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.

In [1]:

```
# READING SOME OF THE USEFUL PACKAGES
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# TO EXTRACT SOME MORE ADVANCED FEATURES AND ANALYSIZE MORE IN DETAIL
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
# nltk.download('punkt')   Package punkt is already up-to-date!
# nltk.download("stopwords")    Package stopwords is already up-to-date!

from nltk.stem import WordNetLemmatizer
# nltk.download('wordnet')  Package wordnet is already up-to-date!

# for visualizaton stuff
from wordcloud import WordCloud,STOPWORDS

# IMPORT FUZZYWUZZY
from fuzzywuzzy import fuzz

# for visualizaton stuff
from sklearn.manifold import TSNE
from sklearn.preprocessing import MinMaxScaler
```

```
C:\Users\Danish\AppData\Roaming\Python\Python37\site-pack
ages\pandas\compat\_optional.py:138: UserWarning: Pandas
requires version '2.7.0' or newer of 'numexpr' (version
'2.6.9' currently installed).
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\fuzzywuzzy\fuz
z.py:11: UserWarning: Using slow pure-python SequenceMatc
her. Install python-Levenshtein to remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher.
Install python-Levenshtein to remove this warning')
```

In [2]:

```
# READING THE DATA
Qd = pd.read_csv(r"Z:\DS DATA\train.csv")
Qd.head()

import time
start_time =time.clock()

print(time.clock()-start_time,"seconds")
```

Out[2]:

	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 [3]:

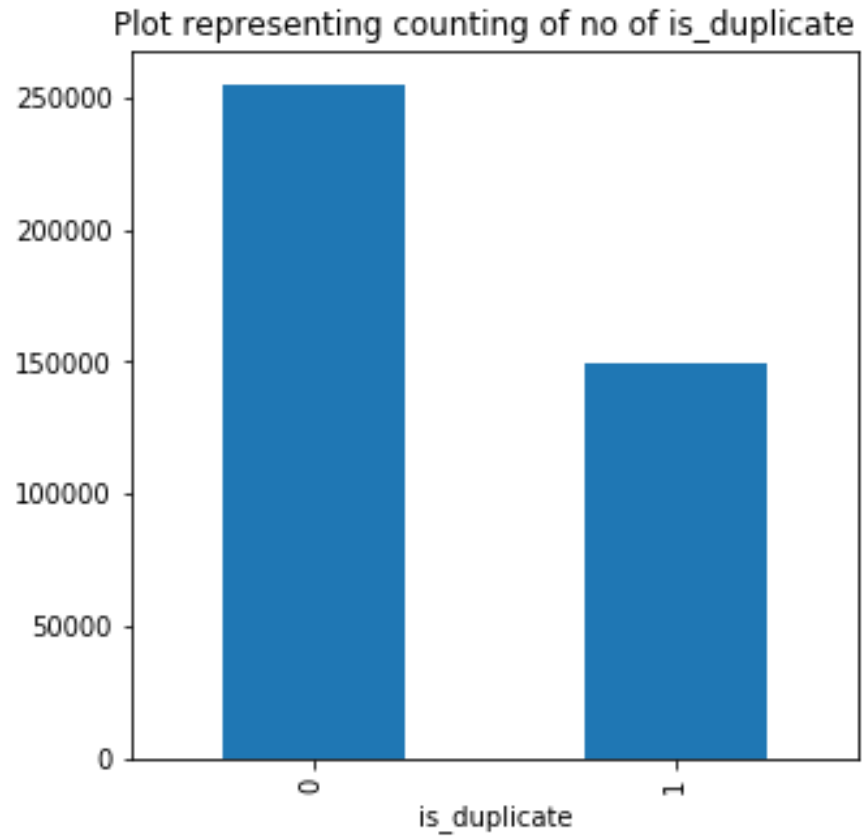
```
Qd.info()
```

```
<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
```

Distribution of Data

In [4]:

```
# RATION OF THE DUPLICATE AND NON DUPLICATE QUESTION
Qd['is_duplicate'].value_counts()
plt.figure(figsize=(5,5))
plt.title ("Plot representing counting of no of is_duplicate  ")
Qd.groupby("is_duplicate")['id'].count().plot.bar()
plt.show()
```



In [5]:

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

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

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

Some basic Analysis on the Bases of qid1 and qid2

In [6]:

```
Qids = pd.Series(Qd['qid1'].tolist()+Qd['qid2'].tolist())
```

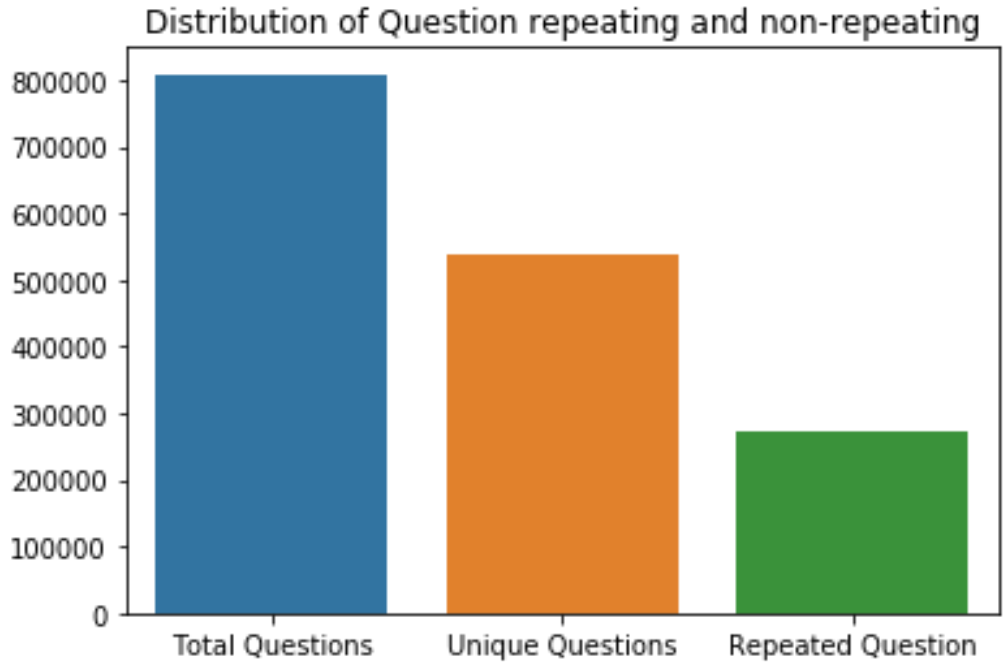
In [7]:

```
print("Total number of question are:",len(Qids),'\n')
print("Total number of question which are unique are: {}".format(len(Qi
print("Total number of question which Occure more than one times: {} : (
    np.sum(Qids.value_counts()>1),
    np.sum(Qids.value_counts()>1)/len(Qids.unique())*100),'\n')
print("Heighest a question occured is:",max(Qids.value_counts()))
```

```
Total number of question are: 808580

Total number of question which are unique are: 537933

Total number of question which Occure more than one time
s: 111780 : (20.77953945937505%)

Heighest a question occured is: 157
```

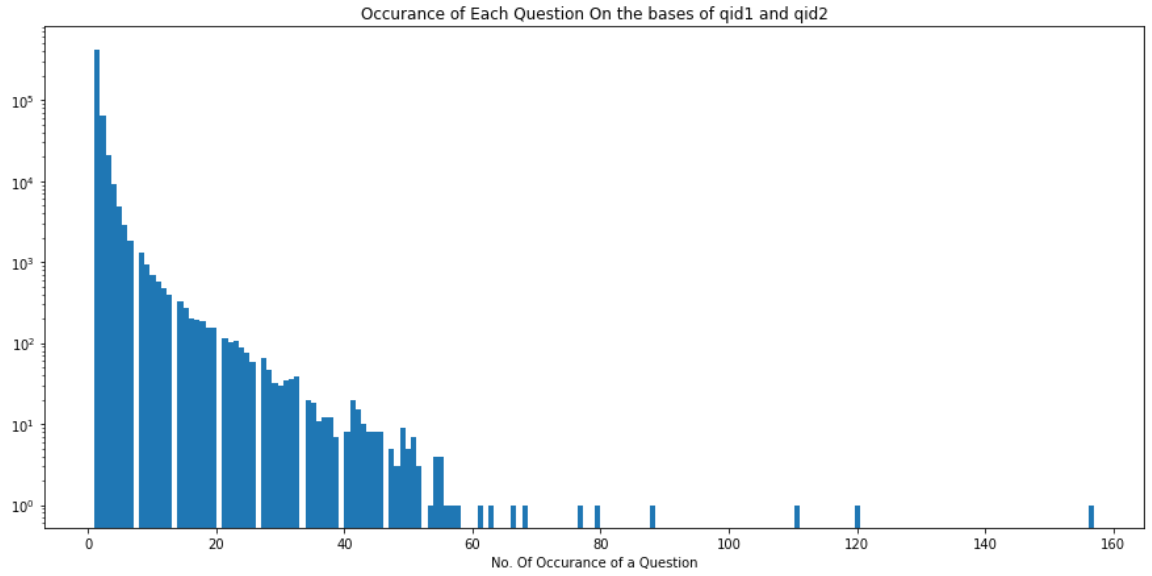
In [8]:

```
x=["Total Questions","Unique Questions","Repeated Question"]
y=[len(Qids),len(Qids.unique()),len(Qids)-len(Qids.unique())]
plt.title("Distribution of Question repeating and non-repeating")
sns.barplot(x,y)
plt.show()
```



In [9]:

```
plt.figure(figsize=(15,7))
plt.hist(Qids.value_counts(),bins=180)
plt.yscale('log', nonposy='clip')
plt.xlabel('No. Of Occurance of a Question')
plt.title("Occurance of Each Question On the bases of qid1 and qid2")
plt.show()
print("{} is the Maximum a single Question has been Repeated".format(ma
```




```
157 is the Maximum a single Question has been Repeated
```

In [10]:

```
Qd['question1']=Qd['question1'].apply(lambda x: str(x).lower())

Qd['question2']=Qd['question2'].apply(lambda x: str(x).lower())
```

Some basic Feature Extraction

Freq_qid1:----> Frequency(count) of Each question id in qid1
Freq_qid2:----> Frequency(count) of Each question id in qid2
q1_alph_len:----> No. of Alphabits in the Question 1
q2_alph_len:----> No. of Alphabits in the Question 2
q1_n_words:----> No. of words in the Question 1
q2_n_words:----> No. of words in the Question 2
total_words:----> No. of words in the Question 1 + Question 2

q1_union_q2:---->length of union of words in Qustion1 and in Question2
q1_intersec_q2:---->Length of Intersection of words in Qustion1 and in Question2
fq1_pls_fq2:---->Sum of frequences of question 1 and question 2
fq1_diff_fq2:---->difference of frequences of question 1 and question 2
inter_union:---->Intersection of Q1 and Q1 is divided by Union of Q1 and Q2

In [11]:

```
# Frequency of qid1 and qid2 in
Qd['freq_qid1'] = Qd.groupby(Qd["qid1"])["qid1"].transform("count")
Qd['freq_qid2'] = Qd.groupby(Qd["qid2"])["qid2"].transform("count")

Qd["q1_total_words"] = Qd['question1'].apply(lambda row: 0 if(type(row)=
Qd["q2_total_words"] = Qd['question2'].apply(lambda row: 0 if(type(row)=

# counts of word in each Questions (unique)
Qd['q1_n_words'] = Qd["question1"].apply(lambda x: len(set(x.split())))
Qd['q2_n_words'] = Qd["question2"].apply(lambda x: len(set(x.split())))

# Union of Question 1 and Question 2
Qd["q1_union_q2"] = Qd.apply(lambda row: (len(set(row['question1'].split
    set(row['question2'].split(" "))))),axis=1)

# Intersection of Question 1 and Question 2
Qd["q1_q2_word_share"] = Qd.apply(lambda w: len(
    set(str(w['question1']).split(" ")) & set(str(w['question2']).split(
),axis=1)

#total number of words in
Qd["total_words"] = Qd.apply(lambda w: (len(set(w["question1"].split()))

#Words share of Intersectioned words outof Union words of Q1 and Q2
Qd['word_share']= (Qd["q1_q2_word_share"]/Qd["total_words"])*1.0


# Intersection Divided by Union of Words----
Qd["intr_by_union"] = (Qd["q1_q2_word_share"]/Qd["q1_union_q2"])*1.0


# sum of Frequency of Q1 and Q2
Qd["fq1_pls_fq2"] = Qd['freq_qid1']+Qd['freq_qid2']

# Difference of Frequency of Q1 and Q2
Qd["fq1_diff_fq2"] = abs(Qd['freq_qid1']-Qd['freq_qid2'])
```

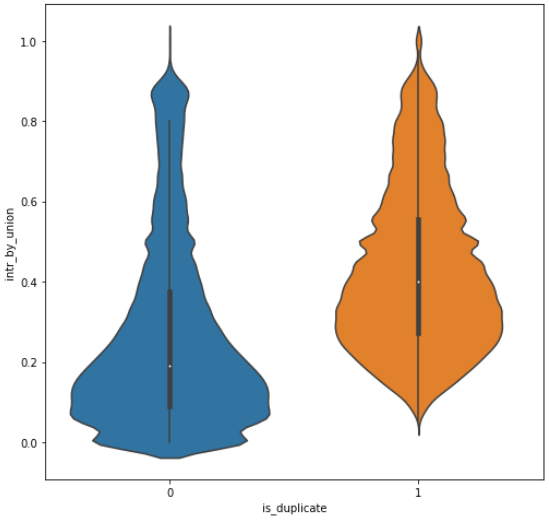
In [12]:

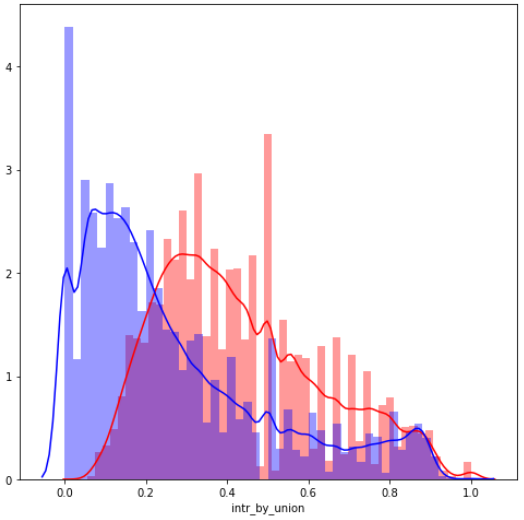
```
# OUR NEW CREATED FEATURES TILL NOW
Qd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 19 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         404290 non-null  object 
 4   question2         404290 non-null  object 
 5   is_duplicate      404290 non-null  int64  
 6   freq_qid1         404290 non-null  int64  
 7   freq_qid2         404290 non-null  int64  
 8   q1_total_words    404290 non-null  int64  
 9   q2_total_words    404290 non-null  int64  
 10  q1_n_words        404290 non-null  int64  
 11  q2_n_words        404290 non-null  int64  
 12  q1_union_q2       404290 non-null  int64  
 13  q1_q2_word_share  404290 non-null  int64  
 14  total_words       404290 non-null  int64  
 15  word_share        404290 non-null  float64
 16  intr_by_union     404290 non-null  float64
 17  fq1_pls_fq2       404290 non-null  int64  
 18  fq1_diff_fq2      404290 non-null  int64  
dtypes: float64(2), int64(15), object(2)
memory usage: 58.6+ MB
```

In [13]:

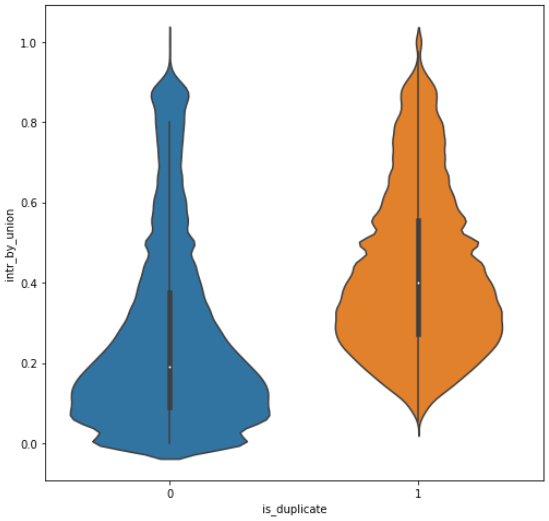
```
plt.figure(figsize=(18, 8))

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

plt.subplot(1,2,2)
sns.distplot(Qd[Qd['is_duplicate'] == 1.0]['intr_by_union'][0:] , label
sns.distplot(Qd[Qd['is_duplicate'] == 0.0]['intr_by_union'][0:] , label
plt.show()
```

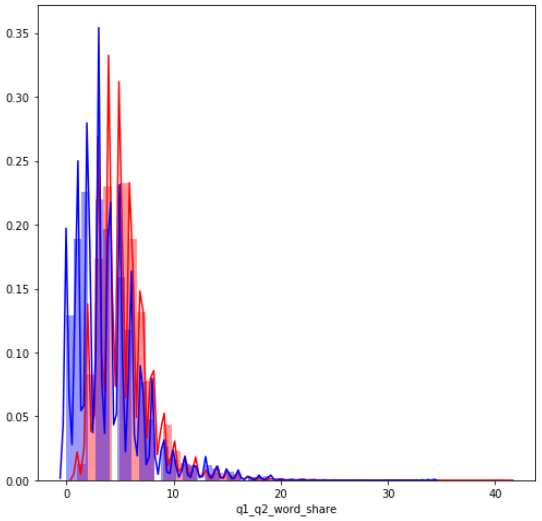




In [14]:

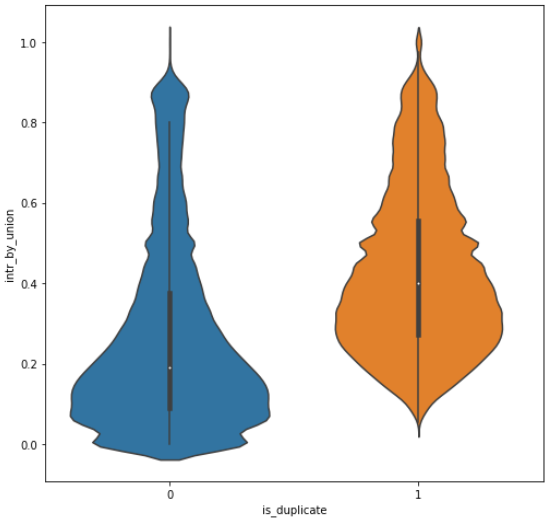
```
plt.figure(figsize=(18, 8))

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

plt.subplot(1,2,2)
sns.distplot(Qd[Qd['is_duplicate'] == 1.0]['q1_q2_word_share'][0:] , lab
sns.distplot(Qd[Qd['is_duplicate'] == 0.0]['q1_q2_word_share'][0:] , lab
plt.show()
```

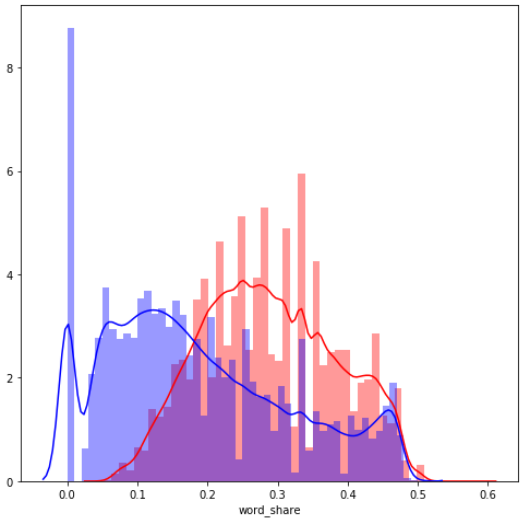




In [15]:

```
plt.figure(figsize=(18, 8))

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

plt.subplot(1,2,2)
sns.distplot(Qd[Qd['is_duplicate'] == 1.0]['word_share'][0:] , label = '
sns.distplot(Qd[Qd['is_duplicate'] == 0.0]['word_share'][0:] , label = '
plt.show()
```





from this it is clear that not all but these features which i created are useful. As they are not seperating text completely but to some extend there is some value that can be useful to classifiy

Some Simple text cleaning

Preprocessing the Text

-Preprocessing:

- Removing Html tags
- Removing Punchutations
- Performing Stemming
- Removing Stopwords
- Expanding contraction etc...

In [16]:

```
LEMATIZE = WordNetLemmatizer()
STOP_WORDS=stopwords.words("english")
```

In [17]:

```
def lematize(x):
    token=word_tokenize(x)
    lema = [LEMATIZE.lemmatize(word) for word in token ]
    y=" ".join(lema)
    return y

Qd['question1']= Qd["question1"].apply(lematize)
Qd["question2"]=Qd["question2"].apply(lematize)
```

In [18]:

```
def pre_process(x):
    #token=word_tokenize(x)
    #lema = [LEMATIZE.lemmatize(word) for word in token ]
    x=re.sub("  *"," ",re.sub("<.*?>"," ",str(x))) #for removing html ta

#     x=re.sub("  *"," ",re.sub('<[a-zA-Z]*>|</[a-zA-Z]*>',' ',str(x))).
    x=x.replace("%","percent").replace("$","dollar").replace("₹","rupees
    "i'm","i am").replace("don't","do not").replace("&","and").replace("
        "you're","you are").replace("they're","they are").replace("he's"
        "have't","have not").replace("has't","has not").replace("is't","
        "didn't","did not").replace("he'll","he will").replace("she'll",
        "it'll","it will").replace(',000',"k").replace("000,000","m").re
        "'ve","have").replace('000',"k").replace("000000","m").replace(
        ')',"").replace("€",'euro').replace("?",'').replace("`",'')

    x = re.sub(r"([0-9]+)000000", r"\1m", x)

    x = re.sub(r"([0-9]+)000", r"\1k", x)

    x=re.sub("[^a-z0-9-]"," ",x)

    return  x

Qd["question1"] = Qd["question1"].apply(pre_process)
Qd["question2"] = Qd["question2"].apply(pre_process)
```

In [19]:

```
def fill_emptly(x):
    if len(str(x['question1']))<=6:
        x['question1']=""
    if len(str(x['question2']))<=6:
        x['question2']=""
    if x["is_duplicate"]==0 and (x['question1']==""  " or x['question2']=
        x['is_duplicate']=1

    if x['question1']==" " or x['question2']==" ":
        print("yes")
    return x

Qd = Qd.apply(fill_emptly,axis=1)
```

Important Terms

-Token-: Each word after splitting the sentence is token
-Stop_Word-: Stops word that are in nltk
-Words-: token that is not in stopwords

Feature Extraction

These are some extra and advanced features

Commom_word_Count_Min (cwc_min) length of Intersection of Q1 and Q2 divide by Minimum length of Q1 and Q2 word
Commom_word_Count_Max (cwc_max) length Intersection of Q1 and Q2 divide by Maximum length of Q1 and Q2 words
Commom_StopWord_Count_min (csc_min) length Intersection of Q1 and Q2 divide by Minimum length of Q1 and Q2 stopwords
Commom_StopWord_Count_Max (cwc_max) length Intersection of Q1 and Q2 divide by Maximum length of Q1 and Q2 stopwords
Commom_Token_Count_Min (ctc_min) length Intersection of Q1 and Q2 divide by

Minimum length of Q1 and Q2 tokenwords

Commom_Token_Count_Max (ctc_max) length Intersection of Q1 and Q2 divide by Maximum length of Q1 and Q2 tokenwords

First_token_Common (fw_com) checking common token in both Questions at first place

Last_token_Common (lw_com) checking common token in both Questions at last place

Absolute difference(abs_diff) Absolute difference b/w length of Q1 and Q2 token

Middle of both question (ratio) middle value of both(Q1+Q2) question

In [20]:

```
# computing common_word_count_min cwc_min
SAFE_DIV = 0.0001
def extract_features(x):

    global SAFE_DIV
    q1_token=word_tokenize(x["question1"])
    q2_token=word_tokenize(x["question2"])

    q1_words = set([word for word in q1_token if word not in STOP_WORDS]
    q2_words = set([word for word in q2_token if word not in STOP_WORDS]

    q1_stops = set([word for word in q1_token if word in STOP_WORDS])
    q2_stops = set([word for word in q2_token if word in STOP_WORDS])

    common_word_count = len(q1_words.intersection(q2_words))
    common_stop_count = len(q1_stops.intersection(q2_stops))
    common_token_count = len(set(q1_token).intersection(set(q2_token)))

    cwc_min=0
    cwc_max=0
#     MIN MAX FOR WORDS
    cwc_max = common_word_count / (max(len(q1_words), len(q2_words)) + S
    cwc_min = common_word_count / (min(len(q1_words), len(q2_words)) + S

#     MIN MAX FOR STOP WORDS
    csc_min=0
    csc_max=0
    csc_max = common_stop_count / (max(len(q1_stops), len(q2_stops)) + S
    csc_min = common_stop_count / (min(len(q1_stops), len(q2_stops)) + S

#     MIN MAX FOR TOKENS
    ctc_min=0
    ctc_max=0
    ctc_max = common_token_count / (max(len(q1_token), len(q2_token)) +
    ctc_min = common_token_count / (min(len(q1_token), len(q2_token)) +

#     LAST WORD FIRST WORD DIFF. LAST WORD DIFF. ABSOLUTE DIFF. RATION E
    if len(q1_token)<=0 or len(q2_token)<=0:
        fw_com =0
        lw_com =0
    else:
```

```
        fw_com = int(q1_token[0]==q2_token[0])
        lw_com = int(q1_token[-1]==q2_token[-1])
    abs_diff= abs(len(q1_token)-len(q2_token))
    ratio = (len(q1_token)+len(q2_token)) / 2

    return cwc_min, cwc_max, csc_min, csc_max, ctc_min, ctc_max, fw_com,

Qd[["cwc_min","cwc_max","csc_min","csc_max","ctc_min",
    "ctc_max","fw_com","lw_com","abs_diff","ratio"]] = Qd.apply(extract_
```

In [21]:

```
def fuzzyfeatures(x):
    token_set_ratio = fuzz.token_set_ratio(x["question1"],x["question2"]
    token_sort_ratio = fuzz.token_sort_ratio(x["question1"],x["question2
    fuzz_ratio = fuzz.QRatio (x["question1"],x["question2"])
    partial_ratio = fuzz.partial_ratio(x["question1"],x["question2"])

    return token_set_ratio,token_sort_ratio,fuzz_ratio,partial_ratio

Qd[["token_set_ratio","token_sort_ratio",
    "fuzz_ratio","partial_ratio"]]=Qd.apply(fuzzyfeatures,axis=1,result_
```

In [22]:

```
Qd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 404290 entries, 0 to 404289
Data columns (total 33 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         404290 non-null  object 
 4   question2         404290 non-null  object 
 5   is_duplicate      404290 non-null  int64  
 6   freq_qid1         404290 non-null  int64  
 7   freq_qid2         404290 non-null  int64  
 8   q1_total_words    404290 non-null  int64  
 9   q2_total_words    404290 non-null  int64  
 10  q1_n_words        404290 non-null  int64  
 11  q2_n_words        404290 non-null  int64  
 12  q1_union_q2       404290 non-null  int64  
 13  q1_q2_word_share  404290 non-null  int64  
 14  total_words       404290 non-null  int64  
 15  word_share        404290 non-null  float64
 16  intr_by_union     404290 non-null  float64
 17  fq1_pls_fq2       404290 non-null  int64  
 18  fq1_diff_fq2      404290 non-null  int64  
 19  cwc_min           404290 non-null  float64
 20  cwc_max           404290 non-null  float64
 21  csc_min           404290 non-null  float64
 22  csc_max           404290 non-null  float64
 23  ctc_min           404290 non-null  float64
 24  ctc_max           404290 non-null  float64
 25  fw_com            404290 non-null  float64
 26  lw_com            404290 non-null  float64
 27  abs_diff          404290 non-null  float64
 28  ratio             404290 non-null  float64
 29  token_set_ratio   404290 non-null  int64  
 30  token_sort_ratio  404290 non-null  int64  
 31  fuzz_ratio        404290 non-null  int64  
 32  partial_ratio     404290 non-null  int64  
dtypes: float64(12), int64(19), object(2)
memory usage: 101.8+ MB
```

Some extra visualization to gain some more insights

In [23]:

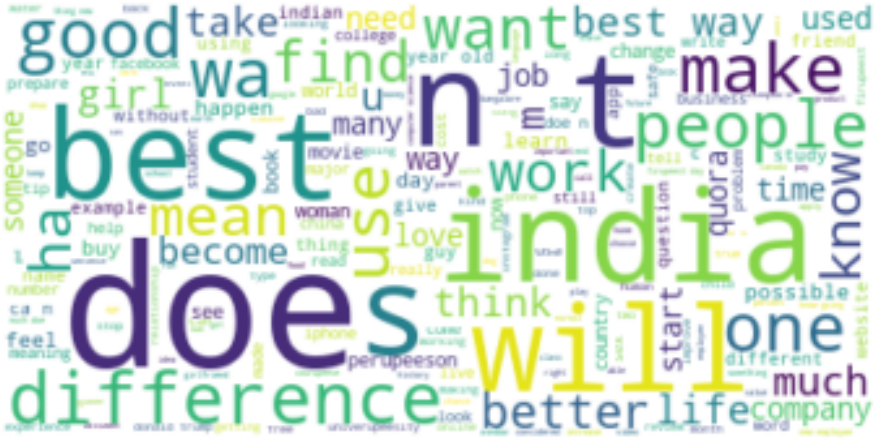
```
df_duplicate = Qd[Qd['is_duplicate']==1]
df_noduplicate = Qd[Qd['is_duplicate']==0]
```

In [24]:

```
q1=np.dstack([df_duplicate["question1"],df_duplicate["question2"]]).flat
wordcloud_spam = WordCloud(background_color="white").generate(" ".join(q
plt.imshow(wordcloud_spam, interpolation='bilinear')
plt.axis("off")
plt.show()
```

In [25]:

```
q1=np.dstack([df_noduplicate["question1"],df_noduplicate["question2"]]).
wordcloud_spam = WordCloud(background_color="white").generate(" ".join(c
# # Lines 2 - 5
# plt.figure(figsize = (10,10))
plt.imshow(wordcloud_spam, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [26]:

```
dfp_subsampled = Qd[0:5000]
X = MinMaxScaler().fit_transform(dfp_subsampled[['cwc_min', 'cwc_max',
                                                 'ctc_max' , 'fw_com',
                                                 'token_set_ratio' , 'tc
                                                 'partial_ratio' ]])
y = dfp_subsampled['is_duplicate'].values
```

In [27]:

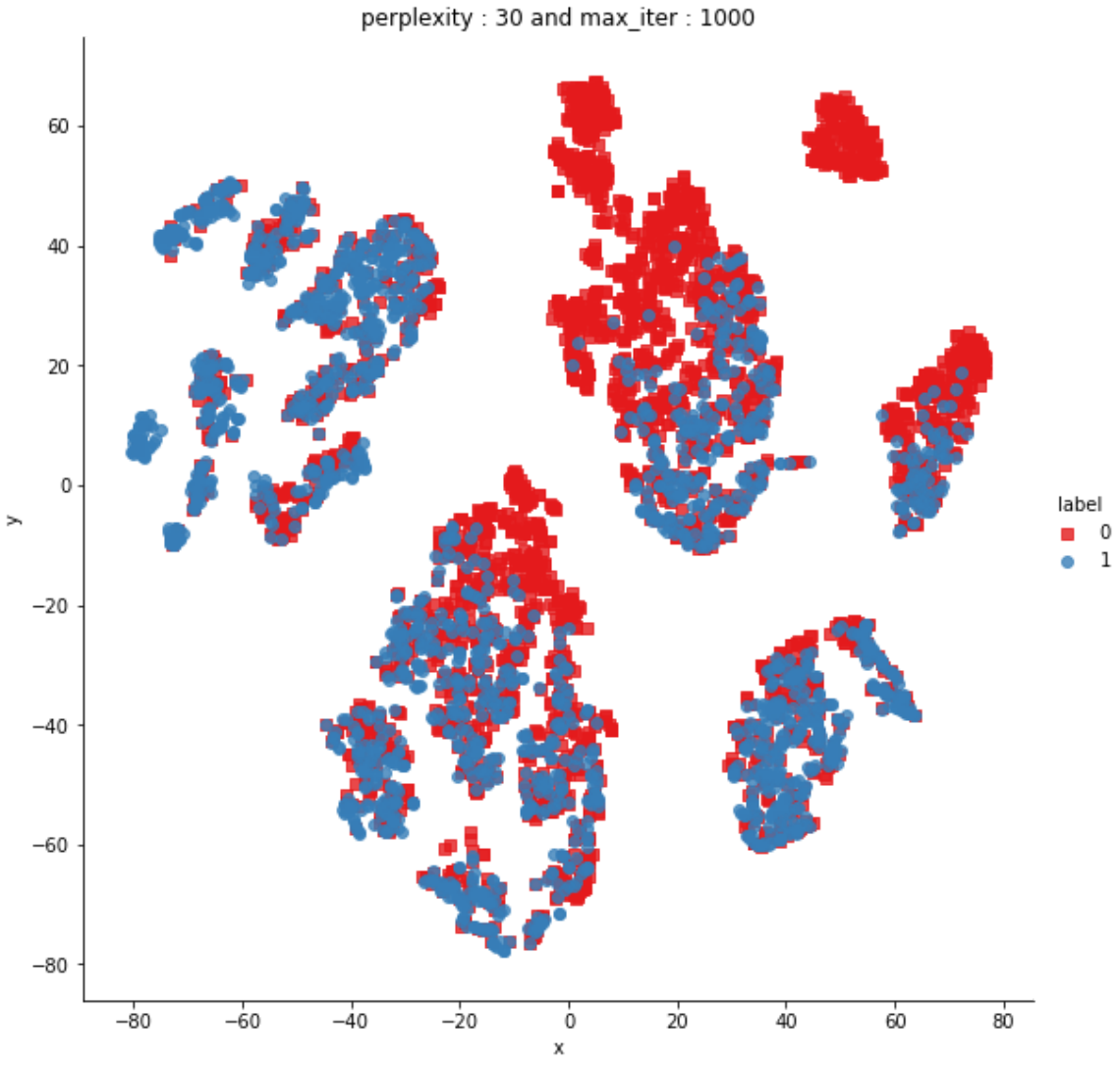
```
tsne2d = TSNE(
    n_components=2,
    init='random', # pca
    random_state=101,
    method='barnes_hut',
    n_iter=1000,
    verbose=2,
    angle=0.5
).fit_transform(X)
```

...

In [28]:

```
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,pal
plt.title("perplexity : {} and max_iter : {}".format(30, 1000))
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\regres
sion.py:546: UserWarning: The `size` paramter has been re
named to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



In [29]:

```
# Qd.to_csv(r"Z:\DS DATA\feature_train.csv",index=False)
Qd.head()
```

...

In [40]:

```
# def preprocess(x):
# #     x=re.sub("[^a-zA-Z0-9]",' ',str(x).lower())
#     x=x.replace("€",'euro').replace("%","percent").replace("$",'dollar
#         "  "," ").replace("₹","rupee").replace("ll",'will').replace("&
#         '00,000','l').replace('000,000','m').replace("don't",'do not')
#         "can't",'can not').replace("cannot",'can not').replace("it's",
#         "have't",'have not').replace("she's",'she is').replace("he's",
#         "i'm",'i am')
#     return x
# Qd['question1']=Qd['question1'].apply(preprocess)
# Qd['question2']=Qd['question2'].apply(preprocess)
# # Qd.head()
```