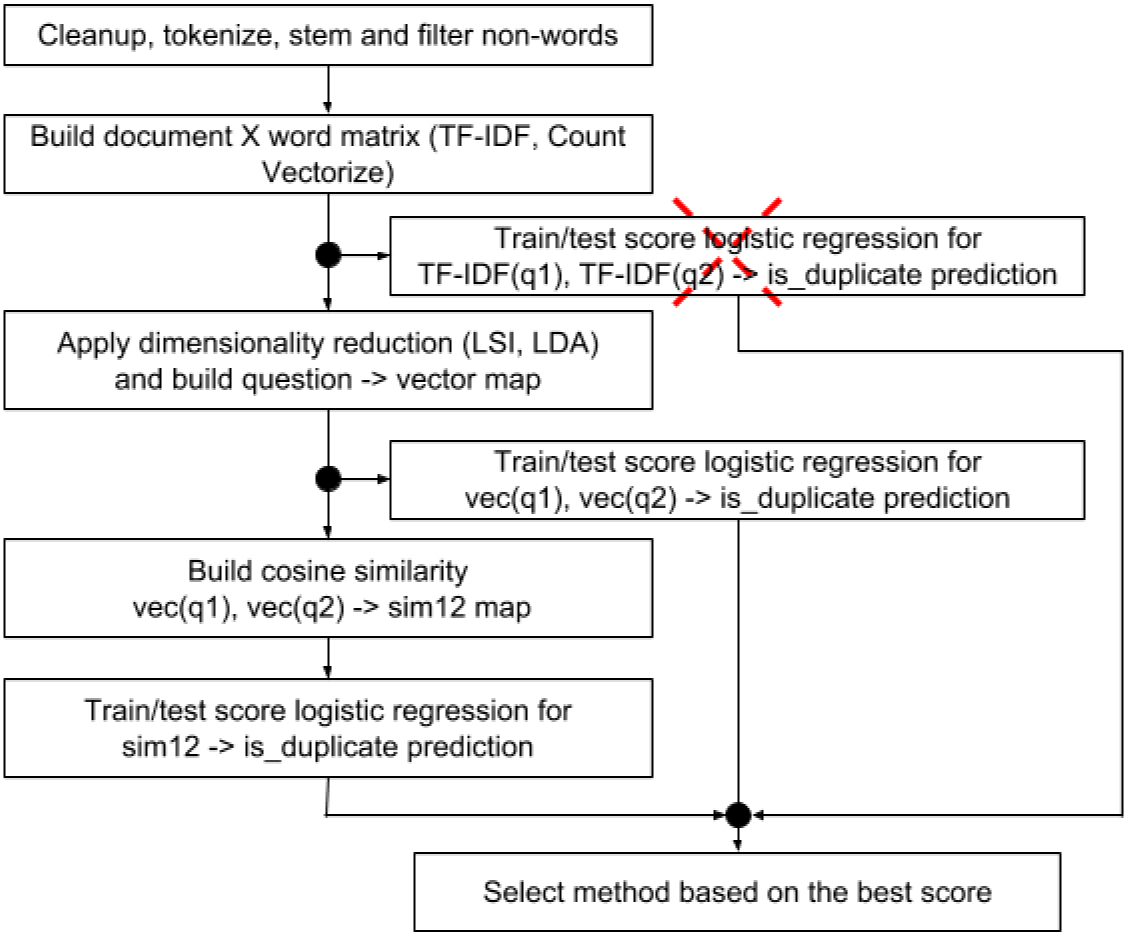
## Exploratory Data Analysis

- 381310 questions
- is_duplicate ratio: .37
- Repeat Q1/Q2 distribution





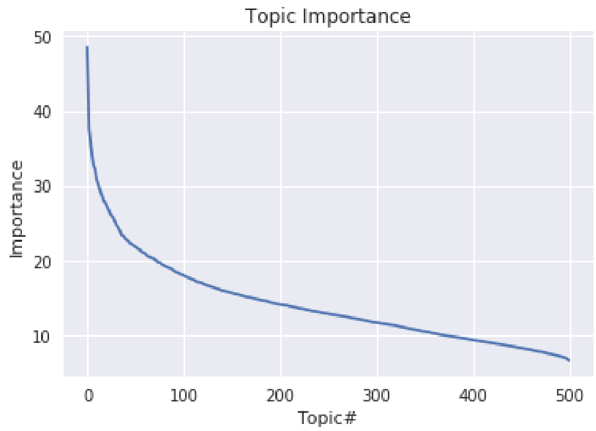
# Workflow


Cleanup, tokenize, stem and filter non-words
Build document X word matrix (TF-IDF, Count Vectorize)
Train/test score logistic regression for TF-IDF(q1), TF-IDF(q2) -> is_duplicate prediction
Apply dimensionality reduction (LSI, LDA) and build question -> vector map
Train/test score logistic regression for vec(q1), vec(q2) -> is_duplicate prediction
Build cosine similarity vec(q1), vec(q2) -> sim12 map
Train/test score logistic regression for sim12 -> is_duplicate prediction
Select method based on the best score

Notes about the workflow:

- Used scores: **accuracy**, relevant for the business use case and **log-loss**, relevant for prediction quality assessment (Kaggle's favorite measure)
- The sooner the logistic regression (LR) is applied along the pipe:
  - The better the chance of getting accurate results
  - The more computational intensive it is
- LR at TF-IDF level turns out to be computationally impractical given the shape of the input matrix: 305048x65172 because LR runs very slow on sparse matrices; for this reason it was skipped

# Number Of Topics Selection



- LSI

Start w/ 500 topics and check the elbow plot. 200 topics should be enough, however 500 may be tried too if  there are enough computing resources

- LDA

200 topics, considering that LDA is better at generating topics

# Results And Method Selection

| **NLP** | **Num Topics** | **Estimator** | **Accuracy**<br>(higher is better) | **Log Loss**<br>(lower is better) |
|---|---|---|---|---|
| LSI | 200 | LR on Topics | 0.721 | 0.559 |
| LSI | 200 | LR on Similarity | 0.639 | 0.625 |
| LSI | 500 | LR on Topics | 0.721 | 0.558 |
| LSI | 500 | LR on Similarity | 0.643 | 0.620 |
| LDA | 200 | LR on Topics | 0.717 | 0.609 |
| LDA | 200 | LR on Similarity | 0.636 | 0.632 |
| Random guessing 1 with  p = .37 (is_duplicate ratio) | | | 0.534 | 0.659 |

The best results were achieved for LSI w/ 200 topics, and LR on topics.

# Factors Impacting The Performance

- No synonym awareness:

```
Q1: 'How much equity should I give to CTO?'
Q2: 'How much equity should I offer a CTO?'
is_dup=1, pred=0, sim=0.28923148979392366
```

- No meaning awareness:

  ```
  Q1: 'What should a person do when everything goes wrong in their life?'
  Q2: 'What should do when nothing goes right in life?'
  is_dup=1, pred=0, sim=0.33404120300891316
  ```

- Number exclusion:

  ```
  Q1: 'What are the best Doctor Who episodes with the 10th Doctor?'
  Q2: 'What are the best Doctor Who episodes with the 5th Doctor?'
  is_dup=0, pred=1, sim=0.998440850819464
  ```

# Future Ideas

- Use Word2Vec because it is better at capturing the context for words
- Cluster Word2Vec representations into synonym groups
- Convert docs into word -> synonym-group representation lists
- Use the converted docs as input for TF-IDF and PCA
- Use a Neural Network for prediction