Quora-1.png

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

Useful Links

- Discussions : https://www.kaggle.com/anokas/data-analysis-xgboost-starter-0-35460-lb/comments (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 (https://www.dropbox.com/sh/93968nfnrzh8bp5/AACZdtsApc1QSTQc7X0H3QZ5a?dl=0)
- Blog 1 : https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning (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 (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 Probelm

2.1 Data

2.1.1 Data Overview

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) Diamon
d?","What would happen if the Indian government stole the Kohi
noor (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 Leaning 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 (https://www.kaggle.com/c/quora-question-pairs#evaluation)

Metric(s):

- log-loss : https://www.kaggle.com/wiki/LogarithmicLoss (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 [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]:

```
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

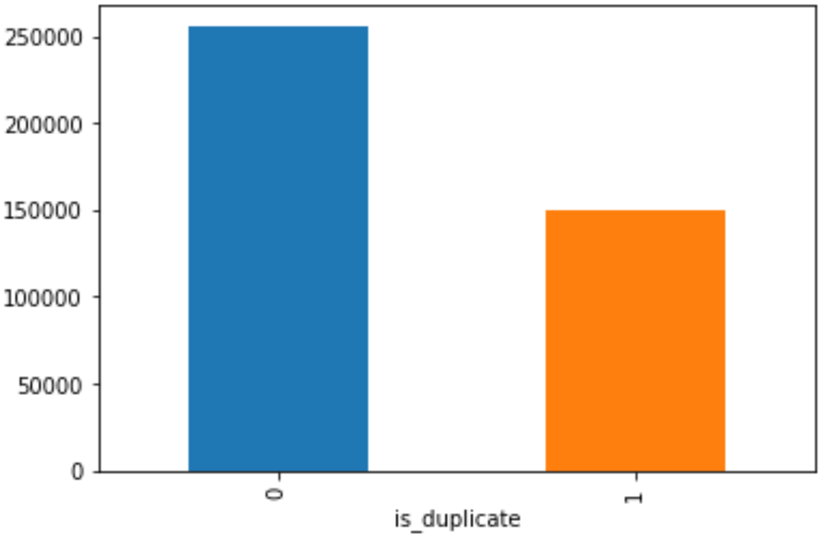
- Number of duplicate(smilar) and non-duplicate(non similar) questions

In [5]:

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

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a254d8eb8>
```



```
In [6]: print('~> Total number of question pairs for training:\n   {}'.format(
```

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

```
In [7]: print('~> Question pairs are not Similar (is_duplicate = 0):\n   {}%'.
        print('\n~> Question pairs are Similar (is_duplicate = 1):\n   {}%'.fo
```

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

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

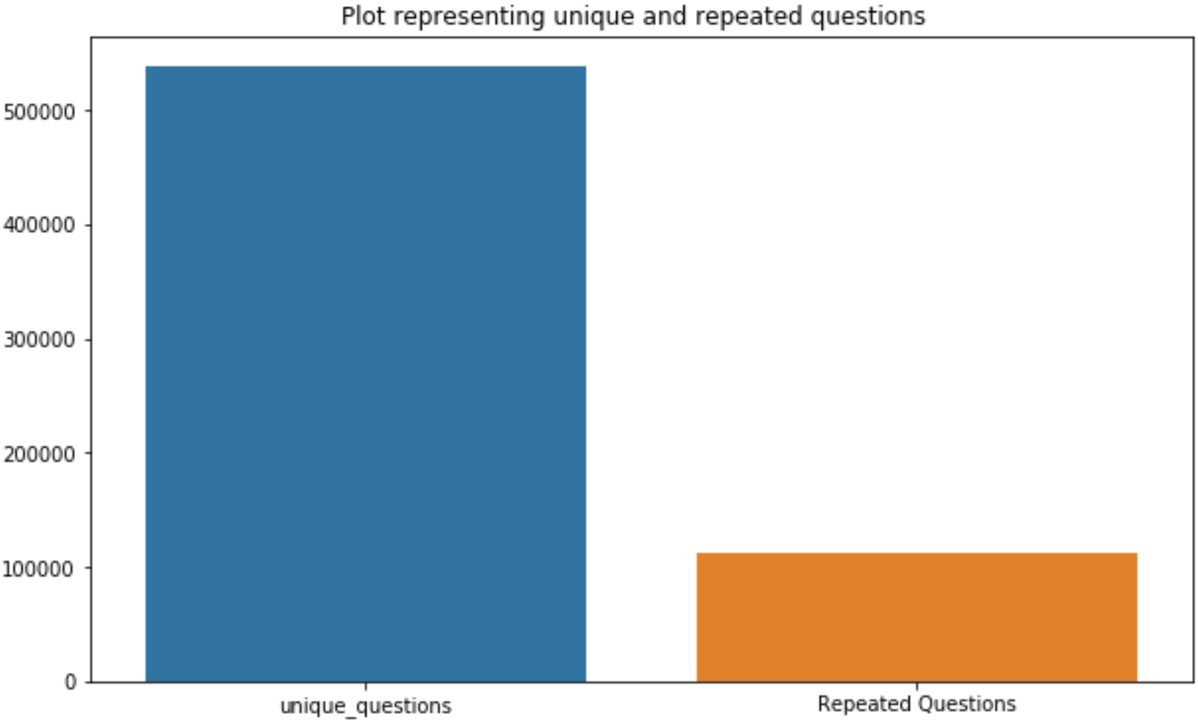
3.2.2 Number of unique questions

```
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: {}

        print ('Max number of times a single question is repeated: {}\n'.forma

        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 (2
0.77953945937505%)

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','

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

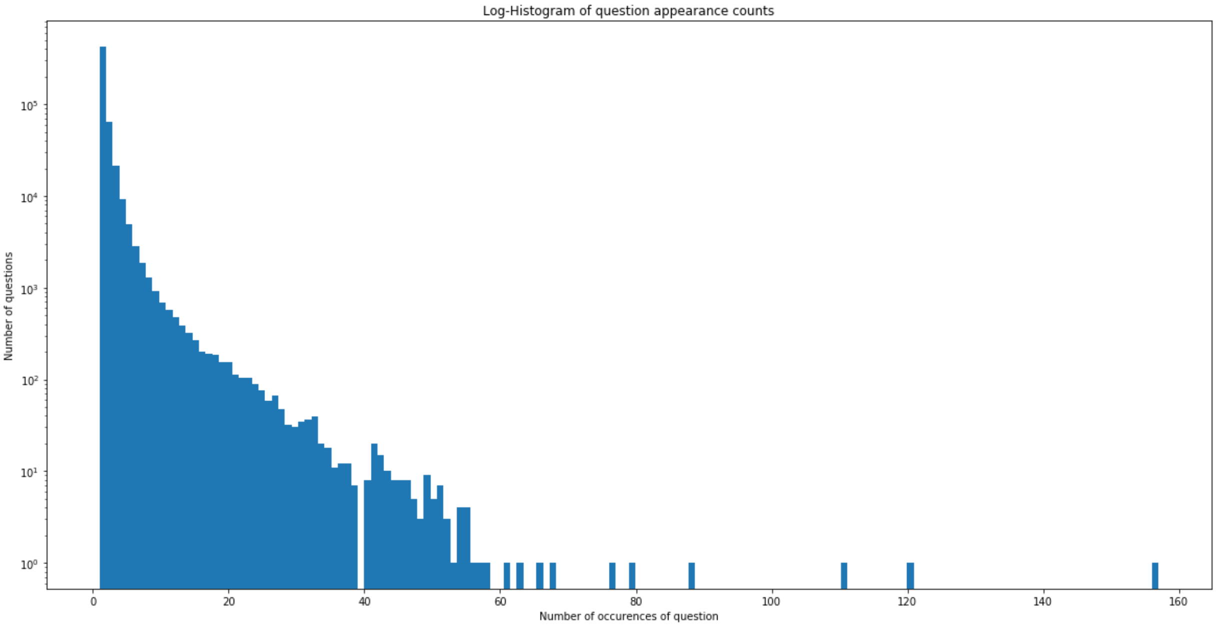
```
Number of duplicate questions 0
```

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 occurences of question')

plt.ylabel('Number of questions')

print ('Maximum number of times a single question is repeated: {}\n'.f
```

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



3.2.5 Checking for NULL values

In [12]:

```
df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
```

Out[12]:

	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
				Why am I mentally very	Find the remainder	

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
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and c...	I'm a triple Capricorn (Sun, Moon and ascendan...	1
6	6	13	14	Should I buy tiago?	What keeps childern active and far from phone ...	0
7	7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1
8	8	17	18	When do you use シ instead of し?	When do you use "&" instead of "and"?	0
9	9	19	20	Motorola (company): Can I hack my Charter Moto...	How do I hack Motorola DCX3400 for free internet?	0
10	10	21	22	Method to find separation of slits using fresn...	What are some of the things technicians can te...	0
11	11	23	24	How do I read and find my YouTube comments?	How can I see all my Youtube comments?	1
12	12	25	26	What can make Physics easy to learn?	How can you make physics easy to learn?	1
13	13	27	28	What was your first sexual experience like?	What was your first sexual experience?	1
14	14	29	30	What are the laws to change your status from a...	What are the laws to change your status from a...	0
15	15	31	32	What would a Trump presidency mean for current...	How will a Trump presidency affect the student...	1
16	16	33	34	What does manipulation mean?	What does manipulation means?	1
17	17	35	36	Why do girls want to be friends with the guy t...	How do guys feel after rejecting a girl?	0
18	18	37	38	Why are so many Quora users posting questions ...	Why do people ask Quora questions which can be...	1
19	19	39	40	Which is the best digital marketing institutio...	Which is the best digital marketing institute ...	0
20	20	41	42	Why do rockets look white?	Why are rockets and boosters painted white?	1
21	21	43	44	What's causing someone to be jealous?	What can I do to avoid being jealous of someone?	0

				someone to be jealous?	someone?	
22	22	45	46	What are the questions should not ask on Quora?	Which question should I ask on Quora?	0
23	23	47	48	How much is 30 kV in HP?	Where can I find a conversion chart for CC to ...	0
24	24	49	50	What does it mean that every time I look at th...	How many times a day do a clock’s hands overlap?	0
25	25	51	52	What are some tips on making it through the jo...	What are some tips on making it through the jo...	0
26	26	53	54	What is web application?	What is the web application framework?	0
27	27	55	56	Does society place too much importance on sports?	How do sports contribute to the society?	0
28	28	57	58	What is best way to make money online?	What is best way to ask for money online?	0
29	29	59	60	How should I prepare for CA final law?	How one should know that he/she completely pre...	1
...	...	...	...	...	...	...
404260	404260	182494	691	Which phone is best under 12000?	What is the best phone to buy below 15k?	0
404261	404261	281150	124172	Who is the overall most popular Game of Throne...	Who is the most popular character in the Game ...	1
404262	404262	537905	466328	How do you troubleshoot a Toshiba laptop?	How do I reset a Toshiba laptop?	0
404263	404263	375195	537906	How does the burning of fossil fuels contribut...	Why does CO2 contribute more to global warming...	0
404264	404264	537907	537908	Is it safe to store an external battery power ...	How do I make a safe and cheap power bank?	0
404265	404265	25994	16064	How can I gain weight on my body?	What should I eat to gain weight?	1
404266	404266	155813	146284	What is the green dot next to the phone icon o...	My boyfriend says he deleted his Facebook Mess...	0
404267	404267	20171	290649	What are the causes of the fall of the Roman E...	What were the most important causes and effect...	1
404268	404268	537909	537910	Why don't we still do great music like in the ...	Should I raise my young child on 80's music?	0
				[illegible]	What Does it Feel Li	

404269	404269	537911	349794	How do you diagnose antisocial personality dis...	What Does It Feel Like to have antisocial pers...	0
404270	404270	537912	35364	What is the difference between who and how?	What is the difference between "&" and "and"?	0
404271	404271	537913	537914	Does Stalin have any grandchildren that are st...	What was Joseph Stalin's 5 year plan? How did ...	0
404272	404272	128018	14005	What are the best new car products or inventio...	What are some mind-blowing vehicles tools that...	1
404273	404273	537915	537916	What happens if you put milk in a coffee maker?	What would happen if I put milk instead of wat...	1
404274	404274	178643	87385	Will the next generation of parenting change o...	What kind of parents will the next generation ...	1
404275	404275	97922	537917	In accounting, why do we debit expenses and cr...	What is a utilities expense in accounting? How...	0
404276	404276	24305	308365	What is copilotsearch.com?	What is ContenVania.com?	0
404277	404277	355668	537918	What does analytics do?	What are analytical people like?	0
404278	404278	537919	169786	How did you prepare for AIIMS/NEET/AIPMT?	How did you prepare for the AIIMS UG entrance ...	0
404279	404279	537920	537921	What is the minimum time required to build a f...	What is a cheaper and quicker way to build an ...	0
404280	404280	537922	537923	What are some outfit ideas to wear to a frat p...	What are some outfit ideas wear to a frat them...	1
404281	404281	99131	81495	Why is Manaphy childish in Pokémon Ranger and ...	Why is Manaphy annoying in Pokemon ranger and ...	1
404282	404282	1931	16773	How does a long distance relationship work?	How are long distance relationships maintained?	1
404283	404283	537924	537925	What do you think of the removal of the MagSaf...	What will the CPU upgrade to the 2016 Apple Ma...	0
404284	404284	537926	537927	What does Jainism say about homosexuality?	What does Jainism say about Gays and Homosexua...	1
404285	404285	433578	379845	How many keywords are there in the Racket prog...	How many keywords are there in PERL Programmin...	0
404286	404286	18840	155606	Do you believe there is	Is it true that there is life	1

				life after death?	after death?	
404287	404287	537928	537929	What is one coin?	What's this coin?	0
404288	404288	537930	537931	What is the approx annual cost of living while...	I am having little hairfall problem but I want...	0
404289	404289	537932	537933	What is like to have sex with cousin?	What is it like to have sex with your cousin?	0

404287 rows × 6 columns

In [13]:

```
pd.isnull(df).sum()
```

Out[13]:

```
id              0
qid1            0
qid2            0
question1       1
question2       2
is_duplicate    0
dtype: int64
```

In [14]:

```
df=df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=Fals
```

- There are two rows with null values in question2

In [15]:

```
# 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 [16]:

```
if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    df = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding=
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
        w2 = set(map(lambda word: word.lower().strip(), row['question2
        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
        w2 = set(map(lambda word: word.lower().strip(), row['question2
        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
        w2 = set(map(lambda word: word.lower().strip(), row['question2
        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[16]:

	id	qid1	qid2	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q
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	
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	
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	
3	3	7	8	Why am I mentally very lonely? How can I solve...	Find the remainder when [math]23^{24}[/math] i...	0	1	1	50	65	
4	4	9	10	Which one dissolve in water quikly sugar, salt...	Which fish would survive in salt water?	0	3	1	76	39	

3.3.1 Analysis of some of the extracted features

- Here are some questions have only one single words.

In [17]:

```
print ("Minimum length of the questions in question1 : " , min(df['q1_n

print ("Minimum length of the questions in question2 : " , min(df['q2_n

print ("Number of Questions with minimum length [question1] :", df[df[
print ("Number of Questions with minimum length [question2] :", df[df[
```

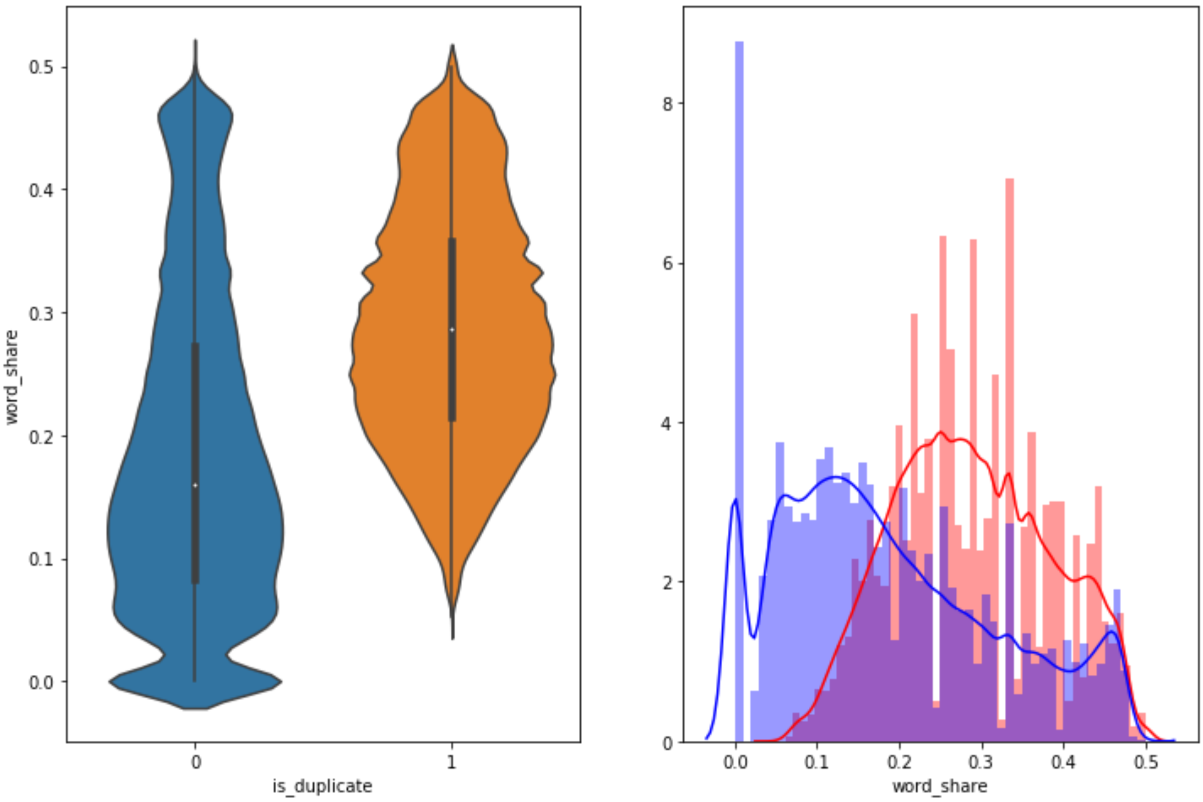
```
Minimum length of the questions in question1 :  1
Minimum length of the questions in question2 :  1
Number of Questions with minimum length [question1] : 67
Number of Questions with minimum length [question2] : 24
```

3.3.1.1 Feature: word_share

In [18]:

```
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 =
sns.distplot(df[df['is_duplicate'] == 0.0]['word_share'][0:] , label =
plt.show()
```

```
/Users/rohitbohra/anaconda3/lib/python3.6/site-packages/scipy/stats/
stats.py:1713: FutureWarning:

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

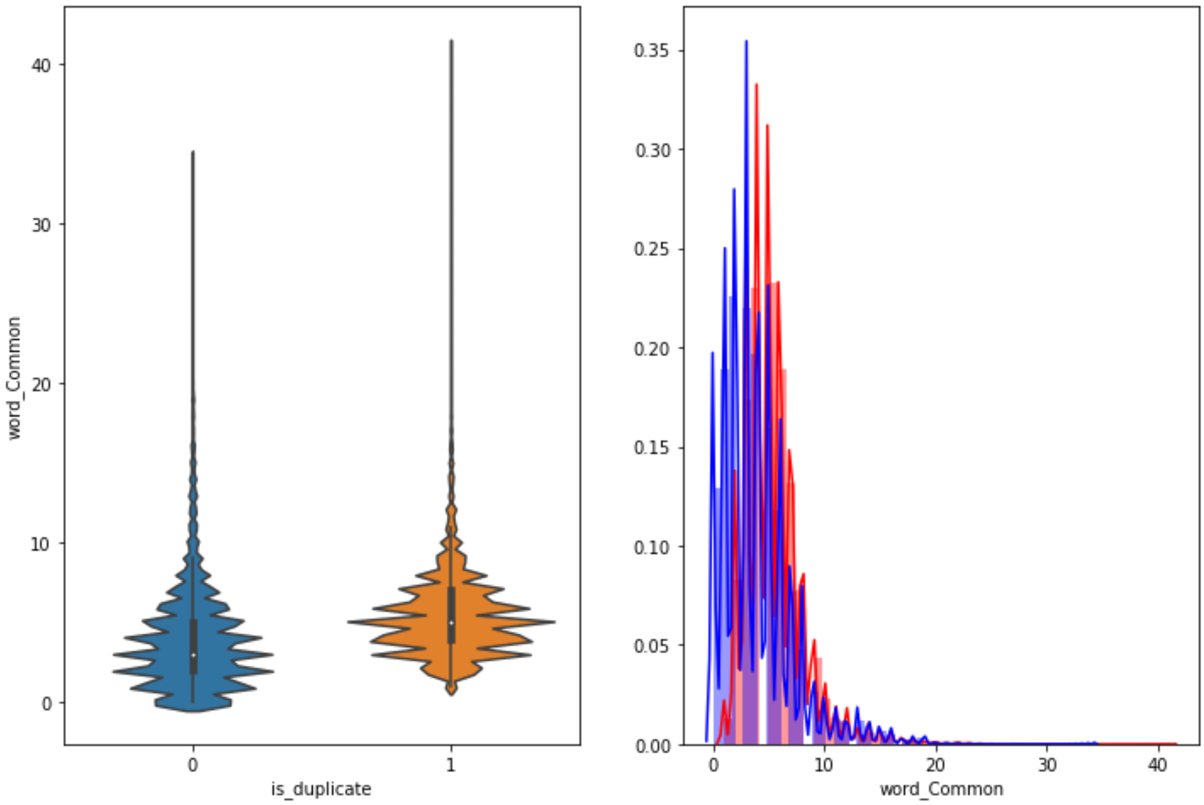


- The distributions for normalized word_share have some overlap on the far right-hand side, i.e., there are quite a lot of questions with high word similarity
- The average word share and Common no. of words of qid1 and qid2 is more when they are duplicate(Similar)

3.3.1.2 Feature: word_Common

In [19]:
```
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
sns.distplot(df[df['is_duplicate'] == 0.0]['word_Common'][0:] , label
plt.show()
```

```
/Users/rohitbohra/anaconda3/lib/python3.6/site-packages/scipy/stats/
stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecat
ed; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this
will be interpreted as an array index, `arr[np.array(seq)]`, which w
ill 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

1.2.1 : EDA: Advanced Feature Extraction.

In [20]:

```
import warnings
warnings.filterwarnings("ignore")
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
import re
from nltk.corpus import stopwords
# This package is used for finding longest common subsequence between
# you can write your own dp code for this
import distance
from nltk.stem import PorterStemmer
from bs4 import BeautifulSoup
from fuzzywuzzy import fuzz
from sklearn.manifold import TSNE
# Import the Required lib packages for WORD-Cloud generation
# https://stackoverflow.com/questions/45625434/how-to-install-wordclou
from wordcloud import WordCloud, STOPWORDS
from os import path
from PIL import Image
```

In [21]:

```
#https://stackoverflow.com/questions/12468179/unicodedecodeerror-utf8-
if os.path.isfile('df_fe_without_preprocessing_train.csv'):
    df = pd.read_csv("df_fe_without_preprocessing_train.csv",encoding=
    df = df.fillna('')
    df.head()
else:
    print("get df_fe_without_preprocessing_train.csv from drive or run
```

In [22]:

```
df.head(2)
```

Out[22]:

	id	qid1	qid2	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_
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	
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	

3.4 Preprocessing of Text

- Preprocessing:
  - Removing html tags
  - Removing Punctuations
  - Performing stemming
  - Removing Stopwords
  - Expanding contractions etc.

In [23]:

```
# To get the results in 4 decemal points
SAFE_DIV = 0.0001

STOP_WORDS = stopwords.words("english")


def preprocess(x):
    x = str(x).lower()
    x = x.replace(",000,000", "m").replace(",000", "k").replace("′", "
                           .replace("won't", "will not").replace("cann
                           .replace("n't", " not").replace("what's", "
                           .replace("'ve", " have").replace("i'm", "i 
                           .replace("he's", "he is").replace("she's", 
                           .replace("%", " percent ").replace("₹", " r
                           .replace("€", " euro ").replace("'ll", " wi
    x = re.sub(r"([0-9]+)000000", r"\1m", x)
    x = re.sub(r"([0-9]+)000", r"\1k", x)


    porter = PorterStemmer()
    pattern = re.compile('\W')

    if type(x) == type(''):
        x = re.sub(pattern, ' ', x)


    if type(x) == type(''):
        x = porter.stem(x)
        example1 = BeautifulSoup(x)
        x = example1.get_text()


    return x
```

- Function to Compute and get the features : With 2 parameters of Question 1 and Question 2

3.5 Advanced Feature Extraction (NLP and Fuzzy Features)

Definition:

- **Token**: You get a token by splitting sentence a space
- **Stop_Word** : stop words as per NLTK.
- **Word** : A token that is not a stop_word

Features:

- **cwc_min** : Ratio of common_word_count to min lenghth of word count of Q1 and Q2
  cwc_min = common_word_count / (min(len(q1_words), len(q2_words))

- **cwc_max** : Ratio of common_word_count to max lenghth of word count of Q1 and Q2
  cwc_max = common_word_count / (max(len(q1_words), len(q2_words))

- **csc_min** : Ratio of common_stop_count to min lenghth of stop count of Q1 and Q2
  csc_min = common_stop_count / (min(len(q1_stops), len(q2_stops))

- **csc_max** : Ratio of common_stop_count to max lenghth of stop count of Q1 and Q2
  csc_max = common_stop_count / (max(len(q1_stops), len(q2_stops))

- **ctc_min** : Ratio of common_token_count to min lenghth of token count of Q1 and Q2
  ctc_min = common_token_count / (min(len(q1_tokens), len(q2_tokens))

- **ctc_max** : Ratio of common_token_count to max lenghth of token count of Q1 and Q2
  ctc_max = common_token_count / (max(len(q1_tokens), len(q2_tokens))

- **last_word_eq** : Check if First word of both questions is equal or not
  last_word_eq = int(q1_tokens[-1] == q2_tokens[-1])

- **first_word_eq** : Check if First word of both questions is equal or not
  first_word_eq = int(q1_tokens[0] == q2_tokens[0])

- **abs_len_diff** : Abs. length difference
  abs_len_diff = abs(len(q1_tokens) - len(q2_tokens))

- **mean_len** : Average Token Length of both Questions
  mean_len = (len(q1_tokens) + len(q2_tokens))/2

- **fuzz_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage
  (https://github.com/seatgeek/fuzzywuzzy#usage)
  http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/
  (http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/)

- **fuzz_partial_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage
  (https://github.com/seatgeek/fuzzywuzzy#usage)
  http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/
  (http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/)

- **token_sort_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage
  (https://github.com/seatgeek/fuzzywuzzy#usage)
  http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/
  (http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/)

- **token_set_ratio** : https://github.com/seatgeek/fuzzywuzzy#usage (https://github.com/seatgeek/fuzzywuzzy#usage) http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/ (http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/)

- **longest_substr_ratio** : Ratio of length longest common substring to min lenghth of token count of Q1 and Q2
  longest_substr_ratio = len(longest common substring) / (min(len(q1_tokens), len(q2_tokens))

In [24]:

```
def get_token_features(q1, q2):
    token_features = [0.0]*10

    # 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_WOR
    q2_words = set([word for word in q2_tokens if word not in STOP_WOR

    #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
    token_features[1] = common_word_count / (max(len(q1_words), len(q2
    token_features[2] = common_stop_count / (min(len(q1_stops), len(q2
    token_features[3] = common_stop_count / (max(len(q1_stops), len(q2
    token_features[4] = common_token_count / (min(len(q1_tokens), len(
    token_features[5] = common_token_count / (max(len(q1_tokens), len(

    # 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])
    token_features[7] = int(q1_tokens[0] == q2_tokens[0])

    token_features[8] = abs(len(q1_tokens) - len(q2_tokens))

    #Average Token Length of both Questions
    token_features[9] = (len(q1_tokens) + len(q2_tokens))/2
    return token_features

# get the Longest Common sub string

def get_longest_substr_ratio(a, b):
    strs = list(distance.lcsubstrings(a, b))
    if len(strs) == 0:
        return 0
    else:
        return len(strs[0]) / (min(len(a), len(b)) + 1)

def extract_features(df):
    # preprocessing each question
    df["question1"] = df["question1"].fillna("").apply(preprocess)
    df["question2"] = df["question2"].fillna("").apply(preprocess)

    print("token features...")

    # Merging Features with dataset

    token_features = df.apply(lambda x: get_token_features(x["question

    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))
    df["abs_len_diff"]  = list(map(lambda x: x[8], token_features))
    df["mean_len"]      = list(map(lambda x: x[9], token_features))

    #Computing Fuzzy Features and Merging with Dataset

    # do read this blog: http://chairnerd.seatgeek.com/fuzzywuzzy-fuzz
    # https://stackoverflow.com/questions/31806695/when-to-use-which-f
    # https://github.com/seatgeek/fuzzywuzzy
    print("fuzzy features..")

    df["token_set_ratio"]       = df.apply(lambda x: fuzz.token_set_ra
    # The token sort approach involves tokenizing the string in questi
    # then joining them back into a string We then compare the transfo
    df["token_sort_ratio"]      = df.apply(lambda x: fuzz.token_sort_r
    df["fuzz_ratio"]            = df.apply(lambda x: fuzz.QRatio(x["que
    df["fuzz_partial_ratio"]    = df.apply(lambda x: fuzz.partial_rati
    df["longest_substr_ratio"]  = df.apply(lambda x: get_longest_subst
    return df
```

In [25]:
```
if os.path.isfile('nlp_features_train.csv'):
    df = pd.read_csv("nlp_features_train.csv",encoding='latin-1')
    df.fillna(' ')
else:
    print("Extracting features for train:")
    df = pd.read_csv("train.csv")
    df = extract_features(df)
    df.to_csv("nlp_features_train.csv", index=False)
df.head(2)
```

Out[25]:

	id	qid1	qid2	question1	question2	is_duplicate	cwc_min	cwc_max	csc_min	csc_ma
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	0.999980	0.833319	0.999983	0.99998
1	1	3	4	what is the story of kohinoor koh i noor dia...	what would happen if the indian government sto...	0	0.799984	0.399996	0.749981	0.59998

2 rows × 21 columns

3.5.1 Analysis of extracted features

3.5.1.1 Plotting Word clouds

- Creating Word Cloud of Duplicates and Non-Duplicates Question pairs
- We can observe the most frequent occuring words

In [26]:

```
df_duplicate = df[df['is_duplicate'] == 1]
dfp_nonduplicate = df[df['is_duplicate'] == 0]

# Converting 2d array of q1 and q2 and flatten the array: like {{1,2},
p = np.dstack([df_duplicate["question1"], df_duplicate["question2"]]).
n = np.dstack([dfp_nonduplicate["question1"], dfp_nonduplicate["questi

print ("Number of data points in class 1 (duplicate pairs) :",len(p))
print ("Number of data points in class 0 (non duplicate pairs) :",len(

#Saving the np array into a text file
np.savetxt('train_p.txt', p, delimiter=' ', fmt='%s')
np.savetxt('train_n.txt', n, delimiter=' ', fmt='%s')
```

```
Number of data points in class 1 (duplicate pairs) : 298526
Number of data points in class 0 (non duplicate pairs) : 510054
```

In [27]:

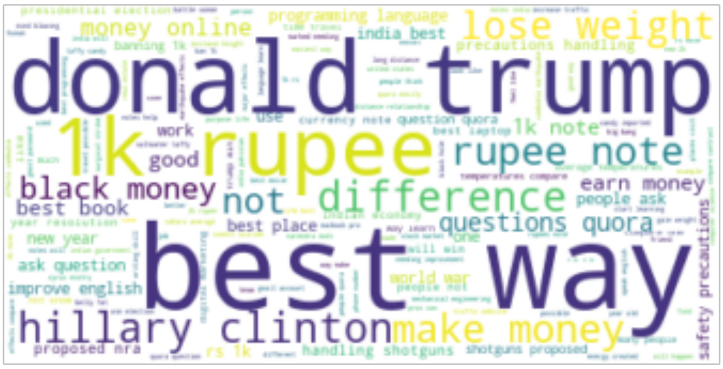
```
# reading the text files and removing the Stop Words:
d = path.dirname('.')

textp_w = open(path.join(d, 'train_p.txt')).read()
textn_w = open(path.join(d, 'train_n.txt')).read()
stopwords = set(STOPWORDS)
stopwords.add("said")
stopwords.add("br")
stopwords.add(" ")
stopwords.remove("not")

stopwords.remove("no")
#stopwords.remove("good")
#stopwords.remove("love")
stopwords.remove("like")
#stopwords.remove("best")
#stopwords.remove("!")
print ("Total number of words in duplicate pair questions :",len(textp_
print ("Total number of words in non duplicate pair questions :",len(te
```

```
Total number of words in duplicate pair questions : 16109886
Total number of words in non duplicate pair questions : 33193130
```

Word Clouds generated from duplicate pair question's text

In [28]:
```
wc = WordCloud(background_color="white", max_words=len(textp_w), stopw
wc.generate(textp_w)
print ("Word Cloud for Duplicate Question pairs")
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

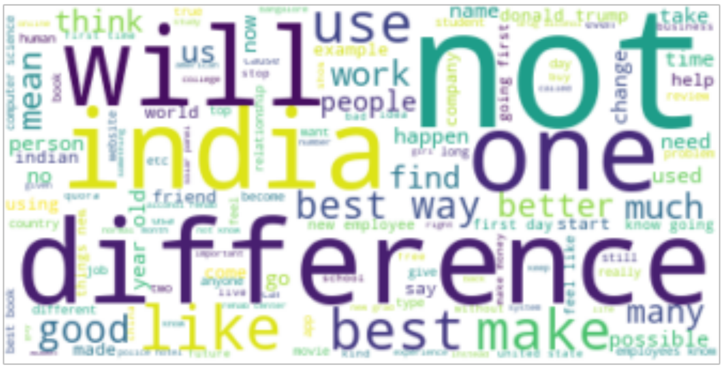
```
Word Cloud for Duplicate Question pairs
```



Word Clouds generated from non duplicate pair question's text

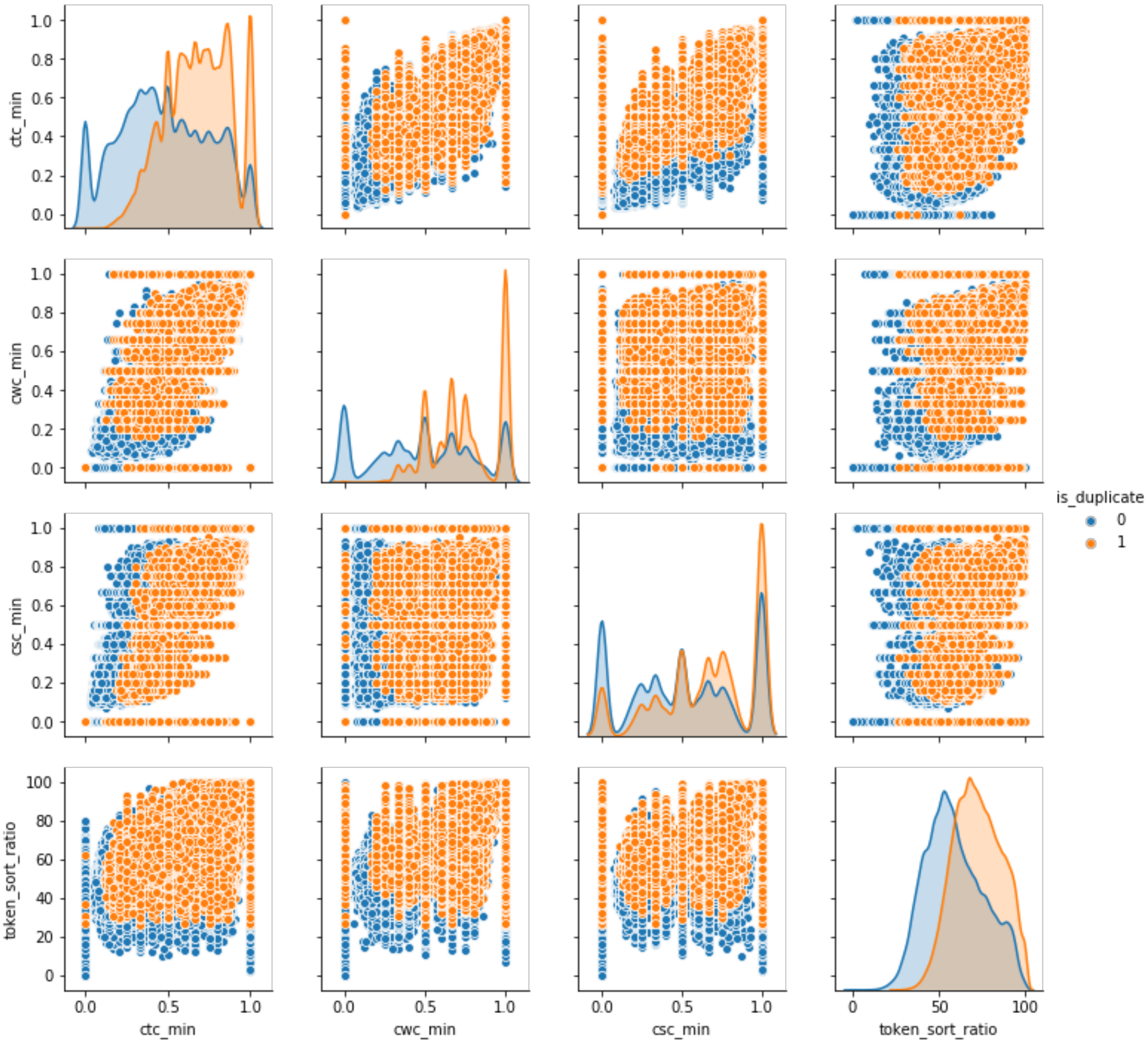
In [29]:
```
wc = WordCloud(background_color="white", max_words=len(textn_w),stopwo
# generate word cloud
wc.generate(textn_w)
print ("Word Cloud for non-Duplicate Question pairs:")
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Word Cloud for non-Duplicate Question pairs:
```

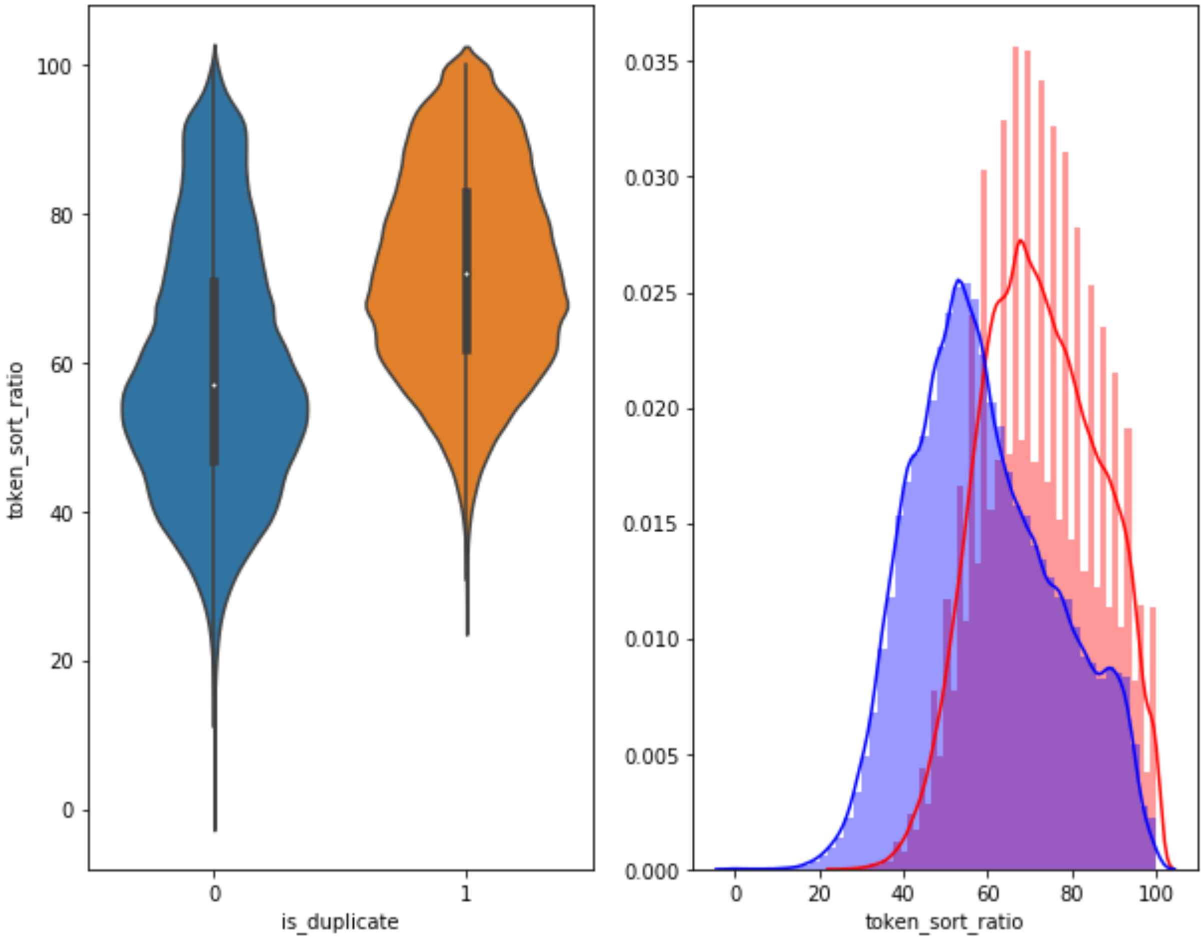


3.5.1.2 Pair plot of features ['ctc_min', 'cwc_min', 'csc_min', 'token_sort_ratio']

```
In [30]: n = df.shape[0]
         sns.pairplot(df[['ctc_min', 'cwc_min', 'csc_min', 'token_sort_ratio', '
         plt.show()
```



In [31]:
```
# Distribution of the token_sort_ratio
plt.figure(figsize=(10, 8))

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

plt.subplot(1,2,2)
sns.distplot(df[df['is_duplicate'] == 1.0]['token_sort_ratio'][0:] , la
sns.distplot(df[df['is_duplicate'] == 0.0]['token_sort_ratio'][0:] , la
plt.show()
```



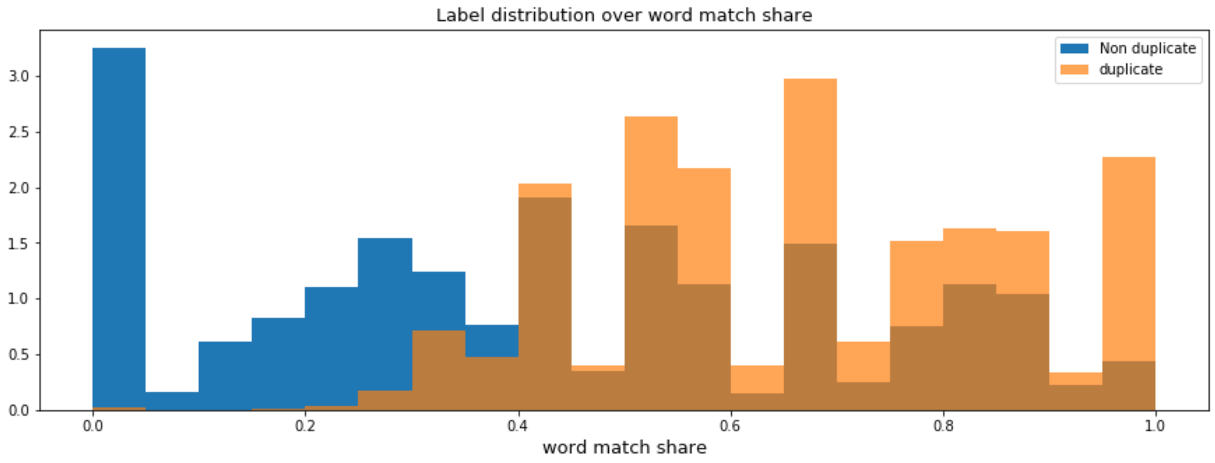
In [33]:

```
# To see the overlap of words in both Duplicate and non Duplicate data

import matplotlib.pyplot as plt
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')

stops = set(stopwords.words("english"))

def word_match_share(row):
    q1words = {}
    q2words = {}
    for word in str(row['question1']).lower().split():
        if word not in stops:
            q1words[word] = 1
    for word in str(row['question2']).lower().split():
        if word not in stops:
            q2words[word] = 1
    if len(q1words) == 0 or len(q2words) == 0:
        # The computer-generated chaff includes a few questions that a
        return 0
    shared_words_in_q1 = [w for w in q1words.keys() if w in q2words]
    shared_words_in_q2 = [w for w in q2words.keys() if w in q1words]
    R = (len(shared_words_in_q1) + len(shared_words_in_q2))/(len(q1wor
    return R

plt.figure(figsize=(15, 5))
train_word_match = df.apply(word_match_share, axis=1, raw=True)
plt.hist(train_word_match[df['is_duplicate'] == 0], bins=20, normed=Tru
plt.hist(train_word_match[df['is_duplicate'] == 1], bins=20, normed=Tru
plt.legend()
plt.title('Label distribution over word match share', fontsize=13)
plt.xlabel('word match share', fontsize=13)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/rohitbohra/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[33]:

```
Text(0.5,0,'word match share')
```



3.5.2 Visualization

In [34]:
```
# Using TSNE for Dimentionality reduction for 15 Features(Generated af
from sklearn.preprocessing import MinMaxScaler
dfp_subsampled = df[0:5000]
X = MinMaxScaler().fit_transform(dfp_subsampled[['cwc_min', 'cwc_max',
y = dfp_subsampled['is_duplicate'].values
```

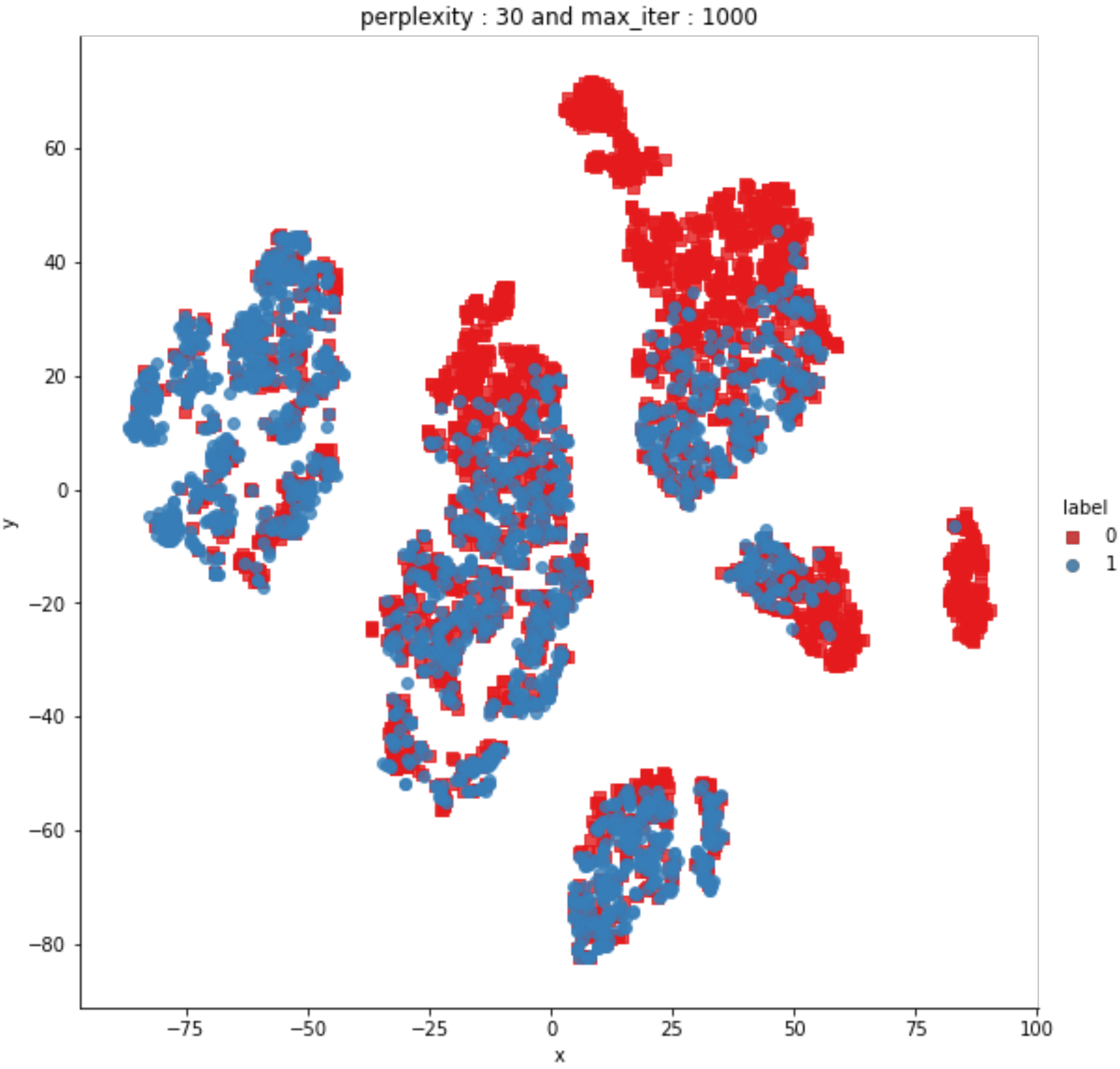
In [35]:
```
tsne2d = TSNE(
    n_components=2,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 5000 samples in 0.373s...
[t-SNE] Computed neighbors for 5000 samples in 0.440s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 0.116557
[t-SNE] Computed conditional probabilities in 0.335s
[t-SNE] Iteration 50: error = 80.9162369, gradient norm = 0.0427600
(50 iterations in 3.524s)
[t-SNE] Iteration 100: error = 70.3915100, gradient norm = 0.0108003
(50 iterations in 2.364s)
[t-SNE] Iteration 150: error = 68.6126938, gradient norm = 0.0054721
(50 iterations in 2.216s)
[t-SNE] Iteration 200: error = 67.7680206, gradient norm = 0.0042246
(50 iterations in 2.460s)
[t-SNE] Iteration 250: error = 67.2733459, gradient norm = 0.0037275
(50 iterations in 2.457s)
[t-SNE] KL divergence after 250 iterations with early exaggeration:
67.273346
[t-SNE] Iteration 300: error = 1.7734827, gradient norm = 0.0011933
(50 iterations in 2.592s)
[t-SNE] Iteration 350: error = 1.3717980, gradient norm = 0.0004826
(50 iterations in 2.486s)
[t-SNE] Iteration 400: error = 1.2037998, gradient norm = 0.0002772
(50 iterations in 2.251s)
[t-SNE] Iteration 450: error = 1.1133003, gradient norm = 0.0001877
(50 iterations in 2.182s)
[t-SNE] Iteration 500: error = 1.0579894, gradient norm = 0.0001429
(50 iterations in 2.182s)
[t-SNE] Iteration 550: error = 1.0220573, gradient norm = 0.0001178
(50 iterations in 2.210s)
```

```
(50 iterations in 2.210s)
[t-SNE] Iteration 600: error = 0.9990303, gradient norm = 0.0001036
(50 iterations in 2.225s)
[t-SNE] Iteration 650: error = 0.9836842, gradient norm = 0.0000951
(50 iterations in 2.214s)
[t-SNE] Iteration 700: error = 0.9732341, gradient norm = 0.0000860
(50 iterations in 2.235s)
[t-SNE] Iteration 750: error = 0.9649901, gradient norm = 0.0000789
(50 iterations in 2.190s)
[t-SNE] Iteration 800: error = 0.9582695, gradient norm = 0.0000745

(50 iterations in 2.192s)
[t-SNE] Iteration 850: error = 0.9525222, gradient norm = 0.0000732
(50 iterations in 2.312s)
[t-SNE] Iteration 900: error = 0.9479918, gradient norm = 0.0000689
(50 iterations in 2.440s)
[t-SNE] Iteration 950: error = 0.9442031, gradient norm = 0.0000651
(50 iterations in 2.630s)
[t-SNE] Iteration 1000: error = 0.9408465, gradient norm = 0.0000590
(50 iterations in 2.514s)
[t-SNE] KL divergence after 1000 iterations: 0.940847
```

In [36]:

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

# draw the plot in appropriate place in the grid
sns.lmplot(data=df, x='x', y='y', hue='label', fit_reg=False, size=8,pa
plt.title("perplexity : {} and max_iter : {}".format(30, 1000))
plt.show()
```



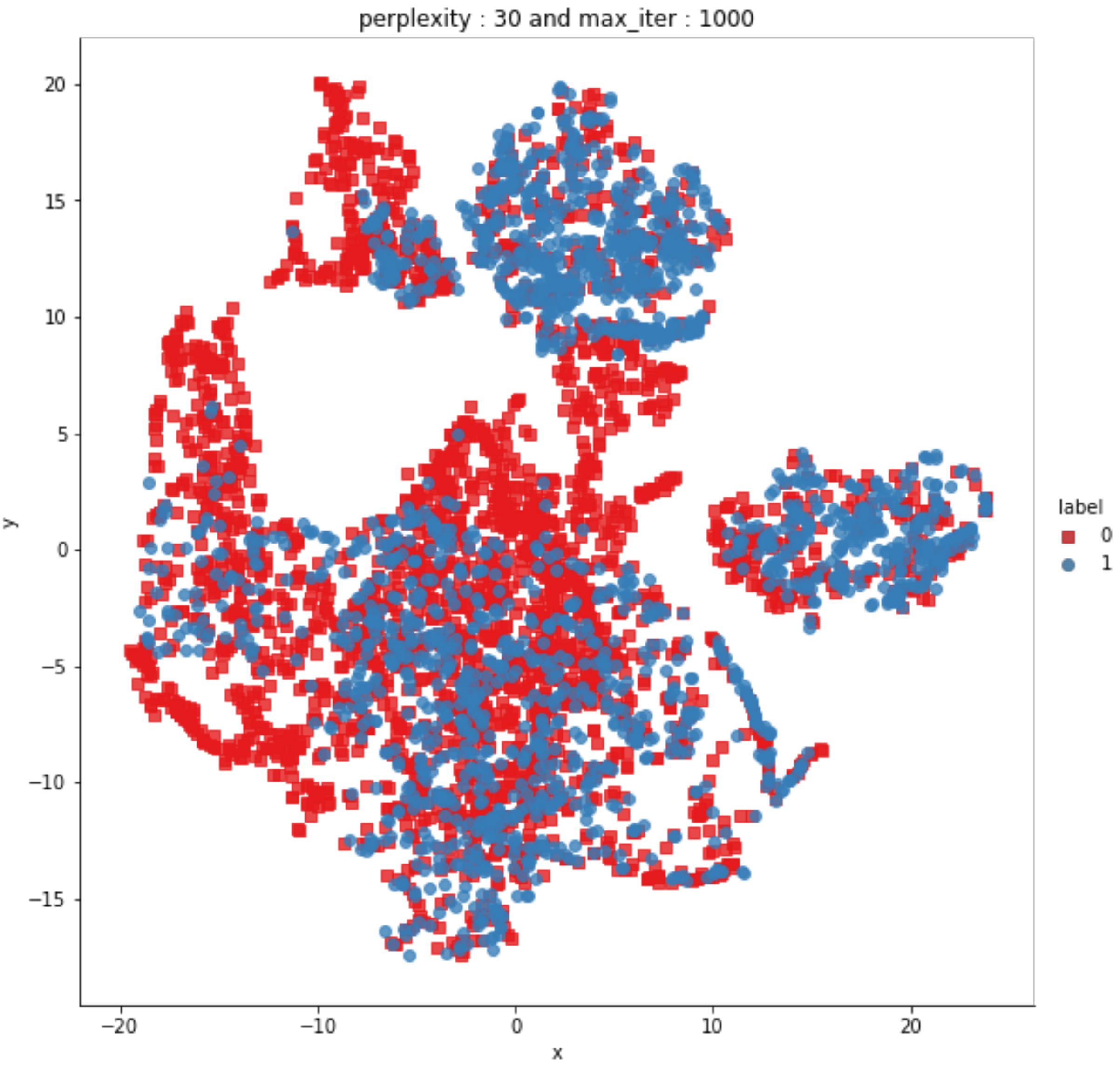
In [37]:

```
from sklearn.manifold import TSNE
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 5000 samples in 0.015s...
[t-SNE] Computed neighbors for 5000 samples in 0.427
```

```
[t-SNE] Computed neighbors for 5000 samples in 0.437s...
[t-SNE] Computed conditional probabilities for sample 1000 / 5000
[t-SNE] Computed conditional probabilities for sample 2000 / 5000
[t-SNE] Computed conditional probabilities for sample 3000 / 5000
[t-SNE] Computed conditional probabilities for sample 4000 / 5000
[t-SNE] Computed conditional probabilities for sample 5000 / 5000
[t-SNE] Mean sigma: 0.116557
[t-SNE] Computed conditional probabilities in 0.221s
[t-SNE] Iteration 50: error = 80.3552017, gradient norm = 0.0329941
(50 iterations in 11.347s)
[t-SNE] Iteration 100: error = 69.1120148, gradient norm = 0.0033901

(50 iterations in 6.557s)
[t-SNE] Iteration 150: error = 67.6176224, gradient norm = 0.0017826
(50 iterations in 5.471s)
[t-SNE] Iteration 200: error = 67.0574570, gradient norm = 0.0014586
(50 iterations in 5.113s)
[t-SNE] Iteration 250: error = 66.7299194, gradient norm = 0.0009065
(50 iterations in 5.822s)
[t-SNE] KL divergence after 250 iterations with early exaggeration:
66.729919
[t-SNE] Iteration 300: error = 1.4958616, gradient norm = 0.0006863
(50 iterations in 7.175s)
[t-SNE] Iteration 350: error = 1.1540339, gradient norm = 0.0001894
(50 iterations in 9.395s)
[t-SNE] Iteration 400: error = 1.0091627, gradient norm = 0.0000964
(50 iterations in 8.106s)
[t-SNE] Iteration 450: error = 0.9373680, gradient norm = 0.0000611
(50 iterations in 7.973s)
[t-SNE] Iteration 500: error = 0.9012471, gradient norm = 0.0000540
(50 iterations in 8.468s)
[t-SNE] Iteration 550: error = 0.8821378, gradient norm = 0.0000498
(50 iterations in 7.790s)
[t-SNE] Iteration 600: error = 0.8697239, gradient norm = 0.0000389
(50 iterations in 7.954s)
[t-SNE] Iteration 650: error = 0.8608552, gradient norm = 0.0000344
(50 iterations in 8.813s)
[t-SNE] Iteration 700: error = 0.8536769, gradient norm = 0.0000326
(50 iterations in 9.149s)
[t-SNE] Iteration 750: error = 0.8485754, gradient norm = 0.0000295
(50 iterations in 8.996s)
[t-SNE] Iteration 800: error = 0.8441855, gradient norm = 0.0000263
(50 iterations in 7.987s)
[t-SNE] Iteration 850: error = 0.8395877, gradient norm = 0.0000260
(50 iterations in 8.790s)
[t-SNE] Iteration 900: error = 0.8356333, gradient norm = 0.0000252
(50 iterations in 8.885s)
[t-SNE] Iteration 950: error = 0.8320156, gradient norm = 0.0000234
(50 iterations in 8.793s)
[t-SNE] Iteration 1000: error = 0.8287079, gradient norm = 0.0000247
(50 iterations in 8.873s)
[t-SNE] KL divergence after 1000 iterations: 0.828708
```

In [38]:
```
df = pd.DataFrame({'x':tsne3d[:,0], 'y':tsne3d[:,1] ,'label':y})

# draw the plot in appropriate place in the grid
sns.lmplot(data=df, x='x', y='y', hue='label', fit_reg=False, size=8,pa
plt.title("perplexity : {} and max_iter : {}".format(30, 1000))
plt.show()
```



4. Featurizing text data with tfidf weighted word-vectors

In [255]:

```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
warnings.filterwarnings("ignore")
import sys
import os
import pandas as pd
import numpy as np
from tqdm import tqdm

# exctract word2vec vectors
# https://github.com/explosion/spaCy/issues/1721
# http://landinghub.visualstudio.com/visual-cpp-build-tools
```

In [256]:

```
import spacy
```

In [257]:

```
# Load Basic Features
df_basic_feature = pd.read_csv("df_fe_without_preprocessing_train.csv"
```

In [258]:

```
# avoid decoding problems
df = pd.read_csv("train.csv")

# encode questions to unicode
# https://stackoverflow.com/a/6812069
# ----------------- python 2 ---------------------
# df['question1'] = df['question1'].apply(lambda x: unicode(str(x),"ut
# df['question2'] = df['question2'].apply(lambda x: unicode(str(x),"ut
# ----------------- python 3 ---------------------
df['question1'] = df['question1'].apply(lambda x: str(x))
df['question2'] = df['question2'].apply(lambda x: str(x))
```

In [259]:
```
df.head()
```

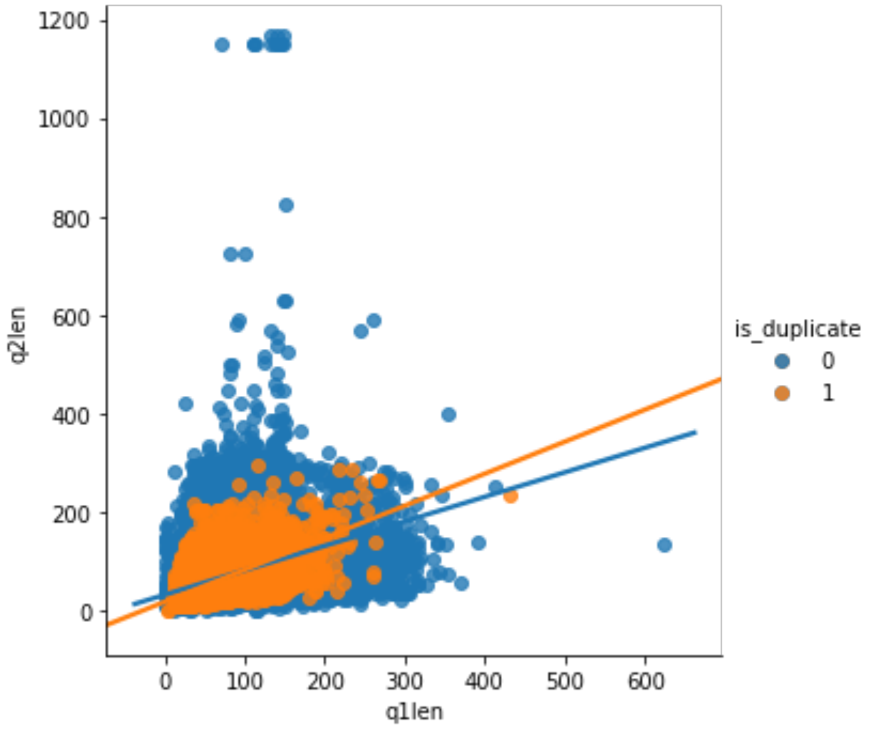
Out[259]:

	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 [260]:
```
#Number of columns in dataframe
len(df_basic_feature.columns)
```

Out[260]:
```
17
```

In [261]:
```
import seaborn as sns;
import matplotlib.pyplot as plt
ax = sns.lmplot(x="q1len", y="q2len",hue="is_duplicate", data=df_basic_
```



In [262]:
```
# Loading the advanced features
df_advance_features = pd.read_csv("nlp_features_train.csv",encoding='l
```

In [263]:
```
# Columns dropped from basic feature dataframe
df_basic_feature = df_basic_feature.drop(['qid1','qid2'],axis=1)

# Columns dropped from advance feature dataframe
df_advance_features = df_advance_features.drop(['qid1','qid2','questio

# Lets add both the truncated dataframe into one dataframe
df_basic_advance_features  = df_basic_feature.merge(df_advance_feature
```

In [264]:
```
list(df_basic_advance_features.columns.values)
```

Out[264]:
```
['id',
 'question1',
 'question2',
 'is_duplicate',
 'freq_qid1',
 'freq_qid2',
 'q1len',
 'q2len',
 'q1_n_words',
 'q2_n_words',
 'word_Common',
 'word_Total',
 'word_share',
 'freq_q1+q2',
 'freq_q1-q2',
 'cwc_min',
 'cwc_max',
 'csc_min',
 'csc_max',
 'ctc_min',
 'ctc_max',
 'last_word_eq',
 'first_word_eq',
 'abs_len_diff',
 'mean_len',
 'token_set_ratio',
 'token_sort_ratio',
 'fuzz_ratio',
 'fuzz_partial_ratio',
 'longest_substr_ratio']
```

In [265]:
```
df1 = df_basic_advance_features.dropna()
df1.isnull().any().sum()
```

Out[265]:
```
0
```

In [266]:
```
df1['is_duplicate'].value_counts()
```

Out[266]:
```
0    255024
1    149263
Name: is_duplicate, dtype: int64
```

In [267]:
```
dff = df1.head(100000)
```

In [269]:
```
dff.columns
```

Out[269]:
```
Index(['id', 'question1', 'question2', 'is_duplicate', 'freq_qid1',
       'freq_qid2', 'q1len', 'q2len', 'q1_n_words', 'q2_n_words',
       'word_Common', 'word_Total', 'word_share', 'freq_q1+q2', 'fre
q_q1-q2',
       'cwc_min', 'cwc_max', 'csc_min', 'csc_max', 'ctc_min', 'ctc_m
ax',
       'last_word_eq', 'first_word_eq', 'abs_len_diff', 'mean_len',
       'token_set_ratio', 'token_sort_ratio', 'fuzz_ratio',
       'fuzz_partial_ratio', 'longest_substr_ratio'],
      dtype='object')
```

In [270]:
```
dff.shape
```

Out[270]:
```
(100000, 30)
```

In [271]:
```
dff.head(2)
```

Out[271]:

	id	question1	question2	is_duplicate	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2
0	0	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	
1	1	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	4	1	51	88	8	

2 rows × 30 columns

In [272]:

```
from sklearn.model_selection import train_test_split
x_train,x_test, y_train, y_test = train_test_split(dff, dff['is_duplica

print(x_train.shape)
print(x_test.shape)
```

```
(70000, 30)
(30000, 30)
```

In [298]:

```
y_train.value_counts()
```

Out[298]:

```
0    43871
1    26129
Name: is_duplicate, dtype: int64
```

In [299]:

```
y_test.value_counts()
```

Out[299]:

```
0    18875
1    11125
Name: is_duplicate, dtype: int64
```

In [273]:

```
x_train.columns
```

Out[273]:

```
Index(['id', 'question1', 'question2', 'is_duplicate', 'freq_qid1',
       'freq_qid2', 'q1len', 'q2len', 'q1_n_words', 'q2_n_words',
       'word_Common', 'word_Total', 'word_share', 'freq_q1+q2', 'fre
q_q1-q2',
       'cwc_min', 'cwc_max', 'csc_min', 'csc_max', 'ctc_min', 'ctc_m
ax',
       'last_word_eq', 'first_word_eq', 'abs_len_diff', 'mean_len',
       'token_set_ratio', 'token_sort_ratio', 'fuzz_ratio',
       'fuzz_partial_ratio', 'longest_substr_ratio'],
      dtype='object')
```

In [274]:

```
# merge texts
questions_train = list(x_train['question1']) + list(x_train['question2

tfidf = TfidfVectorizer(lowercase=False, )
tfidf.fit_transform(questions_train)

# dict key:word and value:tf-idf score
word2tfidf_train = dict(zip(tfidf.get_feature_names(), tfidf.idf_))
```

In [275]:

```
# merge texts
questions_test = list(x_test['question1']) + list(x_test['question2'])

tfidf = TfidfVectorizer(lowercase=False, )
tfidf.fit_transform(questions_test)

# dict key:word and value:tf-idf score
word2tfidf_test = dict(zip(tfidf.get_feature_names(), tfidf.idf_))
```

In [276]:

```
# en_vectors_web_lg, which includes over 1 million unique vectors.
nlp = spacy.load('en_core_web_sm')

vecs1 = []
# https://github.com/noamraph/tqdm
# tqdm is used to print the progress bar
for qu1 in tqdm(list(x_train['question1'])):
    doc1 = nlp(qu1)
    # 384 is the number of dimensions of vectors
    mean_vec1 = np.zeros([len(doc1), 384])
    for word1 in doc1:
        # word2vec
        vec1 = word1.vector
        # fetch df score
        try:
            idf = word2tfidf_train[str(word1)]
        except:
            idf = 0
        # compute final vec
        mean_vec1 += vec1 * idf
    mean_vec1 = mean_vec1.mean(axis=0)
    vecs1.append(mean_vec1)
x_train['q1_feats_m'] = list(vecs1)
```

```
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
1[illegible]
```

In [277]:

```
vecs2 = []
for qu2 in tqdm(list(x_train['question2'])):
    doc2 = nlp(qu2)
    mean_vec2 = np.zeros([len(doc2), 384])
    for word2 in doc2:
        # word2vec
        vec2 = word2.vector
        # fetch df score
        try:
            idf = word2tfidf_train[str(word2)]
        except:
            #print word
            idf = 0
        # compute final vec
        mean_vec2 += vec2 * idf
    mean_vec2 = mean_vec2.mean(axis=0)
    vecs2.append(mean_vec2)
x_train['q2_feats_m'] = list(vecs2)
```

```
  0%|          | 0/70000 [00:00<?, ?it/s]
  0%|          | 3/70000 [00:00<40:27, 28.84it/s]
  0%|          | 9/70000 [00:00<34:55, 33.41it/s]
  0%|          | 15/70000 [00:00<30:52, 37.78it/s]
  0%|          | 20/70000 [00:00<28:38, 40.72it/s]
  0%|          | 26/70000 [00:00<26:46, 43.57it/s]
  0%|          | 32/70000 [00:00<25:08, 46.38it/s]
  0%|          | 38/70000 [00:00<24:14, 48.09it/s]
  0%|          | 44/70000 [00:00<23:13, 50.20it/s]
  0%|          | 50/70000 [00:00<23:05, 50.49it/s]
  0%|          | 56/70000 [00:01<22:47, 51.14it/s]
  0%|          | 62/70000 [00:01<23:21, 49.89it/s]
  0%|          | 68/70000 [00:01<22:17, 52.27it/s]
  0%|          | 74/70000 [00:01<21:55, 53.16it/s]
  0%|          | 80/70000 [00:01<21:55, 53.16it/s]
  0%|          | 86/70000 [00:01<21:51, 53.32it/s]
  0%|          | 92/70000 [00:01<21:52, 53.25it/s]
  0%|          | 98/70000 [00:01<21:39, 53.79it/s]
  0%|          | 104/70000 [00:02<22:03, 52.81it/s]
```

In [278]:
```
# en_vectors_web_lg, which includes over 1 million unique vectors.
nlp = spacy.load('en_core_web_sm')

vecs1 = []
# https://github.com/noamraph/tqdm
# tqdm is used to print the progress bar
for qu1 in tqdm(list(x_test['question1'])):
    doc1 = nlp(qu1)
    # 384 is the number of dimensions of vectors
    mean_vec1 = np.zeros([len(doc1), 384])
    for word1 in doc1:
        # word2vec
        vec1 = word1.vector
        # fetch df score
        try:
            idf = word2tfidf_test[str(word1)]
        except:
            idf = 0
        # compute final vec
        mean_vec1 += vec1 * idf
    mean_vec1 = mean_vec1.mean(axis=0)
    vecs1.append(mean_vec1)
x_test['q1_feats_m'] = list(vecs1)
```

```
  1%|          | 361/30000 [00:07<10:55, 45.19it/s]
  1%|          | 366/30000 [00:07<10:46, 45.83it/s]
  1%|          | 371/30000 [00:07<10:38, 46.40it/s]
  1%|▏         | 377/30000 [00:07<10:14, 48.23it/s]
  1%|▏         | 383/30000 [00:07<09:59, 49.43it/s]
  1%|▏         | 389/30000 [00:08<09:42, 50.85it/s]
  1%|▏         | 395/30000 [00:08<09:39, 51.08it/s]
  1%|▏         | 401/30000 [00:08<09:40, 50.96it/s]
  1%|▏         | 407/30000 [00:08<09:48, 50.27it/s]
  1%|▏         | 413/30000 [00:08<09:44, 50.64it/s]
  1%|▏         | 419/30000 [00:08<09:54, 49.75it/s]
  1%|▏         | 424/30000 [00:08<10:24, 47.38it/s]
  1%|▏         | 429/30000 [00:08<10:25, 47.25it/s]
  1%|▏         | 434/30000 [00:08<10:33, 46.66it/s]
  1%|▏         | 439/30000 [00:09<10:34, 46.59it/s]
  1%|▏         | 444/30000 [00:09<10:26, 47.17it/s]
  1%|▏         | 449/30000 [00:09<10:18, 47.78it/s]
  2%|▏         | 455/30000 [00:09<09:46, 50.37it/s]
  2%|▏         | 461/30000 [00:09<09:56, 49.50it/s]
  2%|▏         | 466/30000 [00:09<10:05, 48.80it/s]
```

In [279]:
```
vecs2 = []
for qu2 in tqdm(list(x_test['question2'])):
    doc2 = nlp(qu2)
    mean_vec2 = np.zeros([len(doc2), 384])
    for word2 in doc2:
        # word2vec
        vec2 = word2.vector
        # fetch df score
        try:
            idf = word2tfidf_test[str(word2)]
        except:
            #print word
            idf = 0
        # compute final vec
        mean_vec2 += vec2 * idf
    mean_vec2 = mean_vec2.mean(axis=0)
    vecs2.append(mean_vec2)
x_test['q2_feats_m'] = list(vecs2)
```

```
  3%|█          | 885/30000 [00:17<09:20, 51.96it/s]
```

In [280]:
```
x_train.shape
```

Out[280]:
```
(70000, 32)
```

In [281]:
```
x_test.shape
```

Out[281]:
```
(30000, 32)
```

In [282]:
```
x_train_q1 = pd.DataFrame(x_train.q1_feats_m.values.tolist(), index= x_
x_train_q2 = pd.DataFrame(x_train.q2_feats_m.values.tolist(), index= x_
```

In [283]:

```
x_test_q1 = pd.DataFrame(x_test.q1_feats_m.values.tolist(), index= x_te
x_test_q2 = pd.DataFrame(x_test.q2_feats_m.values.tolist(), index= x_te
```

In [284]:

```
print(x_train_q1.shape)
print(x_train_q2.shape)
print(x_test_q1.shape)
print(x_test_q2.shape)
```

```
(70000, 384)
(70000, 384)
(30000, 384)
(30000, 384)
```

In [285]:

```
x_train = x_train.drop(['question1','question2','is_duplicate', 'q1_fea
```

In [286]:

```
x_test = x_test.drop(['question1','question2','is_duplicate', 'q1_feats
```

In [287]:

```
x_train.head(2)
```

Out[287]:

	id	freq_qid1	freq_qid2	q1len	q2len	q1_n_words	q2_n_words	word_Common	w
75462	75462	15	2	31	32	6	6	2.0	
41375	41375	1	1	97	109	22	25	9.0	

2 rows × 27 columns

In [288]:

```
print(x_train.shape)
print(x_test.shape)
```

```
(70000, 27)
(30000, 27)
```

In [289]:

```
x_train_q1['id']=x_train['id']
x_train_q2['id']=x_train['id']
x_test_q1['id']=x_test['id']
x_test_q2['id']=x_test['id']
```

In [290]:

```
x_train = x_train.merge(x_train_q1, on='id',how='left')
x_train = x_train.merge(x_train_q2, on='id',how='left')
```

In [291]:

```
x_test = x_test.merge(x_test_q1, on='id',how='left')
x_test = x_test.merge(x_test_q2, on='id',how='left')
```

In [292]:
```
print(x_train.shape)
print(x_test.shape)
```

```
(70000, 795)
(30000, 795)
```

In [304]:
```
from sklearn import preprocessing

# Create the Scaler object
scaler = preprocessing.StandardScaler()
# Fit your data on the scaler object
x_train = scaler.fit_transform(x_train)
```

In [305]:
```
x_test = scaler.fit_transform(x_test)
```

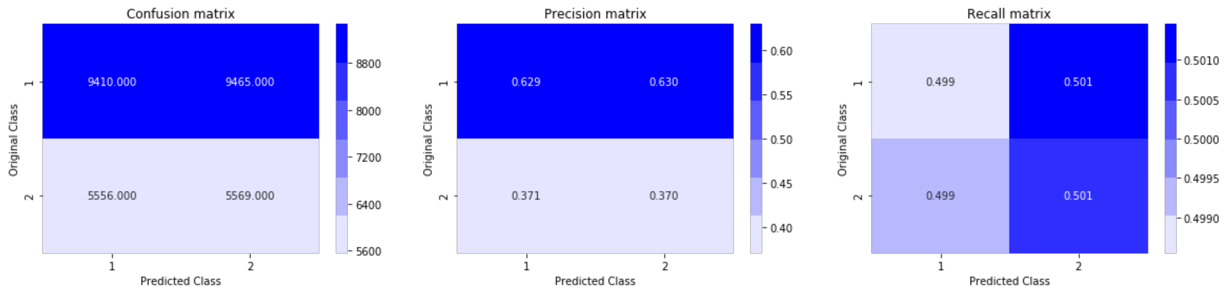
In [307]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
from sklearn.calibration import CalibratedClassifierCV
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
from mlxtend.classifier import StackingClassifier
```

In [317]:
```
from collections import Counter, defaultdict

print("-"*10, "Distribution of output variable in train data", "-"*10)
train_distr = Counter(y_train)
train_len = len(y_train)
print("Class 0: ",int(train_distr[0])/train_len,"Class 1: ", int(train_
print("-"*10, "Distribution of output variable in train data", "-"*10)
test_distr = Counter(y_test)
test_len = len(y_test)
print("Class 0: ",int(test_distr[1])/test_len, "Class 1: ",int(test_di
```

```
---------- Distribution of output variable in train data ----------
Class 0:  0.6267285714285714 Class 1:  0.3732714285714286
---------- Distribution of output variable in train data ----------
Class 0:  0.37083333333333335 Class 1:  0.37083333333333335
```

In [318]:
```
##  Models

# Random Model
predicted_y = np.zeros((len(y_test),2))
for i in range(test_len):
    rand_probs = np.random.rand(1,2)
    predicted_y[i] = ((rand_probs/sum(sum(rand_probs)))[0])
print("Log loss on Test Data using Random Model",log_loss(y_test, pred

predicted_y =np.argmax(predicted_y, axis=1)
plot_confusion_matrix(y_test, predicted_y)
```

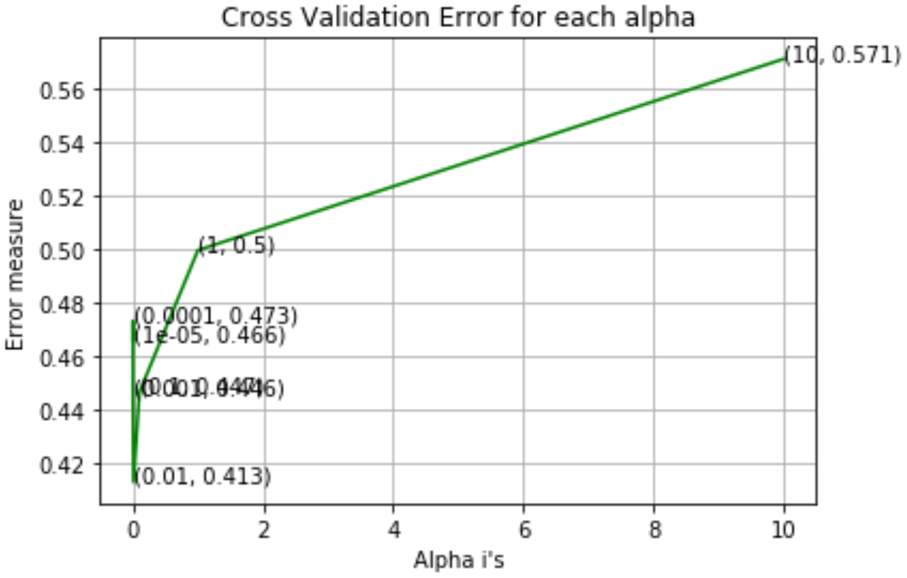
```
Log loss on Test Data using Random Model 0.885374037024287
```



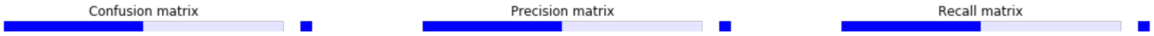
In [308]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie

# read more about SGDClassifier() at http://scikit-learn.org/stable/mo
# -------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, l
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, …]) Fit linear model with Stoc
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------



log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state
    clf.fit(x_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(x_train, y_train)
    predict_y = sig_clf.predict_proba(x_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.class
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_te

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
```

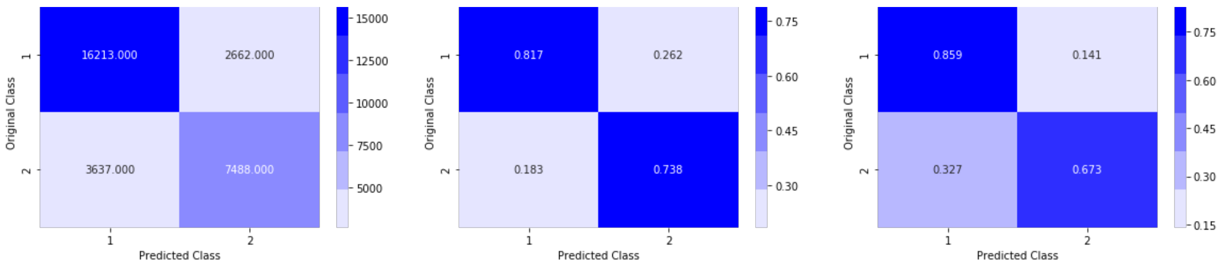
```
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log',
clf.fit(x_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(x_train, y_train)

predict_y = sig_clf.predict_proba(x_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log
predict_y = sig_clf.predict_proba(x_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.4661112573028649
For values of alpha =  0.0001 The log loss is: 0.4733028089896728
For values of alpha =  0.001 The log loss is: 0.44588399023613856
For values of alpha =  0.01 The log loss is: 0.41318361822661187
For values of alpha =  0.1 The log loss is: 0.4466665529061543
For values of alpha =  1 The log loss is: 0.4996575447423112
For values of alpha =  10 The log loss is: 0.5709438505035458
```



```
For values of best alpha =  0.01 The train log loss is: 0.4078668453
1981443
For values of best alpha =  0.01 The test log loss is: 0.41318361822
661187
Total number of data points : 30000
```

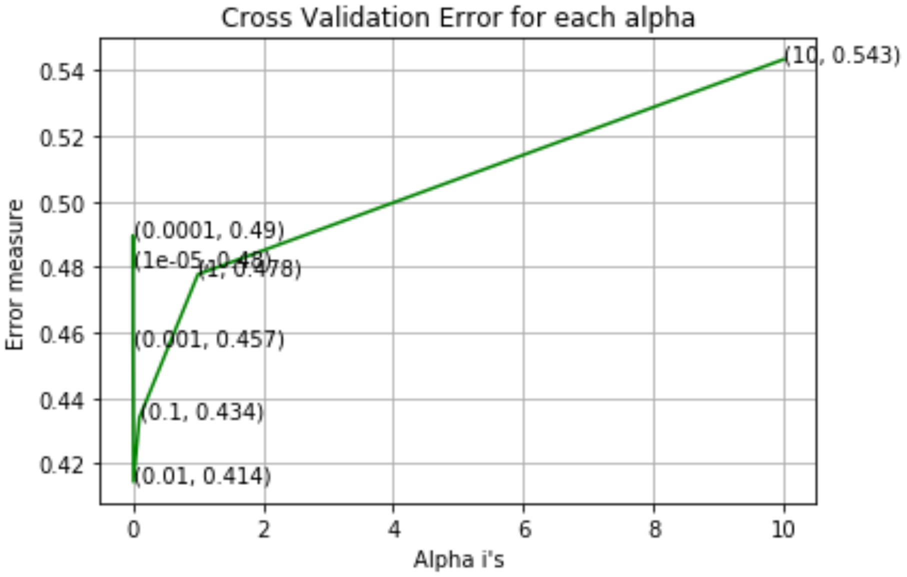




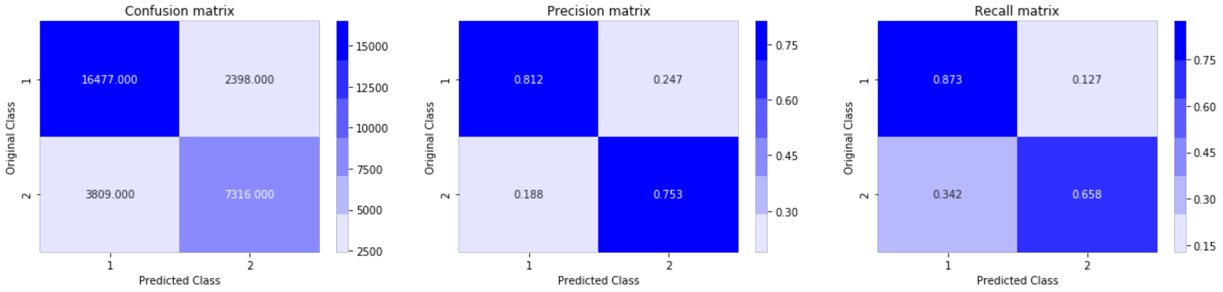
In [310]:
```
alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie

# read more about SGDClassifier() at http://scikit-learn.org/stable/mo
# ------------------------------
# default parameters
# SGDClassifier(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.1
# shuffle=True, verbose=0, epsilon=0.1, n_jobs=1, random_state=None, l
# class_weight=None, warm_start=False, average=False, n_iter=None)

# some of methods
# fit(X, y[, coef_init, intercept_init, …]) Fit linear model with Stoc
# predict(X)    Predict class labels for samples in X.

#-------------------------------
# video link:
#------------------------------


log_error_array=[]
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='hinge', random_sta
    clf.fit(x_train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(x_train, y_train)
    predict_y = sig_clf.predict_proba(x_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.clas
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_te

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='hinge
clf.fit(x_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(x_train, y_train)
```

```
predict_y = sig_clf.predict_proba(x_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log
predict_y = sig_clf.predict_proba(x_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.48017198145677364
For values of alpha =  0.0001 The log loss is: 0.48957143206751197
For values of alpha =  0.001 The log loss is: 0.45664828162701354
For values of alpha =  0.01 The log loss is: 0.4144183019700687
For values of alpha =  0.1 The log loss is: 0.43396867654766225
For values of alpha =  1 The log loss is: 0.4777497996087267
For values of alpha =  10 The log loss is: 0.5434003596204937
```



```
For values of best alpha =  0.01 The train log loss is: 0.4090579081
312148
For values of best alpha =  0.01 The test log loss is: 0.41441830197
00687
Total number of data points : 30000
```



In [311]:

```
import xgboost as xgb
params = {}
params['objective'] = 'binary:logistic'
params['eval_metric'] = 'logloss'
params['eta'] = 0.02
params['max_depth'] = 4
```

```
d_train = xgb.DMatrix(x_train, label=y_train)
d_test = xgb.DMatrix(x_test, label=y_test)

watchlist = [(d_train, 'train'), (d_test, 'valid')]

bst = xgb.train(params, d_train, 400, watchlist, early_stopping_rounds

xgdmat = xgb.DMatrix(x_train,y_train)
predict_y = bst.predict(d_test)
print("The test log loss is:",log_loss(y_test, predict_y, labels=clf.c
```

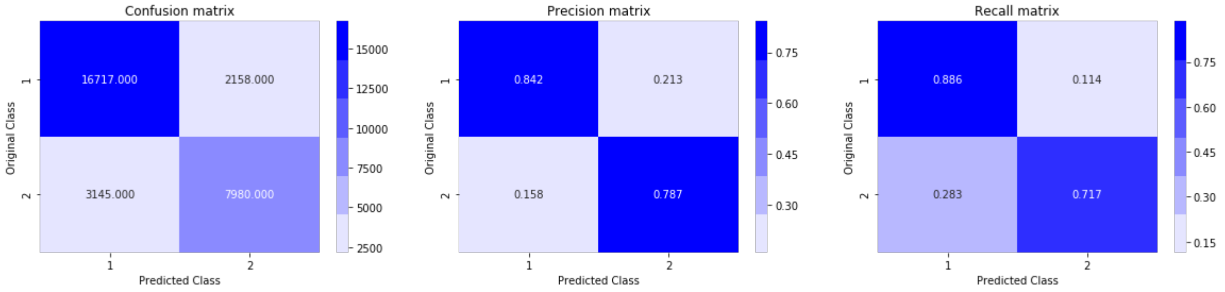
```
[0]     train-logloss:0.684939  valid-logloss:0.684966
Multiple eval metrics have been passed: 'valid-logloss' will be used
for early stopping.

Will train until valid-logloss hasn't improved in 20 rounds.
[10]    train-logloss:0.616287  valid-logloss:0.615908
[20]    train-logloss:0.565479  valid-logloss:0.565055
[30]    train-logloss:0.527642  valid-logloss:0.52718
[40]    train-logloss:0.498356  valid-logloss:0.49805
[50]    train-logloss:0.475679  valid-logloss:0.4754
[60]    train-logloss:0.457495  valid-logloss:0.457406
[70]    train-logloss:0.442861  valid-logloss:0.442937
[80]    train-logloss:0.430932  valid-logloss:0.431162
[90]    train-logloss:0.421187  valid-logloss:0.421589
[100]   train-logloss:0.413169  valid-logloss:0.41366
[110]   train-logloss:0.406411  valid-logloss:0.407012
[120]   train-logloss:0.400822  valid-logloss:0.401585
[130]   train-logloss:0.395954  valid-logloss:0.396923
[140]   train-logloss:0.391969  valid-logloss:0.393096
[150]   train-logloss:0.388152  valid-logloss:0.389551
[160]   train-logloss:0.385025  valid-logloss:0.386673
[170]   train-logloss:0.382018  valid-logloss:0.383889
[180]   train-logloss:0.379543  valid-logloss:0.381665
[190]   train-logloss:0.377243  valid-logloss:0.379644
[200]   train-logloss:0.375094  valid-logloss:0.377768
[210]   train-logloss:0.372815  valid-logloss:0.375859
[220]   train-logloss:0.370559  valid-logloss:0.373965
[230]   train-logloss:0.368436  valid-logloss:0.372145
[240]   train-logloss:0.366444  valid-logloss:0.370576
[250]   train-logloss:0.364676  valid-logloss:0.369217
[260]   train-logloss:0.363119  valid-logloss:0.367988
[270]   train-logloss:0.361679  valid-logloss:0.366896
[280]   train-logloss:0.360072  valid-logloss:0.365611
[290]   train-logloss:0.358545  valid-logloss:0.36446
[300]   train-logloss:0.35719   valid-logloss:0.363421
[310]   train-logloss:0.355819  valid-logloss:0.362491
[320]   train-logloss:0.354554  valid-logloss:0.361487
[330]   train-logloss:0.353299  valid-logloss:0.360564
[340]   train-logloss:0.352283  valid-logloss:0.359872
[350]   train-logloss:0.35125   valid-logloss:0.359225
[360]   train-logloss:0.35012   valid-logloss:0.35843
[370]   train-logloss:0.349063  valid-logloss:0.357693
```

```
[380]	train-logloss:0.34803	valid-logloss:0.357002
[390]	train-logloss:0.3471	valid-logloss:0.356366
[399]	train-logloss:0.346163	valid-logloss:0.355847
The test log loss is: 0.35584939310425545
```

In [312]:

```
predicted_y =np.array(predict_y>0.5,dtype=int)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
Total number of data points : 30000
```



USING ONLY TF-IDF Featurization instead of TF-IDF Weighted W2V technique

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import numpy as np
from nltk.corpus import stopwords
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
warnings.filterwarnings("ignore")
import sys
import os
import pandas as pd
import numpy as np
from tqdm import tqdm
import spacy
# exctract word2vec vectors
# https://github.com/explosion/spaCy/issues/1721
# http://landinghub.visualstudio.com/visual-cpp-build-tools
```

In [179]:

```
# Load Basic Features
df_basic_feature = pd.read_csv("df_fe_without_preprocessing_train.csv"
```

In [180]:

```
#Number of columns in dataframe
len(df_basic_feature.columns)
```

Out[180]:

```
17
```

In [181]:

```
# list of names of columns
list(df_basic_feature.columns.values)
```

Out[181]:

```
['id',
 'qid1',
 'qid2',
 'question1',
 'question2',
 'is_duplicate',
 'freq_qid1',
 'freq_qid2',
 'q1len',
 'q2len',
 'q1_n_words',
 'q2_n_words',
 'word_Common',
 'word_Total',
 'word_share',
 'freq_q1+q2',
 'freq_q1-q2']
```

In [182]:

```
# Loading the advanced features
df_advance_features = pd.read_csv("nlp_features_train.csv",encoding='la
```

In [183]:

```
#Number of columns in dataframe
len(df_advance_features.columns)
```

Out[183]:

```
21
```

In [184]:

```
# list of names of columns
list(df_advance_features.columns.values)
```

Out[184]:

```
['id',
 'qid1',
 'qid2',
 'question1',
 'question2',
 'is_duplicate',
 'cwc_min',
 'cwc_max',
 'csc_min',
 'csc_max',
 'ctc_min',
 'ctc_max',
 'last_word_eq',
 'first_word_eq',
 'abs_len_diff',
 'mean_len',
 'token_set_ratio',
 'token_sort_ratio',
 'fuzz_ratio',
 'fuzz_partial_ratio',
 'longest_substr_ratio']
```

In [185]:

```
# Columns dropped from basic feature dataframe
df_basic_feature = df_basic_feature.drop(['qid1','qid2'],axis=1)

# Columns dropped from advance feature dataframe
df_advance_features = df_advance_features.drop(['qid1','qid2','questio

# Lets add both the truncated dataframe into one dataframe
df_basic_advance_features  = df_basic_feature.merge(df_advance_feature
```

In [186]:
```
list(df_basic_advance_features.columns.values)
```

Out[186]:
```
['id',
 'question1',
 'question2',
 'is_duplicate',
 'freq_qid1',
 'freq_qid2',
 'q1len',
 'q2len',
 'q1_n_words',
 'q2_n_words',
 'word_Common',
 'word_Total',
 'word_share',
 'freq_q1+q2',
 'freq_q1-q2',
 'cwc_min',
 'cwc_max',
 'csc_min',
 'csc_max',
 'ctc_min',
 'ctc_max',
 'last_word_eq',
 'first_word_eq',
 'abs_len_diff',
 'mean_len',
 'token_set_ratio',
 'token_sort_ratio',
 'fuzz_ratio',
 'fuzz_partial_ratio',
 'longest_substr_ratio']
```

In [187]:
```
y_true = df_basic_advance_features['is_duplicate']
```

In [188]:
```
df_basic_advance_features = df_basic_advance_features.drop(['id'],axis
df_basic_advance_features = df_basic_advance_features.drop(['is_duplic
```

In [189]:
```
null_columns=df_basic_advance_features.columns[df_basic_advance_featur
df_basic_advance_features[null_columns].isnull().sum()
```

Out[189]:
```
question1    1
question2    2
dtype: int64
```

In [190]:

```
from nltk.stem import PorterStemmer
from bs4 import BeautifulSoup


# To get the results in 4 decemal points
SAFE_DIV = 0.0001

STOP_WORDS = stopwords.words("english")


def preprocess(x):
    x = str(x).lower()
    x = x.replace(",000,000", "m").replace(",000", "k").replace("′", "
                           .replace("won't", "will not").replace("cann
                           .replace("n't", " not").replace("what's", "
                           .replace("'ve", " have").replace("i'm", "i 
                           .replace("he's", "he is").replace("she's",
                           .replace("%", " percent ").replace("₹", " r
                           .replace("€", " euro ").replace("'ll", " wi
    x = re.sub(r"([0-9]+)000000", r"\1m", x)
    x = re.sub(r"([0-9]+)000", r"\1k", x)


    porter = PorterStemmer()
    pattern = re.compile('\W')

    if type(x) == type(''):
        x = re.sub(pattern, ' ', x)


    if type(x) == type(''):
        x = porter.stem(x)
        example1 = BeautifulSoup(x)
        x = example1.get_text()


    return x
```

In [255]:

```
# preprocessing each question
df_basic_advance_features['question1'] = df_basic_advance_features['qu
df_basic_advance_features['question2'] = df_basic_advance_features['qu
```

In [256]:

```
df_basic_advance_features['question1'] = df_basic_advance_features['qu
df_basic_advance_features['question2'] = df_basic_advance_features['qu
```

In [198]:

```
df_basic_advance_features["question1"] = df_basic_advance_features["qu
df_basic_advance_features["question2"] = df_basic_advance_features["qu
```

In [257]:
```
df_basic_advance_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 404290 entries, 0 to 404289
Data columns (total 28 columns):
question1               404290 non-null object
question2               404290 non-null object
freq_qid1               404290 non-null int64
freq_qid2               404290 non-null int64
q1len                   404290 non-null int64
q2len                   404290 non-null int64
q1_n_words              404290 non-null int64
q2_n_words              404290 non-null int64
word_Common             404290 non-null float64
word_Total              404290 non-null float64
word_share              404290 non-null float64
freq_q1+q2              404290 non-null int64
freq_q1-q2              404290 non-null int64
cwc_min                 404290 non-null float64
cwc_max                 404290 non-null float64
csc_min                 404290 non-null float64
csc_max                 404290 non-null float64
ctc_min                 404290 non-null float64
ctc_max                 404290 non-null float64
last_word_eq            404290 non-null float64
first_word_eq           404290 non-null float64
abs_len_diff            404290 non-null float64
mean_len                404290 non-null float64
token_set_ratio         404290 non-null int64
token_sort_ratio        404290 non-null int64
fuzz_ratio              404290 non-null int64
fuzz_partial_ratio      404290 non-null int64
longest_substr_ratio    404290 non-null float64
dtypes: float64(14), int64(12), object(2)
memory usage: 89.5+ MB
```

In [271]:
```
x_train,x_test, y_train, y_test = train_test_split(df_basic_advance_fe
```

In [272]:
```
print("The shape of train data is ",x_train.shape)
print("The shape of Y train data is ",y_train.shape)
print("The shape of test data is ",x_test.shape)
print("The shape of Y test data is ",y_test.shape)
```

```
The shape of train data is  (283003, 28)
The shape of Y train data is  (283003,)
The shape of test data is  (121287, 28)
The shape of Y test data is  (121287,)
```

In [273]:

```
tfidf = TfidfVectorizer()

train_X = x_train['question1'] + x_train['question2']
question1_question2_train = tfidf.fit_transform(train_X)

test_X = x_test['question1'] + x_test['question2']
question1_question2_test = tfidf.transform(test_X)


print("The shape of test data is ",question1_question2_train.shape)
print("The shape of Y test data is ",question1_question2_test.shape)
```

```
The shape of test data is  (283003, 74077)
The shape of Y test data is  (121287, 74077)
```

In [278]:

```
x_train = x_train.drop(['question1'],axis=1)
```

In [279]:

```
x_test=x_test.drop(['question2'],axis=1)
```

In [281]:

```
x_train = x_train.drop(['question2'],axis=1)
x_test=x_test.drop(['question1'],axis=1)
```

In [282]:

```
x_train = hstack((x_train, question1_question2_train),dtype='float64')
x_test = hstack((x_test, question1_question2_test),dtype='float64').to
```

In [228]:

```
# Instanciate Tfidf Vectorizer
tfidfVectorizer_question1_train = TfidfVectorizer(ngram_range = (1,2),

question1_train = tfidfVectorizer_question1_train.fit_transform(x_trai
```

In [229]:

```
# Instanciate Tfidf Vectorizer
tfidfVectorizer_question2_train = TfidfVectorizer(ngram_range = (1,2),

question2_train = tfidfVectorizer_question2_train.fit_transform(x_trai
```

In [230]:

```
# Instanciate Tfidf Vectorizer
tfidfVectorizer_question1_test = TfidfVectorizer(ngram_range = (1,2),

question1_test = tfidfVectorizer_question1_test.fit_transform(x_test['

tfidfVectorizer_question2_test = TfidfVectorizer(ngram_range = (1,2),

question2_test = tfidfVectorizer_question2_test.fit_transform(x_test['
```

In [231]:

```
question1_question2_train = hstack((question1_train,question2_train))
question1_question2_test = hstack((question1_test,question2_test))
```

In [232]:
```
type(question1_question2_train)
```

Out[232]:
```
scipy.sparse.coo.coo_matrix
```

In [233]:
```
# Drop unnecessary question1 and question2 columns
x_train.drop(['question1','question2'], axis=1, inplace=True)
```

In [234]:
```
# Drop unnecessary question1 and question2 columns
x_test.drop(['question1','question2'], axis=1, inplace=True)
```

In [235]:
```
# Combine all basic, advance and tfidf features
x_train = hstack((X_train, question1_question2_train),format="csr",dty
```

In [236]:
```
# Combine all basic, advance and tfidf features
x_test = hstack((X_test, question1_question2_test),format="csr",dtype=
```

In [283]:
```
x_train.shape
```

Out[283]:
```
(283003, 74103)
```

In [284]:
```
x_test.shape
```

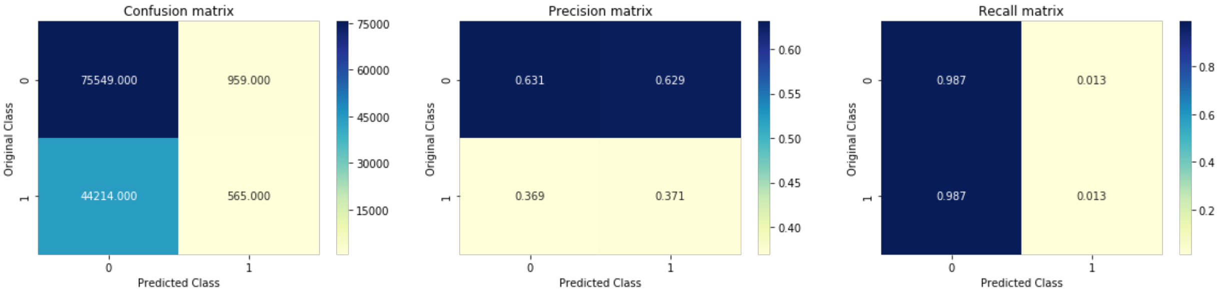
Out[284]:
```
(121287, 74103)
```

In [285]:

```
# This function plots the confusion matrices given y_i, y_i_hat.
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)
    A =(((C.T)/(C.sum(axis=1))).T)
    B =(C/C.sum(axis=0))

    plt.figure(figsize=(20,4))

    labels = [0,1]
    # representing A in heatmap format
    cmap=sns.light_palette("blue")
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap='YlGnBu', fmt=".3f", xticklabels=la
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap='YlGnBu', fmt=".3f", xticklabels=la
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap='YlGnBu', fmt=".3f", xticklabels=la
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")

    plt.show()
```

```
In [286]: ##  Models

          # Random Model
          predicted_y = np.zeros((len(y_test),2))
          for i in range(test_len):
              rand_probs = np.random.rand(1,2)
              predicted_y[i] = ((rand_probs/sum(sum(rand_probs))))[0])
          print("Log loss on Test Data using Random Model",log_loss(y_test, pred

          predicted_y =np.argmax(predicted_y, axis=1)
          plot_confusion_matrix(y_test, predicted_y)
```

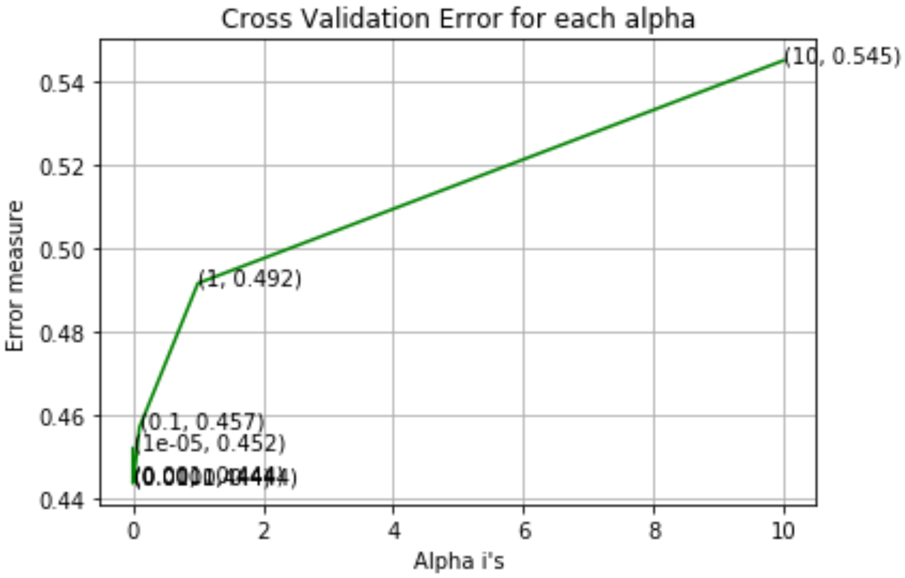
```
Log loss on Test Data using Random Model 0.6978384229962716
```



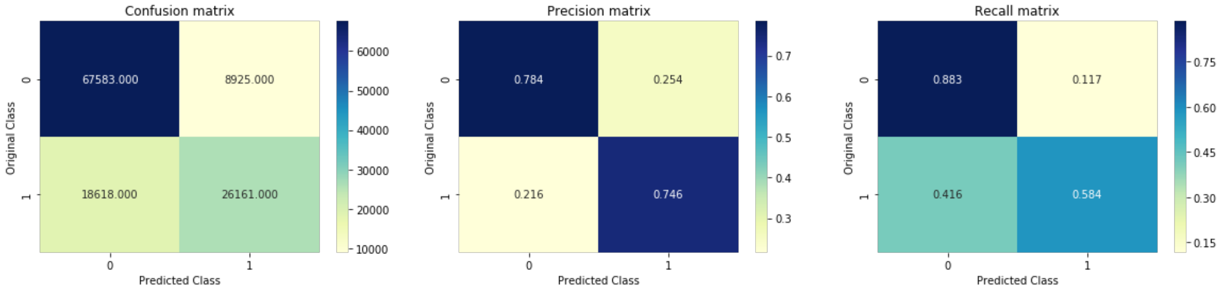
```
In [287]: gistic regression

          a = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifier.
          error_array=[]
          i in alpha:
          clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
          clf.fit(x_train, y_train)
          sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
          sig_clf.fit(x_train, y_train)
          predict_y = sig_clf.predict_proba(x_test)
          log_error_array.append(log_loss(y_test, predict_y, labels=clf.classes_,
          print('For values of alpha = ', i, "The log loss is:",log_loss(y_test,

           ax = plt.subplots()
          lot(alpha, log_error_array,c='g')
          i, txt in enumerate(np.round(log_error_array,3)):
          ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[i]))
          grid()
          title("Cross Validation Error for each alpha")
          xlabel("Alpha i's")
          ylabel("Error measure")
          show()


          _alpha = np.argmin(log_error_array)
          = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='log', rand
          fit(x_train, y_train)
          clf = CalibratedClassifierCV(clf, method="sigmoid")
          clf.fit(x_train, y_train)
```

```
ict_y = sig_clf.predict_proba(x_train)
t('For values of best alpha = ', alpha[best_alpha], "The train log loss
ict_y = sig_clf.predict_proba(x_test)
t('For values of best alpha = ', alpha[best_alpha], "The test log loss
icted_y =np.argmax(predict_y,axis=1)
t("Total number of data points :", len(predicted_y))
_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.45222282457073765
For values of alpha =  0.0001 The log loss is: 0.4438136398168544
For values of alpha =  0.001 The log loss is: 0.44421058215133563
For values of alpha =  0.01 The log loss is: 0.44406531462283577
For values of alpha =  0.1 The log loss is: 0.45716077943043243
For values of alpha =  1 The log loss is: 0.4917796143696626
For values of alpha =  10 The log loss is: 0.5452455477724563
```



```
For values of best alpha =  0.0001 The train log loss is: 0.44614785
34986376
For values of best alpha =  0.0001 The test log loss is: 0.443813639
8168544
Total number of data points : 121287
```

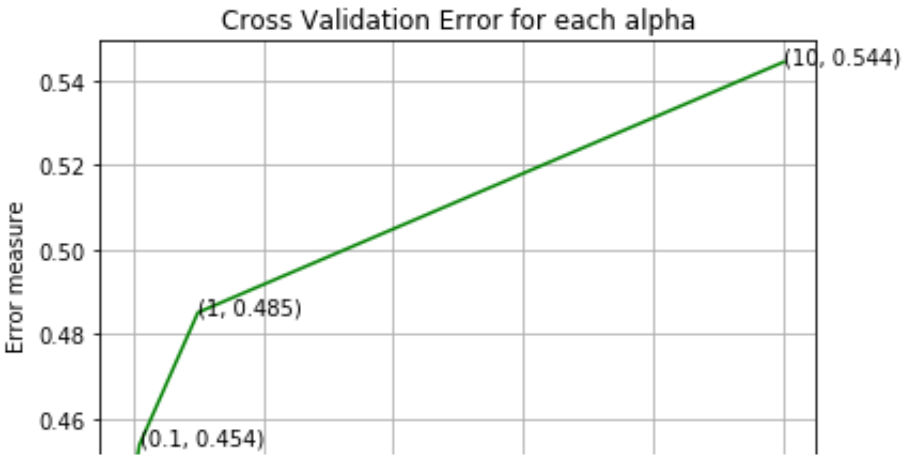


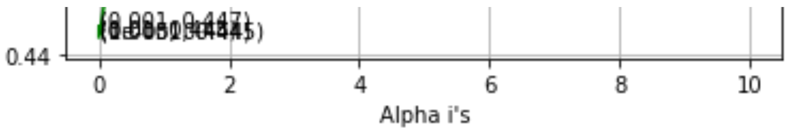
In [290]:

```
#SVM

alpha = [10 ** x for x in range(-5, 2)] # hyperparam for SGD classifie

log_error_array=[]
```

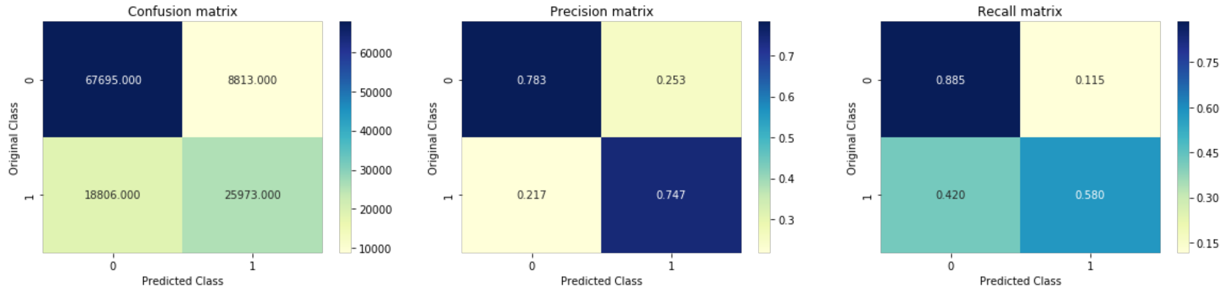
```
for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='hinge', random_st
    clf.fit(train, y_train)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    sig_clf.fit(x_train, y_train)
    predict_y = sig_clf.predict_proba(x_test)
    log_error_array.append(log_loss(y_test, predict_y, labels=clf.clas
    print('For values of alpha = ', i, "The log loss is:",log_loss(y_t

fig, ax = plt.subplots()
ax.plot(alpha, log_error_array,c='g')
for i, txt in enumerate(np.round(log_error_array,3)):
    ax.annotate((alpha[i],np.round(txt,3)), (alpha[i],log_error_array[
plt.grid()
plt.title("Cross Validation Error for each alpha")
plt.xlabel("Alpha i's")
plt.ylabel("Error measure")
plt.show()


best_alpha = np.argmin(log_error_array)
clf = SGDClassifier(alpha=alpha[best_alpha], penalty='l2', loss='hinge
clf.fit(x_train, y_train)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(x_train, y_train)

predict_y = sig_clf.predict_proba(x_train)
print('For values of best alpha = ', alpha[best_alpha], "The train log
predict_y = sig_clf.predict_proba(x_test)
print('For values of best alpha = ', alpha[best_alpha], "The test log
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
For values of alpha =  1e-05 The log loss is: 0.4443182937236933
For values of alpha =  0.0001 The log loss is: 0.4446676661749702
For values of alpha =  0.001 The log loss is: 0.44695080803075365
For values of alpha =  0.01 The log loss is: 0.4449931122995938
For values of alpha =  0.1 The log loss is: 0.45382778060219536
For values of alpha =  1 The log loss is: 0.48516772493285976
For values of alpha =  10 The log loss is: 0.5444155987261744
```





```
For values of best alpha =  1e-05 The train log loss is: 0.446662500
1999097
For values of best alpha =  1e-05 The test log loss is: 0.4443182937
236933
Total number of data points : 121287
```



In [293]:

```
from xgboost import XGBClassifier
import scipy.stats as sc
from sklearn.model_selection import RandomizedSearchCV,StratifiedKFold
```

In [295]:

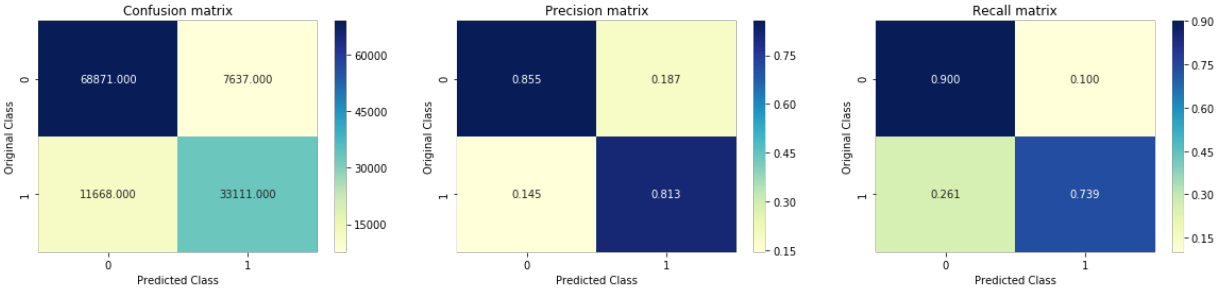
```
# Hyperparameters
learning_rate = sc.uniform(0.01,0.1)
base_learners = sc.randint(10,200)
depth = sc.randint(5,10)
min_child_weight = sc.randint(5,10)

params = {'learning_rate': learning_rate, 'n_estimators':base_learners

xgb_classifier = xgb.XGBClassifier(objective='binary:logistic')
gsv = RandomizedSearchCV(xgb_classifier, params, cv=3, scoring="neg_lo
gsv.fit(x_train,y_train)

print("Best Hyperparameter: ", gsv.best_params_)
print("Best neg_log_loss: %.2f%%",(gsv.best_score_*100))
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent w
orkers.
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 71.1min finish
ed

Best Hyperparameter:  {'learning_rate': 0.0364555612104627, 'max_dep
th': 8, 'min_child_weight': 7, 'n_estimators': 193}
Best neg_log_loss: %.2f%% -33.000159827153645
```

In [296]:

```
predict_y = gsv.predict_proba(x_train)
print("The train log loss is:",log_loss(y_train, predict_y, eps=1e-15)
predict_y = gsv.predict_proba(x_test)
print("The test log loss is:",log_loss(y_test, predict_y, eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points :", len(predicted_y))
plot_confusion_matrix(y_test, predicted_y)
```

```
The train log loss is: 0.31688220098944414
The test log loss is: 0.32862519851050037
Total number of data points : 121287
```



In [319]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ['Model Name', 'Tokenizer', 'Test Log Loss']
x.add_row(["Random model","TFIDF Weighted W2V","0.88537"])
x.add_row(["Logistic Regression","TFIDF Weighted W2V","0.41318"])
x.add_row(["Linear SVM","TFIDF Weighted W2V","0.414418"])
x.add_row(["XG BOOST","TFIDF Weighted W2V","0.35584"])
x.add_row(["Random model","TFIDF","0.69783"])
x.add_row(["Logistic Regression","TFIDF","0.44381"])
x.add_row(["Linear SVM","TFIDF","0.444318"])
x.add_row(["XG BOOST","TFIDF","0.32862"])

print(x)
```

```
+---------------------+--------------------+---------------+
|      Model Name     |     Tokenizer      | Test Log Loss |
+---------------------+--------------------+---------------+
|     Random model    | TFIDF Weighted W2V |    0.88537    |
| Logistic Regression | TFIDF Weighted W2V |    0.41318    |
|      Linear SVM     | TFIDF Weighted W2V |    0.414418   |
|       XG BOOST      | TFIDF Weighted W2V |    0.35584    |
|     Random model    |       TFIDF        |    0.69783    |
| Logistic Regression |       TFIDF        |    0.44381    |
|      Linear SVM     |       TFIDF        |    0.444318   |
|       XG BOOST      |       TFIDF        |    0.32862    |
+---------------------+--------------------+---------------+
```

Steps followed

1) FOR TF-IDF Weighted W2V Featurization

```
   a) We build a random model, which gives a bench mark that t
he other models should perform better than the
random model. The test log loss is 0.88537

   b) Model 1 is built using Logistic Regression algorithm and
we got a Log loss of 0.41318.

   c) Model 2 is built using Linear Regression algorithm and w
e got a Log loss of 0.414418.

   d) Model 3 is built using XG BOOST algorithm and we got a L
og loss of 0.35584.
```

2) For TF-IDF Featurization

```
    a) We build a random model, which gives a bench mark that
the other models should perform better than the
random model. The test log loss is 0.69783

   b) Model 1 is built using Logistic Regression algorithm and
we got a Log loss of 0.44381.

   c) Model 2 is built using Linear Regression algorithm and w
e got a Log loss of 0.444381.

   d) Model 3 is built using XG BOOST algorithm and we got a L
og loss of 0.32862.
```

Conclusion

1) The model was featurized using TFIDF Weighted W2V featurization and TFIDF featurization.

2) In TFIDF Weighted W2V featurization, XG BOOST algorithm performed the best with a log loss of 0.35584.

3) In TFIDF featurization, XG BOOST algorithm performed the best with a log loss of 0.32862.