Project_2_Quara Question Pairs Similarity

Dataset Link - https://www.kaggle.com/c/quora-question-pairs (https://www.kaggle.com/c/quora-question-pairs)

Dataset Description

The goal of this competition is to predict which of the provided pairs of questions contain two questions with the same meaning. The ground truth is the set of labels that have been supplied by human experts. The ground truth labels are inherently subjective, as the true meaning of sentences can never be known with certainty. Human labeling is also a 'noisy' process, and reasonable people will disagree. As a result, the ground truth labels on this dataset should be taken to be 'informed' but not 100% accurate, and may include incorrect labeling. We believe the labels, on the whole, to represent a reasonable consensus, but this may often not be true on a case by case basis for individual items in the dataset.

Please note:

as an anti-cheating measure, Kaggle has supplemented the test set with computer-generated question pairs. Those rows do not come from Quora, and are not counted in the scoring. All of the questions in the training set are genuine examples from Quora.

Data fields

id - the id of a training set question pair qid1, qid2 - unique ids of each question (only available in train.csv) question1, question2 - the full text of each question is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

In [1]:

```
import numpy as np
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import re
from textblob import TextBlob
from wordcloud import WordCloud
import seaborn as sns
import matplotlib.pyplot as plt
import cufflinks as cf
%matplotlib inline
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected = True)
cf.go_offline();
import plotly.graph_objs as go
from plotly.subplots import make_subplots


pd.set_option('display.max_columns', None)


from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer

from collections import Counter
from numpy import where



from sklearn.decomposition import PCA

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn import metrics

from scipy.sparse import hstack, vstack

from prettytable import PrettyTable
from scipy.stats import loguniform # Log-uniform is useful for searching
from sklearn.model_selection import RepeatedStratifiedKFold, RandomizedSe

import warnings
warnings.filterwarnings('ignore')
```

```
C:\ProgramData\anaconda3\Lib\site-packages\paramiko\transport.py:219: Crypto
graphyDeprecationWarning:

Blowfish has been deprecated
```

In [2]:
```
df = pd.read_csv('train.csv')
df.shape
```

Out[2]: (404290, 6)

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 [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

In [4]:
```
1 df.sample(10)
```

Out[4]:

	id	qid1	qid2	question1	question2	is_duplicate
368038	368038	498371	498372	How realistic are Tyler Durden's plans in "Fig...	[SPOILER ALERT] At the end of Fight Club, why ...	0
16732	16732	31869	31870	If goods are being transported from one state ...	If I want to sell goods from one state to anot...	0
82545	82545	139936	98268	What are some things new employees should know...	What are some things new employees should know...	0
253486	253486	368039	368040	Where does GoodRec.com get its data from?	Where does www.magnetic.com get its data from?	0
225230	225230	333516	333517	What was your latest discovery?	What is the latest discovery in finance?	0
357058	357058	9642	154412	How can you make a friend?	How do I make more friends? How do I?	1
11808	11808	22786	22787	What makes you proud to be Canadian?	How do I make perfect CV for job banks for Can...	0
300133	300133	361686	422891	Has India been succesful to divert world atten...	Why is India not conducting a surgical strike ...	0
88130	88130	148290	148291	How did Aditi Saini, once a careless girl (as ...	What are the perks of being an IES officer?	0
233509	233509	343774	343775	What is the use of shell scripting?	Where can I get Shell Script programs?	0

In [5]:
```
1 df.info()
```

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

In [6]:
```
1 df = df.sample(30000,random_state=2)
```

In [7]:
```
# missinig values
df.isnull().sum()
```

Out[7]:
```
id             0
qid1           0
qid2           0
question1      0
question2      0
is_duplicate   0
dtype: int64
```

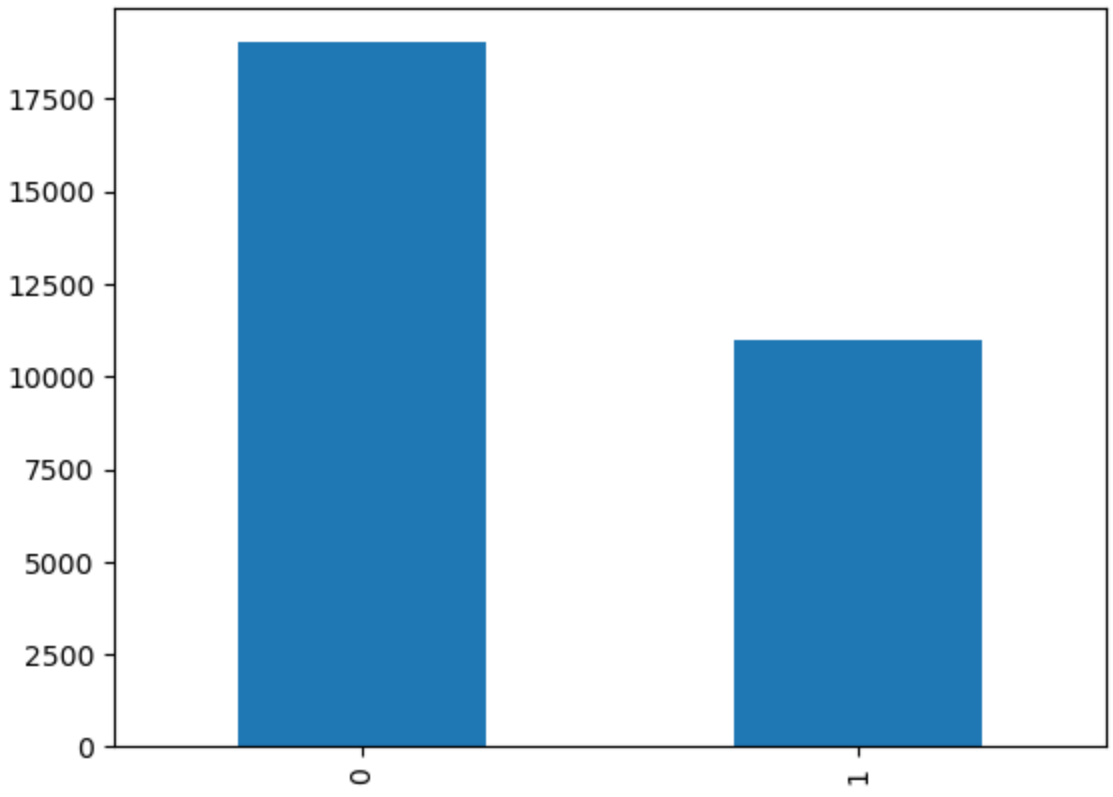
In [8]:
```
# duplicate raws
df.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
# Diistribution of duplicate and non-duplicate questions
print(df['is_duplicate'].value_counts())
print(df['is_duplicate'].value_counts()/df['is_duplicate'].count()*100)
df['is_duplicate'].value_counts().plot(kind = 'bar')
```

```
0    19013
1    10987
Name: is_duplicate, dtype: int64
0    63.376667
1    36.623333
Name: is_duplicate, dtype: float64
```

Out[9]:
```
<Axes: >
```



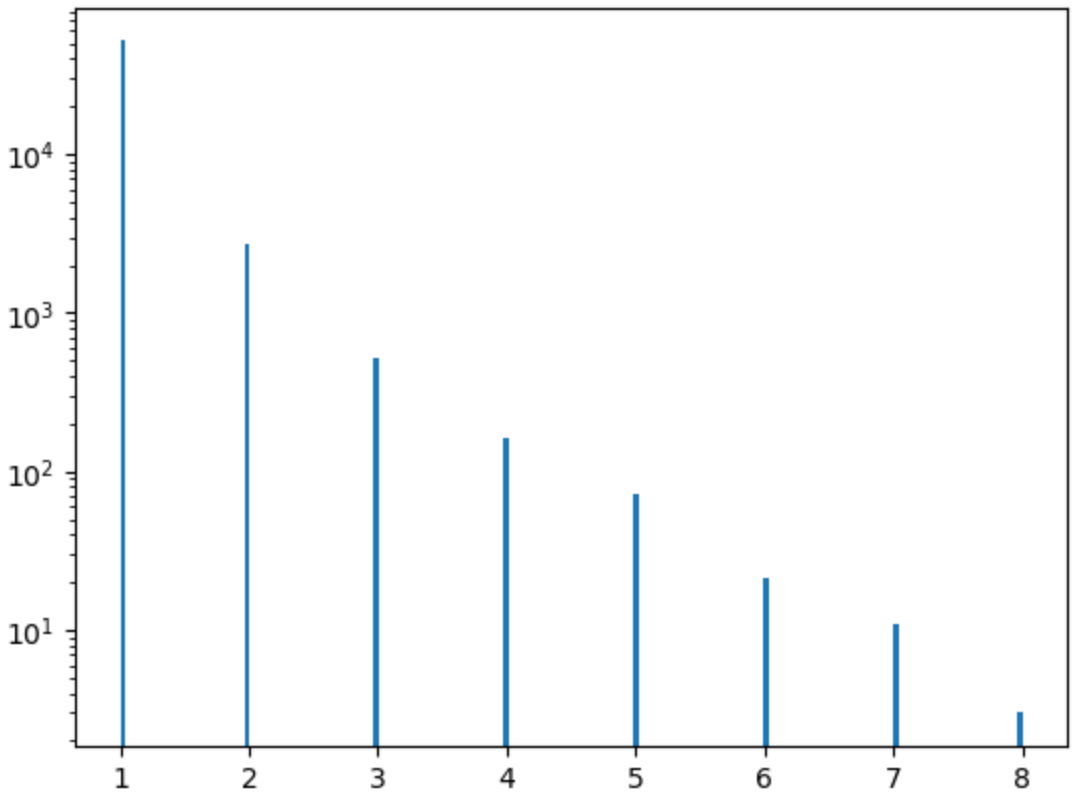
In [10]:
```
# Repeated questions

qid = pd.Series(df['qid1'].tolist() + df['qid2'].tolist())
print('Number of unique questions',np.unique(qid).shape[0])
x = qid.value_counts()>1
print('Number of questions getting repeated',x[x].shape[0])
```

```
Number of unique questions 55299
Number of questions getting repeated 3480
```

In [11]:
```
# Repeated questions histogram

plt.hist(qid.value_counts().values,bins=160)
plt.yscale('log')
plt.show()
```



In [12]:
```
# Feature Engineering

df['q1_len'] = df['question1'].str.len()
df['q2_len'] = df['question2'].str.len()
```

In [13]:
```
1 df.head()
```

Out[13]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len
398782	398782	496695	532029	What is the best marketing automation tool for...	What is the best marketing automation tool for...	1	76	77
115086	115086	187729	187730	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...	0	49	57
327711	327711	454161	454162	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar Univers...	0	105	120
367788	367788	498109	491396	Why do so many people in the U.S. hate the sou...	My boyfriend doesnt feel guilty when he hurts ...	0	59	146
151235	151235	237843	50930	Consequences of Bhopal gas tragedy?	What was the reason behind the Bhopal gas trag...	0	35	50

In [14]:
```
import pandas as pd

# Assuming 'df' is your DataFrame
df['q1_num_words'] = df['question1'].apply(lambda row: len(str(row).split
df['q2_num_words'] = df['question2'].apply(lambda row: len(str(row).split
df.head()

```

Out[14]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nu
398782	398782	496695	532029	What is the best marketing automation tool for...	What is the best marketing automation tool for...	1	76	77	
115086	115086	187729	187730	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...	0	49	57	
327711	327711	454161	454162	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar Univers...	0	105	120	
367788	367788	498109	491396	Why do so many people in the U.S. hate the sou...	My boyfriend doesnt feel guilty when he hurts ...	0	59	146	
151235	151235	237843	50930	Consequences of Bhopal gas tragedy?	What was the reason behind the Bhopal gas trag...	0	35	50	

In [15]:
```
def common_words(row):
    w1 = set(map(lambda word: word.lower().strip(), str(row['question1']))
    w2 = set(map(lambda word: word.lower().strip(), str(row['question2']))
    return len(w1 & w2)

```

In [16]:
```
df['word_common'] = df.apply(common_words, axis=1)
df.head()
```

Out[16]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nu
398782	398782	496695	532029	What is the best marketing automation tool for...	What is the best marketing automation tool for...	1	76	77	
115086	115086	187729	187730	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...	0	49	57	
327711	327711	454161	454162	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar Univers...	0	105	120	
367788	367788	498109	491396	Why do so many people in the U.S. hate the sou...	My boyfriend doesnt feel guilty when he hurts ...	0	59	146	
151235	151235	237843	50930	Consequences of Bhopal gas tragedy?	What was the reason behind the Bhopal gas trag...	0	35	50	

In [17]:
```
def total_words(row):
    w1 = set(map(lambda word: word.lower().strip(), str(row['question1']))
    w2 = set(map(lambda word: word.lower().strip(), str(row['question2']))
    return (len(w1) + len(w2))
```

In [18]:
```
df['word_total'] = df.apply(total_words, axis=1)
df.head()
```

Out[18]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nu
398782	398782	496695	532029	What is the best marketing automation tool for...	What is the best marketing automation tool for...	1	76	77	
115086	115086	187729	187730	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...	0	49	57	
327711	327711	454161	454162	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar Univers...	0	105	120	
367788	367788	498109	491396	Why do so many people in the U.S. hate the sou...	My boyfriend doesnt feel guilty when he hurts ...	0	59	146	
151235	151235	237843	50930	Consequences of Bhopal gas tragedy?	What was the reason behind the Bhopal gas trag...	0	35	50	

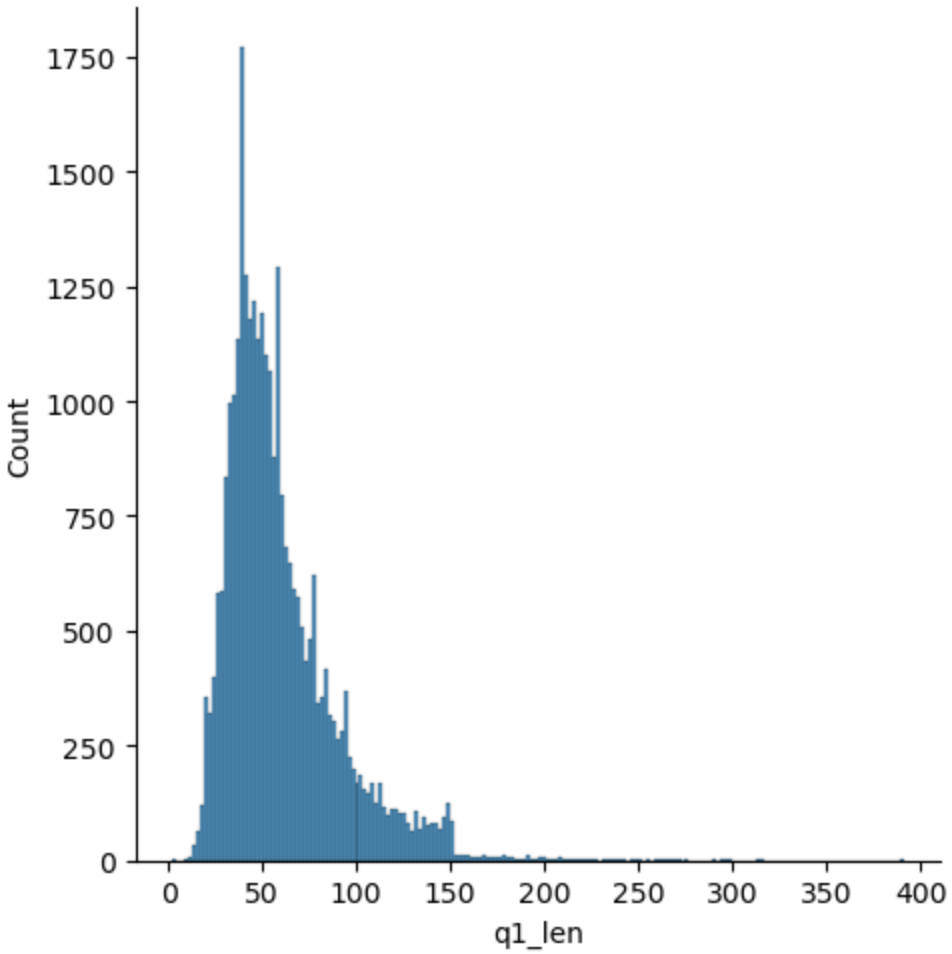
In [19]:
```
df['word_share'] = round(df['word_common']/df['word_total'],2)
df.head()
```

Out[19]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nu
398782	398782	496695	532029	What is the best marketing automation tool for...	What is the best marketing automation tool for...	1	76	77	
115086	115086	187729	187730	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...	0	49	57	
327711	327711	454161	454162	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar Univers...	0	105	120	
367788	367788	498109	491396	Why do so many people in the U.S. hate the sou...	My boyfriend doesnt feel guilty when he hurts ...	0	59	146	
151235	151235	237843	50930	Consequences of Bhopal gas tragedy?	What was the reason behind the Bhopal gas trag...	0	35	50	

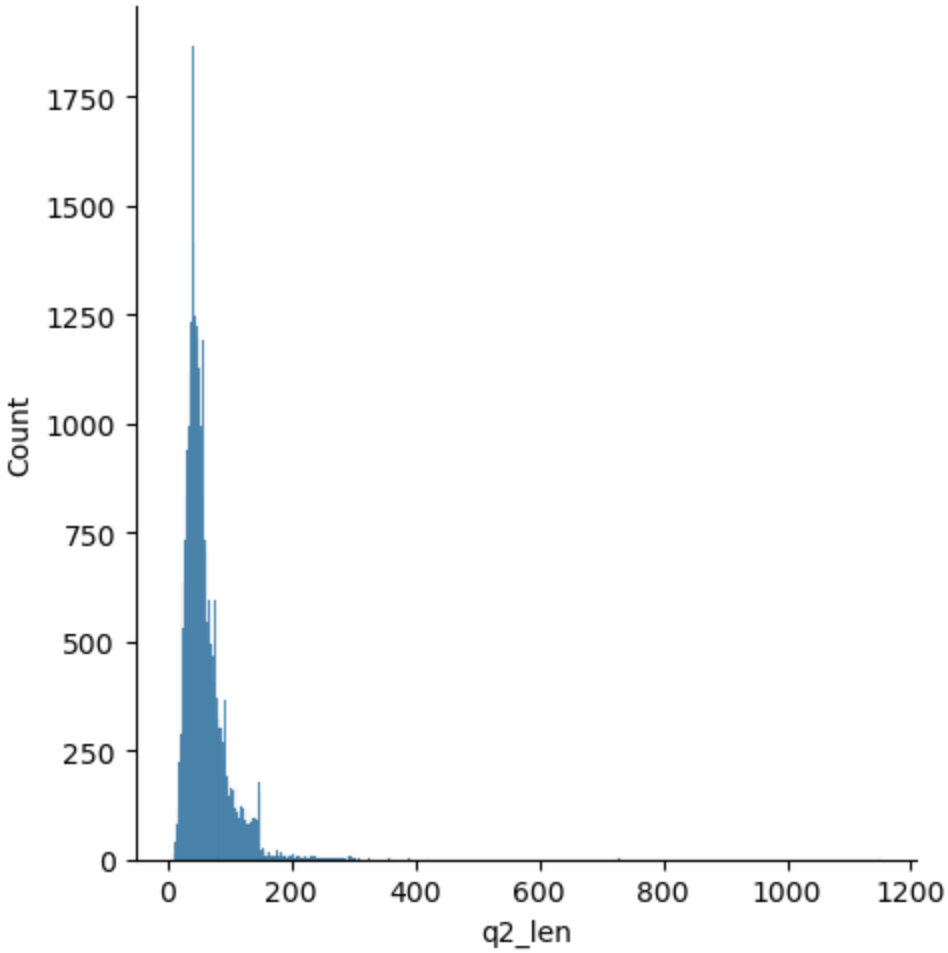
In [20]:
```
# Analysis of features
sns.displot(df['q1_len'])
print('minimum characters',df['q1_len'].min())
print('maximum characters',df['q1_len'].max())
print('average num of characters',int(df['q1_len'].mean()))
```

```
minimum characters 2
maximum characters 391
average num of characters 59
```



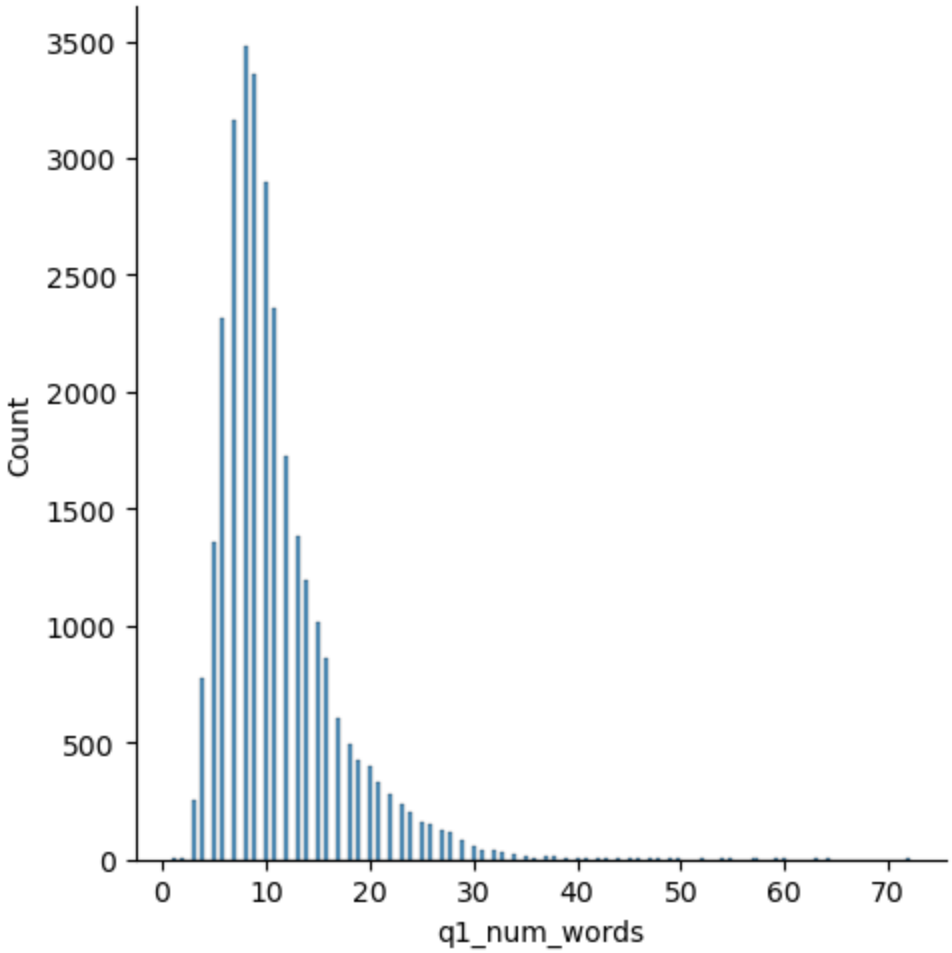
In [21]:
```
sns.displot(df['q2_len'])
print('minimum characters',df['q2_len'].min())
print('maximum characters',df['q2_len'].max())
print('average num of characters',int(df['q2_len'].mean()))
```

```
minimum characters 6
maximum characters 1151
average num of characters 60
```

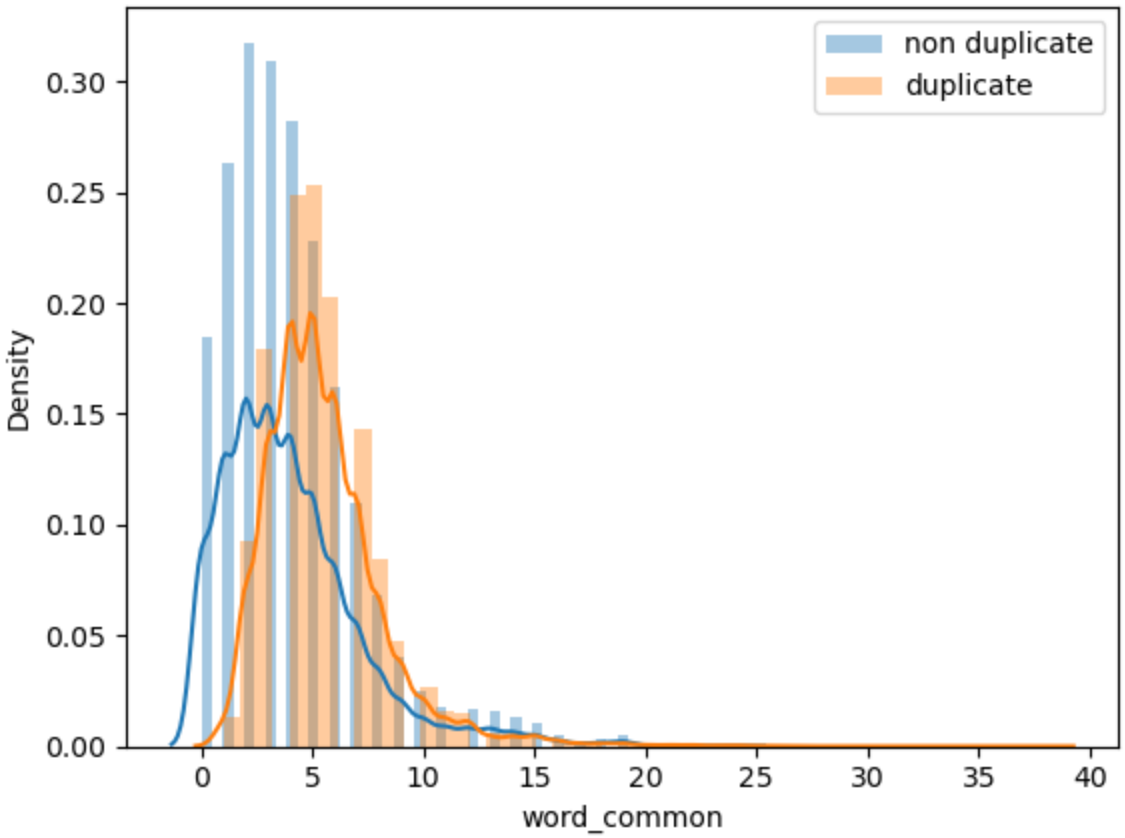


In [22]:
```
sns.displot(df['q1_num_words'])
print('minimum words',df['q1_num_words'].min())
print('maximum words',df['q1_num_words'].max())
print('average num of words',int(df['q1_num_words'].mean()))
```

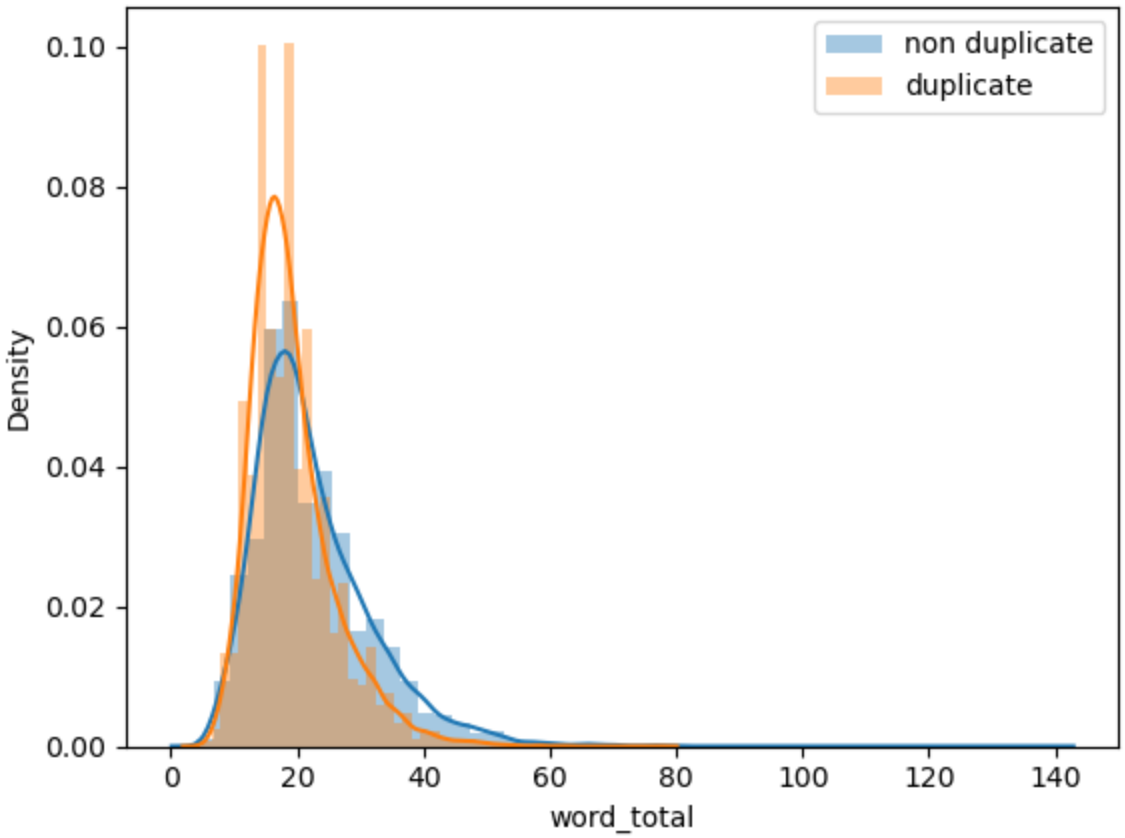
```
minimum words 1
maximum words 72
average num of words 10
```



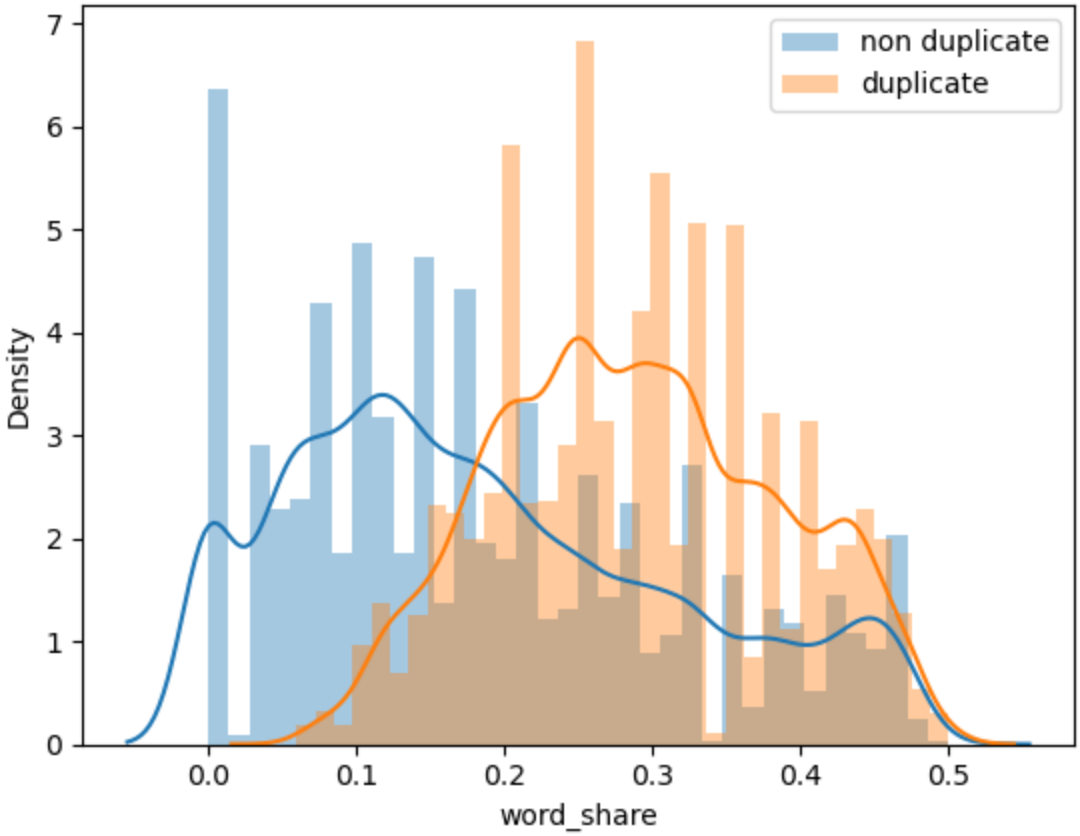
In [23]:
```
# common words
sns.distplot(df[df['is_duplicate'] == 0]['word_common'],label='non duplic
sns.distplot(df[df['is_duplicate'] == 1]['word_common'],label='duplicate'
plt.legend()
plt.show()
```



In [24]:
```
# total words
sns.distplot(df[df['is_duplicate'] == 0]['word_total'],label='non duplica
sns.distplot(df[df['is_duplicate'] == 1]['word_total'],label='duplicate')
plt.legend()
plt.show()
```



In [25]:
```
# word share
sns.distplot(df[df['is_duplicate'] == 0]['word_share'],label='non duplica
sns.distplot(df[df['is_duplicate'] == 1]['word_share'],label='duplicate')
plt.legend()
plt.show()
```



In [26]:
```
ques_df = df[['question1','question2']]
ques_df.head()
```

Out[26]:

	question1	question2
398782	What is the best marketing automation tool for...	What is the best marketing automation tool for...
115086	I am poor but I want to invest. What should I do?	I am quite poor and I want to be very rich. Wh...
327711	I am from India and live abroad. I met a guy f...	T.I.E.T to Thapar University to Thapar Univers...
367788	Why do so many people in the U.S. hate the sou...	My boyfriend doesnt feel guilty when he hurts ...
151235	Consequences of Bhopal gas tragedy?	What was the reason behind the Bhopal gas trag...

In [27]:
```
final_df = df.drop(columns=['id','qid1','qid2','question1','question2'])
print(final_df.shape)
final_df.head()
```

```
(30000, 8)
```

Out[27]:

	is_duplicate	q1_len	q2_len	q1_num_words	q2_num_words	word_common	word_total
398782	1	76	77	12	12	11	24
115086	0	49	57	12	15	7	23
327711	0	105	120	25	17	2	34
367788	0	59	146	12	30	0	32
151235	0	35	50	5	9	3	13

In [28]:
```
from sklearn.feature_extraction.text import CountVectorizer
# merge texts
questions = list(ques_df['question1']) + list(ques_df['question2'])

cv = CountVectorizer(max_features=3000)
q1_arr, q2_arr = np.vsplit(cv.fit_transform(questions).toarray(),2)
```

In [29]:
```
temp_df1 = pd.DataFrame(q1_arr, index= ques_df.index)
temp_df2 = pd.DataFrame(q2_arr, index= ques_df.index)
temp_df = pd.concat([temp_df1, temp_df2], axis=1)
temp_df.shape
```

Out[29]: (30000, 6000)

In [30]:
```
final_df = pd.concat([final_df, temp_df], axis=1)
print(final_df.shape)
final_df.head()
```

```
(30000, 6008)
```

Out[30]:

	is_duplicate	q1_len	q2_len	q1_num_words	q2_num_words	word_common	word_total
398782	1	76	77	12	12	11	24
115086	0	49	57	12	15	7	23
327711	0	105	120	25	17	2	34
367788	0	59	146	12	30	0	32
151235	0	35	50	5	9	3	13

In [31]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(final_df.iloc[:,1:].valu

```

RandomForestClassifier

In [33]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[33]: 0.7716666666666666

XGBClassifier

In [34]:
```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
y_pred = xgb.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[34]: 0.7661666666666667

Advanced Features

In [35]:
```
import re
from bs4 import BeautifulSoup
```

In [36]:

```
def preprocess(q):

    q = str(q).lower().strip()

    # Replace certain special characters with their string equivalents
    q = q.replace('%', ' percent')
    q = q.replace('$', ' dollar ')
    q = q.replace('₹', ' rupee ')
    q = q.replace('€', ' euro ')
    q = q.replace('@', ' at ')

    # The pattern '[math]' appears around 900 times in the whole dataset.
    q = q.replace('[math]', '')

    # Replacing some numbers with string equivalents (not perfect, can be
    q = q.replace(',000,000,000 ', 'b ')
    q = q.replace(',000,000 ', 'm ')
    q = q.replace(',000 ', 'k ')
    q = re.sub(r'([0-9]+)000000000', r'\1b', q)
    q = re.sub(r'([0-9]+)000000', r'\1m', q)
    q = re.sub(r'([0-9]+)000', r'\1k', q)

    # Decontracting words
    # https://en.wikipedia.org/wiki/Wikipedia%3aList_of_English_contracti
    # https://stackoverflow.com/a/19794953
    contractions = {
    "ain't": "am not",
    "aren't": "are not",
    "can't": "can not",
    "can't've": "can not have",
    "'cause": "because",
    "could've": "could have",
    "couldn't": "could not",
    "couldn't've": "could not have",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "hadn't": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "haven't": "have not",
    "he'd": "he would",
    "he'd've": "he would have",
    "he'll": "he will",
    "he'll've": "he will have",
    "he's": "he is",
    "how'd": "how did",
    "how'd'y": "how do you",
    "how'll": "how will",
    "how's": "how is",
    "i'd": "i would",
    "i'd've": "i would have",
    "i'll": "i will",
    "i'll've": "i will have",
    "i'm": "i am",
    "i've": "i have",
    "isn't": "is not",
```

```
    "it'd": "it would",
    "it'd've": "it would have",
    "it'll": "it will",
    "it'll've": "it will have",
    "it's": "it is",
    "let's": "let us",
    "ma'am": "madam",
    "mayn't": "may not",
    "might've": "might have",
    "mightn't": "might not",
    "mightn't've": "might not have",
    "must've": "must have",
    "mustn't": "must not",
    "mustn't've": "must not have",
    "needn't": "need not",
    "needn't've": "need not have",
    "o'clock": "of the clock",
    "oughtn't": "ought not",
    "oughtn't've": "ought not have",
    "shan't": "shall not",
    "sha'n't": "shall not",
    "shan't've": "shall not have",
    "she'd": "she would",
    "she'd've": "she would have",
    "she'll": "she will",
    "she'll've": "she will have",
    "she's": "she is",
    "should've": "should have",
    "shouldn't": "should not",
    "shouldn't've": "should not have",
    "so've": "so have",
    "so's": "so as",
    "that'd": "that would",
    "that'd've": "that would have",
    "that's": "that is",
    "there'd": "there would",
    "there'd've": "there would have",
    "there's": "there is",
    "they'd": "they would",
    "they'd've": "they would have",
    "they'll": "they will",
    "they'll've": "they will have",
    "they're": "they are",
    "they've": "they have",
    "to've": "to have",
    "wasn't": "was not",
    "we'd": "we would",
    "we'd've": "we would have",
    "we'll": "we will",
    "we'll've": "we will have",
    "we're": "we are",
    "we've": "we have",
    "weren't": "were not",
    "what'll": "what will",
    "what'll've": "what will have",
    "what're": "what are",
    "what's": "what is",
```

```
    "what've": "what have",
    "when's": "when is",
    "when've": "when have",
    "where'd": "where did",
    "where's": "where is",
    "where've": "where have",
    "who'll": "who will",
    "who'll've": "who will have",
    "who's": "who is",
    "who've": "who have",
    "why's": "why is",
    "why've": "why have",
    "will've": "will have",
    "won't": "will not",
    "won't've": "will not have",
    "would've": "would have",
    "wouldn't": "would not",
    "wouldn't've": "would not have",
    "y'all": "you all",
    "y'all'd": "you all would",
    "y'all'd've": "you all would have",
    "y'all're": "you all are",
    "y'all've": "you all have",
    "you'd": "you would",
    "you'd've": "you would have",
    "you'll": "you will",
    "you'll've": "you will have",
    "you're": "you are",
    "you've": "you have"
    }

    q_decontracted = []

    for word in q.split():
        if word in contractions:
            word = contractions[word]

        q_decontracted.append(word)

    q = ' '.join(q_decontracted)
    q = q.replace("'ve", " have")
    q = q.replace("n't", " not")
    q = q.replace("'re", " are")
    q = q.replace("'ll", " will")

    # Removing HTML tags
    q = BeautifulSoup(q)
    q = q.get_text()

    # Remove punctuations
    pattern = re.compile('\W')
    q = re.sub(pattern, ' ', q).strip()


    return q
```

In [37]:
```
preprocess("I've already! wasn't <b>done</b>?")
```

Out[37]: 'i have already was not done'

In [38]:
```
df['question1'] = df['question1'].apply(preprocess)
df['question2'] = df['question2'].apply(preprocess)
```

In [39]:
```
df.head()
```

Out[39]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	76	77	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	49	57	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	105	120	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	59	146	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	35	50	

In [40]:
```
df['q1_len'] = df['question1'].str.len()
df['q2_len'] = df['question2'].str.len()
```

In [41]:
```
df['q1_num_words'] = df['question1'].apply(lambda row: len(row.split(" ")
df['q2_num_words'] = df['question2'].apply(lambda row: len(row.split(" ")
df.head()
```

Out[41]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

In [42]:
```
def common_words(row):
    w1 = set(map(lambda word: word.lower().strip(), row['question1'].spli
    w2 = set(map(lambda word: word.lower().strip(), row['question2'].spli
    return len(w1 & w2)
```

In [43]:
```
df['word_common'] = df.apply(common_words, axis=1)
df.head()
```

Out[43]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

In [44]:
```
def total_words(row):
    w1 = set(map(lambda word: word.lower().strip(), row['question1'].spli
    w2 = set(map(lambda word: word.lower().strip(), row['question2'].spli
    return (len(w1) + len(w2))
```

In [45]:
```
df['word_total'] = df.apply(total_words, axis=1)
df.head()
```

Out[45]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

In [46]:
```
df['word_share'] = round(df['word_common']/df['word_total'],2)
df.head()
```

Out[46]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

In [47]:

```
# Advanced Features
from nltk.corpus import stopwords

def fetch_token_features(row):

    q1 = row['question1']
    q2 = row['question2']

    SAFE_DIV = 0.0001

    STOP_WORDS = stopwords.words("english")

    token_features = [0.0]*8

    # Converting the Sentence into Tokens:
    q1_tokens = q1.split()
    q2_tokens = q2.split()

    if len(q1_tokens) == 0 or len(q2_tokens) == 0:
        return token_features

    # Get the non-stopwords in Questions
    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 Questions
    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 Question pair
    common_word_count = len(q1_words.intersection(q2_words))

    # Get the common stopwords from Question pair
    common_stop_count = len(q1_stops.intersection(q2_stops))

    # Get the common Tokens from Question pair
    common_token_count = len(set(q1_tokens).intersection(set(q2_tokens)))


    token_features[0] = common_word_count / (min(len(q1_words), len(q2_wo
    token_features[1] = common_word_count / (max(len(q1_words), len(q2_wo
    token_features[2] = common_stop_count / (min(len(q1_stops), len(q2_st
    token_features[3] = common_stop_count / (max(len(q1_stops), len(q2_st
    token_features[4] = common_token_count / (min(len(q1_tokens), len(q2_
    token_features[5] = common_token_count / (max(len(q1_tokens), len(q2_

    # Last word of both question is same or not
    token_features[6] = int(q1_tokens[-1] == q2_tokens[-1])

    # First word of both question is same or not
    token_features[7] = int(q1_tokens[0] == q2_tokens[0])

    return token_features
```

In [48]:
```
token_features = df.apply(fetch_token_features, axis=1)

df["cwc_min"]       = list(map(lambda x: x[0], token_features))
df["cwc_max"]       = list(map(lambda x: x[1], token_features))
df["csc_min"]       = list(map(lambda x: x[2], token_features))
df["csc_max"]       = list(map(lambda x: x[3], token_features))
df["ctc_min"]       = list(map(lambda x: x[4], token_features))
df["ctc_max"]       = list(map(lambda x: x[5], token_features))
df["last_word_eq"]  = list(map(lambda x: x[6], token_features))
df["first_word_eq"] = list(map(lambda x: x[7], token_features))
```

In [49]:
```
df.head()
```

Out[49]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

In [50]:
```
import distance

def fetch_length_features(row):

    q1 = row['question1']
    q2 = row['question2']

    length_features = [0.0]*3

    # Converting the Sentence into Tokens:
    q1_tokens = q1.split()
    q2_tokens = q2.split()

    if len(q1_tokens) == 0 or len(q2_tokens) == 0:
        return length_features

    # Absolute length features
    length_features[0] = abs(len(q1_tokens) - len(q2_tokens))

    #Average Token Length of both Questions
    length_features[1] = (len(q1_tokens) + len(q2_tokens))/2

    strs = list(distance.lcsubstrings(q1, q2))
    length_features[2] = len(strs[0]) / (min(len(q1), len(q2)) + 1)

    return length_features
```

In [51]:
```
# Apply the fetch_length_features function to each row
length_features = df.apply(fetch_length_features, axis=1)

# Extract features from the length_features list and assign them to new c
df['abs_len_diff'] = list(map(lambda x: x[0], length_features))
df['mean_len'] = list(map(lambda x: x[1], length_features))
df['longest_substr_ratio'] = list(map(lambda x: x[2], length_features))

```

In [52]:
```
1 df.head()
```

Out[52]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

In [53]:
```
# Fuzzy Features
from fuzzywuzzy import fuzz

def fetch_fuzzy_features(row):

    q1 = row['question1']
    q2 = row['question2']

    fuzzy_features = [0.0]*4

    # fuzz_ratio
    fuzzy_features[0] = fuzz.QRatio(q1, q2)

    # fuzz_partial_ratio
    fuzzy_features[1] = fuzz.partial_ratio(q1, q2)

    # token_sort_ratio
    fuzzy_features[2] = fuzz.token_sort_ratio(q1, q2)

    # token_set_ratio
    fuzzy_features[3] = fuzz.token_set_ratio(q1, q2)

    return fuzzy_features
```

In [54]:
```
fuzzy_features = df.apply(fetch_fuzzy_features, axis=1)

# Creating new feature columns for fuzzy features
df['fuzz_ratio'] = list(map(lambda x: x[0], fuzzy_features))
df['fuzz_partial_ratio'] = list(map(lambda x: x[1], fuzzy_features))
df['token_sort_ratio'] = list(map(lambda x: x[2], fuzzy_features))
df['token_set_ratio'] = list(map(lambda x: x[3], fuzzy_features))
```

In [55]:
```
print(df.shape)
df.head()
```

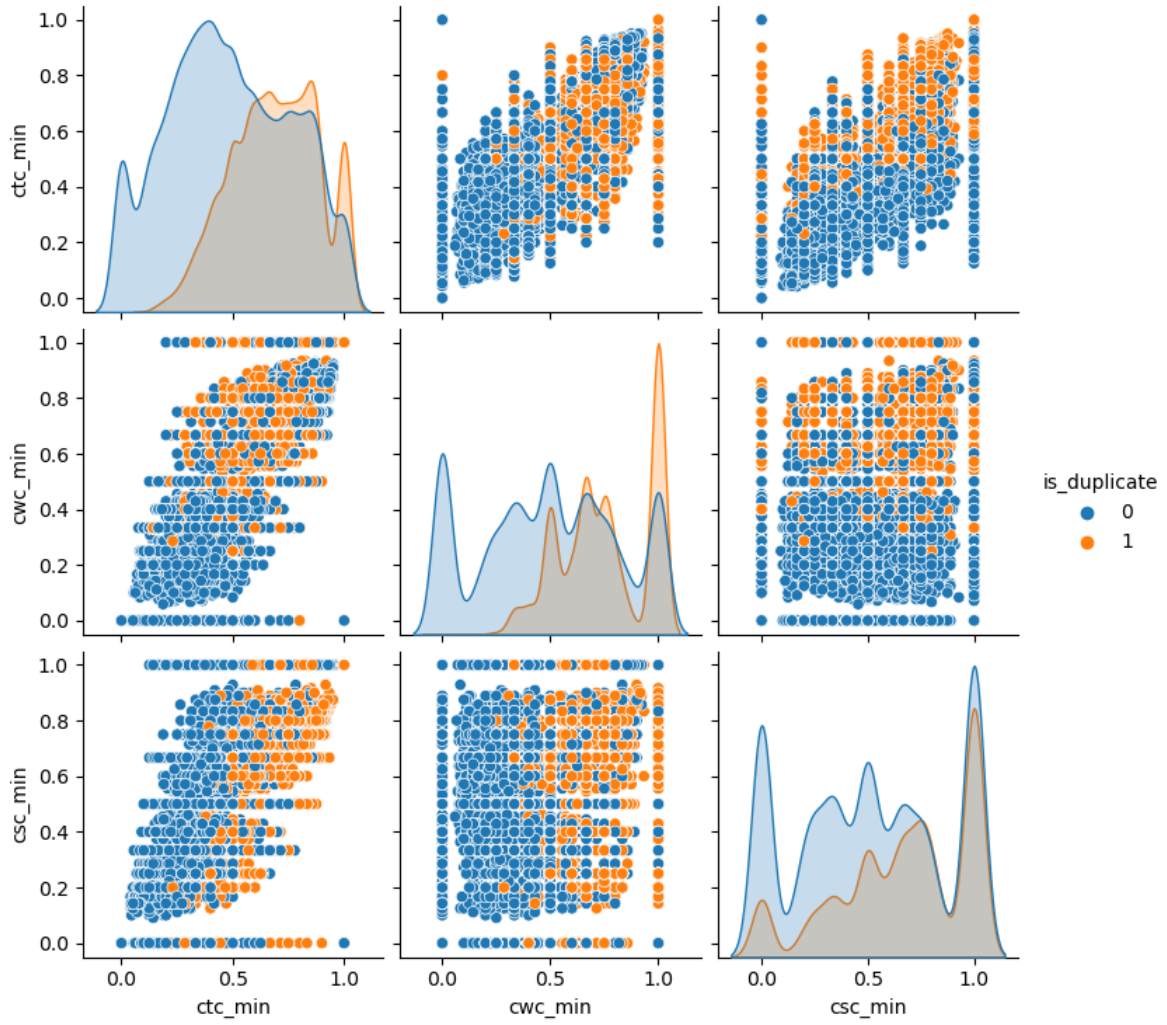
```
(30000, 28)
```

Out[55]:

	id	qid1	qid2	question1	question2	is_duplicate	q1_len	q2_len	q1_nur
398782	398782	496695	532029	what is the best marketing automation tool for...	what is the best marketing automation tool for...	1	75	76	
115086	115086	187729	187730	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...	0	48	56	
327711	327711	454161	454162	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...	0	104	119	
367788	367788	498109	491396	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...	0	58	145	
151235	151235	237843	50930	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy	0	34	49	

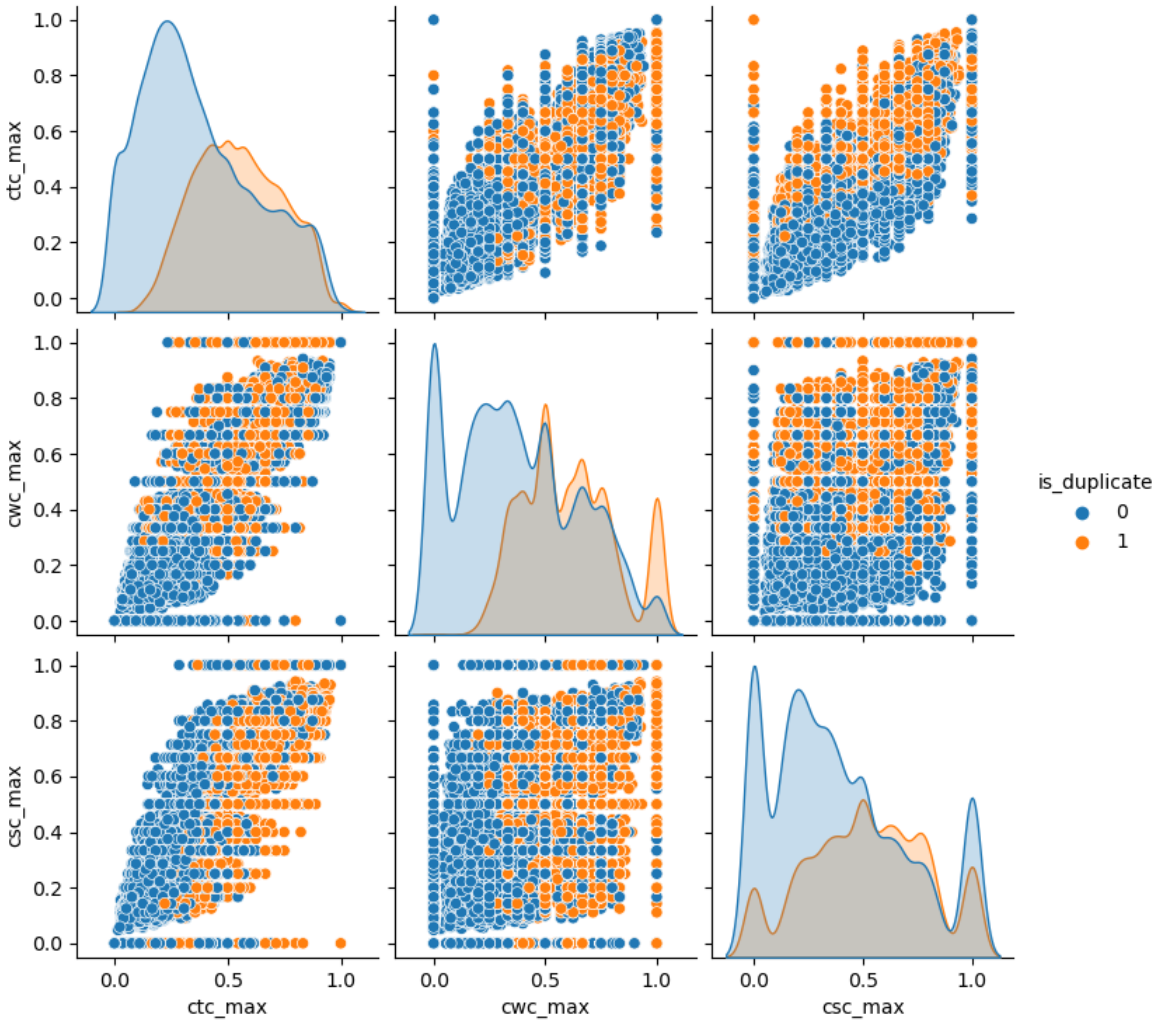
In [56]:
```
1 sns.pairplot(df[['ctc_min', 'cwc_min', 'csc_min', 'is_duplicate']],hue='i
```

Out[56]: <seaborn.axisgrid.PairGrid at 0x2fb06ddbb10>



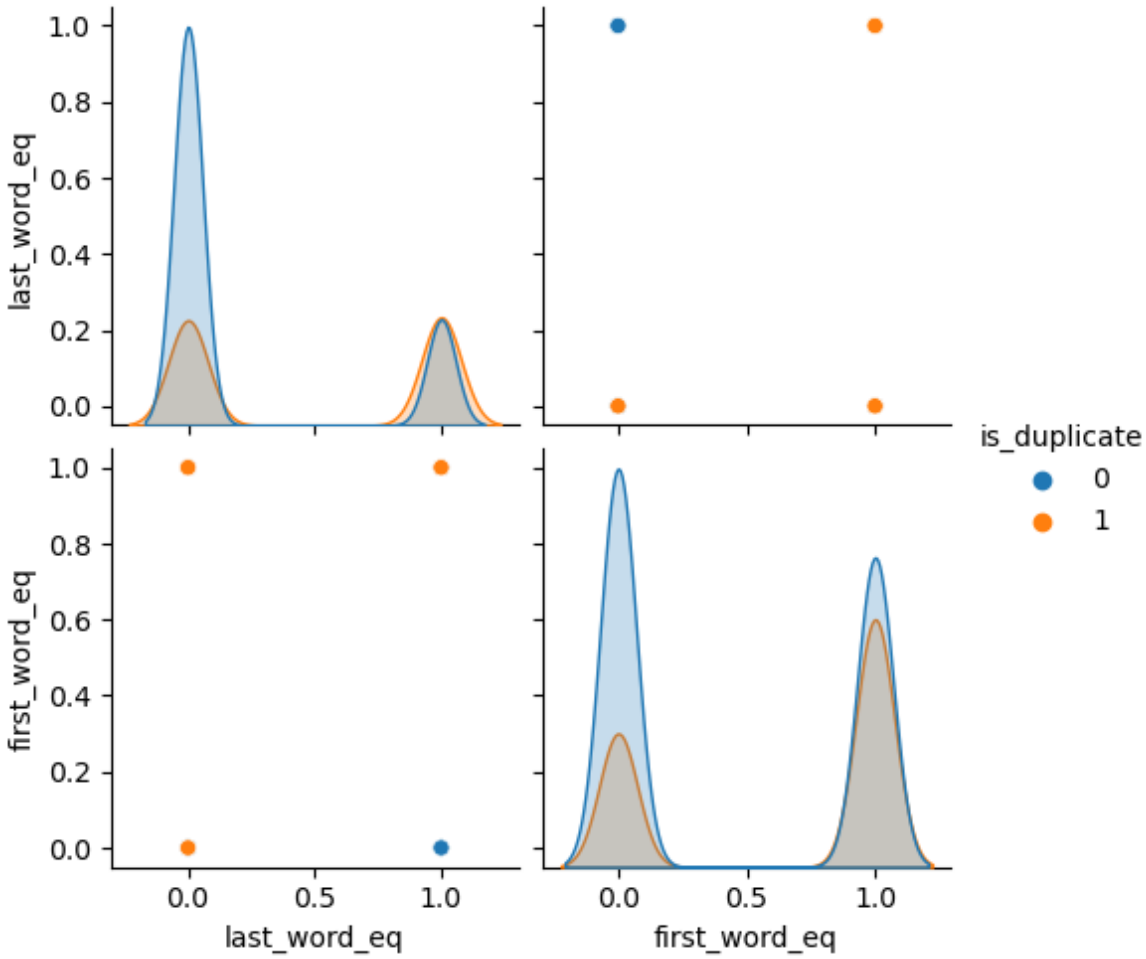
In [57]:
```
1 sns.pairplot(df[['ctc_max', 'cwc_max', 'csc_max', 'is_duplicate']],hue='i
```

Out[57]: <seaborn.axisgrid.PairGrid at 0x2fc584de490>



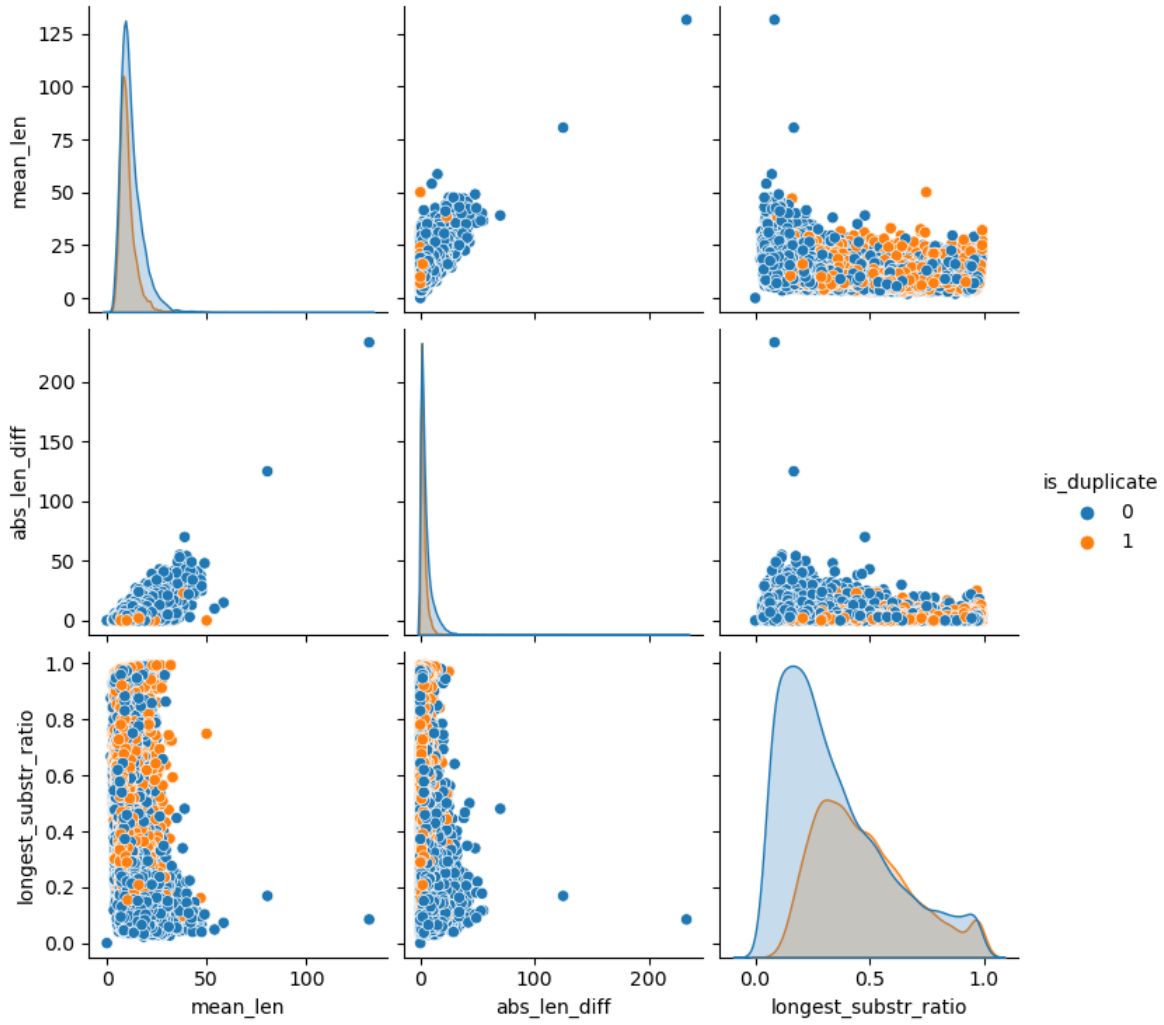
In [58]:
```
1 sns.pairplot(df[['last_word_eq', 'first_word_eq', 'is_duplicate']],hue='i
```

Out[58]: <seaborn.axisgrid.PairGrid at 0x2fb06e1a0d0>



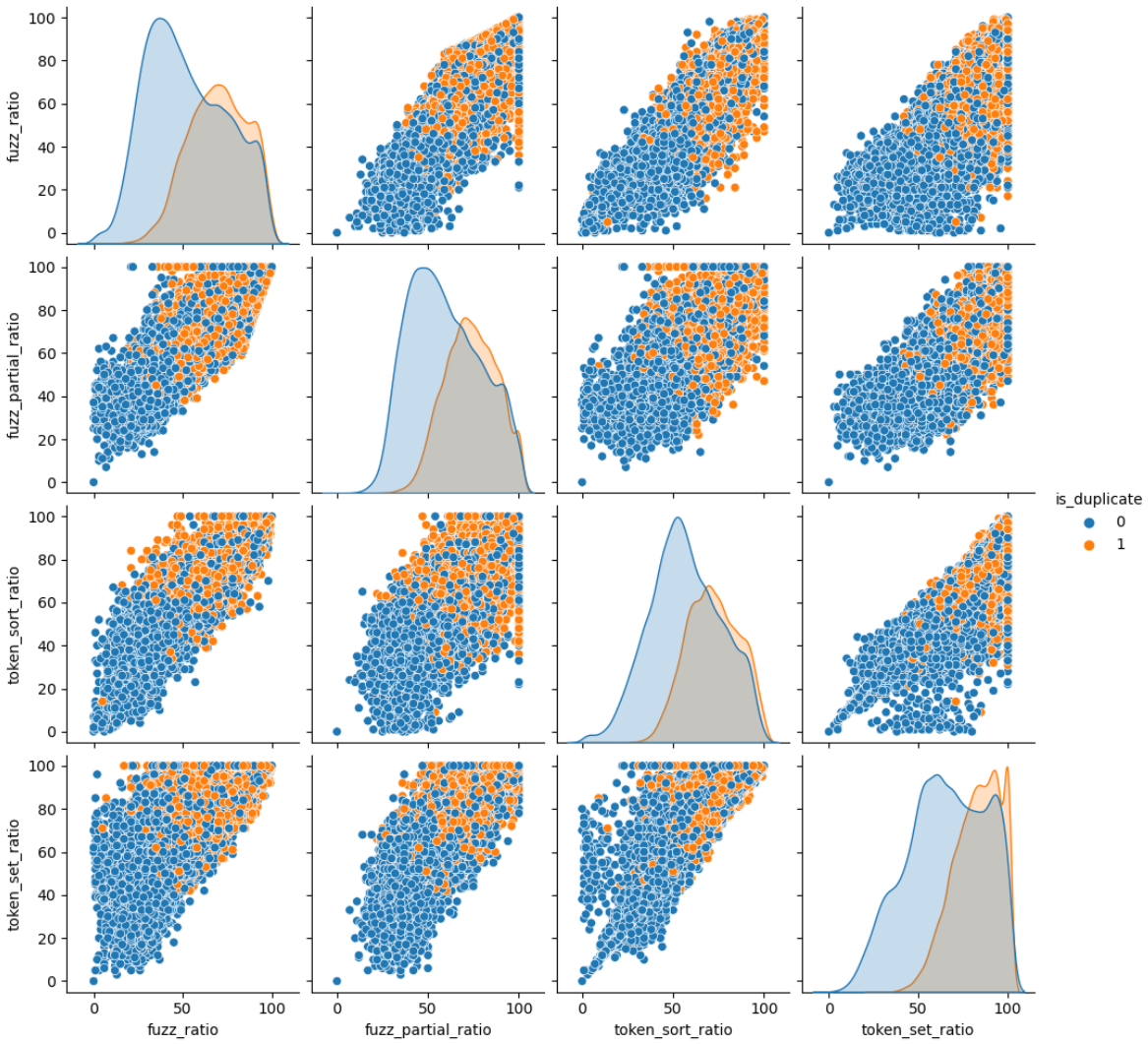
In [59]:
```
1 sns.pairplot(df[['mean_len', 'abs_len_diff','longest_substr_ratio', 'is_d
```

Out[59]: <seaborn.axisgrid.PairGrid at 0x2fb06606490>



In [60]:
```
1 sns.pairplot(df[['fuzz_ratio', 'fuzz_partial_ratio','token_sort_ratio','t
```

Out[60]: <seaborn.axisgrid.PairGrid at 0x2fc5871bd10>



In [61]:
```
1 # Using TSNE for Dimentionality reduction for 15 Features(Generated after
2
3 from sklearn.preprocessing import MinMaxScaler
4
5 X = MinMaxScaler().fit_transform(df[['cwc_min', 'cwc_max', 'csc_min', 'cs
6 y = df['is_duplicate'].values
```

In [62]:

```
from sklearn.manifold import TSNE

tsne2d = TSNE(
    n_components=2,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 30000 samples in 0.068s...
[t-SNE] Computed neighbors for 30000 samples in 5.413s...
[t-SNE] Computed conditional probabilities for sample 1000 / 30000
[t-SNE] Computed conditional probabilities for sample 2000 / 30000
[t-SNE] Computed conditional probabilities for sample 3000 / 30000
[t-SNE] Computed conditional probabilities for sample 4000 / 30000
[t-SNE] Computed conditional probabilities for sample 5000 / 30000
[t-SNE] Computed conditional probabilities for sample 6000 / 30000
[t-SNE] Computed conditional probabilities for sample 7000 / 30000
[t-SNE] Computed conditional probabilities for sample 8000 / 30000
[t-SNE] Computed conditional probabilities for sample 9000 / 30000
[t-SNE] Computed conditional probabilities for sample 10000 / 30000
[t-SNE] Computed conditional probabilities for sample 11000 / 30000
[t-SNE] Computed conditional probabilities for sample 12000 / 30000
[t-SNE] Computed conditional probabilities for sample 13000 / 30000
[t-SNE] Computed conditional probabilities for sample 14000 / 30000
[t-SNE] Computed conditional probabilities for sample 15000 / 30000
[t-SNE] Computed conditional probabilities for sample 16000 / 30000
[t-SNE] Computed conditional probabilities for sample 17000 / 30000
[t-SNE] Computed conditional probabilities for sample 18000 / 30000
[t-SNE] Computed conditional probabilities for sample 19000 / 30000
[t-SNE] Computed conditional probabilities for sample 20000 / 30000
[t-SNE] Computed conditional probabilities for sample 21000 / 30000
[t-SNE] Computed conditional probabilities for sample 22000 / 30000
[t-SNE] Computed conditional probabilities for sample 23000 / 30000
[t-SNE] Computed conditional probabilities for sample 24000 / 30000
[t-SNE] Computed conditional probabilities for sample 25000 / 30000
[t-SNE] Computed conditional probabilities for sample 26000 / 30000
[t-SNE] Computed conditional probabilities for sample 27000 / 30000
[t-SNE] Computed conditional probabilities for sample 28000 / 30000
[t-SNE] Computed conditional probabilities for sample 29000 / 30000
[t-SNE] Computed conditional probabilities for sample 30000 / 30000
[t-SNE] Mean sigma: 0.080413
[t-SNE] Computed conditional probabilities in 0.941s
[t-SNE] Iteration 50: error = 110.0354767, gradient norm = 0.0222168 (50 ite
rations in 15.025s)
[t-SNE] Iteration 100: error = 89.8034515, gradient norm = 0.0077262 (50 ite
rations in 11.897s)
[t-SNE] Iteration 150: error = 85.8413696, gradient norm = 0.0047961 (50 ite
rations in 12.044s)
[t-SNE] Iteration 200: error = 83.9956284, gradient norm = 0.0039413 (50 ite
rations in 10.530s)
[t-SNE] Iteration 250: error = 82.8466949, gradient norm = 0.0029376 (50 ite
rations in 13.284s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 82.84669
5
[t-SNE] Iteration 300: error = 3.3983612, gradient norm = 0.0073742 (50 iter
ations in 9.378s)
[t-SNE] Iteration 350: error = 2.7810087, gradient norm = 0.0073581 (50 iter
ations in 9.022s)
[t-SNE] Iteration 400: error = 2.4468734, gradient norm = 0.0068702 (50 iter
ations in 7.910s)
[t-SNE] Iteration 450: error = 2.2388027, gradient norm = 0.0064638 (50 iter
ations in 7.764s)
[t-SNE] Iteration 500: error = 2.0952334, gradient norm = 0.0061122 (50 iter
ations in 7.882s)
```

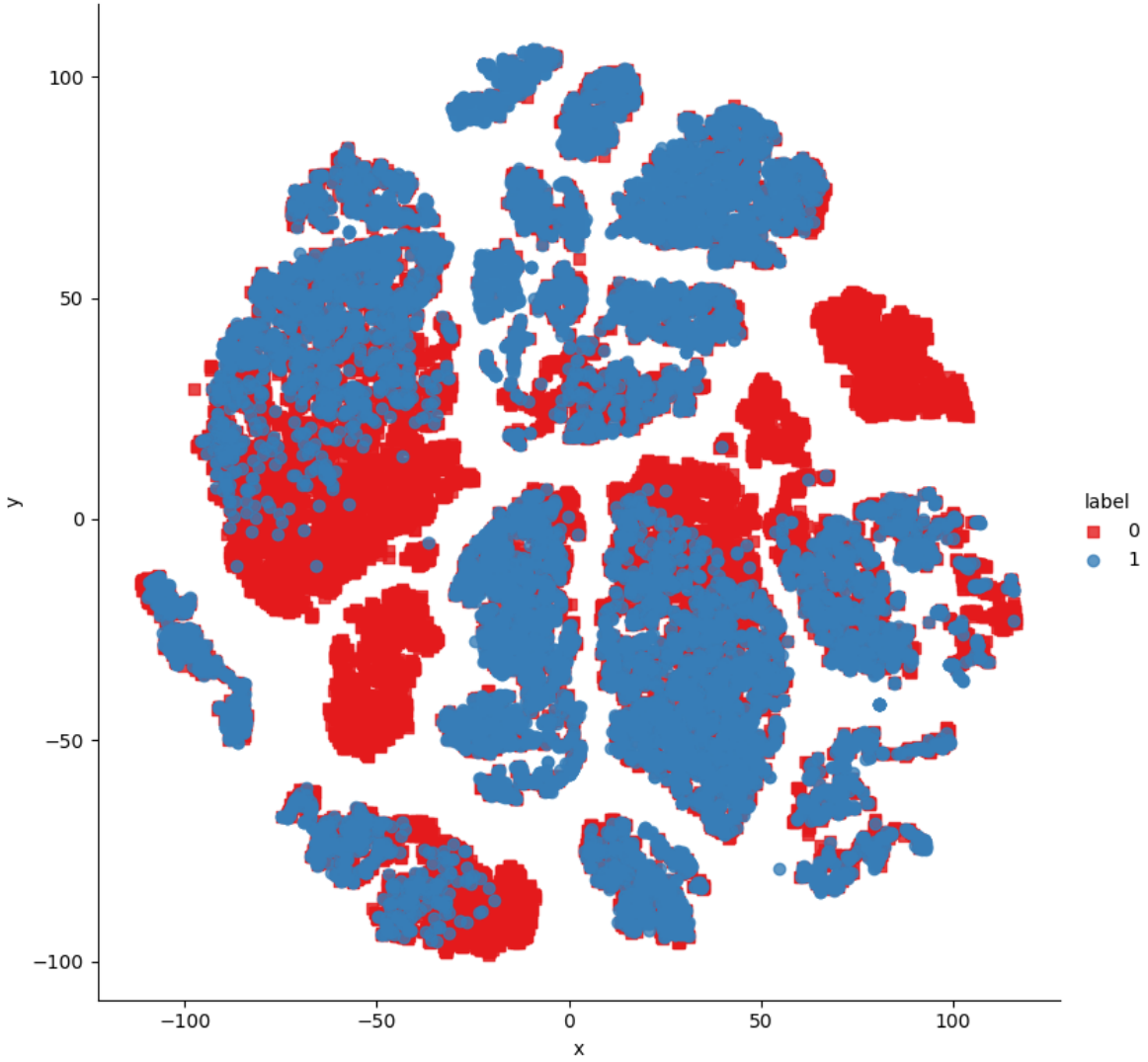
```
[t-SNE] Iteration 550: error = 1.9906163, gradient norm = 0.0057656 (50 iter
ations in 9.103s)
[t-SNE] Iteration 600: error = 1.9118668, gradient norm = 0.0053742 (50 iter
ations in 8.223s)
[t-SNE] Iteration 650: error = 1.8510072, gradient norm = 0.0050041 (50 iter
ations in 7.879s)
[t-SNE] Iteration 700: error = 1.8027258, gradient norm = 0.0046457 (50 iter
ations in 7.946s)
[t-SNE] Iteration 750: error = 1.7636205, gradient norm = 0.0043452 (50 iter
ations in 8.119s)
[t-SNE] Iteration 800: error = 1.7313575, gradient norm = 0.0040350 (50 iter
ations in 8.592s)
[t-SNE] Iteration 850: error = 1.7042049, gradient norm = 0.0037814 (50 iter
ations in 8.236s)
[t-SNE] Iteration 900: error = 1.6811382, gradient norm = 0.0035771 (50 iter
ations in 8.122s)
[t-SNE] Iteration 950: error = 1.6614714, gradient norm = 0.0033116 (50 iter
ations in 8.398s)
[t-SNE] Iteration 1000: error = 1.6444725, gradient norm = 0.0031130 (50 ite
rations in 10.454s)
[t-SNE] KL divergence after 1000 iterations: 1.644472
```

In [63]:
```
x_df = pd.DataFrame({'x': tsne2d[:, 0], 'y': tsne2d[:, 1], 'label': y})

# draw the plot in the appropriate place in the grid
sns.lmplot(data=x_df, x='x', y='y', hue='label', fit_reg=False, height=8,

```

Out[63]: <seaborn.axisgrid.FacetGrid at 0x2fc5cc3a810>



In [64]:

```
tsne3d = TSNE(
    n_components=3,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 30000 samples in 0.071s...
[t-SNE] Computed neighbors for 30000 samples in 4.859s...
[t-SNE] Computed conditional probabilities for sample 1000 / 30000
[t-SNE] Computed conditional probabilities for sample 2000 / 30000
[t-SNE] Computed conditional probabilities for sample 3000 / 30000
[t-SNE] Computed conditional probabilities for sample 4000 / 30000
[t-SNE] Computed conditional probabilities for sample 5000 / 30000
[t-SNE] Computed conditional probabilities for sample 6000 / 30000
[t-SNE] Computed conditional probabilities for sample 7000 / 30000
[t-SNE] Computed conditional probabilities for sample 8000 / 30000
[t-SNE] Computed conditional probabilities for sample 9000 / 30000
[t-SNE] Computed conditional probabilities for sample 10000 / 30000
[t-SNE] Computed conditional probabilities for sample 11000 / 30000
[t-SNE] Computed conditional probabilities for sample 12000 / 30000
[t-SNE] Computed conditional probabilities for sample 13000 / 30000
[t-SNE] Computed conditional probabilities for sample 14000 / 30000
[t-SNE] Computed conditional probabilities for sample 15000 / 30000
[t-SNE] Computed conditional probabilities for sample 16000 / 30000
[t-SNE] Computed conditional probabilities for sample 17000 / 30000
[t-SNE] Computed conditional probabilities for sample 18000 / 30000
[t-SNE] Computed conditional probabilities for sample 19000 / 30000
[t-SNE] Computed conditional probabilities for sample 20000 / 30000
[t-SNE] Computed conditional probabilities for sample 21000 / 30000
[t-SNE] Computed conditional probabilities for sample 22000 / 30000
[t-SNE] Computed conditional probabilities for sample 23000 / 30000
[t-SNE] Computed conditional probabilities for sample 24000 / 30000
[t-SNE] Computed conditional probabilities for sample 25000 / 30000
[t-SNE] Computed conditional probabilities for sample 26000 / 30000
[t-SNE] Computed conditional probabilities for sample 27000 / 30000
[t-SNE] Computed conditional probabilities for sample 28000 / 30000
[t-SNE] Computed conditional probabilities for sample 29000 / 30000
[t-SNE] Computed conditional probabilities for sample 30000 / 30000
[t-SNE] Mean sigma: 0.080413
[t-SNE] Computed conditional probabilities in 0.858s
[t-SNE] Iteration 50: error = 110.5925522, gradient norm = 0.0148754 (50 ite
rations in 33.847s)
[t-SNE] Iteration 100: error = 86.6675262, gradient norm = 0.0046031 (50 ite
rations in 24.531s)
[t-SNE] Iteration 150: error = 83.4906464, gradient norm = 0.0022670 (50 ite
rations in 24.854s)
[t-SNE] Iteration 200: error = 82.2365875, gradient norm = 0.0015197 (50 ite
rations in 25.397s)
[t-SNE] Iteration 250: error = 81.5207214, gradient norm = 0.0011329 (50 ite
rations in 20.254s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 81.52072
1
[t-SNE] Iteration 300: error = 2.9802334, gradient norm = 0.0038841 (50 iter
ations in 27.950s)
[t-SNE] Iteration 350: error = 2.3667295, gradient norm = 0.0030882 (50 iter
ations in 29.651s)
[t-SNE] Iteration 400: error = 2.0555992, gradient norm = 0.0024692 (50 iter
ations in 31.315s)
[t-SNE] Iteration 450: error = 1.8723609, gradient norm = 0.0020611 (50 iter
ations in 29.924s)
[t-SNE] Iteration 500: error = 1.7512760, gradient norm = 0.0017748 (50 iter
ations in 30.053s)
```

```
[t-SNE] Iteration 550: error = 1.6657951, gradient norm = 0.0015336 (50 iter
ations in 27.281s)
[t-SNE] Iteration 600: error = 1.6035148, gradient norm = 0.0013240 (50 iter
ations in 28.066s)
[t-SNE] Iteration 650: error = 1.5577862, gradient norm = 0.0011421 (50 iter
ations in 30.451s)
[t-SNE] Iteration 700: error = 1.5237987, gradient norm = 0.0009500 (50 iter
ations in 28.701s)
[t-SNE] Iteration 750: error = 1.4991173, gradient norm = 0.0007596 (50 iter
ations in 30.606s)
[t-SNE] Iteration 800: error = 1.4807075, gradient norm = 0.0006571 (50 iter
ations in 33.086s)
[t-SNE] Iteration 850: error = 1.4661463, gradient norm = 0.0005565 (50 iter
ations in 36.494s)
[t-SNE] Iteration 900: error = 1.4544582, gradient norm = 0.0004825 (50 iter
ations in 38.770s)
[t-SNE] Iteration 950: error = 1.4449804, gradient norm = 0.0004071 (50 iter
ations in 32.307s)
[t-SNE] Iteration 1000: error = 1.4379606, gradient norm = 0.0003029 (50 ite
rations in 31.926s)
[t-SNE] KL divergence after 1000 iterations: 1.437961
```

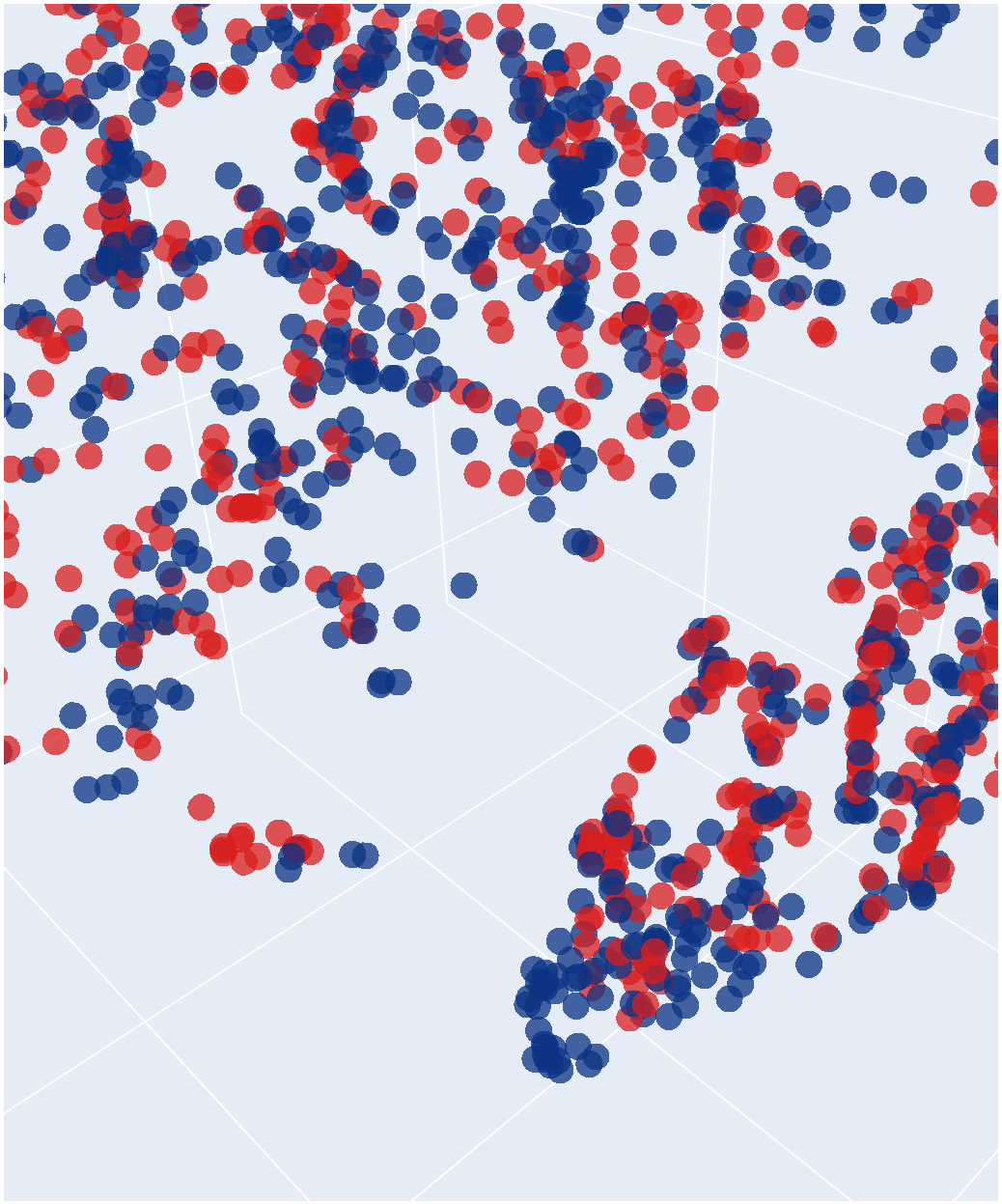
In [65]:

```
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.offline as py
py.init_notebook_mode(connected=True)

trace1 = go.Scatter3d(
    x=tsne3d[:,0],
    y=tsne3d[:,1],
    z=tsne3d[:,2],
    mode='markers',
    marker=dict(
        sizemode='diameter',
        color = y,
        colorscale = 'Portland',
        colorbar = dict(title = 'duplicate'),
        line=dict(color='rgb(255, 255, 255)'),
        opacity=0.75
    )
)

data=[trace1]
layout=dict(height=800, width=800, title='3d embedding with engineered fe
fig=dict(data=data, layout=layout)
py.iplot(fig, filename='3DBubble')
```

3d embedding with engineered features



In [66]:
```
ques_df = df[['question1','question2']]
ques_df.head()
```

Out[66]:

	question1	question2
398782	what is the best marketing automation tool for...	what is the best marketing automation tool for...
115086	i am poor but i want to invest what should i do	i am quite poor and i want to be very rich wh...
327711	i am from india and live abroad i met a guy f...	t i e t to thapar university to thapar univers...
367788	why do so many people in the u s hate the sou...	my boyfriend doesnt feel guilty when he hurts ...
151235	consequences of bhopal gas tragedy	what was the reason behind the bhopal gas tragedy

In [67]:
```
final_df = df.drop(columns=['id','qid1','qid2','question1','question2'])
print(final_df.shape)
final_df.head()
```

```
(30000, 23)
```

Out[67]:

	is_duplicate	q1_len	q2_len	q1_num_words	q2_num_words	word_common	word_total
398782	1	75	76	13	13	12	26
115086	0	48	56	13	16	8	24
327711	0	104	119	28	21	4	38
367788	0	58	145	14	32	1	34
151235	0	34	49	5	9	3	13

In [68]:
```
from sklearn.feature_extraction.text import CountVectorizer
# merge texts
questions = list(ques_df['question1']) + list(ques_df['question2'])

cv = CountVectorizer(max_features=3000)
q1_arr, q2_arr = np.vsplit(cv.fit_transform(questions).toarray(),2)
```

In [69]:
```
temp_df1 = pd.DataFrame(q1_arr, index= ques_df.index)
temp_df2 = pd.DataFrame(q2_arr, index= ques_df.index)
temp_df = pd.concat([temp_df1, temp_df2], axis=1)
temp_df.shape
```

Out[69]: (30000, 6000)

In [70]:
```
final_df = pd.concat([final_df, temp_df], axis=1)
print(final_df.shape)
final_df.head()
```

```
(30000, 6023)
```

Out[70]:

	is_duplicate	q1_len	q2_len	q1_num_words	q2_num_words	word_common	word_total
398782	1	75	76	13	13	12	26
115086	0	48	56	13	16	8	24
327711	0	104	119	28	21	4	38
367788	0	58	145	14	32	1	34
151235	0	34	49	5	9	3	13

In [71]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(final_df.iloc[:,1:].valu

```

KNN

In [72]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train, y_train)
KNeighborsClassifier(n_neighbors=3)
y_pred = knn_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

Out[72]: 0.6956666666666667

Decision Tree

In [73]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
DecisionTreeClassifier()
y_pred = clf.predict(X_test)
accuracy = metrics.accuracy_score(y_test, y_pred)
accuracy
```

Out[73]: 0.7313333333333333

In [ ]:
```
1
```

Random Forest

In [74]:
```
1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn.metrics import accuracy_score
3 rf = RandomForestClassifier()
4 rf.fit(X_train,y_train)
5 y_pred = rf.predict(X_test)
6 accuracy_score(y_test,y_pred)
```

Out[74]: 0.7838333333333334

XGBoost (Extreme Gradient Boosting)

In [75]:
```
1 from xgboost import XGBClassifier
2 xgb = XGBClassifier()
3 xgb.fit(X_train,y_train)
4 y_pred1 = xgb.predict(X_test)
5 accuracy_score(y_test,y_pred1)
```

Out[75]: 0.7946666666666666

Boosting Algorithm

In [79]:
```
1  from sklearn.ensemble import AdaBoostClassifier
2  base_classifier = DecisionTreeClassifier(max_depth=3)
3  adaboost_classifier = AdaBoostClassifier(base_classifier,
4  n_estimators=50, random_state=42)
5  adaboost_classifier.fit(X_train, y_train)
6  AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=1),
7  random_state=42)
8  y_pred = adaboost_classifier.predict(X_test)
9  accuracy = metrics.accuracy_score(y_test, y_pred)
10 accuracy
```

Out[79]: 0.7725

Logistic Regression

In [80]:
```
from sklearn import linear_model
lrg = linear_model.LogisticRegression()
lrg.fit(X_train, y_train)
LogisticRegression()
lrg.score(X_test, y_test)
```

Out[80]: 0.7151666666666666

In [84]:
```
from sklearn.metrics import confusion_matrix
```

In [85]:
```
# for random forest model
confusion_matrix(y_test,y_pred)
```

Out[85]:
```
array([[3191,  621],
       [ 744, 1444]], dtype=int64)
```

In [86]:
```
# for xgboost model
confusion_matrix(y_test,y_pred1)
```

Out[86]:
```
array([[3237,  575],
       [ 657, 1531]], dtype=int64)
```