NLP based Question & Answer for E-commerce- Project

Sr.No.	Name	Email Id	Phone no.	Group No.
1	john victor	johnitsmywish@gmail.com	9629008997	G4
2	Ashish Gore	ashish.gore234@gmail.com	9028885821	
3	Megha Gowda	meghagowda738@gmail.com	7259920307	
4	Swathi katikala	katikalaswathi7@gmail.com	8978180449	
5	Akshay krishnan	akshaykrishnan143@gmail.com	8891617252	

Mentors:

Mr. Karthik

Mr. Parth

09/03/2020

Business Problem:

An e-commerce company wants to build an algorithm to retrieve top 5 Question and Answer based on the user given Keyword.

Objective:

The objective of this analysis is to predict the top five Question and Answer for a user given keyword.

Project Architecture / Project Flow

1. Understanding business objective
2. Getting data set details
3. JSON to CSV
4. EDA: Exploratory Data Analysis
5. Model Building
6. Evaluate the model
7. Data Visualizations
8. Deployment Frame

Json to csv conversion:

```
##conversion of json file to new csv file
import csv
import json
import ast
import os

# Opening JSON file and loading the data into the variable data
data=[]
with open('qa_Electronics.json', encoding= 'utf-8') as json_file:
    for i in json_file:
        c= ast.literal_eval(i)
        data.append(c)

data_file= open('data_file.csv','w')
```

```
# create the csv writer object
csv_writer = csv.writer(data_file)

#Counter variable used for writing headers to the CSV file

header  = ['question Type', 'asin', 'answerTime' , 'unixTime' , 'question' , 'answerType', "answer"]
csv_writer.writerow(header)

for emp in data:
    default_list=['NaN', 'NaN', 'NAN', 'NaN','NaN','NaN' ,'NaN']
    if len(emp)!=0:
        for j in emp:
            if j == 'questionType':
                default_list[0]=emp[j]
            if j == 'asin':
                default_list[1]=emp[j]
            if j == 'answerTime':
                default_list[2]=emp[j]
            if j == 'unixTime':
                default_list[3]=emp[j]
            if j == 'question':
                default_list[4]=emp[j]
            if j == 'answerType':
                default_list[5]=emp[j]
            if j == 'answer':
                default_list[6]=emp[j]

        csv_writer.writerow(default_list)
data_file.close()
```

Data set details

- The dataset is given in json format, prerequisite is to change json data to data frame
- The dataset contains question, answer, question type, asin ID, answer time and Unix time
- The dimension of the data set is 314263 (observations) , 7(number of columns)
- There are 32 missing values observed in question and answer column

Exploratory Data Analysis (EDA) and Feature Engineering:

- Basic Text Processing & Deep processing.
- Question's & Answer's character count and it's word count.
- Removing stop words and whitespaces.
- Removing the punctuations, special characters and numbers. All upper case to lower case.
- Tokenization where sentence is tokenize into words.
- Lemmatization grouping together the different infected form of words.

	question	answer	qus_char_count	ans_char_count	que_word_count	ans_word_count	cleaned_qus	cleaned_ans
0	Is this cover the one that fits the old nook c...	Yes this fits both the nook color and the same...	75	65	16	12	is this cover the one that fits the old nook c...	yes this fits both the nook color and the same...
1	Does it fit Nook GlowLight?	No. The nook color or color tablet	27	34	5	7	does it fit nook glowlight	no the nook color or color tablet
2	Would it fit Nook 1st Edition? 4.9in x 7.7in ?	I don't think so. The nook color is 5 x 8 so n...	46	112	10	24	would it fit nook st edition in in	don think so the nook color is so not su anyt...
3	Will this fit a Nook Color that's 5 x 8?	yes	40	3	10	1	will this fit nook color that x	yes
4	will this fit the Samsung Galaxy Tab 4 Nook 10.1	No, the tab is smaller than the 'color'	48	39	10	8	will this fit the samsung galaxy tab nook	no the tab is smaller than the color

Data Cleaning for Question and Answer

```
####Lemmatization
data1['Question'] = data1['Question'].apply(lambda x: " ".join([lemmatizer.lemmatize(word) for word in x.split()]))
data1['Answer']=data1['Answer'].apply(lambda x: " ".join([lemmatizer.lemmatize(word)for word in x.split()]))
```

```
data1.head()
```

	question	answer	que_char_count	ans_char-count	que_word_count	ans_word_count	Question	Answer
0	Is this cover the one that fits the old nook c...	Yes this fits both the nook color and the same...	75	65	16	12	cover one fit old nook color believe x	yes fit nook color sameshaped nook tablet
1	Does it fit Nook GlowLight?	No. The nook color or color tablet	27	34	5	7	fit nook glowlight	nook color color tablet
2	Would it fit Nook 1st Edition? 4.9in x 7.7in ?	I don't think so. The nook color is 5 x 8 so n...	46	112	10	24	fit nook st edition x	dont think nook color x sure anything smaller ...
3	Will this fit a Nook Color that's 5 x 8?	yes	40	3	10	1	will fit nook color thats x	yes
4	will this fit the Samsung Galaxy Tab 4 Nook 10.1	No, the tab is smaller than the 'color'	48	39	10	8	will fit samsung galaxy tab nook	tab smaller color

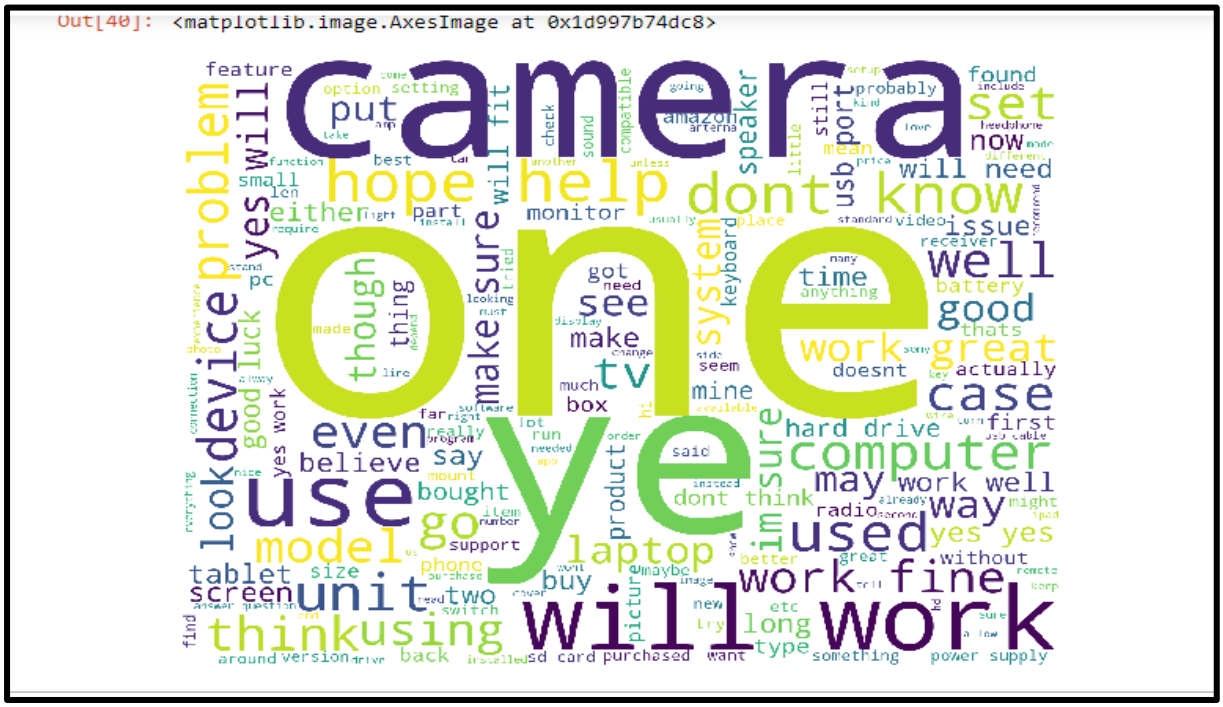
Word Cloud for Question and Answer

```
########### wordcloud generation
####Questuion
Question_text = ' '.join(data1['Question'])
Q_wordcloud=WordCloud(
                    background_color='white',
                    width=1800,
                    height=1400
                   ).generate(Question_text)
fig = plt.figure(figsize = (10, 15))
plt.axis('off')
plt.imshow(Q_wordcloud)
```

```
###Answer
Answer_text = ' '.join(data1['Answer'])
A_wordcloud = WordCloud(
                        background_color='white',
                        width=1800,
                        height=1400
                       ).generate(str(Answer_text))
fig = plt.figure(figsize = (10, 15))
plt.axis('off')
plt.imshow(A_wordcloud )
```

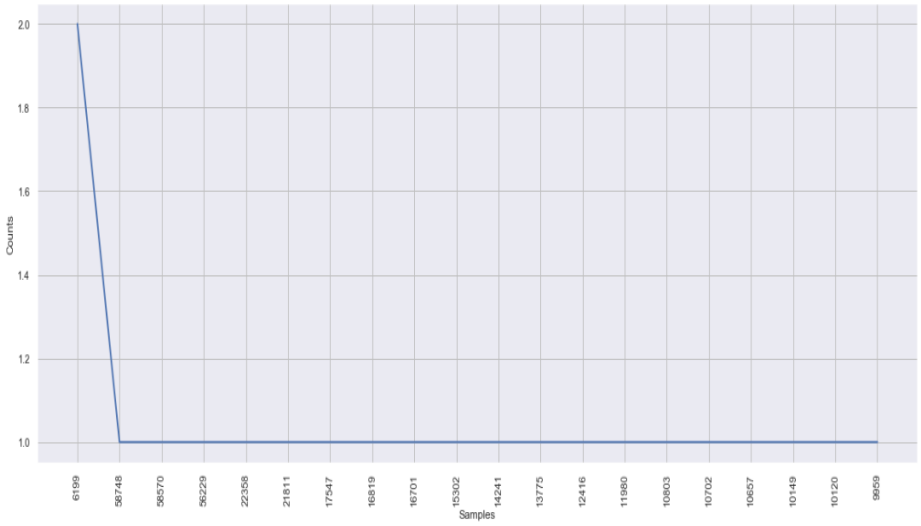
Word Cloud of Question

Word Cloud of Answer



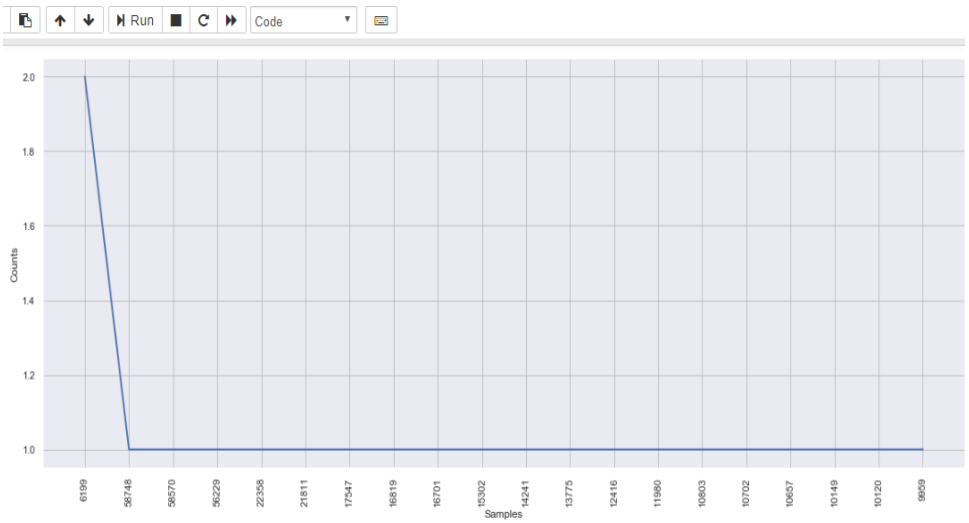
Top20 frequently used words in Question

```
will          58748
can           58570
work          56229
use           22358
fit           21811
camera        17547
one           16819
come          16701
need          15302
battery       14241
tv            13775
anyone        12416
cable         11980
compatible    10803
get           10702
case          10657
know          10149
speaker       10120
card           9959
just           9898
dtype: int64
```



Top 20 frequently used Answers

```
yes           70765
will          63204
work          60015
can           59865
one           43448
use           42029
just          33857
camera        32006
dont          31026
need          26462
get           24484
sure          22024
like          21835
good          21548
cable         21235
fit           20676
battery       20577
know          19790
also          19722
usb           18763
dtype: int64
```



Bigrams

```
################ bigrams for Question and Answers
for i in data1['Question'][0:10]:
    grams=TextBlob(i).ngrams(2)
    print(grams)

counts = collections.Counter()
for i in data1['Question']:
    words1 =i.split()
    counts.update(nltk.bigrams(words1))

common_bigrams = counts.most_common(10)
common_bigrams
```

```
[WordList(['cover', 'one']), WordList(['one', 'fit']), WordList(['fit', 'old']), WordList(['old', 'nook']), WordList(['nook',
'color']), WordList(['color', 'believe']), WordList(['believe', 'x'])]
[WordList(['fit', 'nook']), WordList(['nook', 'glowlight'])]
[WordList(['fit', 'nook']), WordList(['nook', 'st']), WordList(['st', 'edition']), WordList(['edition', 'x'])]
[WordList(['will', 'fit']), WordList(['fit', 'nook']), WordList(['nook', 'color']), WordList(['color', 'thats']), WordList(['th
ats', 'x'])]
[WordList(['will', 'fit']), WordList(['fit', 'samsung']), WordList(['samsung', 'galaxy']), WordList(['galaxy', 'tab']), WordLis
t(['tab', 'nook'])]
[WordList(['flip', 'stand'])]
[WordList(['flip', 'stand'])]
[WordList(['also', 'fit']), WordList(['fit', 'hd'])]
[WordList(['position', 'reader']), WordList(['reader', 'horizontalvertical']), WordList(['horizontalvertical', 'thank']), WordL
ist(['thank', 'kwod'])]
[WordList(['closure', 'mechanism']), WordList(['mechanism', 'band']), WordList(['band', 'magnetic']), WordList(['magnetic', 'et
```

Out[42]:
```
[(('will', 'work'), 19056),
 (('can', 'use'), 8743),
 (('will', 'fit'), 7697),
 (('anyone', 'know'), 3687),
 (('can', 'used'), 2626),
 (('nikon', 'd'), 2619),
 (('hard', 'drive'), 2390),
 (('can', 'get'), 2156),
 (('usb', 'port'), 1928),
 (('sd', 'card'), 1857)]
```

In [43]:
```
##Answer
for i in data1['Answer'][0:10]:
    grams=TextBlob(i).ngrams(2)
    print(grams)

counts = collections.Counter()
for i in data1['Answer']:
    words1 =i.split()
    counts.update(nltk.bigrams(words1))

common_bigrams = counts.most_common(10)
common_bigrams
```

```
[WordList(['tab', 'smaller']), WordList(['smaller', 'color'])]
[WordList(['flip', 'stand']), WordList(['stand', 'pocket']), WordList(['pocket', 'front']), WordList(['front', 'flap']), WordLi
st(['flap', 'nice']), WordList(['nice', 'cover'])]
[WordList(['hi', 'doesnt'])]
[WordList(['size', 'charging']), WordList(['charging', 'port']), WordList(['port', 'place'])]
[]
[WordList(['like', 'normal']), WordList(['normal', 'book']), WordList(['book', 'doesnt']), WordList(['doesnt', 'flop']), WordLi
st(['flop', 'open']), WordList(['open', 'never']), WordList(['never', 'wen']), WordList(['wen', 'issue']), WordList(['issue',
'nook']), WordList(['nook', 'clip']), WordList(['clip', 'secure']), WordList(['secure', 'safe']), WordList(['safe', 'holder']),
WordList(['holder', 'inside']), WordList(['inside', 'small']), WordList(['small', 'convenient']), WordList(['convenient', 'bes
t']), WordList(['best', 'cover']), WordList(['cover', 'ive']), WordList(['ive', 'ever']), WordList(['ever', 'nook']), WordList
(['nook', 'trim']), WordList(['trim', 'functional']), WordList(['functional', 'magnet']), WordList(['magnet', 'elastic']), Word
List(['elastic', 'needed'])]
```

ut[43]:
```
[(('will', 'work'), 10130),
 (('hope', 'help'), 9669),
 (('dont', 'know'), 9051),
 (('work', 'great'), 5331),
 (('yes', 'can'), 5179),
 (('yes', 'will'), 4861),
 (('make', 'sure'), 4750),
 (('im', 'sure'), 4488),
 (('work', 'fine'), 4230),
 (('can', 'use'), 4201)]
```

TD-IDF Vectorizer

In [44]:
```
################ TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(max_features=1000, lowercase=True, analyzer='word',stop_words= 'english',ngram_range=(1,1))
vect = tfidf.fit_transform(data1['Question'])
vect1=tfidf.fit_transform(data1['Answer'])
print(vect)
print(vect1)
vect
```

```
  (0, 150)      0.4678855759378759
  (0, 552)      0.5684647389867761
  (0, 565)      0.43967736343663466
  (0, 288)      0.28049209323274527
  (0, 187)      0.4312063659934782
  (1, 552)      0.8967749067419665
  (1, 288)      0.4424870242593983
  (2, 249)      0.55522628658249
  (2, 823)      0.5564963319850462
  (2, 552)      0.554287859693836
  (2, 288)      0.27349693192252506
  (3, 876)      0.5758369588095471
  (3, 150)      0.4855177756699793
  (3, 552)      0.5898872498183639
  (3, 288)      0.2910623968828244
  (4, 861)      0.4938922744355213
  (4, 307)      0.4402499030331026
```

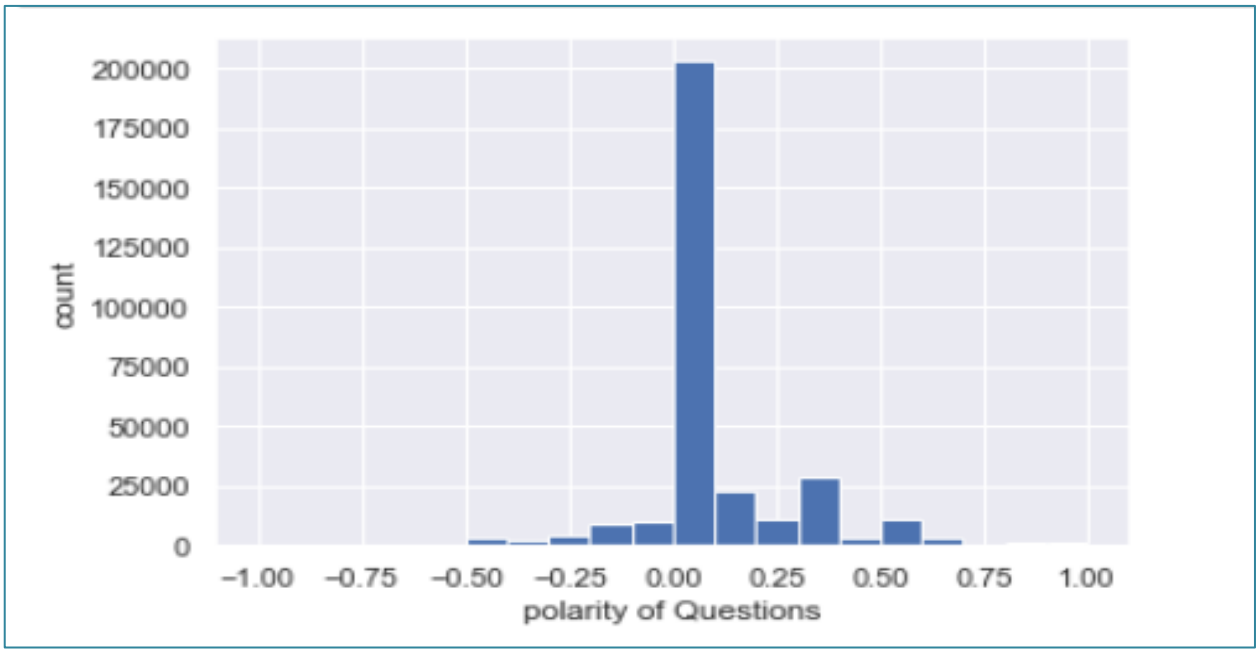
Question and Answer Sentiment

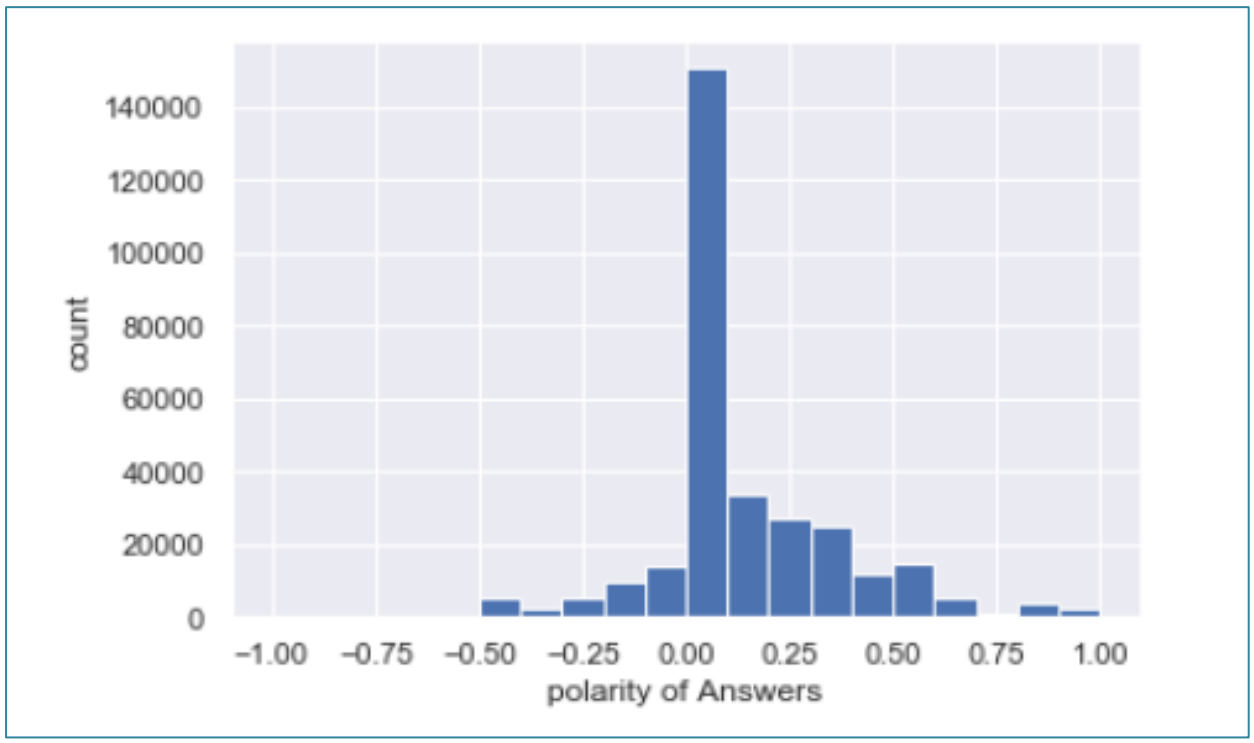
In [39]:
```
##################################### sentiment analysis############################
data1['Q_sentiment'] = data1['Question'].apply(lambda x: TextBlob(x).sentiment[0] )
data1[['Question','Q_sentiment']].head()
data1['A_sentiment']=data1['Answer'].apply(lambda x: TextBlob(x).sentiment[0])
data1[['Answer','A_sentiment']]


def sent_type(text):
    for i in text:
        if i>0:
            print('positive')
        elif i==0:
             print('natural')
        else:
             print('negative')


sent_type(data1['Q_sentiment'])
sent_type(data1['A_sentiment'])
```

Sentimental Analysis of Question & Answer





Model Building

In [25]:
```
stopwords_list = stopwords.words('english')
```

In [26]:
```
lemmatizer = WordNetLemmatizer()
```

In [27]:
```
def my_tokenizer(doc):
    words = word_tokenize(doc)
    pos_tags = pos_tag(words)
    non_stopwords = [w for w in pos_tags if not w[0].lower() in stopwords_list]


    non_punctuation = [w for w in non_stopwords if not w[0] in string.punctuation]

    lemmas = []

    for w in non_punctuation:
        if w[1].startswith('J'):
            pos = wordnet.ADJ
        elif w[1].startswith('V'):
            pos = wordnet.VERB
        elif w[1].startswith('N'):
            pos = wordnet.NOUN
        elif w[1].startswith('R'):
            pos = wordnet.ADV
        else:
            pos = wordnet.NOUN

        lemmas.append(lemmatizer.lemmatize(w[0], pos))

    return lemmas
```

In [28]:
```
tfidf_vectorizer = TfidfVectorizer(tokenizer = my_tokenizer)
tfidf_vectorizer
```

Out[28]:
```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=None,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=<function my_tokenizer at 0x000001F41B5E4048>,
                use_idf=True, vocabulary=None)
```

In [29]:
```
tfidf_matrix = tfidf_vectorizer.fit_transform(tuple(df_final['question']))
```

In [32]:
```
tfidf_matrix
```

Out[32]:
```
<290841x90035 sparse matrix of type '<class 'numpy.float64'>'
        with 2041613 stored elements in Compressed Sparse Row format>
```

In [33]:
```
pickle.dump(tfidf_vectorizer, open('tranform.pkl', 'wb'))
```

Model Evaluating

In [34]:
```
#print(tfidf_matrix.shape)

def ask_question(question):
    query_vect = tfidf_vectorizer.transform([question])

    similarity = cosine_similarity(query_vect, tfidf_matrix)



    top_5_simmi = similarity[0].argsort()[-5:][::-1]


    count =1
    for i in top_5_simmi:
        print('Question:-',count,':' ,df_final.iloc[i]['question'])
        print('Answer: ', df_final.iloc[i]['answer'])
        print('Accuracy is: {:.2%}'.format(similarity[0, i]))
        print('*'*25)
        count+=1


    filename = 'nlp_model.pkl'
    pickle.dump(similarity, open(filename, 'wb'))
```

Model Prediction and Results

Model generated the output of top 5 Question and Answer with respect to the keyword entered.

```
In [35]: ask_question(input('Hello, Please enter the Keywords for /n a question you want to search for: '))

Hello, Please enter the Keywords for /n a question you want to search for: complaint
Question:- 1 : what are the complaints
Answer:  I have a MAC but there are certain programs that seem to run better on PC, (Audible.com, Stamps.com) and i did not wan
t to use Parallels program on the Mac. The Acer works great for me, no problems at all. I am satisfied with the Acer, especiall
y for the price.. I hope this is helpful for you...Chuck
Accuracy is: 100.00%
**************************
Question:- 2 : What was your complaint about these headphones?
Answer:  I really don't have a complaint about them, they're dope.
Accuracy is: 86.30%
**************************
Question:- 3 : Do you guys have any complaints about these speakers?
Answer:  Chris, I have put them in place with 6 different members of my team (A/V and graphic designers). I also am using a set
at my desk. They are great quality, they look great, and we have had no trouble with any of the sets. I would highly recommend
them for the price.
Accuracy is: 72.50%
**************************
Question:- 4 : Any defect complaints?
Answer:  No
Accuracy is: 70.96%
**************************
Question:- 5 : Number one customer complaint
Answer:  Picture slid off into one corner and ended in a spider web fracture.
Accuracy is: 64.48%
**************************
```

Model Deployment using Flask

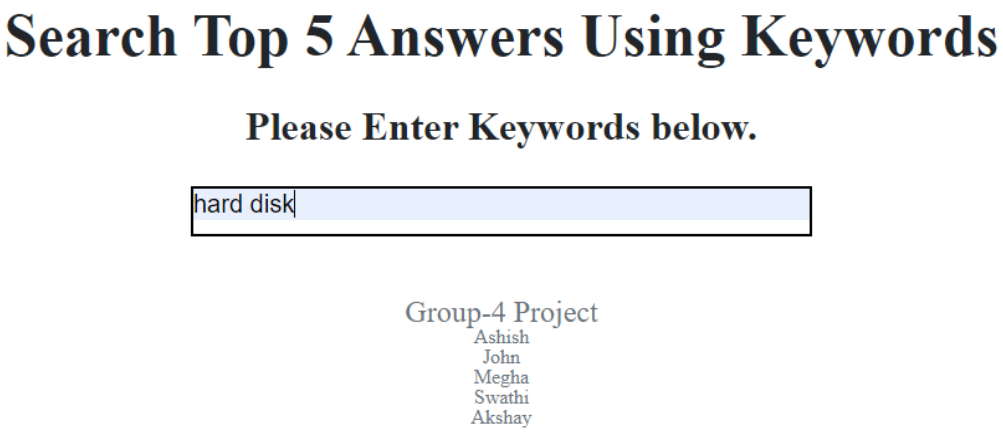
- We used the Heroku platform since its one of the easiest to use for deployment.

- Before we use Heroku, we need to install some command line tools and create our application.

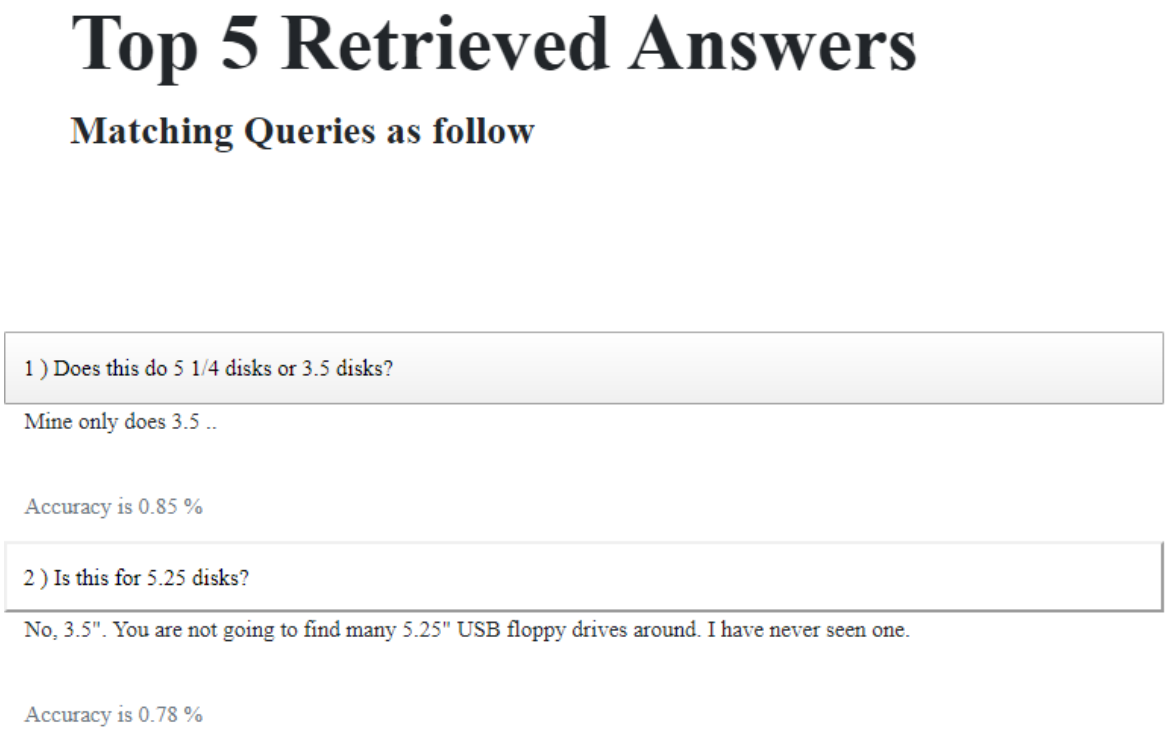
Steps to Create Our Application on Local Machine

1. Create Flask App
2. Install gunicorn
3. Create a requirements.txt file
4. Create a Profile
5. Create a Heroku Account
6. Push and launch the application

Home Template



Results Template



Project Link

https://top5-qa-search.herokuapp.com/