In [28]:
```
import pandas as pd

df = pd.read_csv("../datasets/train_df.csv")

df
```

Out[28]:

	qid	question_text	target
0	dda0b0efc8ba86e81ec4	What are interesting facts about Microsoft his...	0
1	dc708b74a108d0fc0ad9	What are those things which are not gonna happ...	0
2	06a27ec5d82dacd8bfe0	What should I know to avoid being "upsold" whe...	0
3	00cbb6b17e3ceb7c5358	How I add any account with payment bank?	0
4	7c304888973a701585a0	Which Multi level marketing products are actua...	0
...	...	...	...
999995	4bd96088d0b5f0f2c4f4	How is CSE at VIT Chennai?	0
999996	e80edbfc086f7125940f	How can we prevent a holocaust by robots, AI, ...	0
999997	1506dfad6bd340782a1f	How can I help a student remember key steps an...	0
999998	b56c60fd407f2f85553c	What is the difference between lace closure & ...	0
999999	a1b32d315c2782cdbcc3	What happens when you look into a broken mirror?	0

1000000 rows × 3 columns

Word Tokenizer

In [29]:
```
import nltk
nltk.download('punkt')

from nltk.tokenize import word_tokenize

def word_tokenize(sent):
    return nltk.word_tokenize(sent)

print("word tokenizing:",word_tokenize("Life is beautiful so Enjoy everymoment you have."))
```

```
word tokenizing: ['Life', 'is', 'beautiful', 'so', 'Enjoy', 'everymoment', 'you', 'have', '.']
```

```
[nltk_data] Downloading package punkt to /home/btv/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

RegexpTokenizer

In [30]:
```
from nltk.tokenize import RegexpTokenizer


def regex_word_tokenizer(sent):
    tokenizer = RegexpTokenizer(r'\w+')

    sample_word_tokens = tokenizer.tokenize(sent)
    sample_word_tokens = [word.lower() for word in sample_word_tokens]
    return sample_word_tokens

words = regex_word_tokenizer(str("Life is beautiful so Enjoy everymoment you have. Runners run hard to win"))
words
```

Out[30]:
```
['life',
 'is',
 'beautiful',
 'so',
 'enjoy',
 'everymoment',
 'you',
 'have',
 'runners',
 'run',
 'hard',
 'to',
 'win']
```

Stopwords Removal

In [31]:
```
from nltk.corpus import stopwords

def stop_words_removal(words):
    stop_words = [word.lower() for word in stopwords.words('english')]
```

```
    word_tokens = [word for word in words if word.lower() not in stop_words]
    return word_tokens

stop_words_removal(words)
```

Out[31]:
```
['life', 'beautiful', 'enjoy', 'everymoment', 'runners', 'run', 'hard', 'win']
```

Lemmatizer

In [32]:
```
from nltk.stem import WordNetLemmatizer

def Lemmatizer(words):
    lemmatizer = WordNetLemmatizer()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
    return lemmatized_words

Lemmatizer(stop_words_removal(words))
```

Out[32]:
```
['life', 'beautiful', 'enjoy', 'everymoment', 'runner', 'run', 'hard', 'win']
```

Stemming

In [33]:
```
from nltk.stem import PorterStemmer

def stemmer(words):
    ps = PorterStemmer()
    stemmed_words = [ps.stem(w) for w in words]
    return stemmed_words

stemmer(stop_words_removal(words))
```

Out[33]:
```
['life', 'beauti', 'enjoy', 'everymo', 'runner', 'run', 'hard', 'win']
```

In [34]:
```
def format_sentence(sent):
    tokens = regex_word_tokenizer(sent)
    tokens = stop_words_removal(tokens)
    tokens = Lemmatizer(tokens)
    return ({word: True for word in tokens})

format_sentence("Life is beautiful so Enjoy everymoment you have. Runners run hard to win")
```

Out[34]:
```
{'life': True,
 'beautiful': True,
 'enjoy': True,
 'everymoment': True,
 'runner': True,
 'run': True,
 'hard': True,
 'win': True}
```

In [35]:
```
pos = []
neg = []


for index,row in df.iterrows():
    if(row['target']==1):
        neg.append([format_sentence(row['question_text']), 1])
    else:
        pos.append([format_sentence(row['question_text']), 0])
```

In [37]:
```
training = pos[:int((.9)*len(pos))] + neg[:int((.9)*len(neg))]
```

In [38]:
```
test = pos[int((.1)*len(pos)):] + neg[int((.1)*len(neg)):]
```

In [39]:
```
from nltk.classify import NaiveBayesClassifier

classifier = NaiveBayesClassifier.train(training)
```

In [40]:
```
classifier.show_most_informative_features()
```

```
Most Informative Features
            islamophobic = True                1 : 0      =    156.7 : 1.0
                   apist = True                1 : 0      =    136.5 : 1.0
                  drumpf = True                1 : 0      =    116.2 : 1.0
                castrate = True                1 : 0      =    115.9 : 1.0
                   aping = True                1 : 0      =    112.9 : 1.0
                butthurt = True                1 : 0      =    106.1 : 1.0
               castrated = True                1 : 0      =     98.1 : 1.0
                shitting = True                1 : 0      =     96.0 : 1.0
                   wumao = True                1 : 0      =     96.0 : 1.0
                   moron = True                1 : 0      =     95.6 : 1.0
```

In [41]:
```
from nltk.classify.util import accuracy
```

```
print(accuracy(classifier, test))
```

```
0.8404188888888889
```