Find proper cleaning rules

OK we are convinced the better the questions are cleaned, the better the features around Nb common words will be powerful (even if we may have reached a limit)

So, now objective is to reduced as much as possible the fields 'uncommon words' (or extend the field 'common word')

Lot of things are possible

Chirurgical and ad hoc cleaning

- remove the ? at end. tons of common words are missed due to that
- replace all what's, it's, where's by their long versions what is, it is, where is ...
- after that and after that only, replace any punctuation by a blank to get rid of these strange "" we have seen

In [44]:
```
# Ugly incantation to make our framework working
import sys
sys.path.insert(0, r'/SAPDevelop/QuoraPairs/BruteForce/Tools')

#import all our small tools (paths, cache, print,zip,excel, pandas, progress,..)
from Tools.all import *

# setup the name of our experiment
# it will be used to store every result in a unique place
EXPERIMENT='spacy_preprocessing'
print_alert('You will work on experiment %s' %EXPERIMENT)

prepare_environnement(EXPERIMENT)
train_dataframe=load_dataframe(CLEAN_TRAINING_DATA)
challenge_dataframe=load_dataframe(CLEAN_CHALLENGE_DATA)
print_section('Untouched input data has been loaded. Training: %d lines Challenge: %d lines' % (len(trai
n_dataframe),len(challenge_dataframe)))
```

You will work on experiment spacy_preprocessing

Prepare spacy_preprocessing environment in ../spacy_preprocessing

---

Done

Untouched input data has been loaded. Training: 404290 lines Challenge: 2345796 lines

---

First step : as usual lowercase everything

In [45]:
```
# our main tool to add feature
# no progress info as we will starts to play on a small sample
def add_column_from_columns(dataframe,output_column_name,function):
    dataframe[output_column_name]=dataframe.apply(function,axis=1)
    return dataframe[output_column_name]

def add_column_from_column(dataframe,output_column_name,input_column_name,function):
    dataframe[output_column_name]=dataframe[input_column_name].apply(function)
    return dataframe[output_column_name]
```

In [46]:
```
def build_all_lower_data(dataframe):
    print_info('Lower case question1')
    dataframe['question1'] = dataframe['question1'].str.lower()
    print_info('Lower case question2')
    dataframe['question2'] = dataframe['question2'].str.lower()
    return dataframe

print_section('Before')
display(train_dataframe.head(1).transpose())
train_dataframe = load_or_build_dataframe('Lower case everything in training','training_lower',build_all
_lower_data,train_dataframe)
challenge_dataframe = load_or_build_dataframe('Lower case everything in challenge','challenge_lower',bui
ld_all_lower_data,challenge_dataframe)
print_section('After')
display(train_dataframe.head(1).transpose())
```

Before

	0
id	0
qid1	1
qid2	2
question1	What is the step by step guide to invest in share market in india?
question2	What is the step by step guide to invest in share market?
is_duplicate	0

Lower case everything in training: Load or rebuild training_lower

---

!!!!! ../spacy_preprocessing/training_lower.pkl is cached!!!

Done:training_lower contains 404290 lines in 0.1 s

Lower case everything in challenge: Load or rebuild challenge_lower

---

!!!!! ../spacy_preprocessing/challenge_lower.pkl is cached!!!

Done:challenge_lower contains 2345796 lines in 0.9 s

After

	0
id	0
qid1	1
qid2	2
question1	what is the step by step guide to invest in share market in india?
question2	what is the step by step guide to invest in share market?
is_duplicate	0

We will start by exploring our preprocessing on a small dataset

In [47]:
```
#small_train = train_dataframe.sample(20000,random_state=42)
small_train = train_dataframe
```

In [48]:
```
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS


nltk_stopwords = set(stopwords.words('english'))
sk_stopwords = set(ENGLISH_STOP_WORDS)
all_stop_words = nltk_stopwords | sk_stopwords

def preprocess_one_row(q1,q2,stopwords):
    q1 = set([w for w in q1.split() if w not in stopwords])
    len_q1 = len(q1)
    q2 = set([w for w in q2.split() if w not in stopwords])
    len_q2 = len(q2)

    common = q1&q2
    len_common = len(common)

    uncommon_q1 = q1-common
    len_uncommon_q1 = len(uncommon_q1)

    #       0     1           2          3               4
    return common,uncommon_q1,len_common,len_uncommon_q1,len_q1

def initial_preprocess(dataframe):
    print_warning('Compute all features in one shot')
    add_column_from_columns(dataframe,'temp',lambda r: preprocess_one_row(r.question1,r.question2,all_st
op_words))

    print_warning('Extract common words between question1 & question2')
    add_column_from_column(dataframe,'common_words','temp',lambda x: x[0])

    print_warning('Extract uncommon words in question1')
    add_column_from_column(dataframe,'uncommon_words_question1','temp',lambda x: x[1])

    print_warning('Extract Nb common_words between question1 & question2')
    add_column_from_column(dataframe,'nb_common_words','temp',lambda x: x[2])

    print_warning('Extract Nb words in question1 not in common words')
    add_column_from_column(dataframe,'nb_uncommon_words','temp',lambda x: x[3])

    print_warning('Extract nb_words_question1')
    add_column_from_column(dataframe,'nb_words_question1','temp',lambda x: x[4])
    dataframe = dataframe.drop(columns='temp')
    return dataframe

def sniff_changes(dataframe):
    nb_all_common = dataframe['nb_common_words'].sum()
    nb_all_uncommon = dataframe['nb_uncommon_words'].sum()
    if 'new_nb_common_words' in dataframe and 'new_nb_uncommon_words' in dataframe:
        new_nb_all_common = dataframe['new_nb_common_words'].sum()
        new_nb_all_uncommon = dataframe['new_nb_uncommon_words'].sum()
        print_info( "New common %.3f %% New uncommon %.3f %%" % (100.*new_nb_all_common/nb_all_common,10
0.*new_nb_all_uncommon/nb_all_uncommon))
    else:
        print_warning('??')
```

In [49]:
```
small_train = initial_preprocess(small_train)
display(small_train[['common_words','uncommon_words_question1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

Extract nb_words_question1

	common_words	uncommon_words_question1
0	{share, guide, invest, step}	{india?, market}
1	{(koh-i-noor), kohinoor}	{story, diamond?}
2	{internet, speed}	{connection, vpn?, using, increase}
3	{}	{solve, mentally, it?, lonely?}
4	{}	{methane, sugar,, oxide?, water, carbon, dissolve, salt,, di, quikly}
5	{capricorn, say, moon, me?}	{astrology:, cap, rising...what, sun}
6	{}	{tiago?, buy}
7	{geologist?}	{good}
8	{use, instead}	{▯, ▯?}
9	{hack, motorola}	{(company):, motorolla, dcx3400?, charter}

??

- Remove all these question marks
-

In [50]:
```
# pattern to easily iterate

def preprocess_one_row(q1,q2,stopwords):
    q1 = clean_text(q1)
    q2 = clean_text(q2)

    q1 = set([w for w in q1.split() if w not in stopwords])
    len_q1 = len(q1)
    q2 = set([w for w in q2.split() if w not in stopwords])
    len_q2 = len(q2)

    common = q1&q2
    len_common = len(common)

    uncommon_q1 = q1-common
    len_uncommon_q1 = len(uncommon_q1)

    #       0     1           2          3               4
    return common,uncommon_q1,len_common,len_uncommon_q1,len_q1

def new_preprocess(dataframe):
    print_warning('Compute all features in one shot')
    add_column_from_columns(dataframe,'temp',lambda r: preprocess_one_row(r.question1,r.question2,all_st
op_words))

    print_warning('Extract common words between question1 & question2')
    add_column_from_column(dataframe,'new_common_words','temp',lambda x: x[0])

    print_warning('Extract uncommon words in question1')
    add_column_from_column(dataframe,'new_uncommon_words_question1','temp',lambda x: x[1])

    print_warning('Extract Nb common_words between question1 & question2')
    add_column_from_column(dataframe,'new_nb_common_words','temp',lambda x: x[2])

    print_warning('Extract Nb words in question1 not in common words')
    add_column_from_column(dataframe,'new_nb_uncommon_words','temp',lambda x: x[3])

    dataframe = dataframe.drop(columns='temp')
    return dataframe
```

In [51]:
```
import re

def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.329 % New uncommon 88.429 %

In [52]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.364 % New uncommon 88.365 %

In [53]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub("\'s", " ", text)
    text = re.sub(" whats ", " what is ", text)
    text = re.sub("\'ve", " have ", text)
    text = re.sub("can't", "can not", text)
    text = re.sub("n't", " not ", text)
    text = re.sub("i'm", "i am", text)
    text = re.sub("\'re", " are ", text)
    text = re.sub("\'d", " would ", text)
    text = re.sub("\'ll", " will ", text)
    text = re.sub("e\.g\.", " eg ", text)
    text = re.sub("b\.g\.", " bg ", text)
    text = re.sub("e-mail", " email ", text)
    text = re.sub("(the[\s]+|The[\s]+)?U\.S\.A\.", " America ", text)
    text = re.sub("(the[\s]+|The[\s]+)?United State(s)?", " America ", text)
#    text = re.sub("\(s\)", " ", text)
#    text = re.sub("[c-fC-F]\:\/", " disk ", text)
#    text = re.sub("(\d+)(kK)", " \g<1>000 ", text)

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.678 % New uncommon 87.088 %

In [54]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub('’', "'", text) # special single quote
    text = re.sub('`', "'", text) # special single quote
    text = re.sub('“', '"', text) # special double quote
    text = re.sub('▯', '?', text)
    text = re.sub('…', ' ', text)
    text = re.sub('é', 'e', text)

    # shortcuts
    text = re.sub('\'s', ' ', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('(the[\s]+|The[\s]+)?U\.S\.A\.', ' america ', text)
    text = re.sub('(the[\s]+|The[\s]+)?United State(s)?', ' america ', text)
#    text = re.sub('\(s\)', ' ', text)
#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)

    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)
    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.683 % New uncommon 87.086 %

In [55]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' ', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('(the[\s]+|The[\s]+)?U\.S\.A\.', ' america ', text)
    text = re.sub('(the[\s]+|The[\s]+)?United State(s)?', ' america ', text)
#    text = re.sub('\(s\)', ' ', text)
#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.687 % New uncommon 87.083 %

In [56]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' ', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('(the[\s]+|The[\s]+)?U\.S\.A\.', ' america ', text)
    text = re.sub('(the[\s]+|The[\s]+)?United State(s)?', ' america ', text)
#    text = re.sub('\(s\)', ' ', text)
#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    # Quora is very used in India so roupie (rs) is often present
    text = re.sub("(?<=[0-9])rs ", " rs ", text)
    text = re.sub(" rs(?=[0-9])", " rs ", text)
    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.702 % New uncommon 87.082 %

In [57]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' ', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('\(s\)', ' ', text)
#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    # Quora is very used in India so roupie (rs) is often present
    text = re.sub("(?<=[0-9])rs ", " rs ", text)
    text = re.sub(" rs(?=[0-9])", " rs ", text)

    # stolen at kaggle : https://www.kaggle.com/currie32/the-importance-of-cleaning-text

    # very weird !!! these ones decrease the hit % WTF ?

    #text = re.sub(r" (the[\s]+|the[\s]+)?us(a)? ", " usa ", text)
    #text = re.sub('(the[\s]+|the[\s]+)?united state(s)?', ' usa ', text)

    text = re.sub(r" UK ", " england ", text)
    text = re.sub(r" imrovement ", " improvement ", text)
    text = re.sub(r" intially ", " initially ", text)
    text = re.sub(r" dms ", " direct messages ", text)
    text = re.sub(r" demonitization ", " demonetization ", text)
    text = re.sub(r" actived ", " active ", text)
    text = re.sub(r" kms ", " kilometers ", text)
    text = re.sub(r" cs ", " computer science ", text)
    text = re.sub(r" upvote", " up vote", text)
    text = re.sub(r" iphone ", " phone ", text)
    text = re.sub(r" \0rs ", " rs ", text)
    text = re.sub(r" calender ", " calendar ", text)
    text = re.sub(r" ios ", " operating system ", text)
    text = re.sub(r" programing ", " programming ", text)
    text = re.sub(r" bestfriend ", " best friend ", text)
    text = re.sub(r" iii ", " 3 ", text)
    text = re.sub(r" banglore ", " bangalore ", text)
    text = re.sub(r" j k ", " jk ", text)
    text = re.sub(r" J\.K\. ", " jk ", text)

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 111.770 % New uncommon 87.056 %

In [58]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' ', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('\(s\)', ' ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    # Quora is very used in India so roupie (rs) is often present
    text = re.sub("(?<=[0-9])rs ", " rs ", text)
    text = re.sub(" rs(?=[0-9])", " rs ", text)

    # stolen at kaggle : https://www.kaggle.com/currie32/the-importance-of-cleaning-text

#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)
    # very weird !!! these ones decrease the hit % WTF ?

    #text = re.sub(r" (the[\s]+|the[\s]+)?us(a)? ", " usa ", text)
    #text = re.sub('(the[\s]+|the[\s]+)?united state(s)?', ' usa ', text)

    text = re.sub(r" uk ", " england ", text)
    text = re.sub(r" imrovement ", " improvement ", text)
    text = re.sub(r" intially ", " initially ", text)
    text = re.sub(r" dms ", " direct messages ", text)
    text = re.sub(r" demonitization ", " demonetization ", text)
    text = re.sub(r" actived ", " active ", text)
    text = re.sub(r" kms ", " kilometers ", text)
    text = re.sub(r" cs ", " computer science ", text)
    text = re.sub(r" upvote", " up vote", text)
    text = re.sub(r" iphone ", " phone ", text)
    text = re.sub(r" \0rs ", " rs ", text)
    text = re.sub(r" calender ", " calendar ", text)
    text = re.sub(r" ios ", " operating system ", text)
    text = re.sub(r" programing ", " programming ", text)
    text = re.sub(r" bestfriend ", " best friend ", text)
    text = re.sub(r" iii ", " 3 ", text)
    text = re.sub(r" banglore ", " bangalore ", text)
    text = re.sub(r" j k ", " jk ", text)
    text = re.sub(r" J\.K\. ", " jk ", text)


    # some others
    text = re.sub(r"60k", " 60000 ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"\s{2,}", " ", text)
    text = re.sub(r" usa ", " America ", text)
    text = re.sub(r" u s ", " America ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r" m ", " am ", text)
    # Now we can remove punctuation but not all !
    # we keep - and @ for later maybe
    text = ''.join([c for c in text if c not in '!"#$%&\'()*+,./:;<=>?[\\]^_`{|}~'])
    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 115.516 % New uncommon 82.359 %

In [59]:
```
def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("▯", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' is', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    # this one is tricky do it in order
    text = re.sub("wouldn't", 'would not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('\(s\)', ' ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    # Quora is very used in India so roupie (rs) is often present
    text = re.sub("(?<=[0-9])rs ", " rs ", text)
    text = re.sub(" rs(?=[0-9])", " rs ", text)

    # stolen at kaggle : https://www.kaggle.com/currie32/the-importance-of-cleaning-text

#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)
    # very weird !!! these ones decrease the hit % WTF ?

    #text = re.sub(r" (the[\s]+|the[\s]+)?us(a)? ", " usa ", text)
    #text = re.sub('(the[\s]+|the[\s]+)?united state(s)?', ' usa ', text)

    text = re.sub(r" uk ", " england ", text)
    text = re.sub(r" imrovement ", " improvement ", text)
    text = re.sub(r" intially ", " initially ", text)
    text = re.sub(r" dms ", " direct messages ", text)
    text = re.sub(r" demonitization ", " demonetization ", text)
    text = re.sub(r" actived ", " active ", text)
    text = re.sub(r" kms ", " kilometers ", text)
    text = re.sub(r" cs ", " computer science ", text)
    text = re.sub(r" upvote", " up vote", text)
    text = re.sub(r" iphone ", " phone ", text)
    text = re.sub(r" \0rs ", " rs ", text)
    text = re.sub(r" calender ", " calendar ", text)
    text = re.sub(r" ios ", " operating system ", text)
    text = re.sub(r" programing ", " programming ", text)
    text = re.sub(r" bestfriend ", " best friend ", text)
    text = re.sub(r" iii ", " 3 ", text)
    text = re.sub(r" banglore ", " bangalore ", text)
    text = re.sub(r" j k ", " jk ", text)
    text = re.sub(r" J\.K\. ", " jk ", text)


    # some others
    text = re.sub(r"60k", " 60000 ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"\s{2,}", " ", text)
    text = re.sub(r" usa ", " America ", text)
    text = re.sub(r" u s ", " America ", text)
    text = re.sub(r"'m ", " am ", text)

    # Now we can remove punctuation but not all !
    # we keep - and @ for later maybe
    text = ''.join([c for c in text if c not in '!"#$%&\'()*+,./:;<=>?[\\]^_`{|}~'])

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_q
uestion1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 115.516 % New uncommon 82.360 %

I changed my mind

Instead of replacing punctuation by nothing, we should surround it with blanks. Otherwise we ar just generating more unknown words

Example monday,tuesday i will go swimming.sure! -> MondayTuesday i will go swimmingsure monday,tuesday i will go swimming.sure! -> monday tuesday i will go swimming sure

In [61]:
```python
FINAL_PUNC_CLEANER = str.maketrans(dict([ (c,' ') for c in '!"#$%&\'()*+,./:;<=>?[\\]^_`{|}~-@']))

def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    # will generate more ' so do it first
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("□", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' is', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    # this one is tricky do it in order
    text = re.sub("wouldn't", 'would not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('\(s\)', ' ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    # Quora is very used in India so roupie (rs) is often present
    text = re.sub("(?<=[0-9])rs ", " rs ", text)
    text = re.sub(" rs(?=[0-9])", " rs ", text)

    # stolen at kaggle : https://www.kaggle.com/currie32/the-importance-of-cleaning-text

#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)
    # very weird !!! these ones decrease the hit % WTF ?

    #text = re.sub(r" (the[\s]+|the[\s]+)?us(a)? ", " usa ", text)
    #text = re.sub('(the[\s]+|the[\s]+)?united state(s)?', ' usa ', text)

    text = re.sub(r" uk ", " england ", text)
    text = re.sub(r" imrovement ", " improvement ", text)
    text = re.sub(r" intially ", " initially ", text)
    text = re.sub(r" dms ", " direct messages ", text)
    text = re.sub(r" demonitization ", " demonetization ", text)
    text = re.sub(r" actived ", " active ", text)
    text = re.sub(r" kms ", " kilometers ", text)
    text = re.sub(r" cs ", " computer science ", text)
    text = re.sub(r" upvote", " up vote", text)
    text = re.sub(r" iphone ", " phone ", text)
    text = re.sub(r" \0rs ", " rs ", text)
    text = re.sub(r" calender ", " calendar ", text)
    text = re.sub(r" ios ", " operating system ", text)
    text = re.sub(r" programing ", " programming ", text)
    text = re.sub(r" bestfriend ", " best friend ", text)
    text = re.sub(r" iii ", " 3 ", text)
    text = re.sub(r" banglore ", " bangalore ", text)
    text = re.sub(r" j k ", " jk ", text)
    text = re.sub(r" J\.K\. ", " jk ", text)


    # some others
    text = re.sub(r"60k", " 60000 ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"\s{2,}", " ", text)
    text = re.sub(r" usa ", " America ", text)
    text = re.sub(r" u s ", " America ", text)
    text = re.sub(r"'m ", " am ", text)

    # will blank any of !"#$%&\'()*+,./:;<=>?[\\]^_`{|}~-@
    # Note the @ is dubious : won't we loose some emails
    text = text.translate(FINAL_PUNC_CLEANER)

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_question1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 118.768 % New uncommon 82.289 %

In [64]:
```python
# I do special stuff with $ and roupie char
FINAL_PUNC_CLEANER = str.maketrans(dict([ (c,' ') for c in '!"#%&\'()*+,./:;<=>?[\\]^_`{|}~-@']))

def clean_text(text):
    text = re.sub('\?',' ',text) # ?
    # odd chars
    # will generate more ' so do it first
    text = re.sub("’", "'", text) # special single quote
    text = re.sub("`", "'", text) # special single quote
    text = re.sub("“", '"', text) # special double quote
    text = re.sub("□", "?", text)
    text = re.sub("…", " ", text)
    text = re.sub("é", "e", text)

    # shortcuts
    text = re.sub('\'s', ' is', text)
    text = re.sub(' whats ', ' what is ', text)
    text = re.sub('\'ve', ' have ', text)
    text = re.sub("can't", 'can not', text)
    # this one is tricky do it in order
    text = re.sub("wouldn't", 'would not', text)
    text = re.sub("n't", ' not ', text)
    text = re.sub("i'm", 'i am', text)
    text = re.sub('\'re', ' are ', text)
    text = re.sub('\'d', ' would ', text)
    text = re.sub('\'ll', ' will ', text)
    text = re.sub('e\.g\.', ' eg ', text)
    text = re.sub('b\.g\.', ' bg ', text)
    text = re.sub('e-mail', ' email ', text)
    text = re.sub('\(s\)', ' ', text)

    # Numbers and measures are a true mess
    # 12,000 -> 12000
    text = re.sub('(?<=[0-9])\,(?=[0-9])', '', text)

    # Quora is very used in India so roupie (rs) is often present
    text = re.sub("(?<=[0-9])rs ", " rs ", text)
    text = re.sub(" rs(?=[0-9])", " rs ", text)

    # stolen at kaggle : https://www.kaggle.com/currie32/the-importance-of-cleaning-text

#    text = re.sub('[c-fC-F]\:\/', ' disk ', text)
#    text = re.sub('(\d+)(kK)', ' \g<1>000 ', text)
    # very weird !!! these ones decrease the hit % WTF ?

    #text = re.sub(r" (the[\s]+|the[\s]+)?us(a)? ", " usa ", text)
    #text = re.sub('(the[\s]+|the[\s]+)?united state(s)?', ' usa ', text)

    text = re.sub(r" uk ", " england ", text)
    text = re.sub(r" imrovement ", " improvement ", text)
    text = re.sub(r" intially ", " initially ", text)
    text = re.sub(r" dms ", " direct messages ", text)
    text = re.sub(r" demonitization ", " demonetization ", text)
    text = re.sub(r" actived ", " active ", text)
    text = re.sub(r" kms ", " kilometers ", text)
    text = re.sub(r" cs ", " computer science ", text)
    text = re.sub(r" upvote", " up vote", text)
    text = re.sub(r" iphone ", " phone ", text)
    text = re.sub(r" \0rs ", " rs ", text)
    text = re.sub(r" calender ", " calendar ", text)
    text = re.sub(r" ios ", " operating system ", text)
    text = re.sub(r" programing ", " programming ", text)
    text = re.sub(r" bestfriend ", " best friend ", text)
    text = re.sub(r" iii ", " 3 ", text)
    text = re.sub(r" banglore ", " bangalore ", text)
    text = re.sub(r" j k ", " jk ", text)
    text = re.sub(r" J\.K\. ", " jk ", text)


    # some others
    text = re.sub(r"60k", " 60000 ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"\s{2,}", " ", text)
    text = re.sub(r" usa ", " America ", text)
    text = re.sub(r" u s ", " America ", text)
    text = re.sub(r"'m ", " am ", text)

    # units
    text = re.sub(r"(\d+)kgs ", lambda m: m.group(1) + ' kg ', text)        # e.g. 4kgs => 4 kg
    text = re.sub(r"(\d+)kg ", lambda m: m.group(1) + ' kg ', text)         # e.g. 4kg => 4 kg
    text = re.sub(r"(\d+)k ", lambda m: m.group(1) + '000 ', text)          # e.g. 4k => 4000
    text = re.sub(r"\$(\d+)", lambda m: m.group(1) + ' dollar ', text)
    text = re.sub(r"(\d+)\$", lambda m: m.group(1) + ' dollar ', text)
    # This one is important in 2017
    text = re.sub(' donald trump',' trump ',text)
    text = re.sub(' dollars',' dollar ',text)
    text = re.sub(' quaro',' quora ',text)
    text = re.sub(r"googling", " google ", text)
    text = re.sub(r"googled", " google ", text)
    text = re.sub(r"googleable", " google ", text)
    text = re.sub(r"googles", " google ", text)

    text = re.sub(r"₹", " rs ", text)      # □□□
    text = re.sub(r"\$", " dollar ", text)

    # will blank any of !"#%&\'()*+,./:;<=>?[\\]^_`{|}~-@
    # Note the @ is dubious : won't we loose some emails
    # and $ is replaced by dollar before
    text = text.translate(FINAL_PUNC_CLEANER)

    return text

small_train = new_preprocess(small_train)
#display(small_train[['common_words','new_common_words','uncommon_words_question1','new_uncommon_words_question1']].head(10))
sniff_changes(small_train)
```

Compute all features in one shot

Extract common words between question1 & question2

Extract uncommon words in question1

Extract Nb common_words between question1 & question2

Extract Nb words in question1 not in common words

New common 118.681 % New uncommon 82.213 %

Question that may have impact next steps How many unicode chars are present ?

In [65]:
```python
# Ugly but easy
def sniff_unicode(s):
    if re.sub('[^\x00-\x7F]+', '', s) != s:
        return 1
    else:
        return 0

nb_unicode_train = train_dataframe['question1'].progress_apply(sniff_unicode).sum()/len(train_dataframe)
nb_unicode_train += train_dataframe['question2'].progress_apply(sniff_unicode).sum()/len(train_dataframe)
nb_unicode_challenge = challenge_dataframe['question1'].progress_apply(sniff_unicode).sum()/len(challenge_dataframe)
nb_unicode_challenge += challenge_dataframe['question2'].progress_apply(sniff_unicode).sum()/len(challenge_dataframe)
print_warning('Unicode train: %.3f challenge: %.3f' %(nb_unicode_train,nb_unicode_challenge))
```

Unicode train: 0.022 challenge: 0.021

Humpf, 0.02% of questions has some words fully in Unicode:

- awfully small.
- we are pushing the limits of this 'nb common words' feature and every little bit of information matters for the kaggle result
- Not clear if it does worth it ...
- Not clear also if we just replace it with nothing or with a generic placeholder like 'unicode_text' ...

I decide to not go so far (and a quick test showed kaggle score was slightly decreasing)