DUPLICATE QUESTION IDENTIFICATION

USING NATURAL LANGUAGE PROCESSING METHODS

CAPSTONE PROJECT BY SIRI SURABATHULA
MENTOR - SRDJAN SANTIC

SPRINGBOARD DATA SCIENCE CAREER TRACK

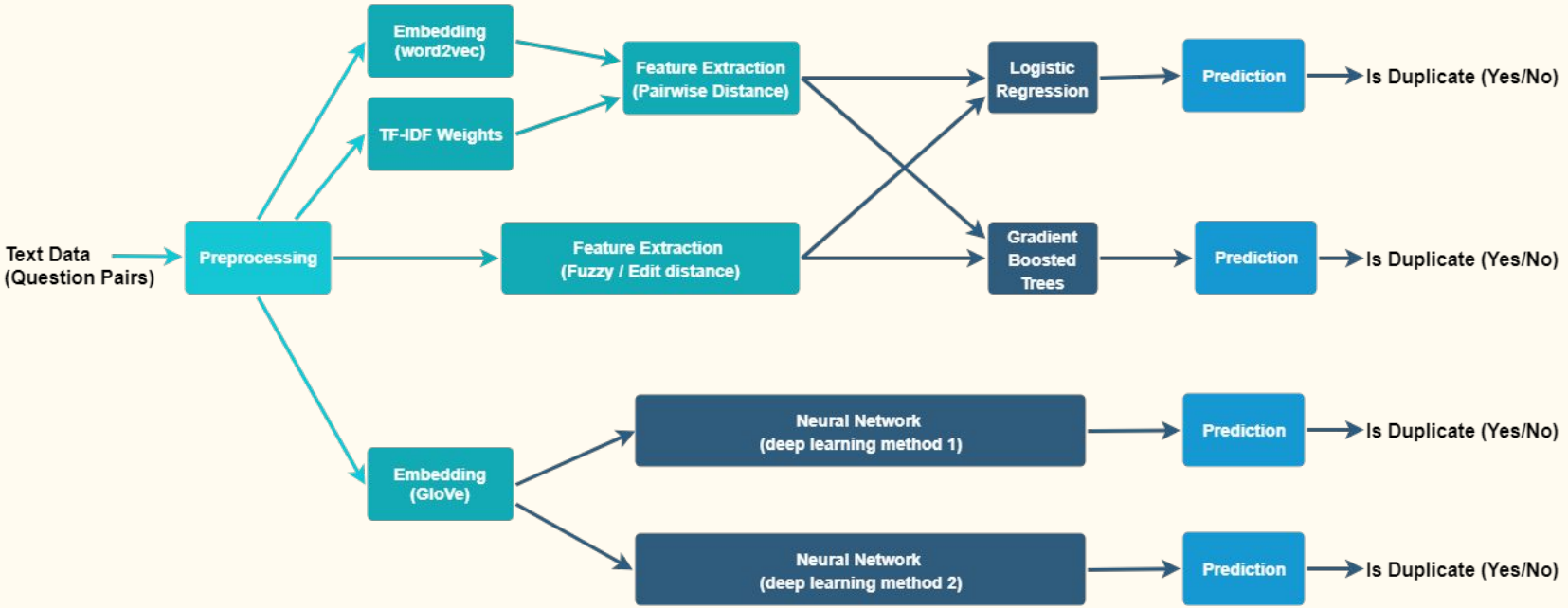
PROBLEM

Question-answering and knowledge-sharing platforms like Quora and Stack-Exchange require mechanisms to group questions in their database based on similar intent. The questions ‘What is the best way to travel from Houston to Atlanta ?’ and ‘Should I drive, fly or take a bus from Houston to Atlanta ?’, are worded differently but have the same intent.

User experience is greatly improved if Quora is able to identify duplicate questions as this enables them to find questions that have already been answered and also avoids the need to answer the same question multiple times.

PROBLEM SOLVING APPROACH

This project explores different classification methods to solve the problem of duplicate identification.



DATA

The data contains a human-labeled training set, and a test set. Each record in the training set represents a pair of questions and a binary label indicating if it is a duplicate or not.

Pair ID	Q1 ID	Q2 ID	Question 1	Question 2	Is Duplicate
3214	678	679	Is talent nature or nurture ?	Are people talented by birth or can it be developed / learnt ?	1
3215	734	735	Is Haskell better than Clojure ?	How do I learn Haskell or CLojure ?	0

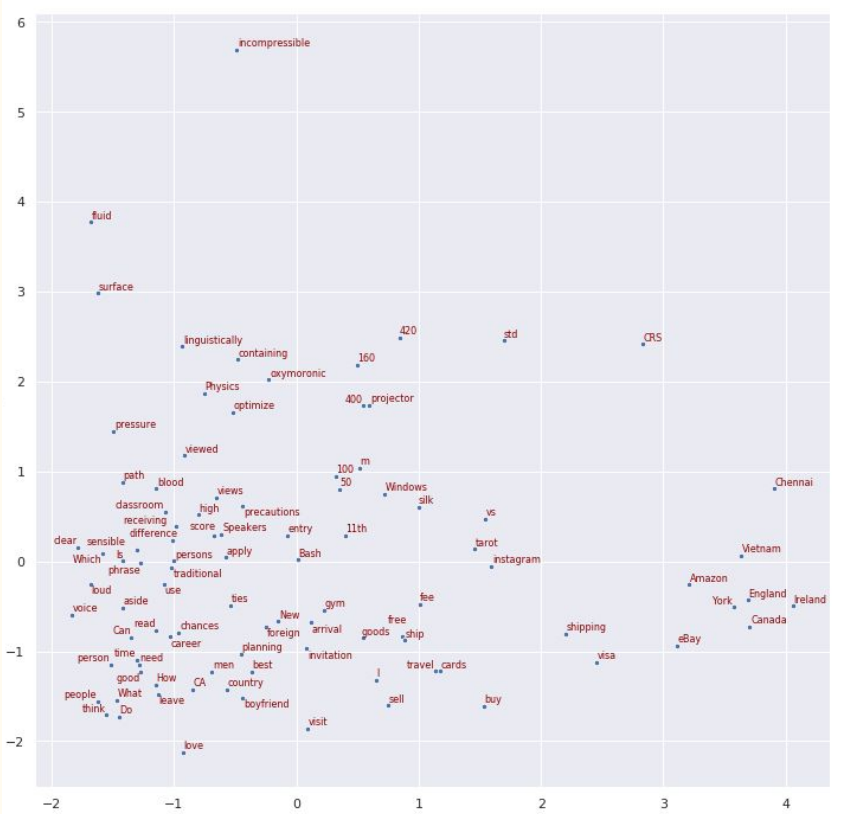
PREPROCESSING

Text data typically requires some cleanup before it can be processed further and fed to a model. The question data was cleaned as follows,

- Tag Removal - "Hello World!" was converted to "Hello World!"
- Repeating Whitespace (spaces, tabs, line breaks) Removal
- Stopwords Removal (stopwords include the most commonly occuring words in a language like ‘the’, ‘on’, ‘is’ etc.)
- Conversion to lowercase and stemming (stemming reduces inflections like ‘fishing’, ‘fished’ and ‘fisher’ to the root ‘fish’)

EMBEDDING - TEXT TO NUMBERS

Embedding converts words in a text document to a set of numbers (called word-vectors). Often, embeddings that are pre-trained on large text datasets like Wikipedia and Common Crawl are used successfully to solve NLP problems. This project uses two such embeddings (from Fasttext and Spacy*)

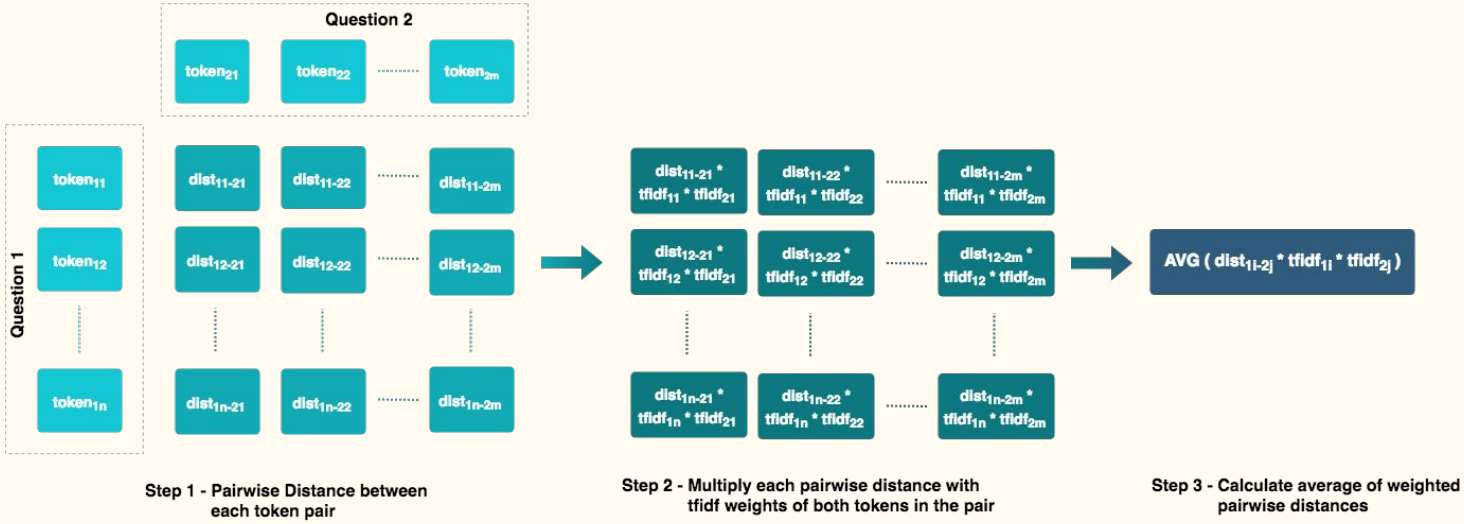


Left : The plot shows Spacy word-vectors for words from 20 questions from the Quora dataset.

*Spacy and Fasttext are utility libraries for performing a wide range of NLP tasks. They also provide pre-trained vector embeddings

FEATURE EXTRACTION

Feature Set 1 - Similarity was computed using word-to-word (pairwise) distance which was weighted with TF-IDF* weights and then averaged over all word-pairs across the two questions in each question-pair (Question 1 & Question 2 below).

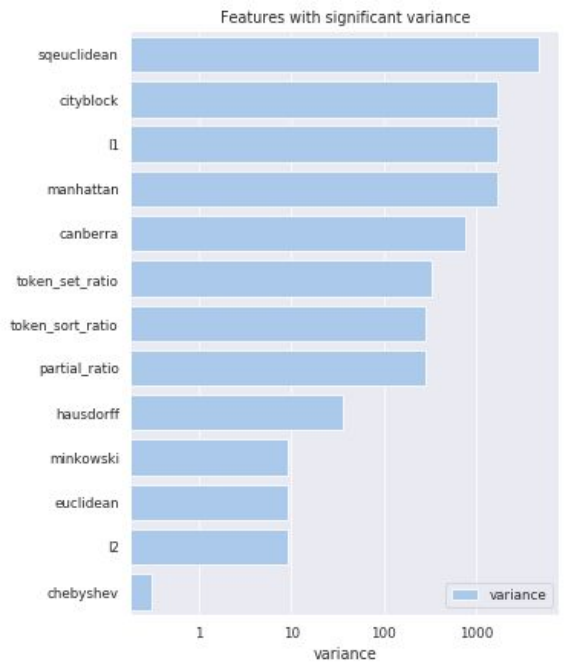


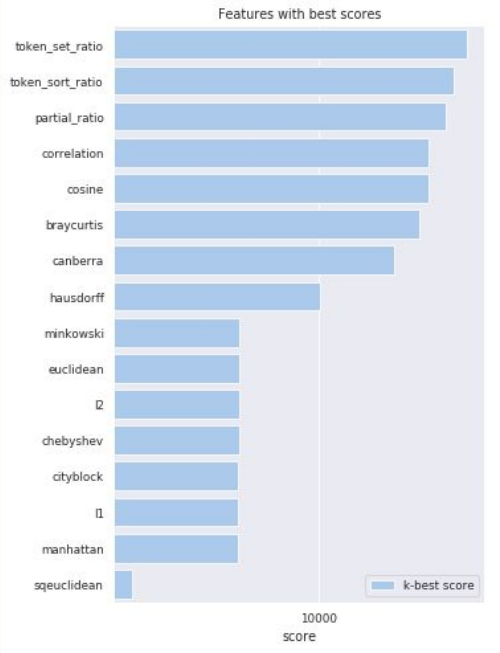
*TF-IDF or Term Frequency - Inverse Document Frequency is a weighting statistic that increases proportionally with the number of times a word appears in a document and is offset by the number of documents in the corpus that contain the word

FEATURE EXTRACTION

Feature Set 2 - Similarity metrics were computed using Levenshtein* (edit) distance.

Feature Set 3 - Each question was converted into a sequence of word-vectors using pre-trained embeddings which formed the features in set 3. These features contain substantial information about the question pairs but are not as interpretable as features in sets 1 & 2.



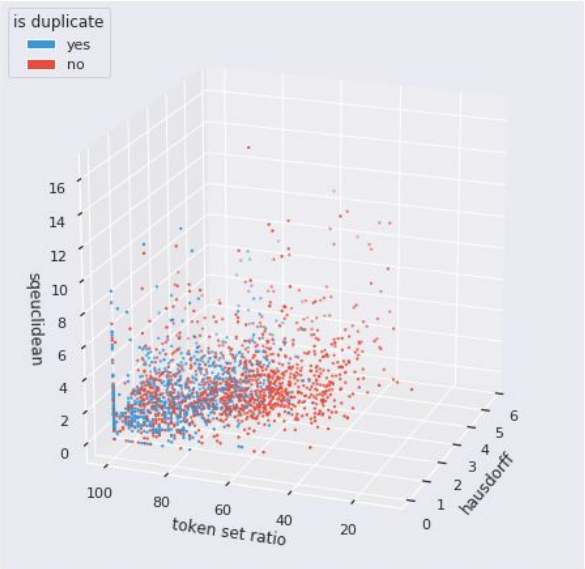


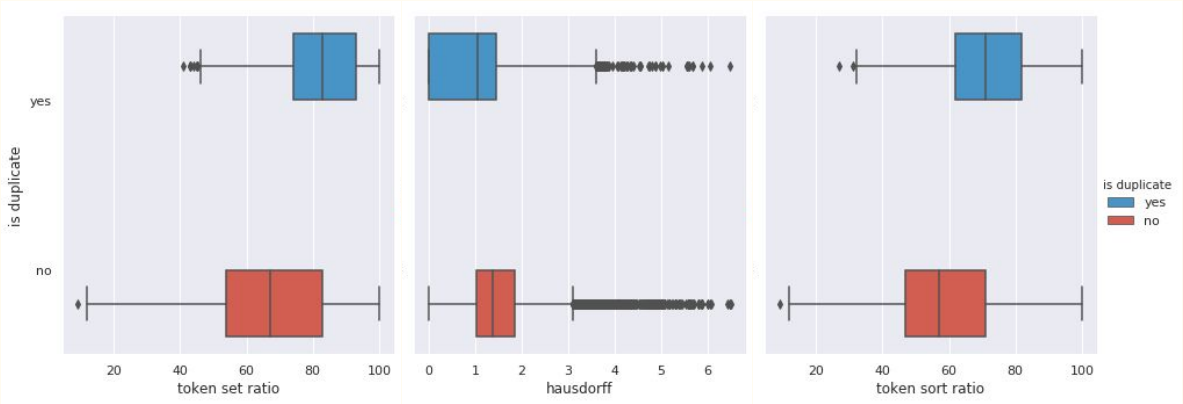
Right : Feature Importance plots rank the best features (for feature-sets 1 & 2) based on variance and F-value score

*This is a method for measuring the difference between two text sequences based on the number of single character edits (insertions, deletions and substitutions) it takes to change one sequence to another.

FEATURE COMPARISON

The plots below give a qualitative idea about how hand-engineered features (sets 1 & 2) might contribute to the classification. It is obvious from these plots that a few hand-engineered features are not sufficient, and it is necessary to include feature-set 3 (embedding vectors)





Above : Box-and-whisker plots show the 3 quantiles and extreme values for the levenshtein features token-sort-ratio and token-set-ratio and pairwise distance feature hausdorff

Left : A 3-D scatter plot shows how duplicate(blue) and non-duplicate(red) question-pairs vary with square euclidean distance, token sort ratio and hausdorff distance.

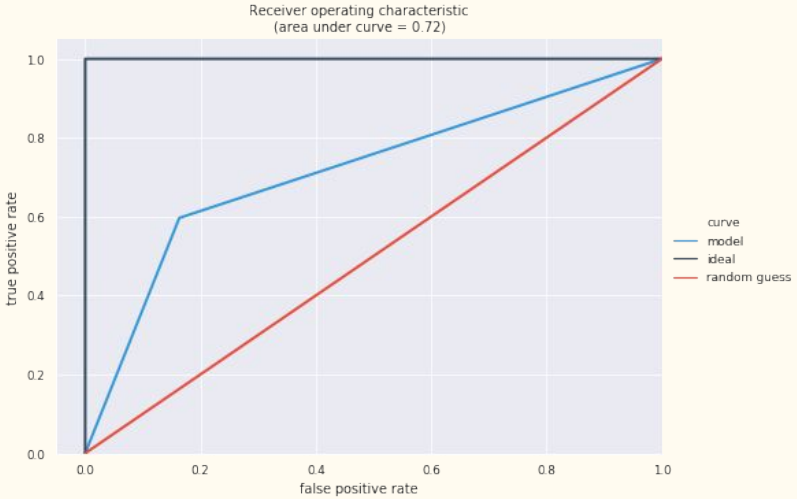
CLASSIFICATION - LOGISTIC REGRESSION

This is a linear model for classification, where the probabilities are modeled with a logistic function*. Overfitting** was countered with L2 regularization and 5-fold cross-validation*** was used to tune the regularization factor and other model parameters.

	precision	recall	f1-score
not duplicate	0.79	0.83	0.81
duplicate	0.68	0.61	0.65

Above : Model accuracy metrics computed on the test data

Right : Receiver Operating Characteristic Curve



*The logistic function is a sigmoid function, which takes any real input and outputs a value between 0 and 1

**When a model learns the training data too closely, it fails to fit new data or predict unseen observations reliably

***To counter overfitting, and find optimum parameters (like regularization factor), a validation set is held-out from the training data (for each fold or run). The model is then trained on the remaining training data and evaluated on the validation set. The best evaluation run determines the optimum parameters.

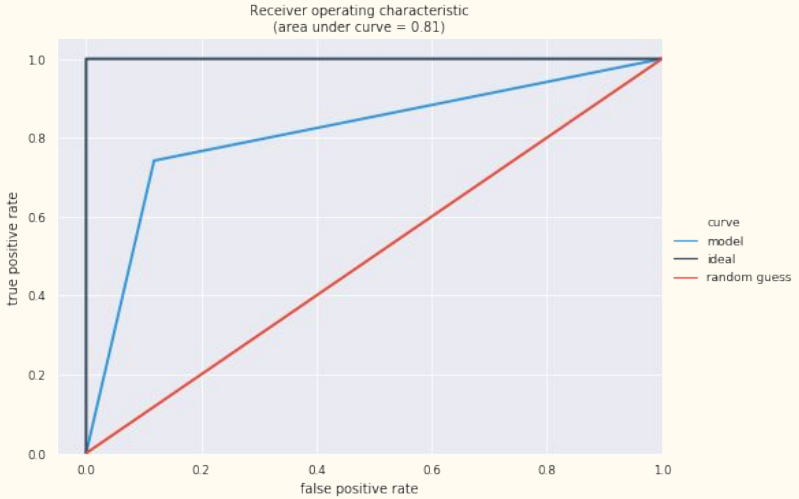
CLASSIFICATION - GRADIENT BOOSTING

XGBoost, extreme gradient boosting, finds an optimum ensemble (combination) of several decision trees*. Overfitting is countered by limiting the depth of the individual trees and by applying L2 regularization to the leaf-weights of individual trees. 5-fold cross-validation was used to tune the regularization factor, tree-depth, number of trees and other model parameters.

	precision	recall	f1-score
not duplicate	0.87	0.88	0.87
duplicate	0.79	0.76	0.78

Above : Model accuracy metrics computed on the test data

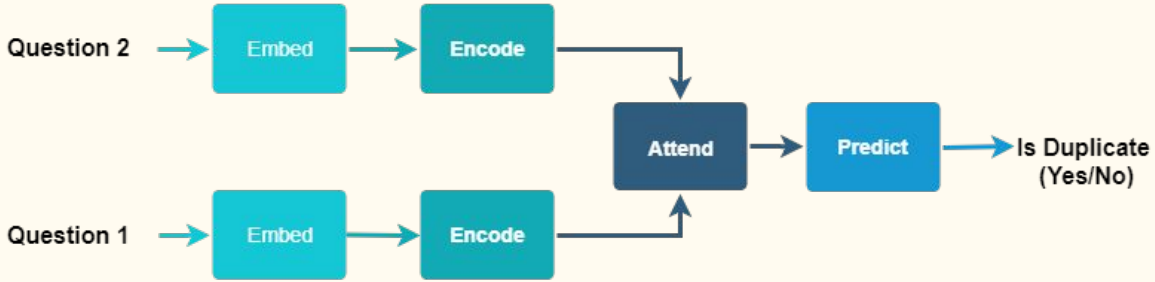
Right : Receiver Operating Characteristic Curve



*gradient boosting computes an objective function (loss) every time it adds a tree to the ensemble and finds the best tree structure by minimizing this objective function sequentially for every split in the tree.

CLASSIFICATION - NEURAL NETWORK 1

This ‘Decomposable Attention’ model uses attention* mechanism. A crucial advantage is provided by the fact that sentence-to-vector reduction operates over the pair of sentences jointly, as opposed to encoding the sentences into vectors independently. This captures the relationship between the two questions in a pair.



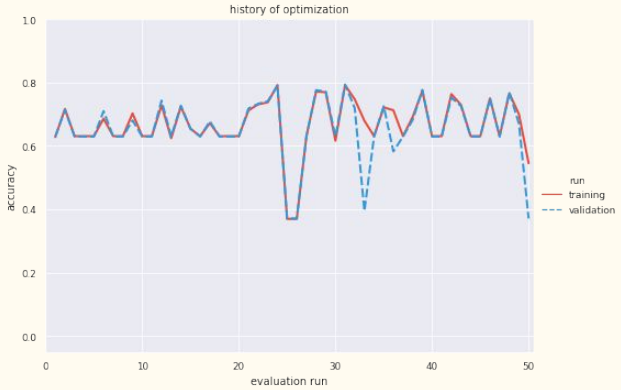
Above : High level components of the neural network. The encode layer performs soft alignment** with tokens in the other question to compute attention weights for the subsequent attention layer

*attention here literally means ‘to pay attention’ to words (one-at-a-time) in the second question while encoding each word in the first question. The encoding is done by computing dot products of the word-vectors of first question with the vectors of the second one.

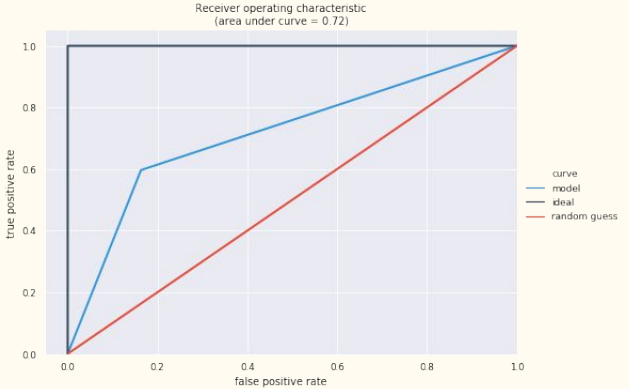
**the encoded weights (normalized dot-products) represent the sub-phrases in each question that are ‘aligned’ to every word in the other question.

CLASSIFICATION - NEURAL NETWORK 1

Overfitting was countered with dropout regularization* and early stopping**. The model parameters were optimized using a sequential optimization technique called Tree Structured Parzen Estimator (TPE)***.



	precision	recall	f1-score
not duplicate	0.85	0.82	0.84
duplicate	0.72	0.76	0.74



Left : The variation of training and validation accuracy during the hyperparameter search (50 evaluations)

Middle: Model accuracy metrics computed on the test data

Right : Receiver Operating Characteristic Curve

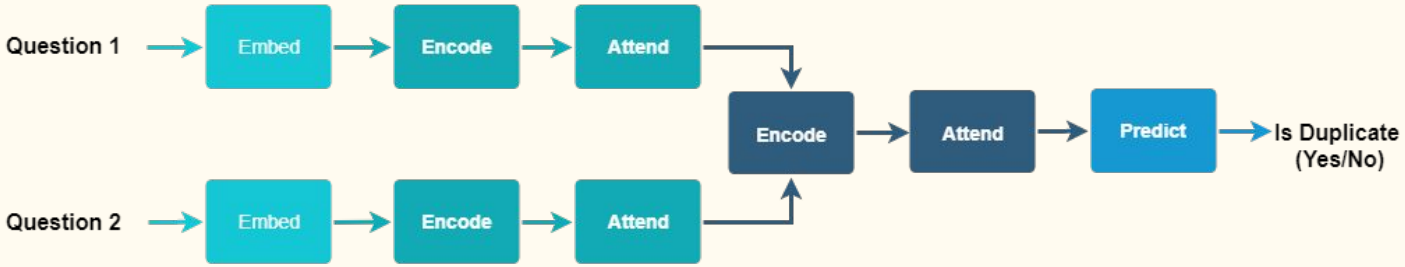
*Dropout tackles overfitting by randomly 'dropping-out' or ignoring neurons during training.

**Early stopping is another strategy to prevent overfitting, that stops training when a given validation metric stops improving (therefore stopping when the model begins to overfit)

***TPE is used for efficiently searching in high-dimensional spaces. This is a Sequential Model Based Optimization (SMBO) method, which builds and evaluates models sequentially, based on historic values of hyperparameters

CLASSIFICATION - NEURAL NETWORK 2

This ‘Hierarchical Attention’ model uses attention to reduce an encoded matrix (representing a sequence of words) to a vector. It does this by learning context vectors for the two attention steps (one for words and the other for sentences)*

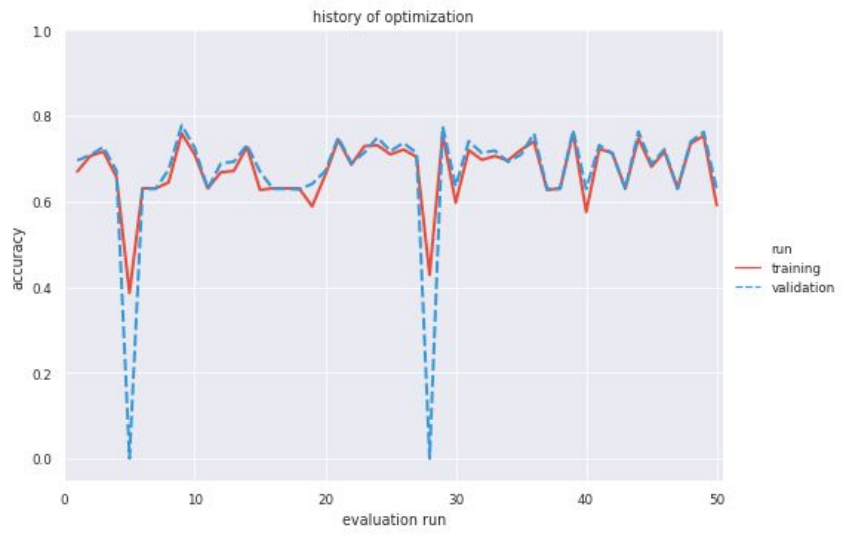


Above : High level components of the neural network. Unlike the previous NN, this model encodes and applies attention to each question individually before encoding and applying attention to the combined question-pair.

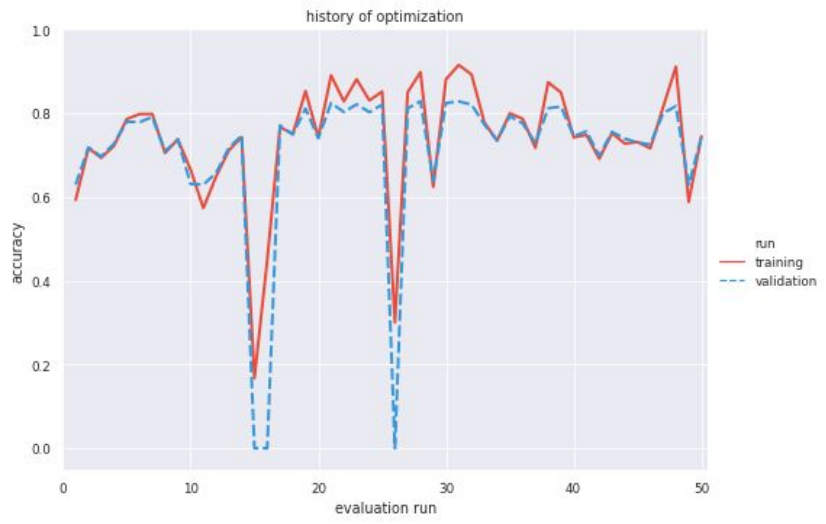
*Applying the word and sentence context-vector analogy to a generic NLP problem, the attention steps can be seen as an effective feature extraction process, which in the case of duplicate question identification, can be employed to learn similarity between words and sentences belonging to a pair of questions.

CLASSIFICATION - NEURAL NETWORK 2

Overfitting was countered with dropout regularization and early stopping. The model parameters were optimized using TPE. Two optimization searches were performed, a quicker coarse-grained run and a more fine-grained run, building on the results of the first one.



Left : The variation of training and validation accuracy during the first hyperparameter search (50 evaluations)



Right : The variation of training and validation accuracy during the second hyperparameter search (50 evaluations)

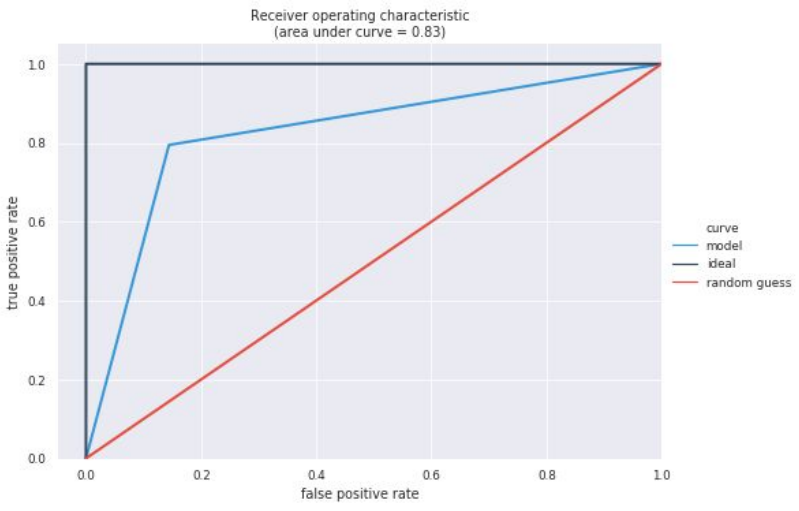
CLASSIFICATION - NEURAL NETWORK 2

The performance of this model is better than the previous NN method and at par with the gradient boosted machine.

	precision	recall	f1-score
not duplicate	0.88	0.86	0.87
duplicate	0.76	0.79	0.78

Above : Model accuracy metrics computed on the test data

Right : Receiver Operating Characteristic Curve



CONCLUSION & FURTHER EXPLORATION

The model evaluation results indicate that gradient boosted trees and neural networks (with Hierarchical Attention) are effective ways to solve the problem of duplicate identification.

Further exploration could include exploring alternative neural network architectures and experimenting with modification of layers and further tuning of the existing models.

REFERENCES

NLP

1. On word embeddings - Part 1 by Sebastian Ruder
2. GloVe - On word embeddings - Part 3 by Sebastian Ruder
3. Spacy
4. Gensim
5. Fasttext
6. FuzzyWuzzy

Modeling

1. Logistic Regression
2. XGBoost
3. Hyper-parameter Tuning
4. Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models, by Matthew Honnibal
5. Hierarchical Attention Networks for Document Classification by Yang et al.
6. A Decomposable Attention Model for Natural Language Inference by Parikh et al.

REFERENCES

Deep Learning

1. Keras
2. TensorFlow
3. Hyperas
4. Hyperopt tutorial for Optimizing Neural Networks' Hyperparameters
5. Hyperopt