```
!pip install -U sentence-transformers
```

```
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.7
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: torchvision in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-pacl
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: huggingface-hub in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: transformers<5.0.0,>=4.6.0 in /usr/local/lib/pyth
Requirement already satisfied: torch>=1.6.0 in /usr/local/lib/python3.7/dist-pacl
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dis
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in /usr/local/l
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dis
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/di
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/l
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pacl
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/di
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.7
```

```
!pip install pyspellchecker
```

```
Collecting pyspellchecker
  Downloading pyspellchecker-0.6.3-py3-none-any.whl (2.7 MB)
     |████████████████████████████████| 2.7 MB 9.8 MB/s
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.6.3
```

```
#Connecting to the drive to access data
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call dr
```

Imports and Loadings

```
#Importing the necessary libraries

import numpy as np
import pandas as pd
import regex as re
import nltk
import matplotlib.pyplot as plt
from matplotlib_venn import venn2, venn3
from transformers import AutoTokenizer
import seaborn as sns
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
nltk.download('wordnet')
from nltk.corpus import brown, stopwords
from gensim.models import Word2Vec
from tqdm.notebook import tqdm
from scipy.stats import uniform, randint, loguniform
from sklearn.preprocessing import LabelEncoder
from scipy.sparse import csr_matrix
from collections import Counter
from nltk.stem import PorterStemmer
from wordcloud import STOPWORDS
from prettytable import PrettyTable
import warnings
import math
from sklearn.neighbors import NearestNeighbors
from sklearn.neighbors import BallTree
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from nltk import sent_tokenize
from nltk.stem import WordNetLemmatizer
from gensim.models import Word2Vec
from numpy.linalg import norm
import pickle

!pip install rank_bm25
!pip install nltk
nltk.download('stopwords')
warnings.filterwarnings('ignore')
nltk.download('brown')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Requirement already satisfied: rank_bm25 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: nltk in /usr/local/lib/python3.7/dist-packages (3
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (fr
[nltk_data] Downloading package stopwords to /root/nltk_data...
```

```
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Package brown is already up-to-date!
True
```

```
# importing dataset for product catalogue
df_product = pd.read_csv('/content/drive/My Drive/data/product_catalogue.csv')
```

```
df_product = df_product[df_product["product_locale"]=='us']
```

```
df_product.head(5)
```

	product_id	product_title	product_description	product_bullet_point	product
0	B0188A3QRM	Amazon Basics Woodcased #2 Pencils, Unsharpene...	NaN	144 woodcase #2 HB pencils made from high-qual...	Amazc
1	B075VXJ9VG	BAZIC Pencil #2 HB Pencils, Latex Free Eraser,...	BACK TO BAZIC Our go...	⭐ UN-SHARPENED #2 PREMIUM PENCILS. Each...	BAZIC
2	B07G7F6JZ6	Emraw Pre Sharpened Round Primary Size No 2 Ju...	<b>Emraw Pre-Sharpened #2 HB Wood Pencils -...	✓ PACK OF 8 NUMBER 2 PRESHARPENED BEGINNERS PE...	
		Emraw Pre			

The code containes data collected from US and Japan. The data collected from Japan is in Japanese language while the one from the US is in English. We will be working with data from the US only

```
# importing training dataset
df = pd.read_csv('/content/drive/My Drive/data/train.csv')
```

```
df = df[df["query_locale"]=='us']
```

```
df.sample(10)
```

	query_id	query	query_locale	product_id	esci_label
261534	11511	the dude big lebowski	us	B07B3XWCSQ	irrelevant
98567	3966	elderly remote roku	us	B00CRJJFYO	complement
313067	13950	cedar gazebo with aluminum roof	us	B083FN5WQ8	substitute
169335	7277	mailbox with locks with keys	us	B01CEUXCQC	complement
22970	773	acrylic paints for painting	us	B01G4AULFS	substitute
139066	5869	ilan stavans	us	0271082305	exact
165028	7078	lol doll	us	B07PMN4NJX	exact
367070	16392	bounce house without blower	us	B08736PY7S	exact

The train data only has product id but no further information on the product. So we map the product description, product brands and product product_bullet_points (that describes the product attributes) from Product dataframe to training data with respect to its product id

```
# Merging data from train and product dataframe
df = (df.merge(df_product, left_on='product_id', right_on='product_id')
          .reindex(columns=['query_id', 'query', 'query_locale', 'product_id','product
df = df.drop('query_locale', 1)
df.head(10)
```

	query_id	query	product_id	product_title	product_brand	product_bull
0	0	# 2 pencils not sharpened	B0000AQO0O	Ticonderoga Beginner Pencils, Wood-Cased #2 HB...	Ticonderoga	Round wood latex-free era

```
print(df['esci_label'].value_counts());
```

```
exact         181856
substitute    147654
irrelevant     71125
complement     19095
Name: esci_label, dtype: int64
```

HB...

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 419730 entries, 0 to 419729
Data columns (total 8 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   query_id              419730 non-null  int64
 1   query                 419730 non-null  object
 2   product_id            419730 non-null  object
 3   product_title         419658 non-null  object
 4   product_brand         399197 non-null  object
 5   product_bullet_point  371076 non-null  object
 6   product_color_name    290488 non-null  object
 7   esci_label            419730 non-null  object
dtypes: int64(1), object(7)
memory usage: 28.8+ MB
```

```
df.describe(include='object')
```

	query	product_id	product_title	product_brand	product_bullet_point	p
count	419730	419730	419658	399197	371076	
unique	20888	352028	348284	95642	274088	
top	airpods	B01HFFXLNA	Episode 1	Nike	Used Book in Good Condition	

```
len(df['product_id'].unique())
```

```
352028
```

```
# Checking for null values in training data
df.isnull().sum()
```

```
query_id                     0
query                        0
product_id                   0
product_title               72
product_brand            20533
product_bullet_point     48654
product_color_name      129242
esci_label                   0
dtype: int64
```

As observed from the dataframe product_df, the product bullet points have elements of product title and product titles can come close to what has been used as queries. So, we will be filling the null values in product_title with corresponding queries and product_bullet_point with corresponding product titles

```
df.product_title.fillna(df.query, inplace=True)
```

```
df.product_bullet_point.fillna(df.product_title, inplace=True)
```

Also, for brand name, according to the data trend the first work in the product generally is the band's name that manufactutred the product. So we will be filling the null spaces for brand name with the first word of the product title

```
df.product_brand.fillna(str(df.product_title.str.split()[0]), inplace=True)
```

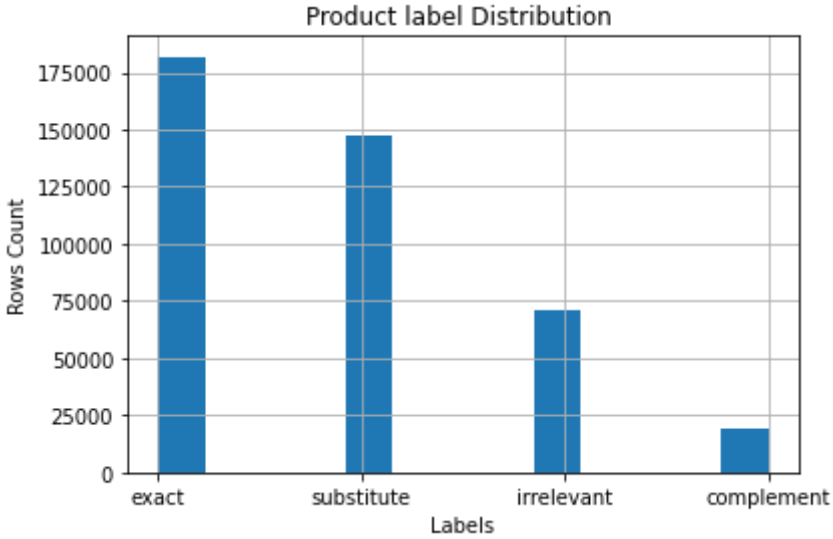
```
df.isnull().sum()
```

```
query_id                     0
query                        0
product_id                   0
product_title                0
product_brand                0
product_bullet_point         0
product_color_name      129242
esci_label                   0
dtype: int64
```

```
sample_data=df.iloc[101]
sample_data
print('query \n',sample_data.query)
print('\nProduct Title \n',sample_data.product_title)
print('\nProduct Brand \n',sample_data.product_brand)
print('\nProduct Colour \n',sample_data.product_color_name)
print('\nProduct Description \n',sample_data.product_bullet_point)
```

```
query
 # mom life

Product Title
 MACCHIASHINE Unique Sleeveless MOM Life Printed Shirt Womens Cotton Tank Tops(G

Product Brand
 MACCHIASHINE

Product Colour
 Green

Product Description
 Material: Soft Cotton,comfortable touching
Lightweight 0.12kg; (Necklace NOT included)
Features: MOM LIFE printed fashion sleeveless ovesized tank tops,poll-on Casual
Size Details: S.M.L.XL. Pls Carefully Refer to the Size Details below before pla
Suitable Occasion: Casual/Date/Home/Party/Work/Shopping/Exercise
```

```
df['esci_label'].hist(bins=13);
plt.xlabel("Labels");
plt.ylabel("Rows Count");
plt.title('Product label Distribution ');
```



Finding top 10 frequently searched products

```
frequently_used_query = pd.DataFrame(df.groupby('query')['esci_label'].count())
most_common_query = frequently_used_query.sort_values('esci_label', ascending=False)
most_common_query.head(10)
```

query	esci_label
airpods	188
printer	136
face mask	109
apple watch series 3	98
apple headphones	94
barbie dolls	87
echo dot	84
prime movies	84

BASIC PREPROCESSING

Our goal here is to find the similarity between the query used to search the product and the actual text used for product. In order to do this it is necessary for us to clean text so that a rubust relationship chan be established between query and product title.

The measuring units in the products are not uniform. for example for unit pounds some title uses `lb` while the others use `lbs` or `pounds`. This non uniformity makes it hard to establish relationship between sentences to compare similarities.

Similarly spacing is added between numeric values and alphabets. In instances like `5vitamins` and `5 vitamins` refer to same product but no similarity can be found in them. so the numbers and alphabets are spaced in such case. Another case we need to take care of are spaces befor and after `-`s and `\`s to resolve similar case.

Beside these case specific cleaning, we are also changing all text to lowercase and removing extra white spaces to get clean data before we can perform further operations

```
import re

def pre_process_text(product_title):
  if type(product_title) is str:
    keywords = product_title.replace('-', ' - ')
    keywords = product_title.replace('/', ' / ')
    keywords = re.split(r'\W+', product_title)
    keywords = [keyword.lower() for keyword in keywords] #converting all text to lower
    clean_words = re.sub("[A-Za-z]+", lambda w: " " + w[0] + " ", ' '.join(keywords))
    processed_text = " " + clean_words + " "
    # bringing varied way of showing mesurement units to same standard
```

```
    processed_text = re.sub('( pcs | pc )',' piece ',processed_text)
    processed_text = re.sub('( foot | ft | feets )',' foot ',processed_text)
    processed_text = re.sub('( in | inches )',' inch ',processed_text)
    processed_text = re.sub('( qt | quart )',' quarter ',processed_text)
    #replace '' or " with inch and ' with feet for measurement
    processed_text = re.sub("^[0-9]+\"|[0-9]+'",' inch ',processed_text)
    rocessed_text = re.sub("^[0-9]+'",' foot ',processed_text)
    # strip whitespaces
    processed_text = ' '.join(processed_text.split())
    return processed_text

df['product_title'] = df['product_title'].apply(lambda text : pre_process_text(text))
df['product_brand'] = df['product_brand'].apply(lambda text : pre_process_text(text))
df['product_bullet_point'] = df['product_bullet_point'].apply(lambda text : pre_proces
df['query'] = df['query'].apply(lambda text : pre_process_text(text))
```

```
df.head(6)
```

	query_id	query	product_id	product_title	product_brand	product_bull
0	0	2 pencils not sharpened	B0000AQO0O	ticonderoga beginner pencils wood cased 2 hb s...	ticonderoga	round wood latex free e
1	8799	pencils for kindergarteners	B0000AQO0O	ticonderoga beginner pencils wood cased 2 hb s...	ticonderoga	round wood latex free e
2	14844	2 dixon oriole pencils not sharpened	B0000AQO0O	ticonderoga beginner pencils wood cased 2 hb s...	ticonderoga	round wood latex free e
3	15768	2 pencils with erasers sharpened not soft	B0000AQO0O	ticonderoga beginner pencils wood cased 2 hb s...	ticonderoga	round wood latex free e

Lemmatization and removal of stop words

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

def stem_text(text):
```

```
  if type(text) is str:
    data_texts = text.split()
    data_texts = [txt for txt in data_texts if not txt in stop_words]
    data_texts = [lemmatizer.lemmatize(word) for word in data_texts]
    return ' '.join(data_texts)

df['product_title'] = df['product_title'].apply(lambda txt : stem_text(txt))
df['product_brand'] = df['product_brand'].apply(lambda txt : stem_text(txt))
df['product_bullet_point'] = df['product_bullet_point'].apply(lambda txt : stem_text(t
df['query'] = df['query'].apply(lambda txt : stem_text(txt))
```

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

```
df.sample(6)
```

	query_id	query	product_id	product_title	product_brand	product_bu
230208	8547	orange short	B00LIB1BBM	stretch comfort woman cotton stretch workout b...	stretch comfort	perfect woman bike
11948	15814	mini refrigerator without freezer	B08DJ2W7SC	mini fridge teccpo 1 7 cu foot small refrigera...	teccpo	compact despite sma
184334	6455	khaki skirt woman knee length	B014QWMX3O	kosher casual woman modest knee length lightwe...	kosher casual	95 cotton 5 new novem
24762	13324	2 x 72 belt grinder	B07KL4BQF5	wen 6515 1 inch x 30 inch belt sander 5 inch s...	wen	two inc machine

```
#fill null values
df.product_title.fillna(df.query, inplace=True)
df.product_bullet_point.fillna(df.product_title, inplace=True)
```

CORRECT TYPOS

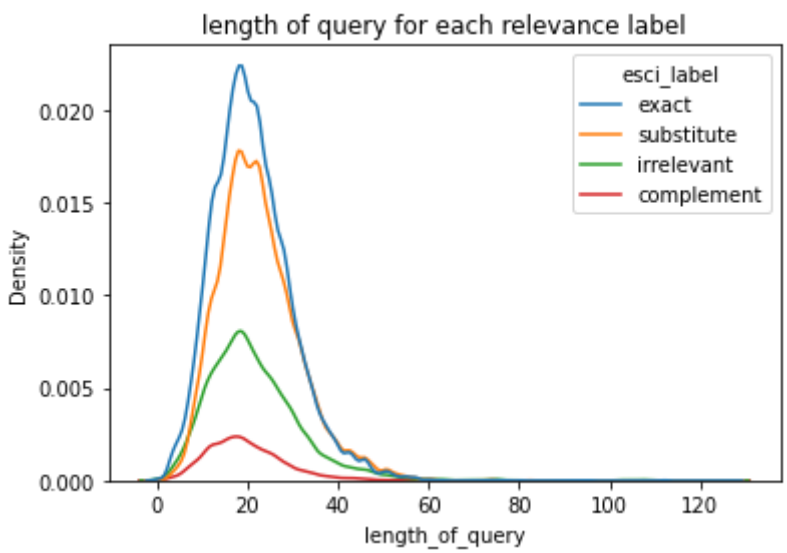
While the information have been obtained from the products in amazon, the `query` is user input. The user input may likely have typos which when corrected give better search results

```
from spellchecker import SpellChecker
check_spelling = SpellChecker()
```

```
def correct_spelling(text, correction = 0):
    words = []
    for word in text.split(' '):
        if word not in stop_words:
            if correction == 1:
                if len(check_spelling.unknown([word])):
                    word = check_spelling.correction(word)
            words.append(word)
    return ' '.join(words)

df['query'] = df['query'].apply(lambda txt : correct_spelling(txt, correction = 1))
```

```
# this splits the text by uppercase alphabets but only if the uppercase letter is not
# eg : Harry Potter VolI becomes Harry Potter Vol I
df['query'] = df['query'].apply(lambda x: ' '.join(re.findall(r'[A-Z]?[^A-Z\s]+|[A-Z]+
```

```
df['length_of_query'] = df['query'].str.len()
sns.kdeplot(data=df, x="length_of_query", hue="esci_label")
plt.title('length of query for each relevance label')
plt.show()
```



Here we check if query length has influence on finding relevant product.Most queries have length between 10-30. The distribution is almost similar for all 4 relevance label. The query length doesnt seem to have much meaningful impact

```
#using count vectorizer to find unigrams and bigrams from bag of words
def N_gram(data_texts, N_gram=None):
    count_vectors = CountVectorizer(ngram_range=(N_gram, N_gram)).fit(data_texts)
    wordsum = count_vectors.transform(data_texts).sum(axis=0)
    frequency_of_words = [(txt, wordsum[0, index])
                  for txt, index in count_vectors.vocabulary_.items()]
```

```
    frequency_of_words =sorted(frequency_of_words, key = lambda word: word[1], reverse
    return frequency_of_words
```

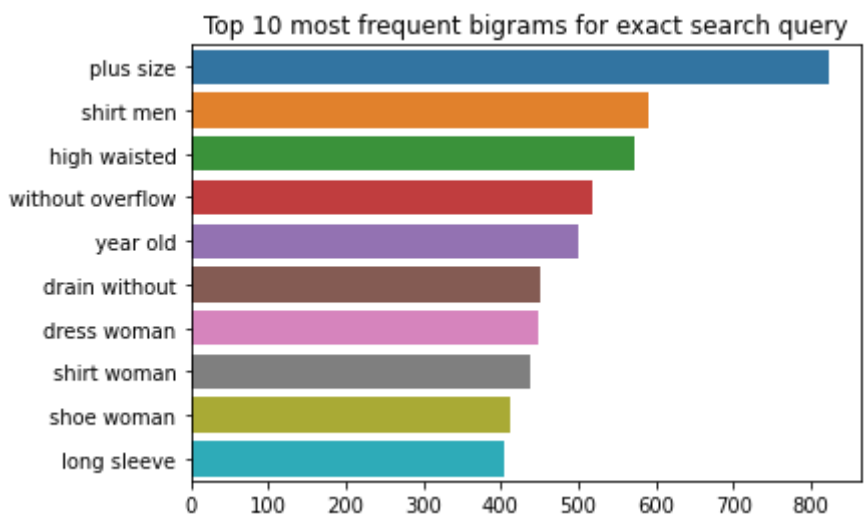
```
df_exact = df[df['esci_label']=='exact']
df_substitute = df[df['esci_label']=='substitute']
df_irrelevant = df[df['esci_label']=='irrelevant']
df_complement = df[df['esci_label']=='complement']
```

Bigram in descending order of frequency in search query for each class

```
bigrams_exact=N_gram(df_exact['query'],2)
print(bigrams_exact)
```

```
[('plus size', 824), ('shirt men', 591), ('high waisted', 572), ('without overfl
```

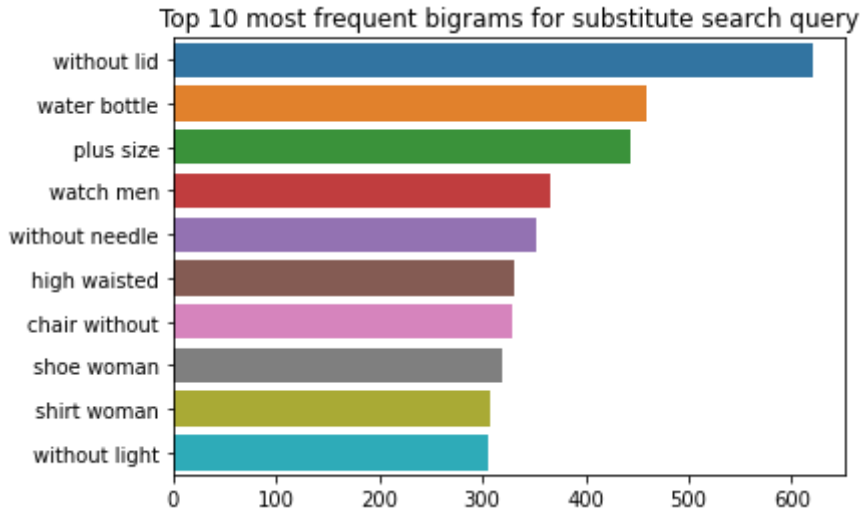
```
count, bigram =map(list,zip(*bigrams_exact[:10]))
plt.title('Top 10 most frequent bigrams for exact search query')
sns.barplot(x=bigram,y=count)
plt.show()
```




```
bigrams_substitute=N_gram(df_substitute['query'],2)
print(bigrams_substitute)
```

```
[('without lid', 620), ('water bottle', 459), ('plus size', 443), ('watch men',
```

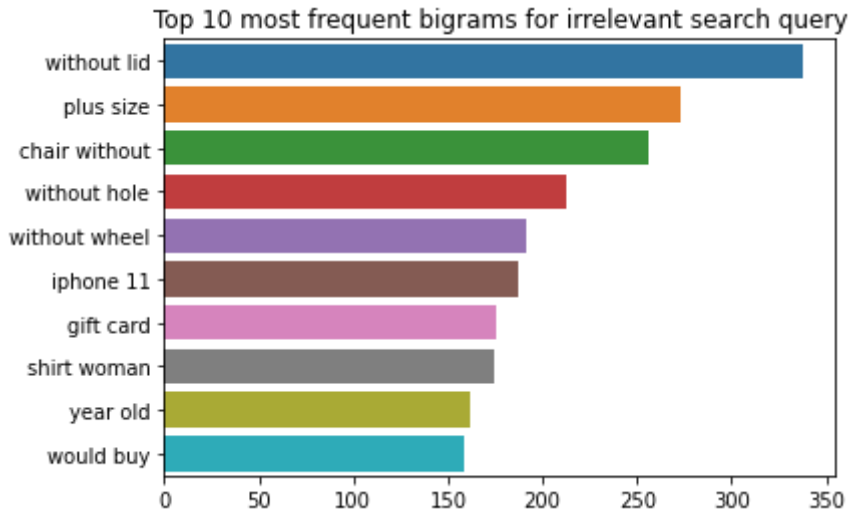
```
count, bigram =map(list,zip(*bigrams_substitute[:10]))
plt.title('Top 10 most frequent bigrams for substitute search query')
sns.barplot(x=bigram,y=count)
plt.show()
```




```
bigrams_irrelevant=N_gram(df_irrelevant['query'],2)
print(bigrams_irrelevant)
```

```
[('without lid', 338), ('plus size', 273), ('chair without', 256), ('without hol
```

```
count, bigram =map(list,zip(*bigrams_irrelevant[:10]))
plt.title('Top 10 most frequent bigrams for irrelevant search query')
sns.barplot(x=bigram,y=count)
plt.show()
```

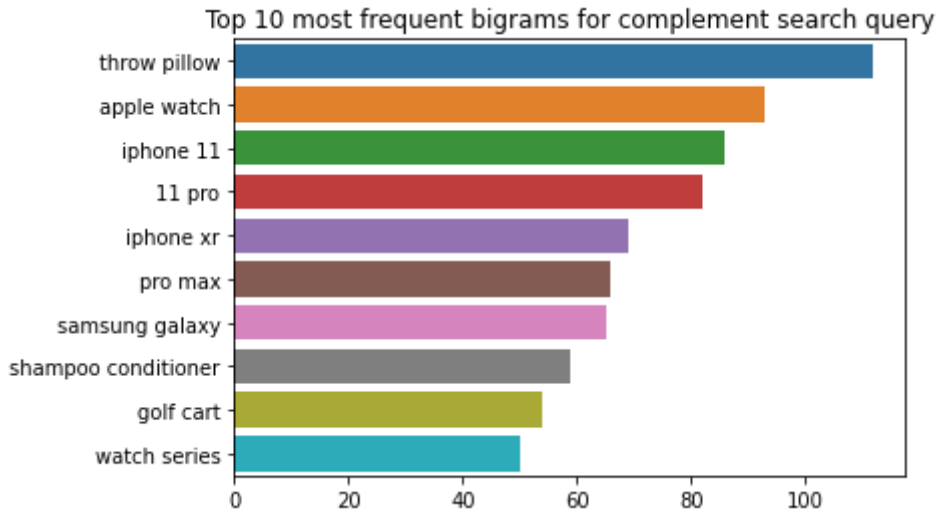



```
bigrams_complement=N_gram(df_complement['query'],2)
print(bigrams_complement)
```

```
[('throw pillow', 112), ('apple watch', 93), ('iphone 11', 86), ('11 pro', 82),
```

```
count, bigram =map(list,zip(*bigrams_complement[:10]))
```

```
plt.title('Top 10 most frequent bigrams for complement search query')
sns.barplot(x=bigram,y=count)
plt.show()
```



LABEL ENCODING

```
# Encoding categorical label into numerical
relevance_label = ('exact','substitute','irrelevant','complement')
label_df = pd.DataFrame(relevance_label, columns=['relevance_label'])
labelencoder = LabelEncoder()
label_df
```

	relevance_label
0	exact
1	substitute
2	irrelevant
3	complement

```
search_relevance = ('exact','substitute','irrelevant','complement')
labelencoder = LabelEncoder()
df['esci_label'] = labelencoder.fit_transform(df['esci_label'])
df.sample(6)
```

	query_id	query	product_id	product_title	product_brand	product_bu
279297	10929	squirell cage	B07VHQN82L	aveen 3 tier small cat cage playpen box kennel...	aveen	pet expert tier cat cag
290519	11541	lady vanishes dvd	B00000C0QO	lady vanishes	cobra entertainment llc	
3722	307	10 mm syringe without needle sealable	B07PPFLKRD	depepe 2 piece 100 ml large plastic syringe 2 ...	depepe	package inc 100 ml
129844	4048	entertaining evil	B003V1WTJ0	lady light shadow tairen soul book	ticonderoga beginner pencil wood cased 2 hb	lady light s

```
df.drop(['length_of_query'], axis = 1, inplace = True)
```

		bag heat		pack chic 4 x 6		spec

```
df.product_title.fillna(df.query, inplace=True)
df.product_bullet_point.fillna(df.product_title, inplace=True)
df.product_color_name.fillna('', inplace=True) # product color is secondary, hence for
```

The query text need to match with the product text to return a match. the product has its title name, description and brand that can be associated with the query text. So we combine the text related to product information to make the comparison easier.

```
columns = ['product_title', 'product_bullet_point', 'product_brand', 'product_color_na
df['product_text'] = df[columns].apply(lambda row: ' '.join(row.values.astype(str)), a
```

```
df.drop(['product_title', 'product_bullet_point', 'product_brand', 'product_color_name
df.sample(5)
```

	query_id	query	product_id	esci_label	product_text
63033	6463	kid chapter book cd	099895800X	3	listening body guide helping kid understand co...
317561	12988	work tote	B0817H1D9Y	3	aldo nusz top handle satchel cognac adjustable...
219511	15059	ffp 2 kn 95	B08F2XXNCZ	3	coast kn 95 ce certified face mask individuall

CREATING VECTORS USING Word2vec FOR INFORMATION RETRIEVAL USING VECTOR SPACE MODEL

Since we have text data, it needs to be represented a numbers i.e inl mean vector form. `Word2vec` is being used here to achieve that as it converts words into vectors while keeping the semantical and syntactical relationship of words intact.

```
# Creating data for the model training
corpus= []

query = df['query'].values
product_text = df['product_text'].values

def create_corpus(text):
  for i in range(len(query)):
    corpus.append(query[i].split())

create_corpus(query)
create_corpus(product_text)
```

```
# Training model from the  corpus we created from our dataset
search_model = Word2Vec(corpus, size=250, min_count=2,window=5, sg=1,workers=4)
```

```
print('Size of the vocab:', len(search_model.wv.vocab))
```

```
Size of the vocab: 13431
```

```
# reperesentating texts as vectors
def get_word_vectors(search_tokens):
    wrd_vec_embed = []
    if len(search_tokens)<1:
        return np.zeros(250)
    else:
        for token in search_tokens:
            if token in search_model.wv.vocab:
                wrd_vec_embed.append(search_model.wv.word_vec(token))
            else:
                wrd_vec_embed.append(np.random.rand(250))
        return np.mean(wrd_vec_embed, axis=0)    # calculate the mean of vectos of eac

df['query_vec'] = df['query'].apply(lambda w :get_word_vectors(w.split()) if type(w) i
df['product_text_vec'] = df['product_text'].apply(lambda w :get_word_vectors(w.split()
```

```
df.head(5)
```

	query_id	query	product_id	esci_label	product_text	query_vec	
0	0	2 pencil sharpened	B0000AQO0O	1	ticonderoga beginner pencil wood cased 2 hb so...	[-0.1540759, -0.3330954, -0.49804088, 0.559671...	[-0.0 -0.01
1	8799	pencil kindergartener	B0000AQO0O	1	ticonderoga beginner pencil wood cased 2 hb so...	[-0.10089663, -0.22997263, -0.05785732, 0.3223...	[-0. 0.0
2	14844	2 dixon oriole pencil	B0000AQO0O	3	ticonderoga beginner pencil wood cased 2	[-0.0697052, -0.18165305, -0.32074255	[-0. 0.

```
from sklearn.metrics.pairwise import cosine_similarity

# Calculate similarity in quesy and product and calculate
def average_precision(query_id,query, product):

  # Get ground truth and product vectors
  ground_truth =df.loc[df['query_id']==query_id,['query','product_text','query_vec','p

  # Ranking documents for the query
  df['similarity']=ground_truth['query_vec'].apply(lambda x: cosine_similarity(np.arra
  df.sort_values(by='similarity',ascending=False,inplace=True)

  # Ranking Top 10 products by similarity score
  rank=df.head(10)['similarity'].values

  # presicion calculation
  mean_avg_precision=[]
  for i in range(1,8):
    if rank[i-1]:
      mean_avg_precision.append(np.sum(rank[:i])/i)

  if mean_avg_precision==[]:
    return 0

  return np.mean(mean_avg_precision)

# Calculating average precision for all queries in the test set
df['AP']=df[1:5000].apply(lambda x: average_precision(x['query_id'],x['query_vec'], x|

# Finding Mean Average Precision
print('Mean Average Precision=>',df['AP'].mean())
```

```
Mean Average Precision=> 0.6119489025339347
```

---------- INCREMENT 2 ----------

```
df = df.drop('AP', 1)
df.head(5)
```

	query_id	query	product_id	esci_label	product_text	query_vec	pr
1916	11557	old strong wither patch	B015G0EWLG	2	feelin good tee opinion offended adult humor s...	[0.16285768, 0.3682381, -0.38387278, 0.2461527...	[0.08 -0.06
2120	11557	old strong wither patch	B076FTC6FQ	2	got back graphic novelty sarcastic funny shirt...	[0.16285768, 0.3682381, -0.38387278, 0.2461527...	[0.08 -0.06
2249	11557	old strong wither patch	B078WZPYDV	2	f scott fitzgerald worth 11 x 14 minimalist in...	[0.16285768, 0.3682381, -0.38387278, 0.2461527...	[0.07 0.011
		old				[0.16285768,	

```
def retrieve_ranked_porducts(query_text):

  # pre-process Query
  query_text=query_text.lower()
  query_text=pre_process_text(query_text)
  query_text=stem_text(query_text)
  query_text=correct_spelling(query_text)
  query_text=re.sub(' +',' ',query_text)

  # generating vector
  vector=get_word_vectors(query_text.split())

  # ranking documents
  documents=df[['query_id','query','product_text']].copy()
  documents['similarity']=df['product_text_vec'].apply(lambda x: cosine_similarity(np.
  documents.sort_values(by='similarity',ascending=False,inplace=True)

  return documents.head(10).reset_index(drop=True)


retrieve_ranked_porducts('pan')[['product_text','similarity']]
```

	product_text	similarity
0	bushcraft takibi deep frying pan bushcraft tak...	0.702840
1	copper chef 10 inch diamond fry pan round fryi...	0.693440
2	avacraft 18 10 tri ply stainless steel frying ...	0.688362
3	copper chef 9 5 inch diamond fry pan square fr...	0.687114
4	small ceramic rectangular dish baking dish han...	0.685135
5	14 inch 1 multi use copper chef wonder cooker ...	0.682696

```
def get_products(query, N):
  query = pre_process_text(query) # preprocessing the query text
  query = correct_spelling(query) #correcting speling for query text
  query = stem_text(query)  #removing stop words and lemmatizing text
  query_tokens = query.split(" ") # breaking query text to tokens
  query = get_word_vectors(query_tokens) # converting tokens to vectors
  relevant_product_candidates = retrieve_ranked_porducts(query) #storing similar produ
  relevant_product_candidates['search_term'] = query
  cols = ['query_id ', 'query', 'product_text', 'similarity']
  return relevant_product_candidates[cols] #returns a new dataframe


#function to display title of ranked documents where n is the top n numbers of product
def get_relevant_products(query,n):
  product_set = retrieve_ranked_porducts(query)
  return product_set.sort_values('similarity', ascending=False).iloc[:n]['product_text
```

Checking for product search resuts ranked by their relevance

```
query = 'air conditioner'
print(f'The relevent suggestion for "{query}" are:')
for idx, product in enumerate(get_products(query,10)):
  print(idx+1, product)
```

```
he most relevant products to the search "air conditioner" are:
 portable air conditioner fan greatssly personal evaporative air cooler small roo
 dometic air conditioner b 59196 xx 1 c 0 brisk air ii heat pump 15 roof product
 evaporative air cooler portable air conditioner fan tower fan circle ping pong d
 electricity refrigeration heating air conditioning electricity refrigeration hea
 ingersoll rand garage mate 2 hp 5 5 cfm model p 1 5 iu 9 single stage electric c
 dometic 3314998 000 rv air conditioner replacement part non ducted heat strip rv
 rv c ducted air grille duo therm air conditioner grille replace dometic 3104928
 rv c ducted air grille duo therm air conditioner grille replace dometic 3104928
 ground stand mini split air conditioner g 380 candycane 3 ground stand mini spli
0 scent bomb air freshener scent bomb air freshener scent bomb
```

Checking for queries with incorrect spelling

```
query = 'aer conditioner'
print(f'The relevent suggestion for "{query}" are:')
for idx, product in enumerate(get_products(query,10)):
  print(idx+1, product)
```

```
The relevent suggestion for "aer conditioner" are:
1 portable air conditioner fan greatssly personal evaporative air cooler small r
2 dometic air conditioner b 59196 xx 1 c 0 brisk air ii heat pump 15 roof produc
3 evaporative air cooler portable air conditioner fan tower fan circle ping pong
4 electricity refrigeration heating air conditioning electricity refrigeration h
5 ingersoll rand garage mate 2 hp 5 5 cfm model p 1 5 iu 9 single stage electric
6 dometic 3314998 000 rv air conditioner replacement part non ducted heat strip
7 rv c ducted air grille duo therm air conditioner grille replace dometic 310492
8 rv c ducted air grille duo therm air conditioner grille replace dometic 310492
9 ground stand mini split air conditioner g 380 candycane 3 ground stand mini sp
10 scent bomb air freshener scent bomb air freshener scent bomb
```

Checking for query with number

```
query = '10'
print(f'The relevent suggestion for "{query}" are:')
for idx, product in enumerate(get_products(query, 10)):
  print(idx+1, product)
```

```
The most relevant products to the search "10" are:
1 rikon c 10 109 guide bearing 10 315 10 320 10 321 10 325 6 pack rikon c 10 109
2 10 second deodorant disinfectant pack 2 10 second 10 second deodorant disinfec
3 ecco flowt luxe buckle slide 10 10 5 ecco flowt luxe buckle slide 10 10 5 ecco
4 10 miracle deep conditioner plus keratin unisex 17 5 ounce 10 10 miracle deep
5 hot wheel 2019 larry wood 50 1969 2019 walmart exclusive 1 10 purple passion b
6 cisco sfp transceiver module 10 gigabit ethernet sfp 10 g sr x sfp 10 g sr x c
7 10 x 10 10 x 10 ticonderoga beginner pencil wood cased 2 hb soft eraser yellow
8 cisco sfp 10 g sr x 10 gbase sr sfp module extended temp range mpn sfp 10 g sr
9 apple juice organic 10 ounce case 24 10 ounce martinelli
10 gazebo universal replacement mosquito netting viragzas adjustable screen side
```

✓ 1m 36s completed at 2:06 PM