DUPLICATION DETECTION IN QUESTION PAIRS

-Deepak Munagala (2014035)
-Purusharth Dwivedi (2014081)
-Avneet Kaur (2014027)

PROBLEM STATEMENT:
Given a pair of questions, the aim is to tell, whether they are semantically equivalent or not.

DATASET: Question pair dataset from Quora
Nature Of Data:

	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

METHODOLOGY:

Various textual features were extracted from the training data and fed into an XGBoost model for training.

FEATURE EXTRACTION:
The following measures were used:
→ **Semantic Similarity using WordNet corpus statistics**
→ **Tf-Idf** : Tf-Idf representations of both the questions were calculated and cosine similarity was used as a measure.
→ **Bag-Of-Words :**

→ **Word Overlap**: The number of shared words between both the questions normalised by the total number of the words in both questions.

Unigram, Bigram, Trigram common ratios were calculated by finding the common unigrams, bigrams, and trigrams between a pair of questions

WordToVec was used to get the vector representations of both the questions and following measures were calculated

→ **WORD MOVER'S DISTANCE**:

This distance between the two documents is the minimum cumulative distance that all words in document 1 need to travel to exactly match document 2.given 2 sentences w1= a b c, and w2= d e f. WMD=Find min distance from *a* to *d,e,f*. Do same for *b* and *c.* Take sum of all these min distances.

→ **COSINE SIMILARITY**:

Dot product of the two vectors divided by the product of the magnitude. (-1 to 1)

→ **CITYBLOCK DISTANCE**:

Difference between the two vectors.(~manhattan distance)

→ **EUCLIDEAN DISTANCE**:

sum of squares of the individual differences between the elements of the vector.

→ **CANBERRA DISTANCE**:

Difference of the vectors, normalised by the sum of length of the vectors.

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|}$$

→ **MINKOWSKI DISTANCE** - p = 3

$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

→ **BRAY CURTIS DISSIMILARITY**: close to 0 - similar, close to 1, dissimilar

$$d_{ij} = \frac{\sum_{k=1}^{n} \left|x_{ik} - x_{jk}\right|}{\sum_{k=1}^{n} \left(x_{ik} + x_{jk}\right)}$$

x_i x_j are two vectors

→ **Skew Of Question**
→ **Kurtosis**

→ **Fuzzy Features:**

Q-Ratio
W-Ratio
Partial Ratio
Partial Token Set Ratio
First Token Set Ratio

Semantic similarity using WordNet and corpus statistics:

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

This method derives text similarity from **semantic** and **syntactic** information of both the sentences. This method forms a joint word set only using all the distinct words in the pair of sentences, dynamically. For each sentence, the following is computed:

→ A raw semantic vector

→ A word order vector

Both of these are derived using the lexical database, in this case WordNet.
Semantic similarity is computed based on the two semantic vectors. An order similarity is calculated using the two order vectors. Finally, the sentence similarity is derived by combining semantic similarity and order similarity.

Now let us see, how each of the above mentioned, works.
In order to compute the similarity between the sentences, we first need to compute the similarity between a pair of words.
This is done as follows:

→ **Word Similarity**:

Words are organized into synonym sets (synsets) in the knowledge base - WordNet. Therefore, we can take the minimum length of path connecting the two words in the hierarchical semantic network. For eg, the shortest path between boy and girl is through person which is the subsumer(nearest word that absorbs both the words).

$s(w1,w2) = f(L)f(D)$

L - shortest path length

D - depth of hierarchical tree

→ **Sentence Similarity**:

Given two sentences T1 and T2, a joint set T = T1 union T2 is computed and a semantic vector is derived from it. Each entry of the semantic vector corresponds to a word in the joint word set. Value is determined by the semantic similarity.

The similarity b/w two sentences is the cosine similarity only.

S = s1.s2 / ||s1|| ||s2||

→ **Word Order Similarity:**

Consider 2 sentences as follows:

A quick brown fox jumped over the lazy dog

A quick brown dog jumped over the lazy fox

Now these two sentences are different but any bag-of words method would give the decision that they are the same. Therefore, the dissimilarity is based on word order. So this is taken into account by the following measure:

Each word in the sentence is given an order , final representations are:

r1 : {1,2,3,4,5,6,7,8,9}

r2: {1,2,3,9,5,6,7,8,4}

1 - (||r1 - r2||) / (||r1 + r2||)

→ **Overall similarity**:

S(T1,T2) = d * sentence_similarity + (1-d)word_order_similarity

Training Of Model:

All these features were fed as different columns to an **XGBoost Model** and gave the following results.

Results:

A Log-Loss of 0.3047 was obtained.

We got a log-loss of 0.304, when we used only the mean, we got a log loss of 0.55, and when we used a combination of wmd and wordnet features, only we got a log loss of 0.356. With a combination of all the above we were able to minimise it to 0.30