

# Quora Question Pairs

## 1. Business Problem

### 1.1 Description

Quora is a place to gain and share knowledge—about anything. It's a platform to ask questions and connect with people who contribute unique insights and quality answers. This empowers people to learn from each other and to better understand the world.

Over 100 million people visit Quora every month, so it's no surprise that many people ask similarly worded questions. Multiple questions with the same intent can cause seekers to spend more time finding the best answer to their question, and make writers feel they need to answer multiple versions of the same question. Quora values canonical questions because they provide a better experience to active seekers and writers, and offer more value to both of these groups in the long term.

Credits: Kaggle

#### Problem Statement

- Identify which questions asked on Quora are duplicates of questions that have already been asked.
- This could be useful to instantly provide answers to questions that have already been answered.
- We are tasked with predicting whether a pair of questions are duplicates or not.

### 1.2 Sources/Useful Links

- Source : <https://www.kaggle.com/c/quora-question-pairs>

#### Useful Links

- Discussions : <https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb/comments>
- Kaggle Winning Solution and other approaches: <https://www.dropbox.com/sh/93968nfnrzh8bp5/AACZdtsApc1QSTQc7X0H3QZ5a?dl=0>
- Blog 1 : <https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning>
- Blog 2 : <https://towardsdatascience.com/identifying-duplicate-questions-on-quora-top-12-on-kaggle-4c1cf93f1c30>

### 1.3 Real world/Business Objectives and Constraints

1. The cost of a mis-classification can be very high.
2. You would want a probability of a pair of questions to be duplicates so that you can choose any threshold of choice.
3. No strict latency concerns.
4. Interpretability is partially important.

## 2. Machine Learning Problem

### 2.1 Data

#### 2.1.1 Data Overview

- Data will be in a file Train.csv
- Train.csv contains 5 columns : qid1, qid2, question1, question2, is\_duplicate
- Size of Train.csv - 60MB
- Number of rows in Train.csv = 404,290

### 2.1.2 Example Data point

```
"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"
"1","3","4","What is the story of Kohinoor (Koh-i-Noor) Diamond?","What would happen if the
Indian government stole the Kohinoor (Koh-i-Noor) diamond back?","0"
"7","15","16","How can I be a good geologist?","What should I do to be a great
geologist?","1"
"11","23","24","How do I read and find my YouTube comments?","How can I see all my Youtube
comments?","1"
```

## 2.2 Mapping the real world problem to an ML problem

### 2.2.1 Type of Machine Learning Problem

It is a binary classification problem, for a given pair of questions we need to predict if they are duplicate or not.

### 2.2.2 Performance Metric

Source: <https://www.kaggle.com/c/quora-question-pairs#evaluation>

Metric(s):

- log-loss : <https://www.kaggle.com/wiki/LogarithmicLoss>
- Binary Confusion Matrix

## 2.3 Train and Test Construction

We build train and test by randomly splitting in the ratio of 70:30 or 80:20 whatever we choose as we have sufficient points to work with.

## 3. Exploratory Data Analysis

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from subprocess import check_output
%matplotlib inline
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import os
import gc

import re
from nltk.corpus import stopwords
import distance
from nltk.stem import PorterStemmer
from bs4 import BeautifulSoup
```

## 3.1 Reading data and basic stats

In [2]:

```
df = pd.read_csv("train.csv")

print("Number of data points:", df.shape[0])
```

Number of data points: 404290

In [3]:

```
df.head()
```

Out[3]:

	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 $23^{24}$ i...	0
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 6 columns):
id                404290 non-null int64
qid1              404290 non-null int64
qid2              404290 non-null int64
question1         404289 non-null object
question2         404288 non-null object
is_duplicate      404290 non-null int64
dtypes: int64(4), object(2)
memory usage: 18.5+ MB
```

We are given a minimal number of data fields here, consisting of:

- id: Looks like a simple rowID
- qid{1, 2}: The unique ID of each question in the pair
- question{1, 2}: The actual textual contents of the questions.
- is\_duplicate: The label that we are trying to predict - whether the two questions are duplicates of each other.

### 3.2.1 Distribution of data points among output classes

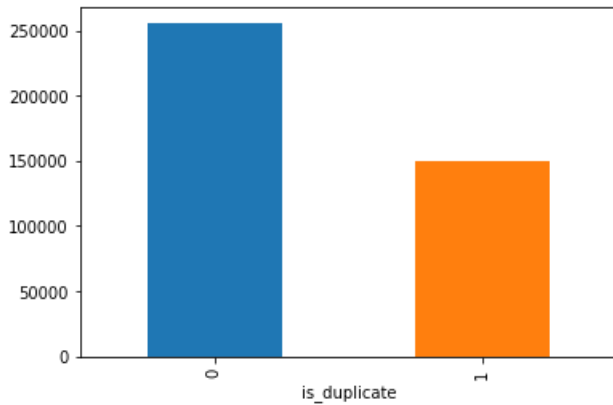
- Number of duplicate(similar) and non-duplicate(non similar) questions

In [5]:

```
df.groupby("is_duplicate")["id"].count().plot.bar()
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1eb59360208>
```



In [6]:

```
print('~> Total number of question pairs for training:\n {}'.format(len(df)))
```

```
~> Total number of question pairs for training:
404290
```

In [7]:

```
print('~> Question pairs are not Similar (is_duplicate = 0):\n {}'.format(100 -
round(df['is_duplicate'].mean()*100, 2)))
print('\n~> Question pairs are Similar (is_duplicate = 1):\n {}'.format(round(df['is_duplicate']
].mean()*100, 2)))
```

```
~> Question pairs are not Similar (is_duplicate = 0):
63.08%
```

```
~> Question pairs are Similar (is_duplicate = 1):
36.92%
```

### 3.2.2 Number of unique questions

In [8]:

```
qids = pd.Series(df['qid1'].tolist() + df['qid2'].tolist())
unique_qs = len(np.unique(qids))
qs_morethan_onetime = np.sum(qids.value_counts() > 1)
print ('Total number of Unique Questions are: {}\n'.format(unique_qs))
#print len(np.unique(qids))

print ('Number of unique questions that appear more than one time: {}
({}%)\n'.format(qs_morethan_onetime,qs_morethan_onetime/unique_qs*100))

print ('Max number of times a single question is repeated: {}\n'.format(max(qids.value_counts()))))

q_vals=qids.value_counts()

q_vals=q_vals.values
```

```
Total number of Unique Questions are: 537933
```

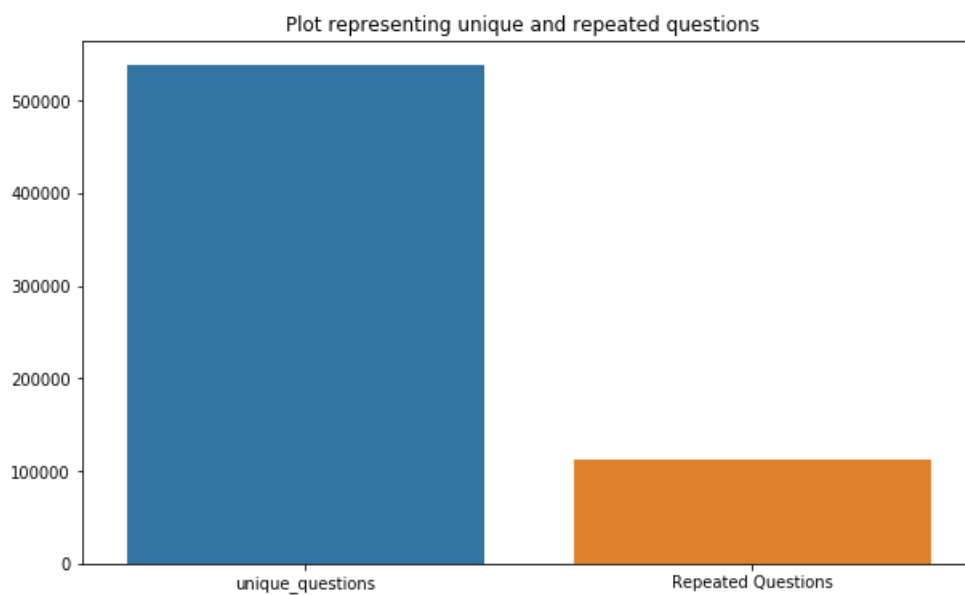
```
Number of unique questions that appear more than one time: 111780 (20.77953945937505%)
```

```
Max number of times a single question is repeated: 157
```

In [9]:

```
x = ["unique_questions" , "Repeated Questions"]
y = [unique_qs , qs_morethan_onetime]
```

```
plt.figure(figsize=(10, 6))
plt.title ("Plot representing unique and repeated questions ")
sns.barplot(x,y)
plt.show()
```



### 3.2.3 Checking for Duplicates

In [10]:

```
#checking whether there are any repeated pair of questions

pair_duplicates =
df[['qid1','qid2','is_duplicate']].groupby(['qid1','qid2']).count().reset_index()

print ("Number of duplicate questions", (pair_duplicates).shape[0] - df.shape[0])
```

Number of duplicate questions 0

### 3.2.4 Number of occurrences of each question

In [11]:

```
plt.figure(figsize=(20, 10))

plt.hist(qids.value_counts(), bins=160)

plt.yscale('log', nonposy='clip')

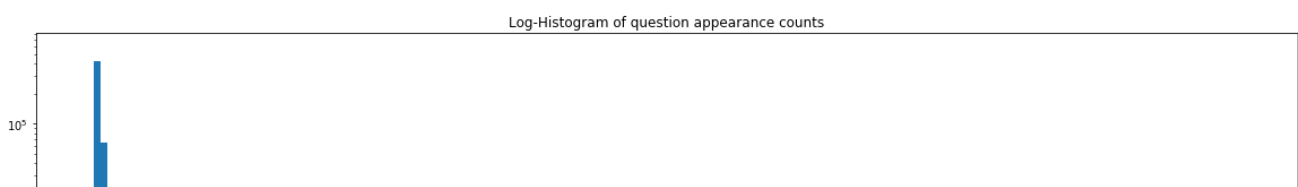
plt.title('Log-Histogram of question appearance counts')

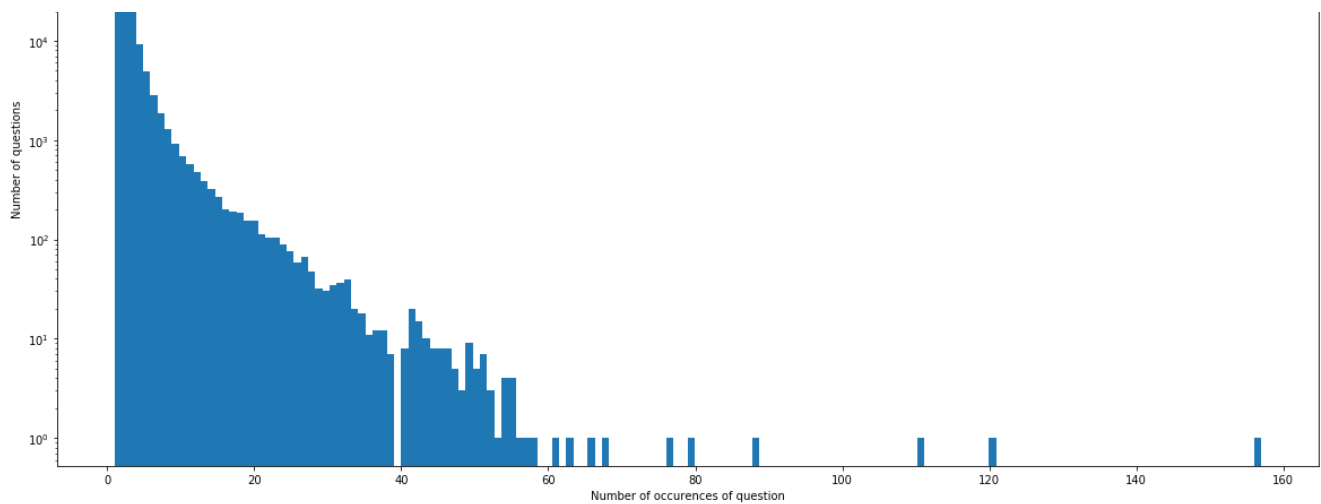
plt.xlabel('Number of occurrences of question')

plt.ylabel('Number of questions')

print ('Maximum number of times a single question is repeated: {}'.format(max(qids.value_counts(
))))
```

Maximum number of times a single question is repeated: 157





### 3.2.5 Checking for NULL values

In [12]:

```
#Checking whether there are any rows with null values
nan_rows = df[df.isnull().any(1)]
print (nan_rows)
```

	id	qid1	qid2	question1 \	question2	is_duplicate
105780	105780	174363	174364	How can I develop android app?		
201841	201841	303951	174364	How can I create an Android app?		
363362	363362	493340	493341	NaN		
105780					NaN	0
201841					NaN	0
363362				My Chinese name is Haichao Yu. What English na...		0

- There are two rows with null values in question2

In [13]:

```
# Filling the null values with ' '
df = df.fillna(' ')
nan_rows = df[df.isnull().any(1)]
print (nan_rows)
```

Empty DataFrame  
Columns: [id, qid1, qid2, question1, question2, is\_duplicate]  
Index: []

## 3.3 Basic Feature Extraction (before cleaning)

Let us now construct a few features like:

- **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+freq\_q2** = sum total of frequency of qid1 and qid2
- **freq\_q1-freq\_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)

df.head()
```

Out[14]:

	id	qid1	qid2	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_
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	66	57	14	12	10.0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	4	1	51	88	8	13	4.0
2	2	5	6	How can I increase the speed of my internet co...	How can Internet speed be increased by hacking...	0	1	1	73	59	14	10	4.0
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when $23^{24}$ i...	0	1	1	50	65	11	9	0.0
				Which one dissolve in water	Which fish								

4	4	9	10	water	would survive	0	3	1	76	39	13	7	2.0
id	qid1	qid2	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word	
			salt...										

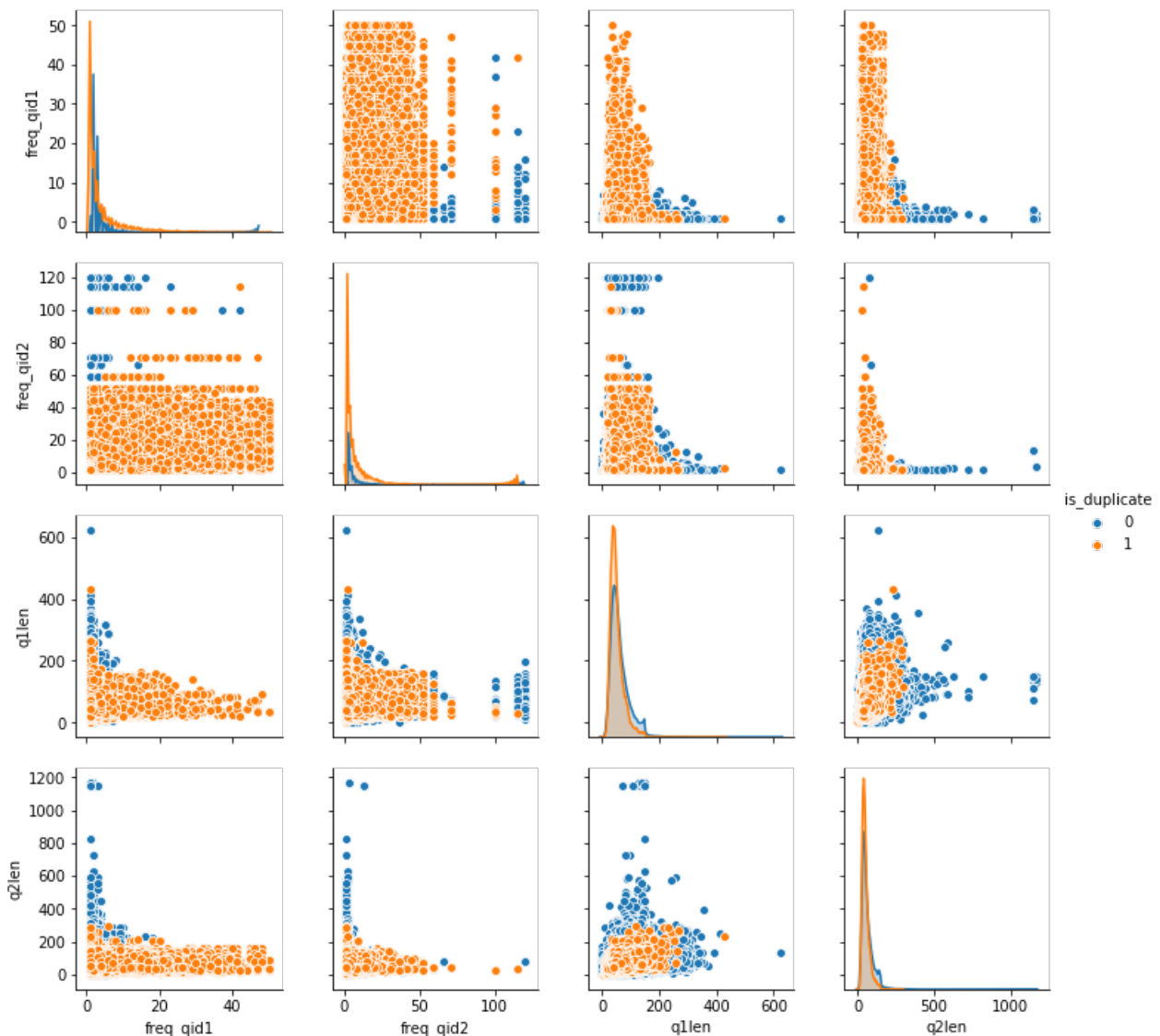
## Pairplots

In [15]:

```
n = df.shape[0]
sns.pairplot(df[['freq_qid1', 'freq_qid2', 'q1len', 'q2len', 'is_duplicate']][0:n], hue='is_duplicate', vars=['freq_qid1', 'freq_qid2', 'q1len', 'q2len'])
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.



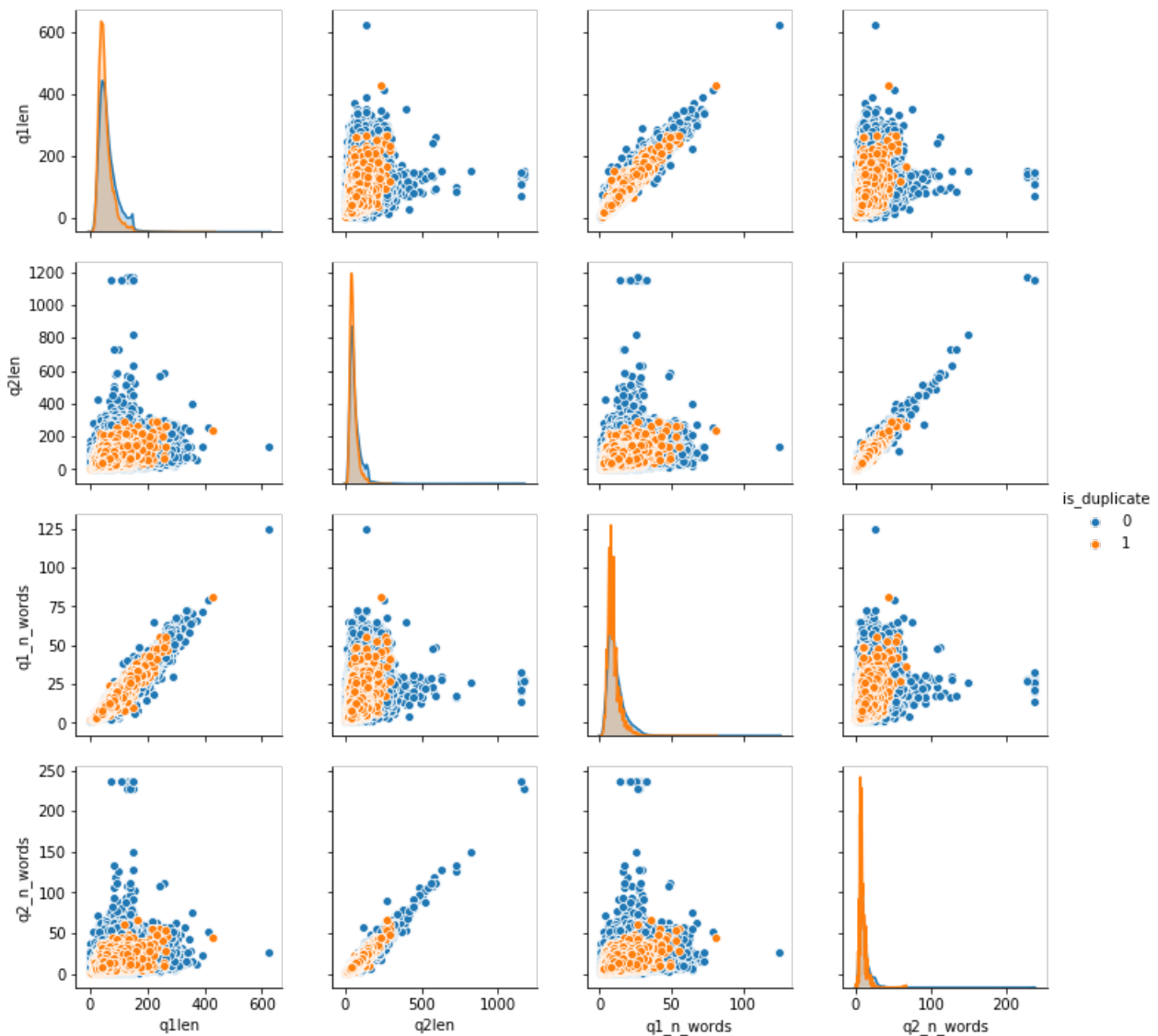
In [16]:

```
n = df.shape[0]
sns.pairplot(df[['q1len', 'q2len', 'q1_n_words', 'q2_n_words', 'is_duplicate']][0:n], hue='is_duplicate', vars=['q1len', 'q2len', 'q1_n_words', 'q2_n_words'])
plt.show()
```



C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.

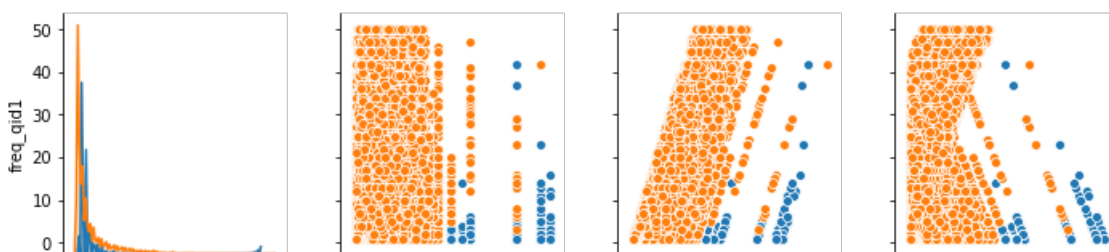


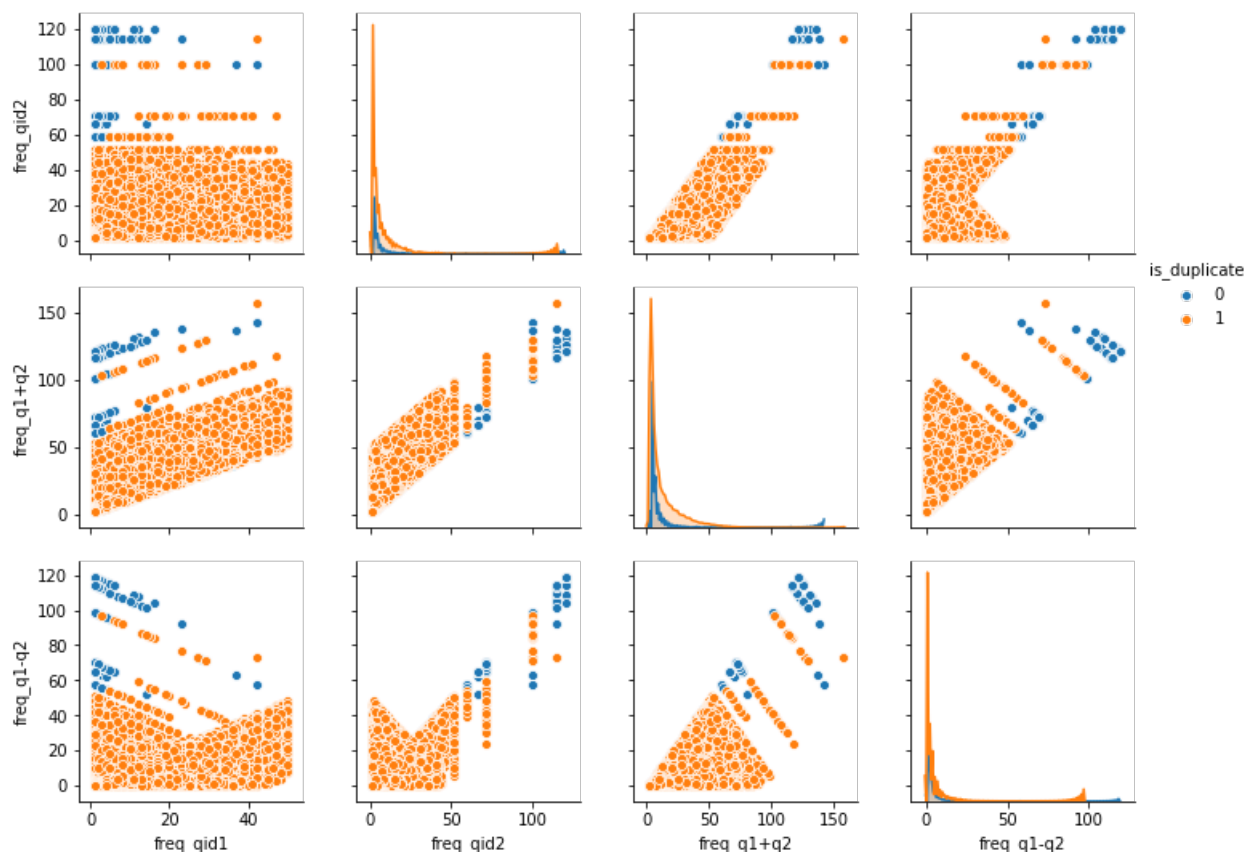
In [17]:

```
n = df.shape[0]
sns.pairplot(df[['freq_qid1', 'freq_qid2', 'freq_q1+q2', 'freq_q1-q2', 'is_duplicate']][0:n], hue='is_duplicate', vars=['freq_qid1', 'freq_qid2', 'freq_q1+q2', 'freq_q1-q2'])
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.



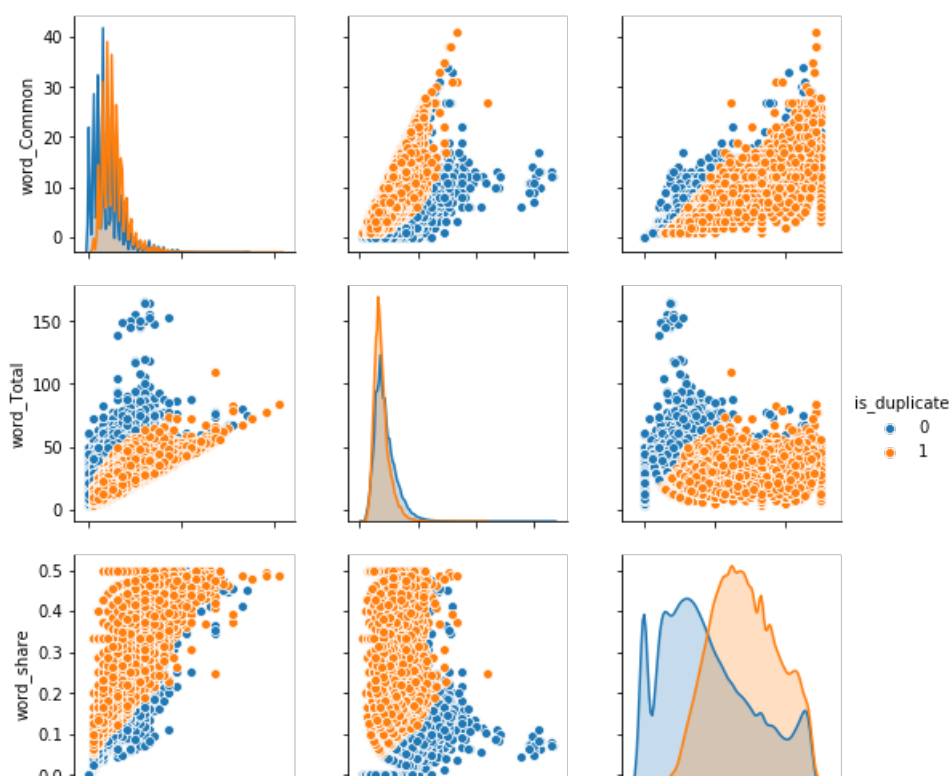


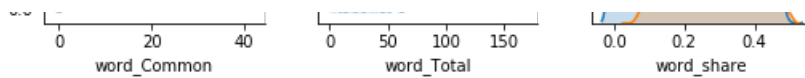
In [18]:

```
n = df.shape[0]
sns.pairplot(df[['word_Common', 'word_Total', 'word_share', 'q1_n_words', 'q2_n_words',
                'is_duplicate']], hue='is_duplicate', vars=['word_Common', 'word_Total', 'word_share'])
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.





In [19]:

```
n = df.shape[0]
sns.pairplot(df[['freq_qid1', 'freq_qid2', 'q1len', 'q2len', 'word_Common', 'word_Total', 'word_share', 'freq_q1+q2', 'freq_q1-q2', 'is_duplicate']][0:n], hue='is_duplicate', vars=['freq_qid1', 'freq_qid2', 'q1len', 'q2len', 'word_Common', 'word_Total', 'word_share', 'freq_q1+q2', 'freq_q1-q2'])
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.



### 3.3.1 Analysis of some of the extracted features

- Here are some questions have only one single words.

In [20]:

```
print ("Minimum length of the questions in question1 : " , min(df['q1_n_words']))

print ("Minimum length of the questions in question2 : " , min(df['q2_n_words']))

print ("Number of Questions with minimum length [question1] :", df[df['q1_n_words']== 1].shape[0])
print ("Number of Questions with minimum length [question2] :", df[df['q2_n_words']== 1].shape[0])
```

```
Minimum length of the questions in question1 : 1
Minimum length of the questions in question2 : 1
Number of Questions with minimum length [question1] : 67
Number of Questions with minimum length [question2] : 24
```

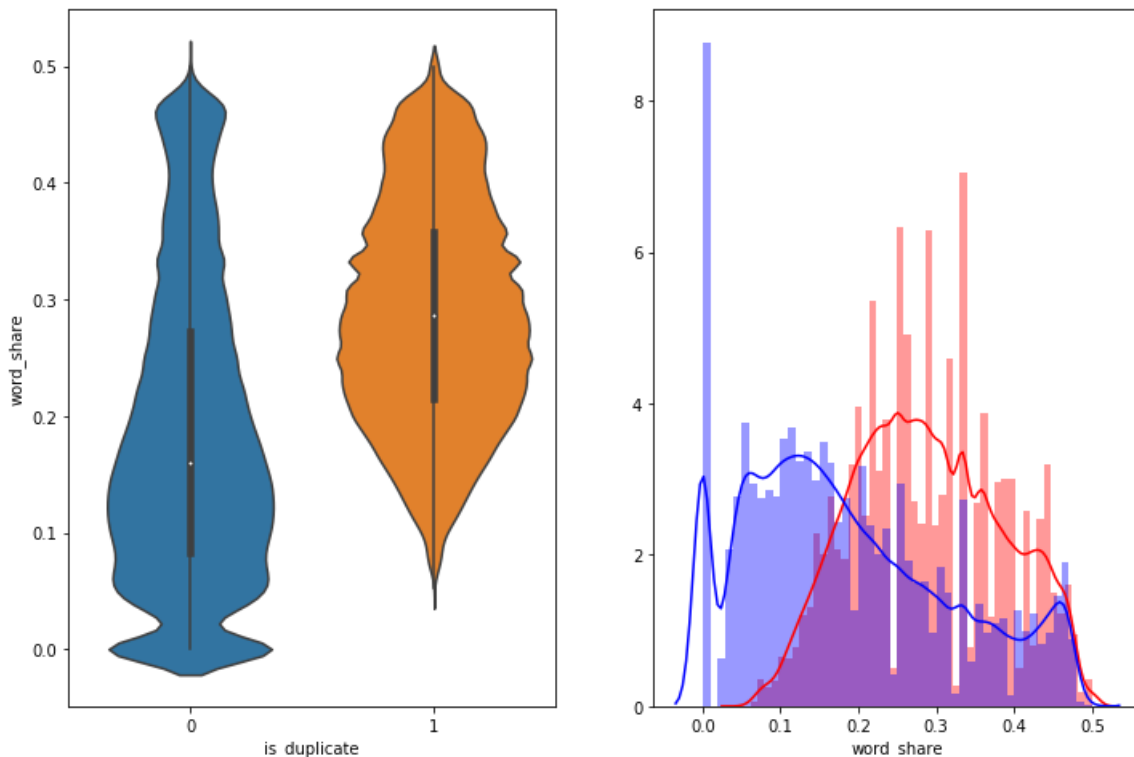
### 3.3.1.1 Feature: word\_share

In [17]:

```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'word_share', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_share'][0:], label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_share'][0:], label = "0" , color = 'blue' )
plt.show()
```



In [22]:

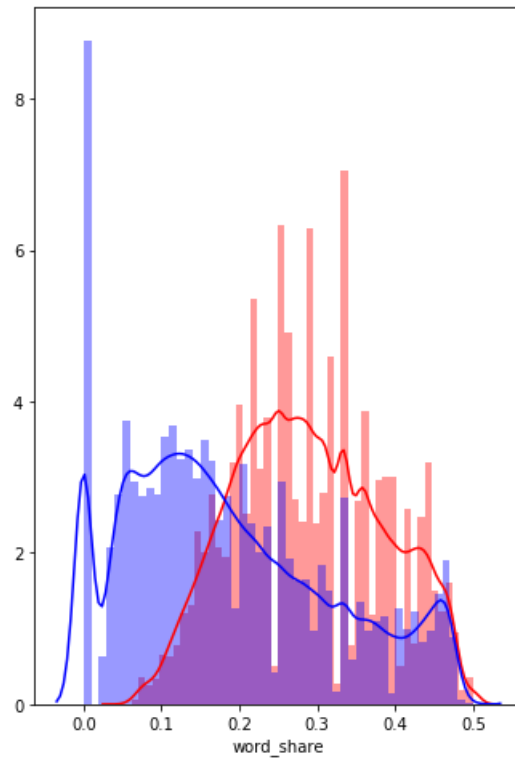
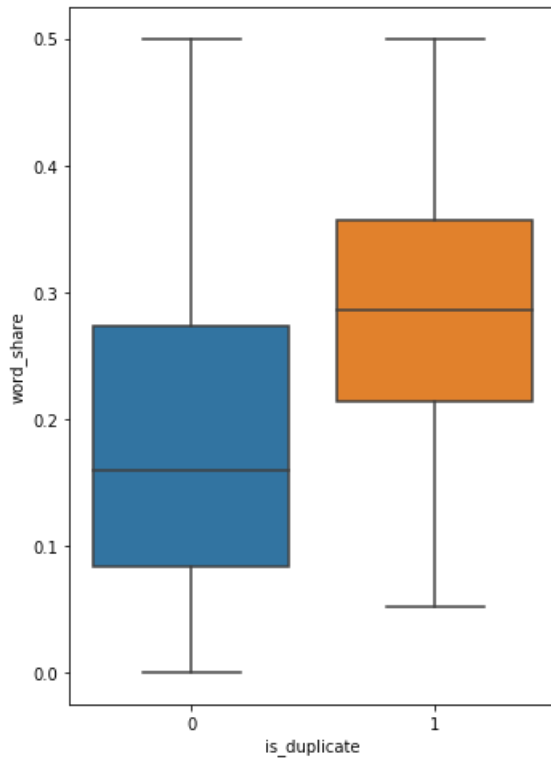
```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.boxplot(x = 'is_duplicate', y = 'word_share', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_share'][0:], label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_share'][0:], label = "0" , color = 'blue' )
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.



- The distributions for normalized word\_share have some overlap on the far right-hand side, i.e., there are quite a lot of questions with high word similarity
- The average word share and Common no. of words of qid1 and qid2 is more when they are duplicate(Similar)

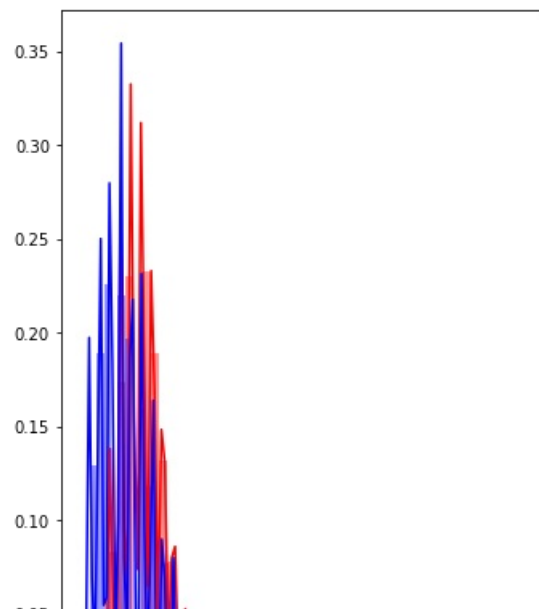
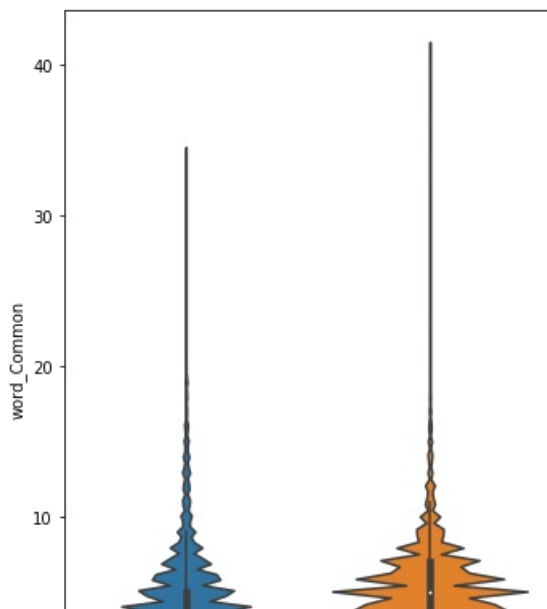
### 3.3.1.2 Feature: word\_Common

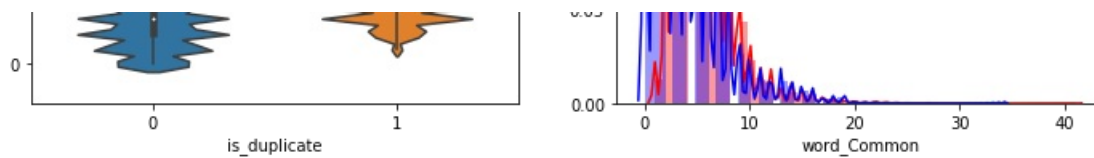
In [18]:

```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'word_Common', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_Common'][0:], label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_Common'][0:], label = "0", color = 'blue' )
plt.show()
```





In [23]:

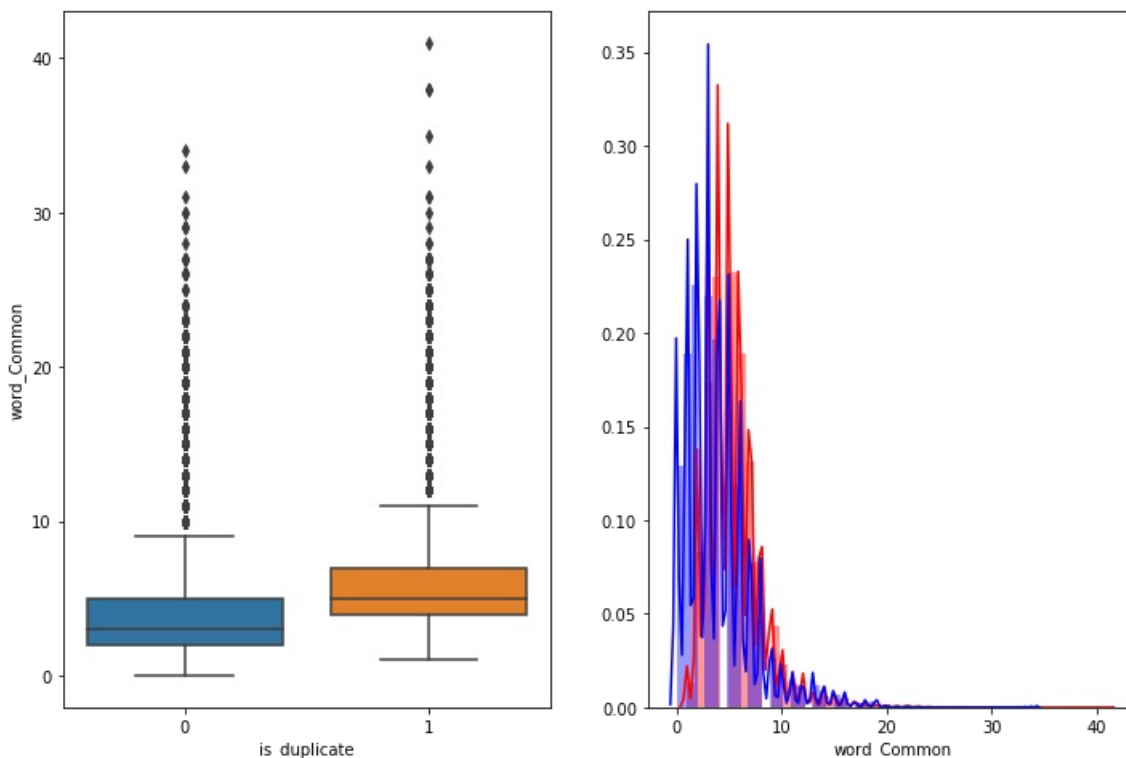
```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.boxplot(x = 'is_duplicate', y = 'word_Common', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_Common'][0:], label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_Common'][0:], label = "0", color = 'blue' )
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.



In [24]:

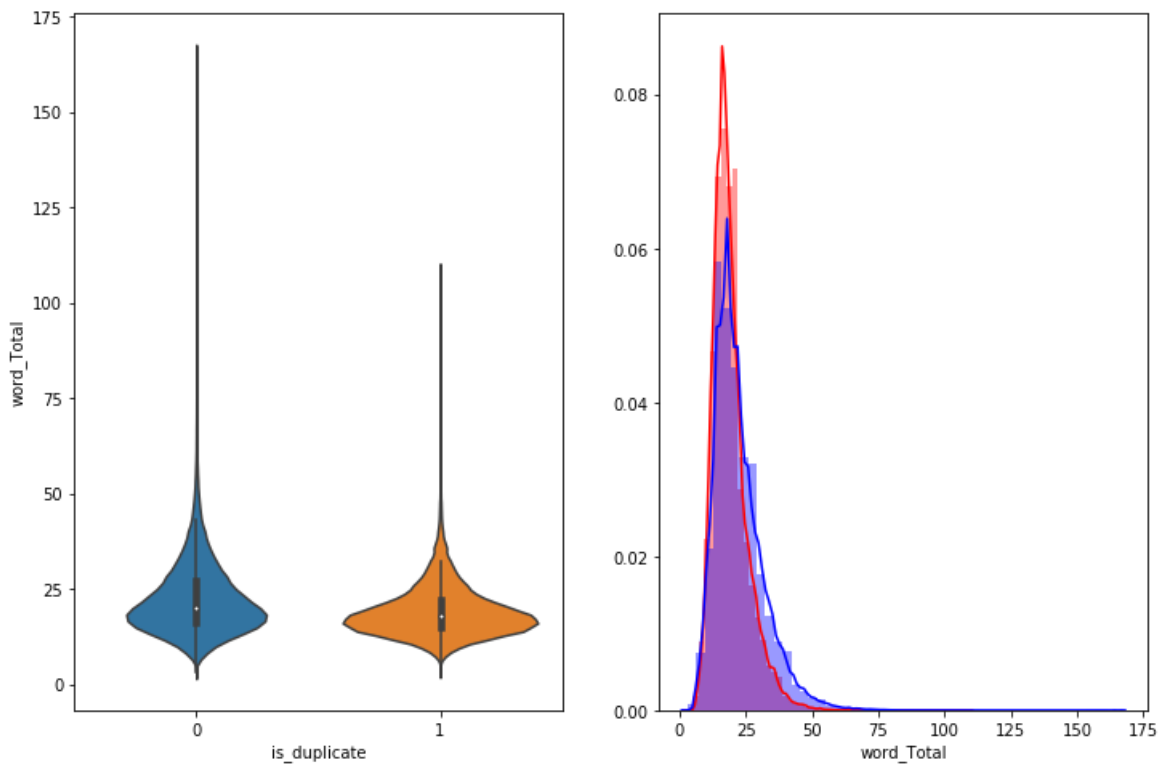
```
plt.figure(figsize=(12, 8))

plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'word_Total', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['word_Total'][0:], label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['word_Total'][0:], label = "0", color = 'blue' )
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.



In [25]:

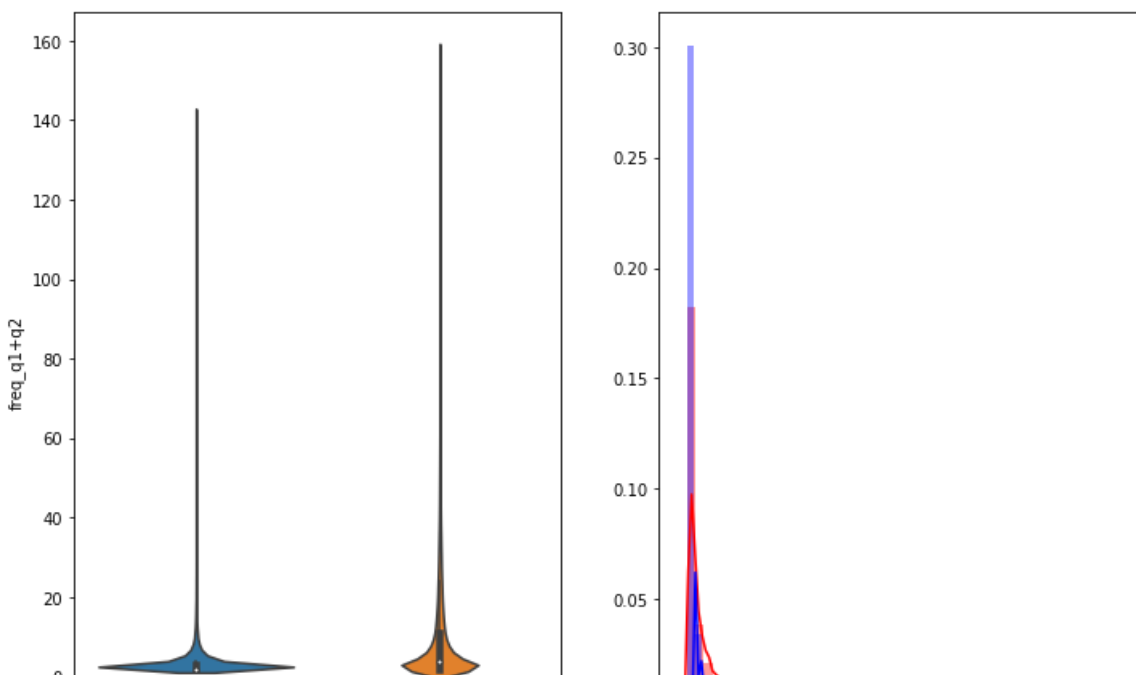
```
plt.figure(figsize=(12, 8))

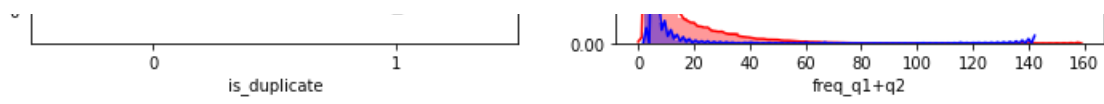
plt.subplot(1,2,1)
sns.violinplot(x = 'is_duplicate', y = 'freq_q1+q2', data = df[0:])

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['freq_q1+q2'][0:], label = "1", color = 'red')
sns.distplot(df[df['is_duplicate'] == 0.0]['freq_q1+q2'][0:], label = "0", color = 'blue' )
plt.show()
```

C:\Users\krush\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.





The distributions of the word\_Common feature in similar and non-similar questions are highly overlapping