Quora Question Pairs

Sukanya Behera [160714733008]
Chandrabhatta Sriram [160713733042]
Abul Faiz Mohammed Abdul Hadi Mouzzam [160714733108]

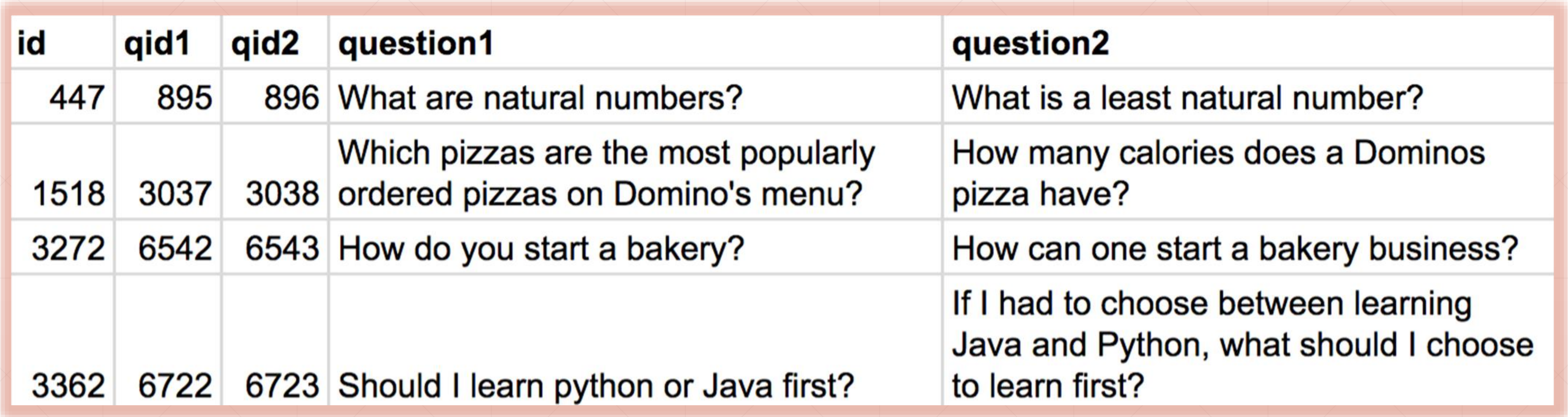
Project Guide: E. Shailaja [Asst. Prof. Dept. of CSE]
Project Co-ordinator: R. Sandeep [Asst. Prof. Dept. of CSE]

Methodist College of Engineering & Technology

Introduction [Problem Definition]

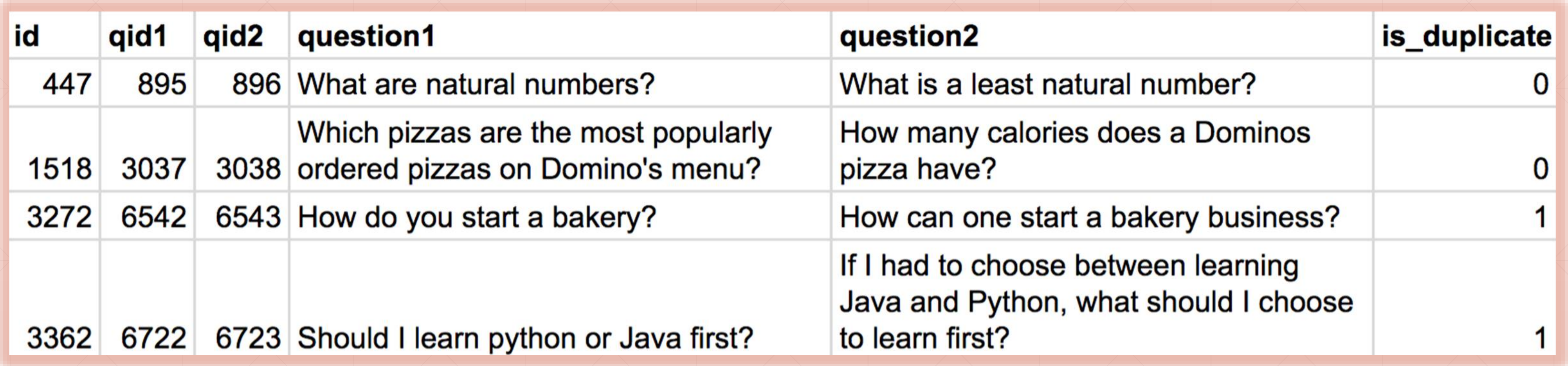
- More formally, the duplicate question detection problem can be defined as follows: given a pair of questions *q1* and *q2,* train a model that learns the function:
  - $f(q1, q2) \rightarrow 0 \text{ or } 1.$
  - where *1* represents that *q1* and *q2* have the same intent and *0* otherwise.
  - NOTE: *q1* and *q2* are given as string data.

Problem Input [test.csv]



id	qid1	qid2	question1	question2
447	895	896	What are natural numbers?	What is a least natural number?
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?

Problem Output [Expected]

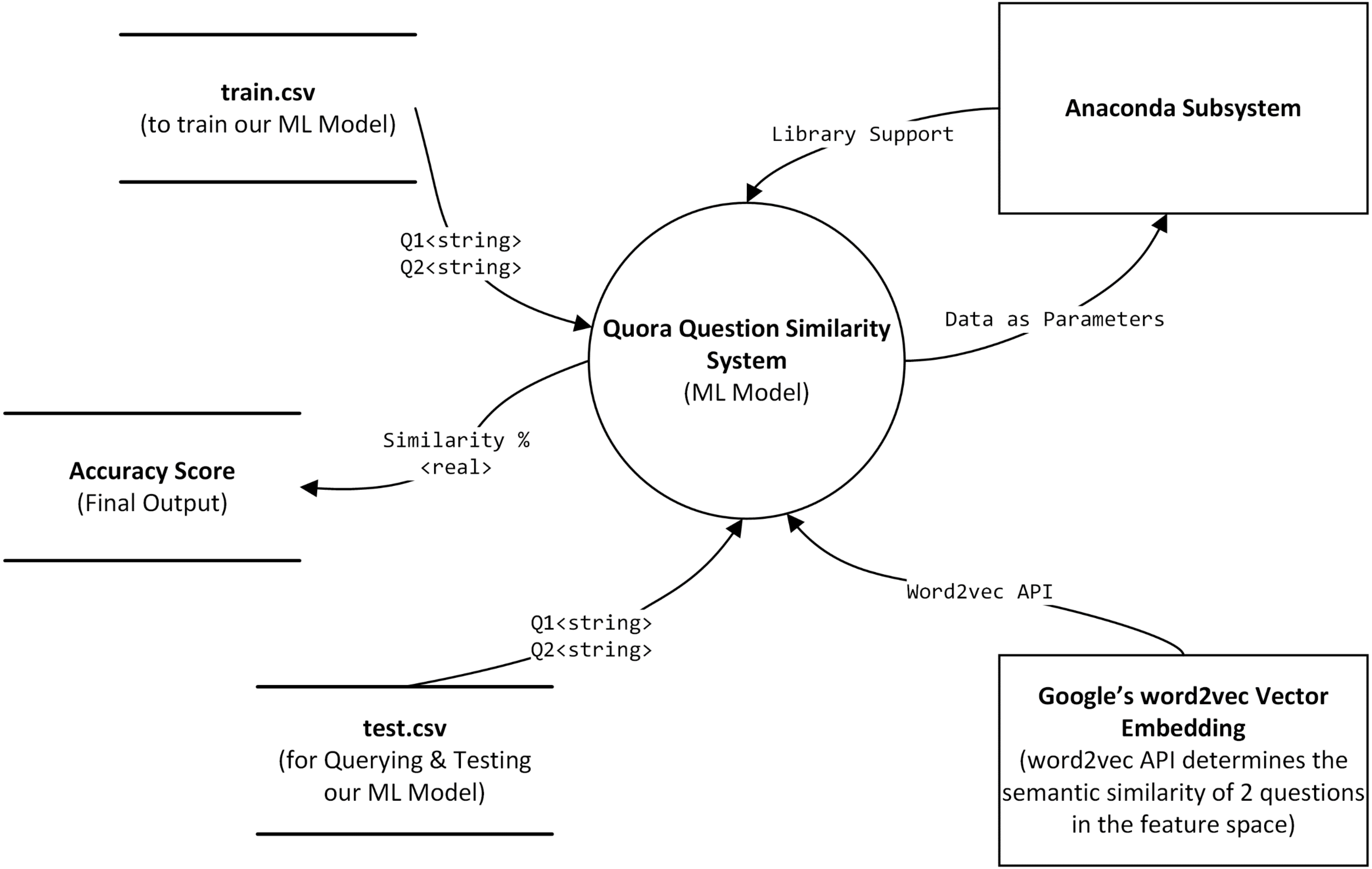


id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

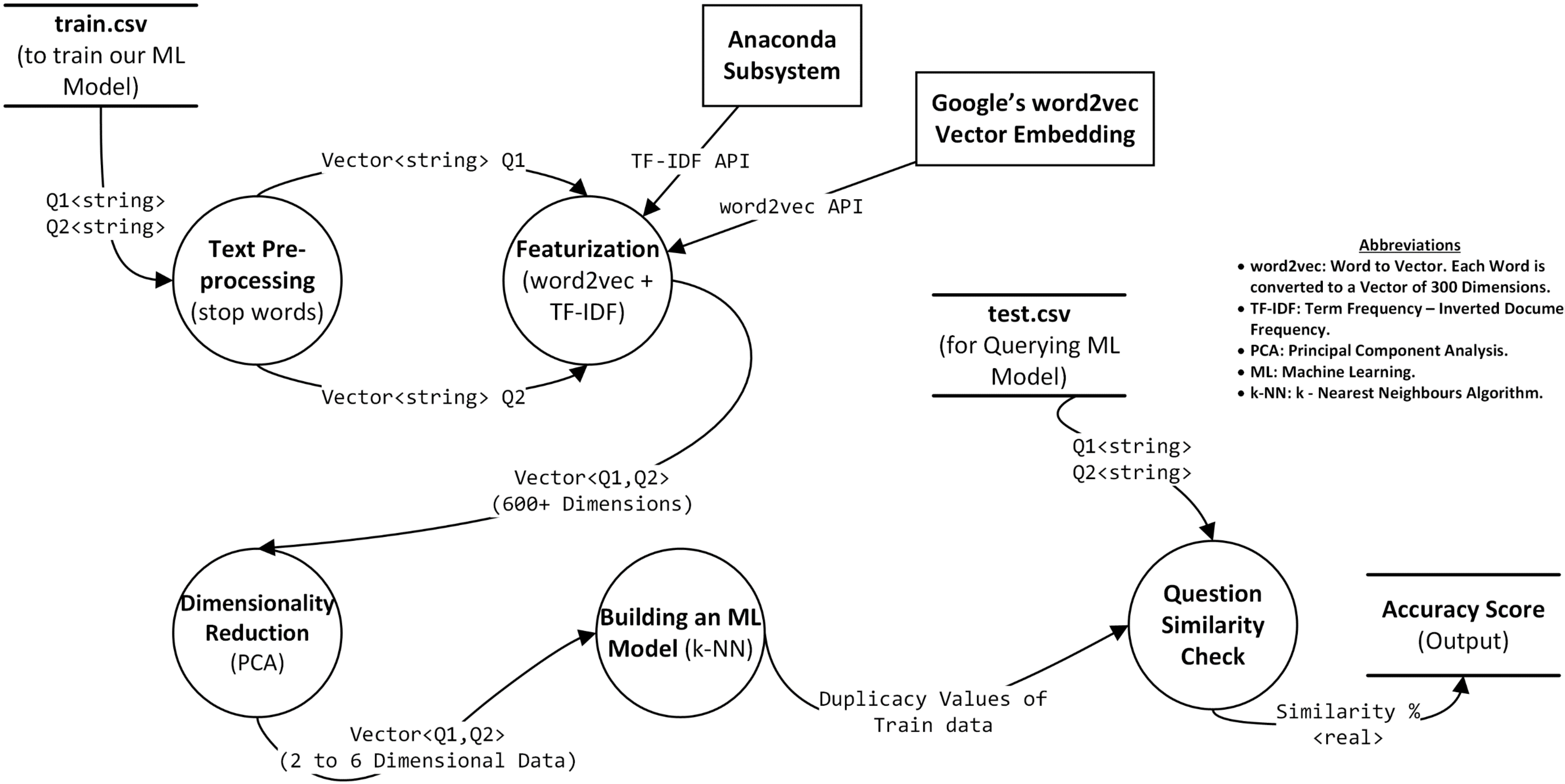
Modules

- Text Pre-processing: *text tokenization*, *uniform text casing*, *stop word removal*, *lemmatization*, *stemming*, etc.
- Featurization / Vectorization of the texts [*q1* and *q2*]: Convert the given *q1* and *q2* into Word Vectors using word2vec. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.
- Apply a Machine Learning Model [using k – Nearest Neighbours algorithm] to get the Expected Output.

Data Flow Diagram – Level 0


train.csv
(to train our ML Model)
Q1<string>
Q2<string>
Library Support
Anaconda Subsystem
Quora Question Similarity System
(ML Model)
Data as Parameters
Similarity %
<real>
Accuracy Score
(Final Output)
Q1<string>
Q2<string>
Word2vec API
test.csv
(for Querying & Testing our ML Model)
Google’s word2vec Vector Embedding
(word2vec API determines the semantic similarity of 2 questions in the feature space)

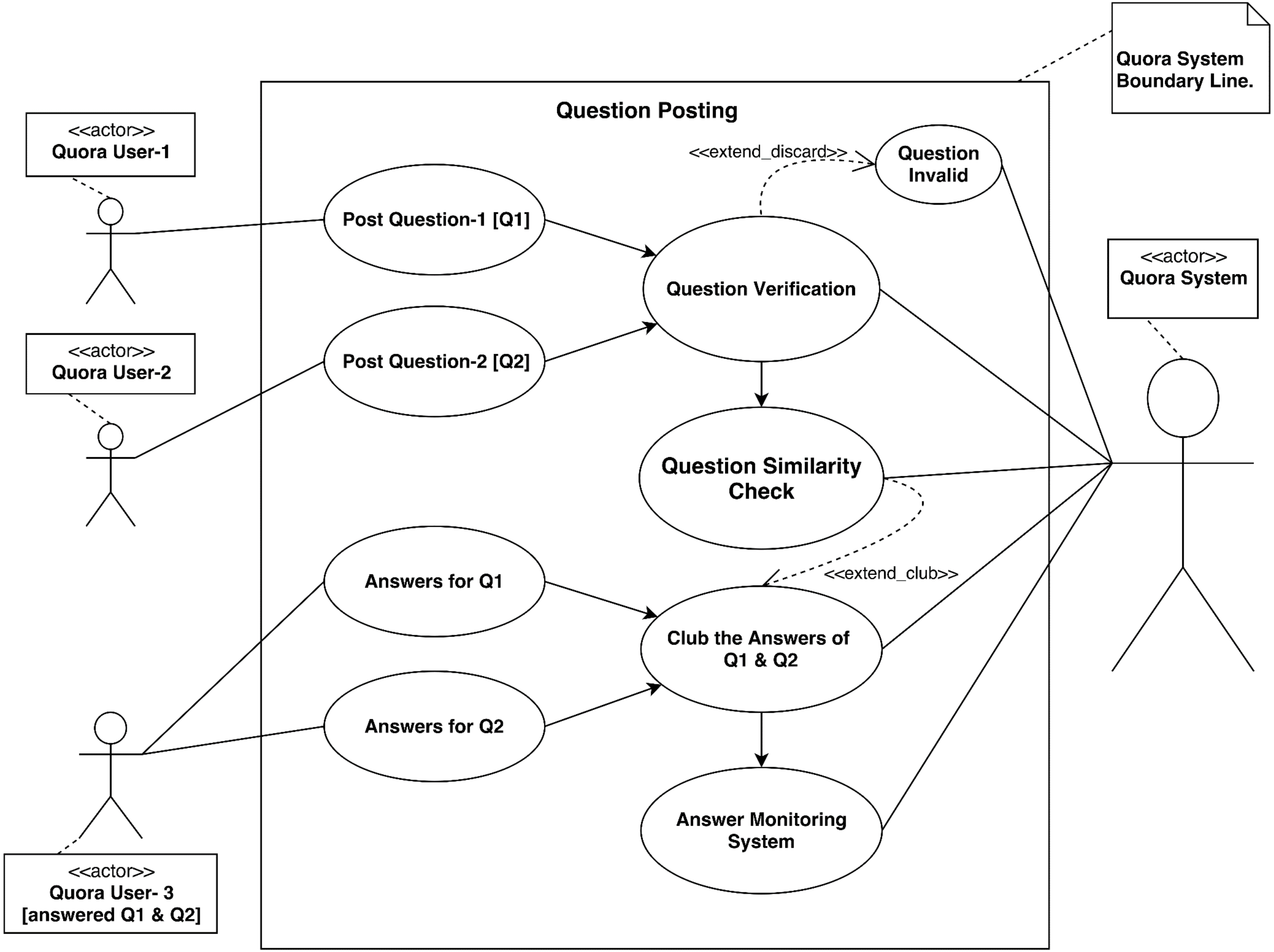
Data Flow Diagram – Level 1
Quora Question Similarity System



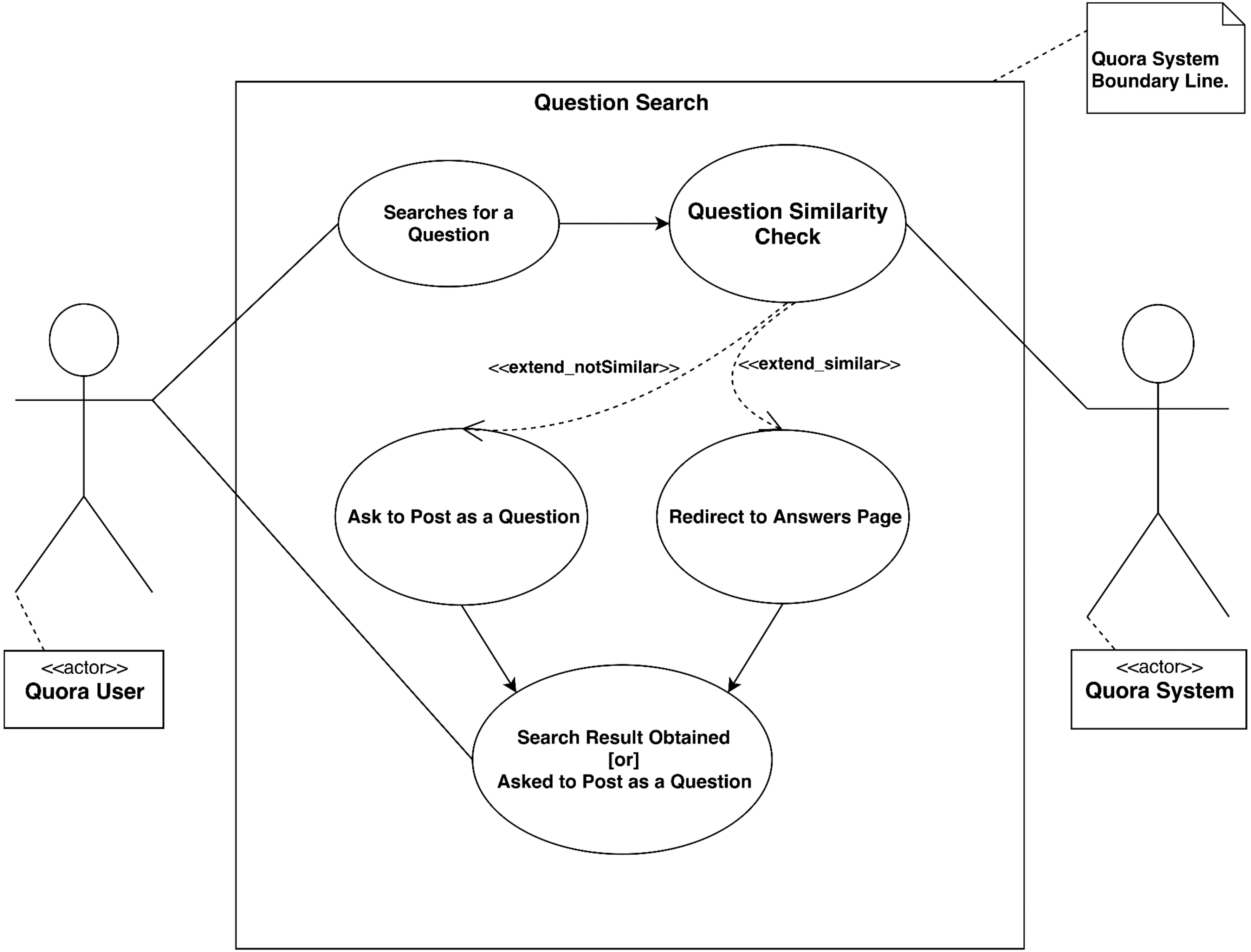
Abbreviations

- word2vec: Word to Vector. Each Word is converted to a Vector of 300 Dimensions.
- TF-IDF: Term Frequency – Inverted Document Frequency.
- PCA: Principal Component Analysis.
- ML: Machine Learning.
- k-NN: k - Nearest Neighbours Algorithm.

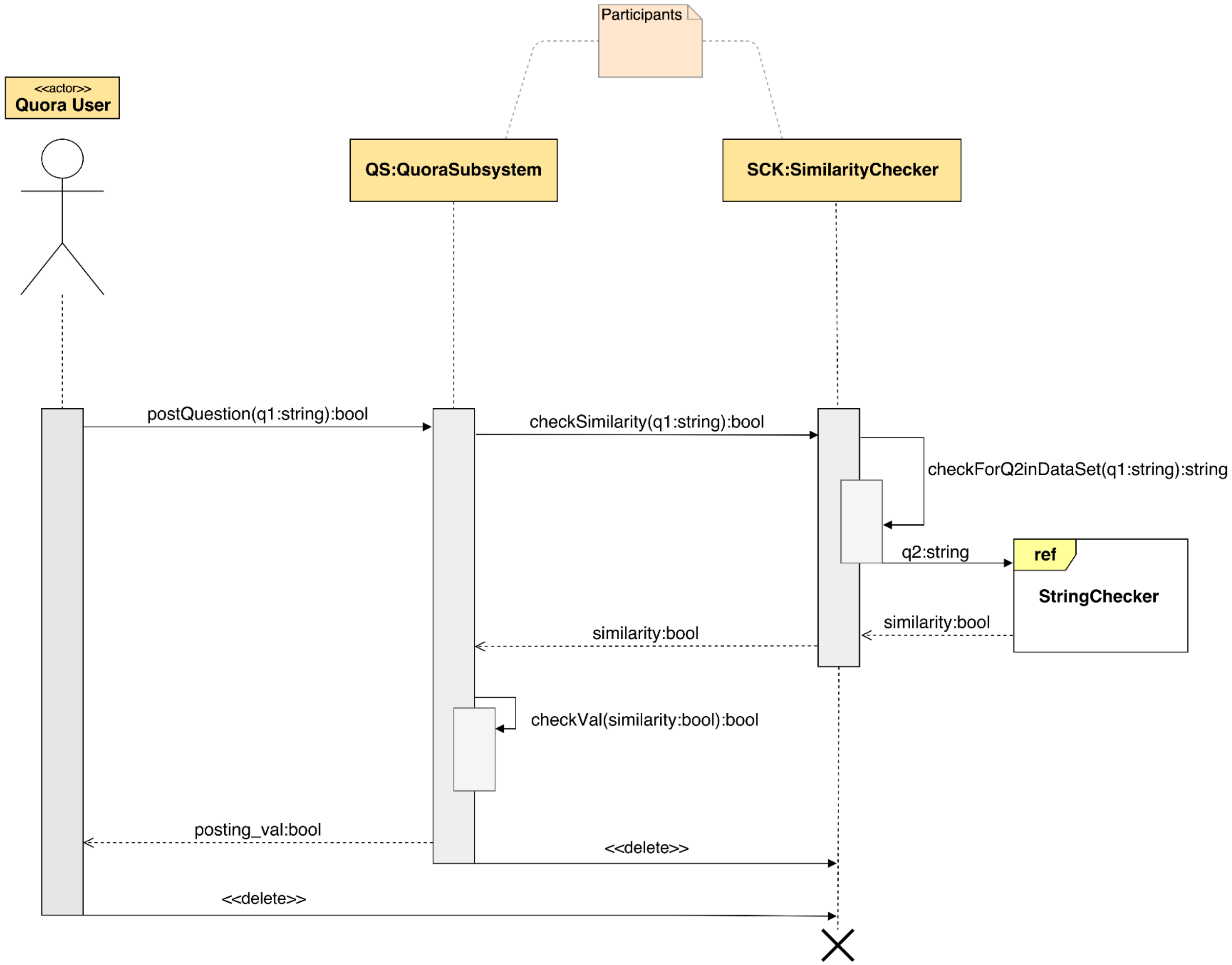
Use Case Diagram – 1 [Question Posting]

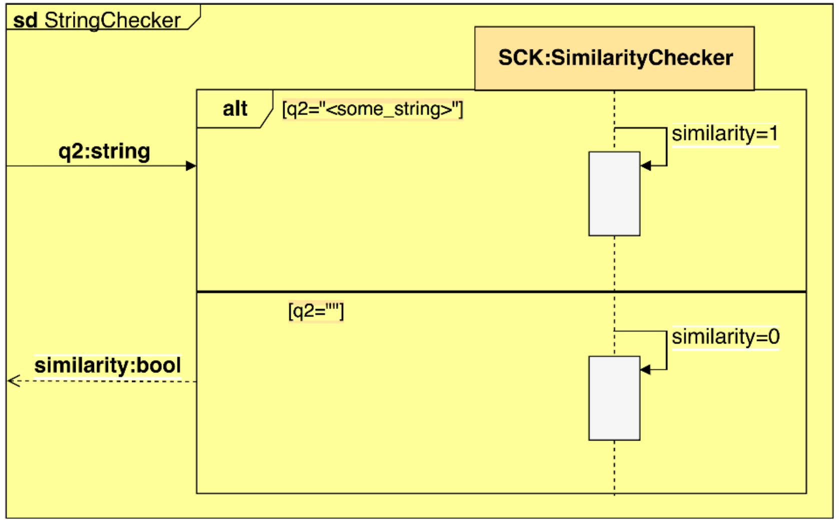

Quora System Boundary Line.
Question Posting
<<actor>>
Quora User-1
<<actor>>
Quora User-2
<<actor>>
Quora User- 3
[answered Q1 & Q2]
<<actor>>
Quora System
Post Question-1 [Q1]
Post Question-2 [Q2]
<<extend_discard>>
Question Invalid
Question Verification
Question Similarity Check
<<extend_club>>
Answers for Q1
Answers for Q2
Club the Answers of Q1 & Q2
Answer Monitoring System

Use Case Diagram – 2 [Question Search]

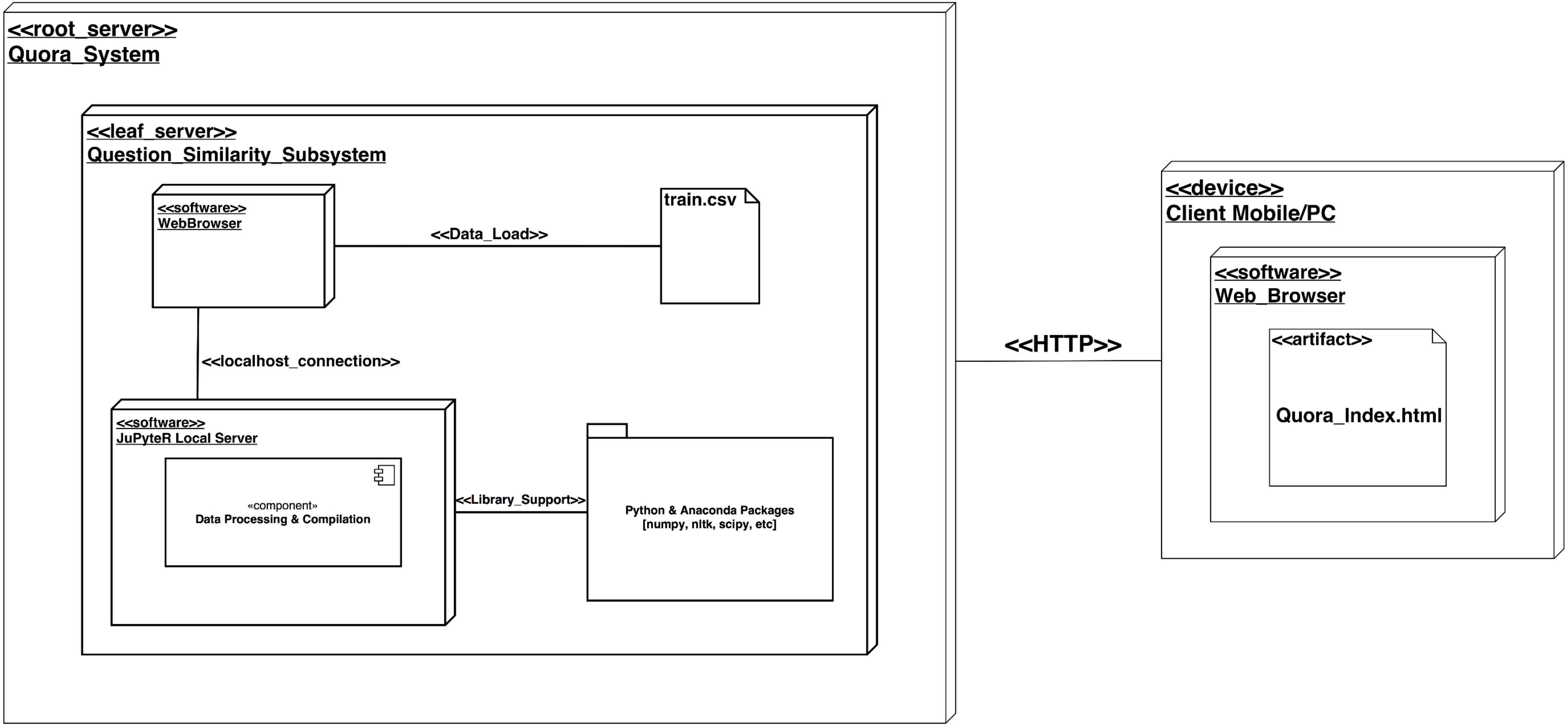

Quora System
Boundary Line.
Question Search
Searches for a Question
Question Similarity Check
<<extend_notSimilar>>
<<extend_similar>>
Ask to Post as a Question
Redirect to Answers Page
Search Result Obtained
[or]
Asked to Post as a Question
<<actor>>
Quora User
<<actor>>
Quora System

Sequence Diagram


Participants
<<actor>>
Quora User
QS:QuoraSubsystem
SCK:SimilarityChecker
postQuestion(q1:string):bool
checkSimilarity(q1:string):bool
checkForQ2inDataSet(q1:string):string
q2:string
ref
StringChecker
similarity:bool
similarity:bool
checkVal(similarity:bool):bool
posting_val:bool
<<delete>>
<<delete>>


sd StringChecker
SCK:SimilarityChecker
alt
[q2="<some_string>"]
similarity=1
q2:string
[q2=""]
similarity=0
similarity:bool

Deployment Diagram

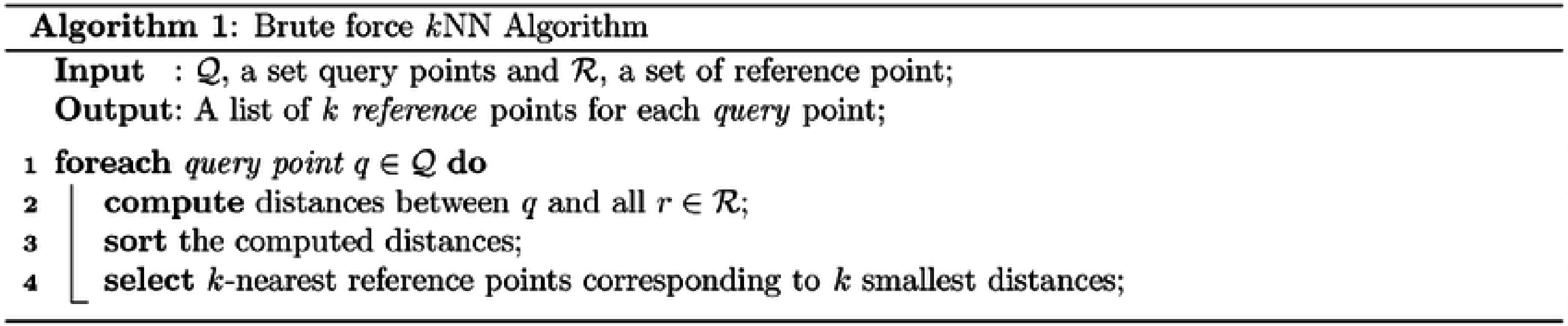

<<root_server>>
Quora_System
<<leaf_server>>
Question_Similarity_Subsystem
<<software>>
WebBrowser
<<Data_Load>>
train.csv
<<localhost_connection>>
<<software>>
JuPyteR Local Server
«component»
Data Processing & Compilation
<<Library_Support>>
Python & Anaconda Packages
[numpy, nltk, scipy, etc]
<<HTTP>>
<<device>>
Client Mobile/PC
<<software>>
Web_Browser
<<artifact>>
Quora_Index.html

Algorithm Information

- In ***k*-NN** classification, the output is a class membership. An object is classified by a majority vote of its neighbours, with the object being assigned to the class most common among its ***k*** nearest neighbours (***k*** is a positive integer, typically small). If ***k*** = *1*, then the object is simply assigned to the class of that single nearest neighbour.

- Distance between 2 points in the feature space, can be found by taking Euclidean Distance (L^2 Norm) between those 2 points.

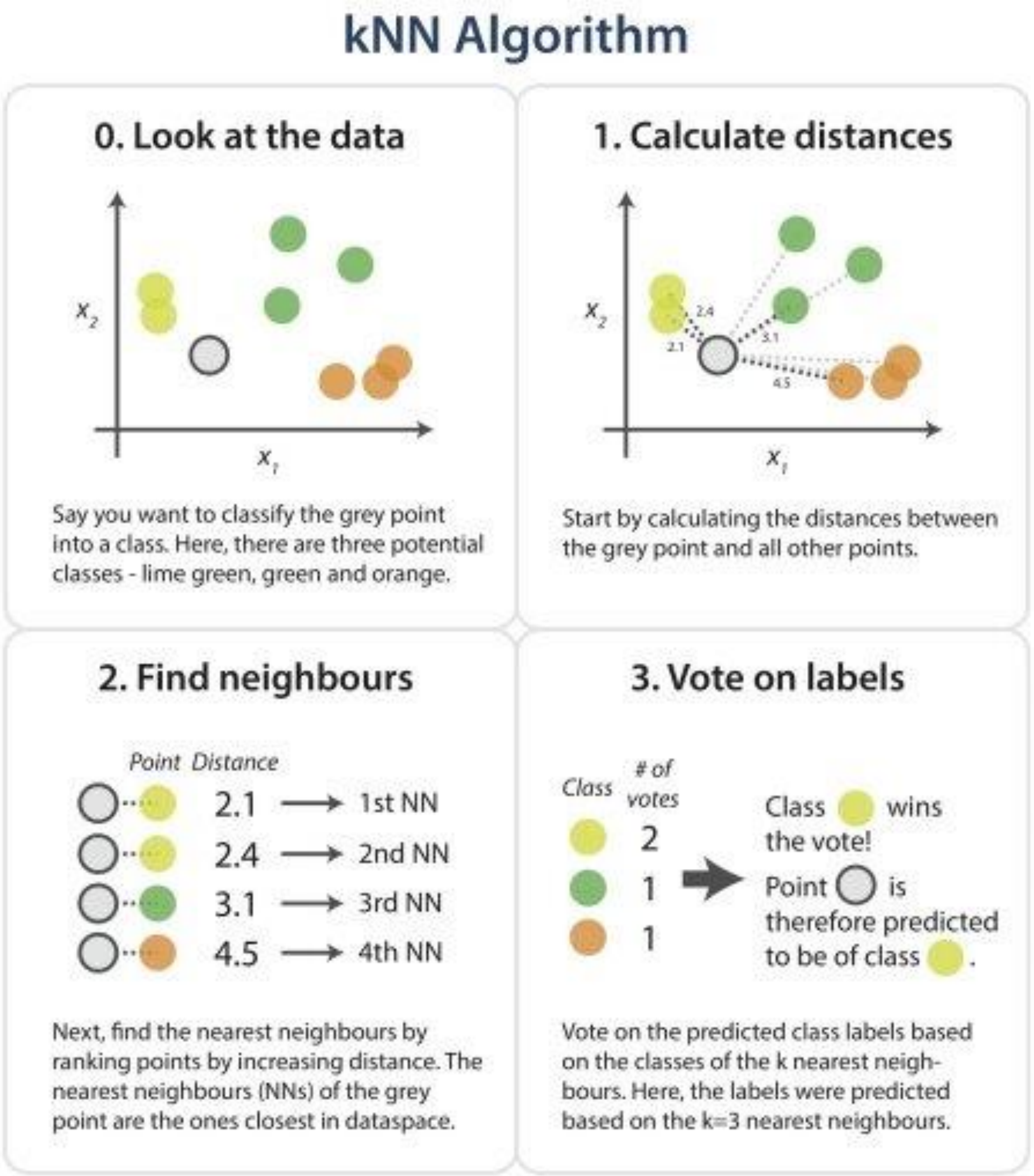
k-NN Brute Force Pseudo Code



Algorithm 1: Brute force kNN Algorithm

```
Input  : $\mathcal{Q}$, a set query points and $\mathcal{R}$, a set of reference point;
Output: A list of $k$ reference points for each query point;
foreach query point $q \in \mathcal{Q}$ do
    compute distances between $q$ and all $r \in \mathcal{R}$;
    sort the computed distances;
    select $k$-nearest reference points corresponding to $k$ smallest distances;
```

Complexity: **O (mnd), which makes it a Polynomial Time Algorithm.**



kNN Algorithm

0. Look at the data

Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances

Start by calculating the distances between the grey point and all other points.

2. Find neighbours

Point	Distance	
○··●	2.1	⟶ 1st NN
○··●	2.4	⟶ 2nd NN
○··●	3.1	⟶ 3rd NN
○··●	4.5	⟶ 4th NN

Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels

Class	# of votes
●	2
●	1
●	1

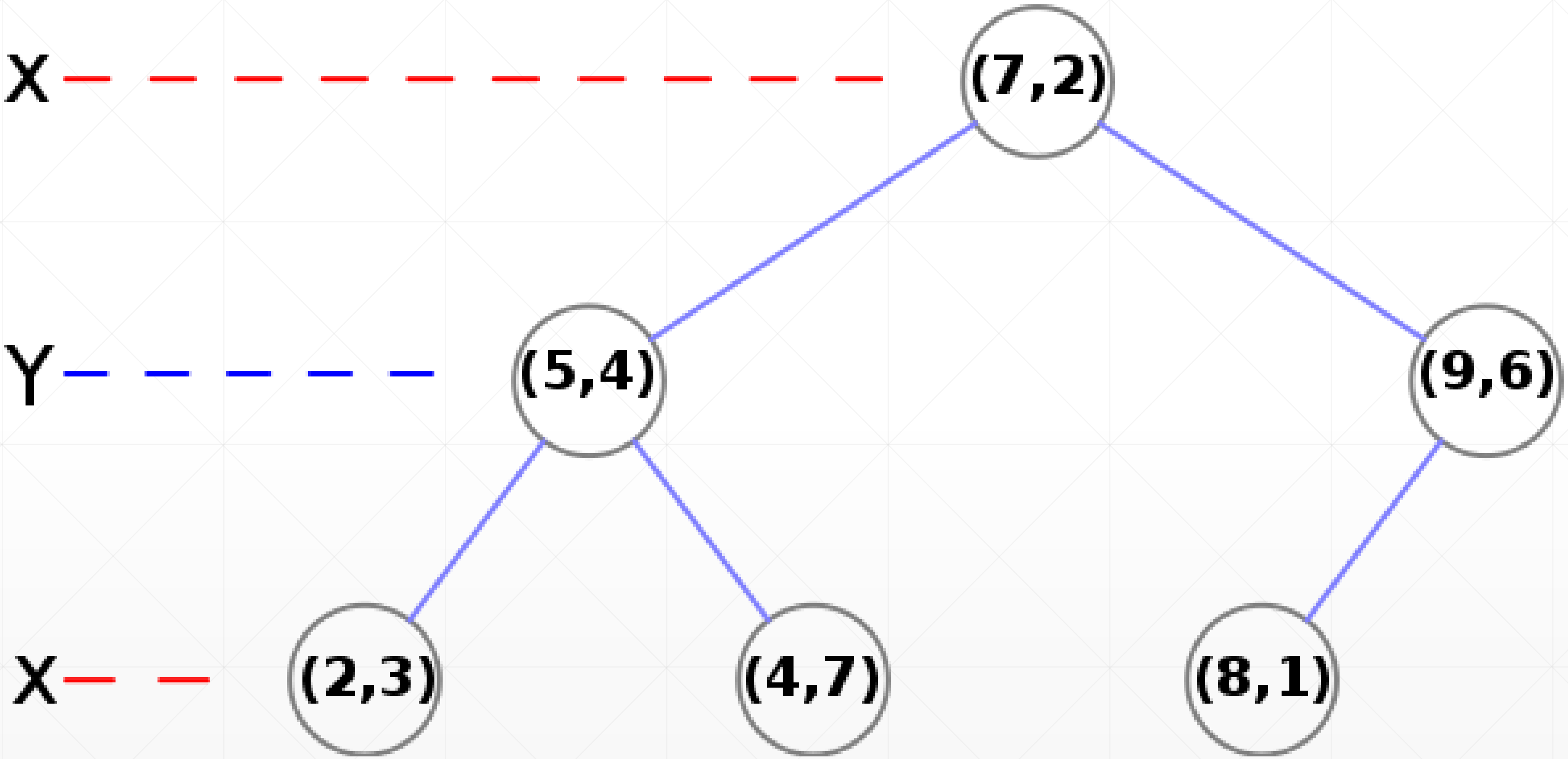
Class ● wins the vote! Point ○ is therefore predicted to be of class ●.

Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

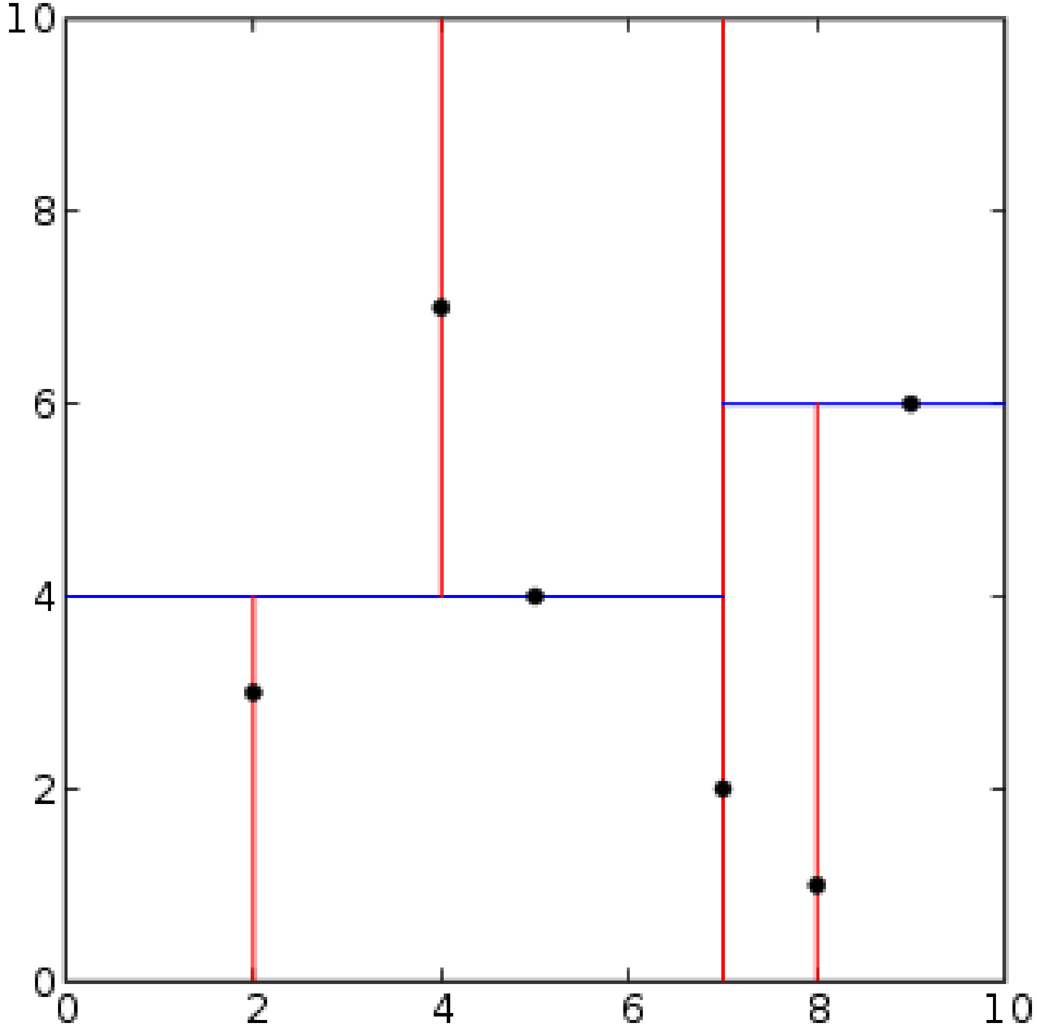
KD-Tree Algorithm for k-NN

```
function kdtree (list of points pointList, int depth)
{
    // Select axis based on depth so that axis cycles through all valid values
    var int axis := depth mod k;

    // Sort point list and choose median as pivot element
    select median by axis from pointList;

    // Create node and construct subtree
    node.location := median;
    node.leftChild := kdtree(points in pointList before median, depth+1);
    node.rightChild := kdtree(points in pointList after median, depth+1);
    return node;
}
```

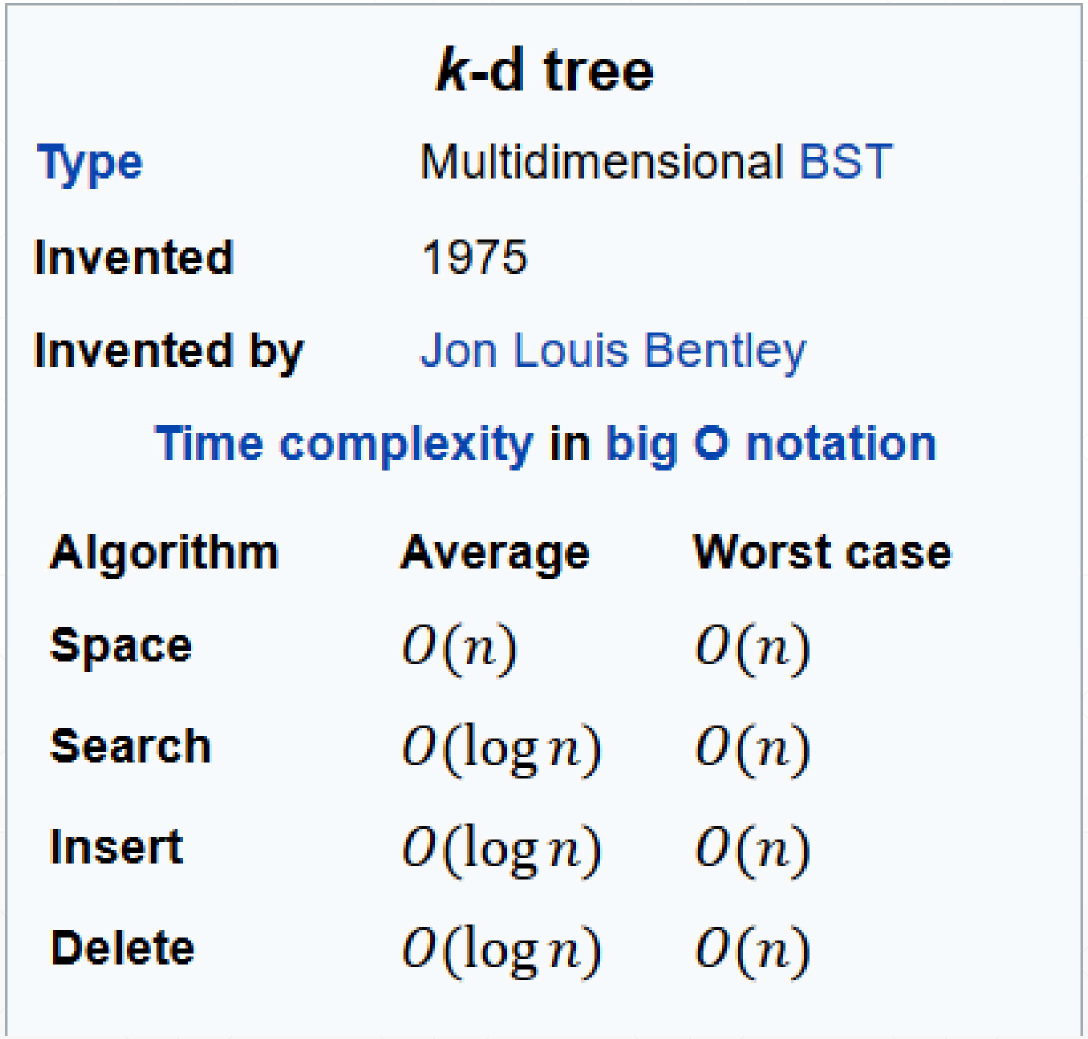
KD-Tree Visualization for k-NN



k-d tree decomposition for the point set (2,3), (5,4), (9,6), (4,7), (8,1), (7,2).



KD-Tree Algorithm Details



_k_-d tree

Type	Multidimensional BST
Invented	1975
Invented by	Jon Louis Bentley

Time complexity in big O notation

Algorithm	Average	Worst case
Space	$O(n)$	$O(n)$
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$

Software & Hardware Requirements

- Software Requirements
  - OS: Windows 7/8/10, Linux (Cent/Ubuntu).
  - Packages: Python 3.6, Anaconda Package for Data Science & Machine Learning.

- Hardware Requirements:
  - Minimum System Requirements:
    - 8GB RAM, 5GB Disk Space, 1GHz Processor.
  - Recommended System Requirements:
    - 16GB RAM, 15GB Disk Space, 3.5GHz Multi Core Processor.

Code Snippets

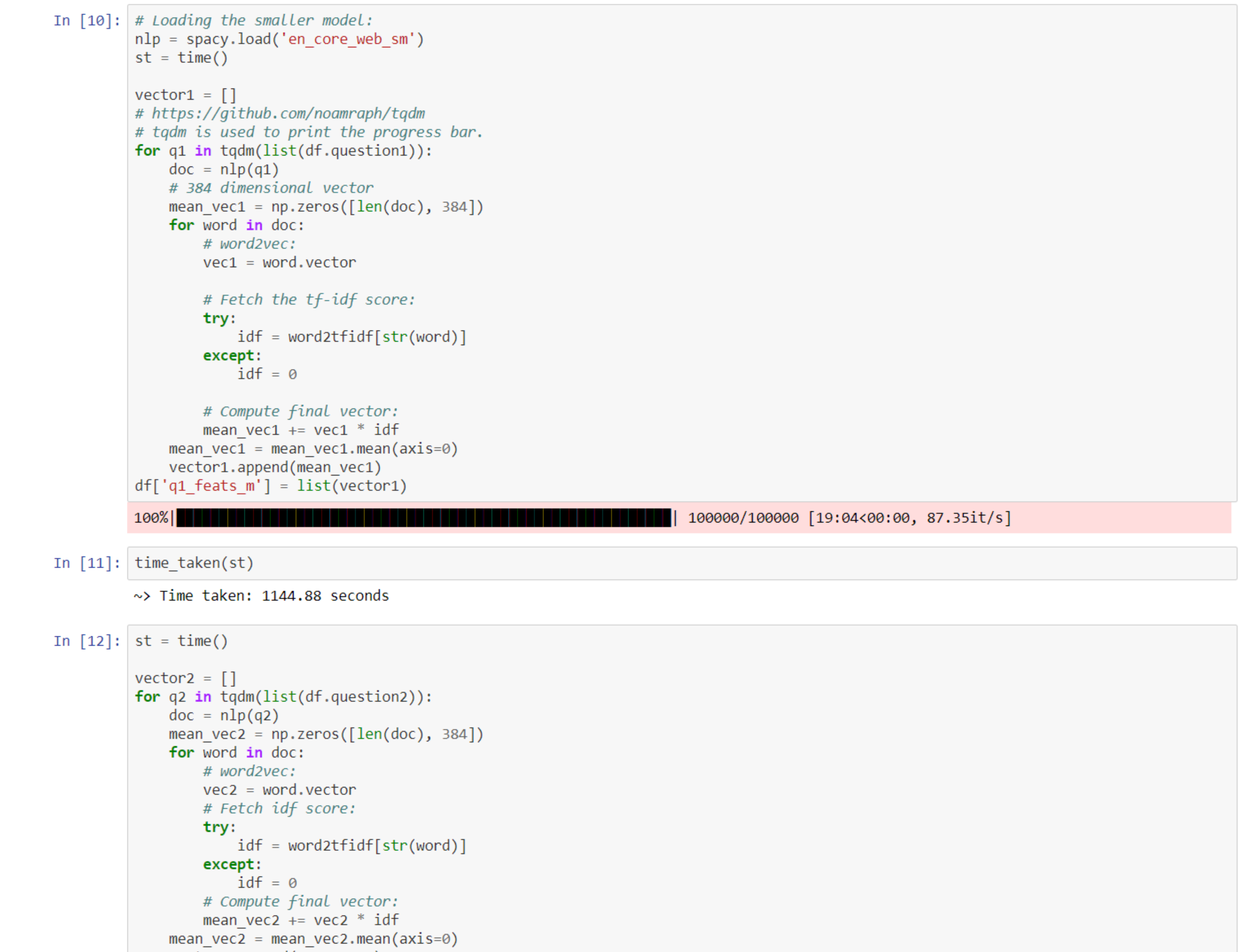
- **freq_qid1** = Frequency of qid1's
- **freq_qid2** = Frequency of qid2's
- **q1len** = Length of q1
- **q2len** = Length of q2
- **q1_n_words** = Number of words in Question 1
- **q2_n_words** = Number of words in Question 2
- **word_Common** = (Number of common unique words in Question 1 and Question 2)
- **word_Total** =(Total num of words in Question 1 + Total num of words in Question 2)
- **word_share** = (word_common)/(word_Total)
- **freq_q1+q2** = sum total of frequency of qid1 and qid2
- **freq_q1+q2** = absolute difference of frequency of qid1 and qid2

In [14]:

```
if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    df = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding='latin-1')
else:
    df['freq_qid1'] = df.groupby('qid1')['qid1'].transform('count')
    df['freq_qid2'] = df.groupby('qid2')['qid2'].transform('count')
    df['q1len'] = df['question1'].str.len()
    df['q2len'] = df['question2'].str.len()
    df['q1_n_words'] = df['question1'].apply(lambda row: len(row.split(" ")))
    df['q2_n_words'] = df['question2'].apply(lambda row: len(row.split(" ")))

    def normalized_word_Common(row):
        w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
        w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
        return 1.0 * len(w1 & w2)
    df['word_Common'] = df.apply(normalized_word_Common, axis=1)

    def normalized_word_Total(row):
        w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
        w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
        return 1.0 * (len(w1) + len(w2))
    df['word_Total'] = df.apply(normalized_word_Total, axis=1)

    def normalized_word_share(row):
        w1 = set(map(lambda word: word.lower().strip(), row['question1'].split(" ")))
        w2 = set(map(lambda word: word.lower().strip(), row['question2'].split(" ")))
        return 1.0 * len(w1 & w2)/(len(w1) + len(w2))
    df['word_share'] = df.apply(normalized_word_share, axis=1)

    df['freq_q1+q2'] = df['freq_qid1']+df['freq_qid2']
    df['freq_q1-q2'] = abs(df['freq_qid1']-df['freq_qid2'])

    df.to_csv("df_fe_without_preprocessing_train.csv", index=False)

# print the first 5 rows of the modified data frame:
df.head()
```

Out[14]:

```python
def get_token_features(q1,q2):
    """
    Function to get the first 10 Normal Features.
    """

    # Since we have 10 Normal Features:
    token_features = [0.0]*10

    # Converting the sentence into tokens:
    q1_tokens = q1.split()
    q2_tokens = q2.split()

    # If either q1 or q2 is empty, return token_features
    if len(q1_tokens) == 0 or len(q2_tokens) == 0:
        return token_features

    # Get the non-stopwords in q1 and q2:
    q1_words = set([word for word in q1_tokens if word not in STOP_WORDS])
    q2_words = set([word for word in q2_tokens if word not in STOP_WORDS])

    # Get the stopwords in q1 and q2:
    q1_stops = set([word for word in q1_tokens if word in STOP_WORDS])
    q2_stops = set([word for word in q2_tokens if word in STOP_WORDS])

    # Get the common non-stopwords from q1 & q2:
    common_word_count = len(q1_words.intersection(q2_words))
    # common_word_count = len(q1_words & q2_words)

    # Get the common stopwords from q1 & q2:
    common_stop_count = len(q1_stops.intersection(q2_stops))
    # common_stop_count = len(q1_stops & q2_stops)

    # Get the common tokens from q1 & q2:
    common_token_count = len(set(q1_tokens).intersection(set(q2_tokens)))
    # common_token_count = len(set(q1_tokens) & set(q2_tokens))

    # cwc_min:
    token_features[0] = common_word_count / (min(len(q1_words), len(q2_words)) + SAFE_DIV)
    # cwc_max:
    token_features[1] = common_word_count / (max(len(q1_words), len(q2_words)) + SAFE_DIV)


    # csc_min:
    token_features[2] = common_stop_count / (min(len(q1_stops), len(q2_stops)) + SAFE_DIV)
    # csc_max:
    token_features[3] = common_stop_count / (max(len(q1_stops), len(q2_stops)) + SAFE_DIV)


    # ctc_min:
    token_features[4] = common_token_count / (min(len(q1_tokens), len(q2_tokens)) + SAFE_DIV)
    # ctc_max:
```



In [10]:
```python
# Loading the smaller model:
nlp = spacy.load('en_core_web_sm')
st = time()

vector1 = []
# https://github.com/noamraph/tqdm
# tqdm is used to print the progress bar.
for q1 in tqdm(list(df.question1)):
    doc = nlp(q1)
    # 384 dimensional vector
    mean_vec1 = np.zeros([len(doc), 384])
    for word in doc:
        # word2vec:
        vec1 = word.vector

        # Fetch the tf-idf score:
        try:
            idf = word2tfidf[str(word)]
        except:
            idf = 0

        # Compute final vector:
        mean_vec1 += vec1 * idf
    mean_vec1 = mean_vec1.mean(axis=0)
    vector1.append(mean_vec1)
df['q1_feats_m'] = list(vector1)
```

```
100%|██████████████████████████████████████████████████| 100000/100000 [19:04<00:00, 87.35it/s]
```

In [11]:
```python
time_taken(st)
```

```
~> Time taken: 1144.88 seconds
```

In [12]:
```python
st = time()

vector2 = []
for q2 in tqdm(list(df.question2)):
    doc = nlp(q2)
    mean_vec2 = np.zeros([len(doc), 384])
    for word in doc:
        # word2vec:
        vec2 = word.vector
        # Fetch idf score:
        try:
            idf = word2tfidf[str(word)]
        except:
            idf = 0
        # Compute final vector:
        mean_vec2 += vec2 * idf
    mean_vec2 = mean_vec2.mean(axis=0)
```

```
 0.358  0.358  0.357  0.358  0.357  0.358  0.356  0.358  0.359  0.359
 0.357  0.356  0.357  0.357  0.357  0.356  0.358  0.359  0.358  0.36 ]
```

From the plot above, we can see that the lowest value of Misclassification error is generated in between k=[20, 21, ..., 40]. That's the reason, we got our optimal_k to be 27.

Let us see the accuracy score after querying the k-NN model with the test data.

In [40]:

```
# KNN with k = optimal_k
st = time()
# Configured parameters are:-
#
# 1. algorithm = 'auto':
#    automatically choose the algorithm (KDTree, BallTree or Brute Force)
#
# 2. metric = 'minkowski', p = 2:
#    Use L2 Minkowski Distance which is nothing but Euclidean Distance.
#
# 3. n_jobs = -1:
#    Use all the CPU cores to apply KNN Classfication.

# Instantiate the learning model:
knn_optimal = KNeighborsClassifier(
    n_neighbors = optimal_k,
    algorithm = 'auto',
    metric = 'minkowski',
    p = 2,
    n_jobs = 3
)

# Fitting the model on train:
knn_optimal.fit(X_train, y_train)

# Predict the response on test:
predict_y_test = knn_optimal.predict(X_test)

# Evaluate the test accuracy:
acc_test = accuracy_score(predict_y_test, y_test, normalize=True) * float(100)
print('''\nThe Accuracy of k-NN classifier on Quora Question Pairs Dataset
for predicting whether two given questions have the same intent or not with
k={} is {}%'''.format(optimal_k, acc_test))

time_taken(st)
```

```
The Accuracy of k-NN classifier on Quora Question Pairs Dataset
for predicting whether two given questions have the same intent or not with
k=27 is 65.70666666666666%

Runtime: 111.65 seconds
```

Conclusion

- Successfully applied k-Nearest Neighbours Algorithm on the Quora Question Pair Similarity Dataset as two variants:
  - k-NN after Simple Cross Validation [Train-Test Split: 70-30 of 25k Sampled Data Points]
  - k-NN after K-fold Cross Validation [Here, ‘K’ is not the same ‘k’ in k-NN].
- The Accuracy of k-NN classifier on a sample of 25,000 Data Points of Quora Question Pairs Dataset for predicting whether two given questions have the same intent or not with k=27 is 65.70666666666666%.
- Due to a computational discrepancy, only 25k Data Points were taken. But, if we apply the same algorithm on the entire Dataset, the Accuracy Score will be near to ~90%.

Thank You!

Any Questions/Queries?