SECTION 1: CORPUS DETAILS AND SOURCE

Title : Amazon review polarity dataset

Our dataset contains 34,686,770 Amazon reviews from 6,643,669 users on 2,441,053 products, from the Stanford Network Analysis Project (SNAP).This subset contains 1,800,000 training samples and 200,000 testing samples in each polarity sentiment.The Amazon reviews polarity dataset is constructed by taking review score 1 and 2 as negative, and 4 and 5 as positive. Samples of score 3 are ignored. In the dataset, class 1 is the negative and class 2 is the positive.

We have used 10000 samples from the training data for building our corpus.We obtained the dataset from kaggle.

Preprocessing:Removing unnecessary columns,Tokenization,stop words removal and lemmatization was done during our preprocessing process.

1.Removing unnecessary columns

```
#droping unneccesary columns
df.drop('t_id', axis=1, inplace=True)
df.drop('Polarity', axis=1, inplace=True)
move = df.pop('Doc_ID')
df.insert(0,'Doc_Id',move)
df.head()
```

	Doc_Id	Title	Review
0	0	Love it	Love this product. Great shine even for a any ...
1	1	eye-opening...	I'd just like say that this book is absolutely...
2	2	A Beautiful Travelography	Alan Booth has the most wonderful way of makin...
3	3	It was very helpful.	I read this book a few years ago and used the ...
4	4	Goodle	A very nice introduction to large format. Give...

```
[ ] #converting sentences into lower case
    df["Review"] = df["Review"].str.lower()
    df.head()
```

	Doc_Id	Title	Review
0	0	Love it	love this product. great shine even for a any ...
1	1	eye-opening...	i'd just like say that this book is absolutely...
2	2	A Beautiful Travelography	alan booth has the most wonderful way of makin...
3	3	It was very helpful.	i read this book a few years ago and used the ...

2.Tokenization:

```
[ ] # tokenizing words
    df1=df['Review']
    l1=list()
    for line in df1:
      tokens=word_tokenize(line)
      l1.append(tokens)
```

```
[ ] #creating column for word tokens
    df1=df['Review']
    df['word_token']=l1
    df.head()
```

	Doc_Id	Title	Review	sentence_token	word_token
0	0	Love it	love this product. great shine even for a any ...	[love this product., great shine even for a an...	[love, this, product, ., great, shine, even, f...
1	1	eye-opening...	i'd just like say that this book is absolutely...	[i'd just like say that this book is absolutel...	[i, 'd, just, like, say, that, this, book, is,...
2	2	A Beautiful Travelography	alan booth has the most wonderful way of makin...	[alan booth has the most wonderful way of maki...	[alan, booth, has, the, most, wonderful, way, ...
3	3	It was very helpful.	i read this book a few years ago and used the ...	[i read this book a few years ago and used the...	[i, read, this, book, a, few, years, ago, and,...
4	4	Goodle	a very nice introduction to large format. give...	[a very nice introduction to large format., gi...	[a, very, nice, introduction, to, large, forma...

3.Stop words removal

```
[ ] # STOP WORDS REMOVAL
    import nltk
    nltk.download('stopwords')
    from nltk.corpus import stopwords
    stoplist= stopwords.words('english')
```

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

```
stoplist=set(stoplist)
l2=list()
for i in l1:
  output = [w for w in i if not w in stoplist]
  l2.append(output)
df['stop_words_removed']=l2
df.head()
```

	Doc_Id	Title	Review	sentence_token	word_token	stop_words_removed
0	0	Love it	love this product. great shine even for a any ...	[love this product., great shine even for a an...	[love, this, product, ., great, shine, even, f...	[love, product, ., great, shine, even, nail, p...
1	1	eye-opening...	i'd just like say that this book is absolutely...	[i'd just like say that this book is absolutel...	[i, 'd, just, like, say, that, this, book, is,...	['d, like, say, book, absolutely, fantastic, !...
2	2	A Beautiful Travelography	alan booth has the most wonderful way of makin...	[alan booth has the most wonderful way of maki...	[alan, booth, has, the, most, wonderful, way, ...	[alan, booth, wonderful, way, making, reader, ...
3	3	It was very helpful.	i read this book a few years ago and used the ...	[i read this book a few years ago and used the...	[i, read, this, book, a, few, years, ago, and,...	[read, book, years, ago, used, information, re...
4	4	Goodle	a very nice introduction to large format. give...	[a very nice introduction to large format., gi...	[a, very, nice, introduction, to, large, forma...	[nice, introduction, large, format, ., gives, ...

4.Lemmatization

```
# LEMMATIZATION
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
final_train_lemma_word = []
wordnet_lemmatizer = WordNetLemmatizer()
for line in df['stop_words_removed']:
  lemma_word=[];
  for w in line:
      word1 = wordnet_lemmatizer.lemmatize(w, pos = "n")
      word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
      word3 = wordnet_lemmatizer.lemmatize(word2, pos = ("a"))
      lemma_word.append(word1)
  lemma_word= [word for word in lemma_word if word.isalnum()]
  final_train_lemma_word.append(lemma_word)
```

Results of lemmatized words :

```
[ ] df['lemmatized_words']=final_train_lemma_word
    df.head()
```

	Doc_Id	Title	Review	sentence_token	word_token	stop_words_removed	lemmatized_words
0	0	Love it	love this product. great shine even for a any ...	[love this product., great shine even for a an...	[love, this, product, ., great, shine, even, f...	[love, product, ., great, shine, even, nail, p...	[love, product, great, shine, even, nail, poli...
1	1	eye-opening...	i'd just like say that this book is absolutely...	[i'd just like say that this book is absolutel...	[i, 'd, just, like, say, that, this, book, is,...	['d, like, say, book, absolutely, fantastic, !...	[like, say, book, absolutely, fantastic, well,...
2	2	A Beautiful Travelography	alan booth has the most wonderful way of makin...	[alan booth has the most wonderful way of maki...	[alan, booth, has, the, most, wonderful, way, ...	[alan, booth, wonderful, way, making, reader, ...	[alan, booth, wonderful, way, making, reader, ...
3	3	It was very helpful.	i read this book a few years ago and used the ...	[i read this book a few years ago and used the...	[i, read, this, book, a, few, years, ago, and,...	[read, book, years, ago, used, information, re...	[read, book, year, ago, used, information, res...
4	4	Goodle	a very nice introduction to large format. give...	[a very nice introduction to large format., gi...	[a, very, nice, introduction, to, large, forma...	[nice, introduction, large, format, ., gives, ...	[nice, introduction, large, format, give, mome...

SECTION-2 : DATA STRUCTURES USED

- All the tokens generated after lemmatization are stored in a list. The data structure of 'lemmatized_words' is a list of lists, where each primary list represents a document and the secondary list represents the tokens of that document.
- For creating an inverted index, we have used a dictionary where the keys are the terms and the values are a list of document IDs along with their TF-IDF score. Dictionary data structure helps it easier to map terms to the documents it belongs to and the list helps in keeping track of a lot of elements(document IDs). Inverted index is used for Boolean retrieval model and to perform wild card queries.
- We have created a positional index to perform phrase queries. We have implemented a positional index in the form of a dictionary. The keys represent the terms and the values are another dictionary with document IDs as keys and position of the term as values. Since we needed to map the position of the terms to documents, it was necessary to use a dictionary as a secondary index rather than a simple list as in inverted index. Positional index is used to perform phrase queries.

SECTION 3.1 : RESULT OF BOOLEAN RETRIEVAL ON FREE-TEXT QUERIES

The Boolean retrieval model can perform 3 operations namely, "AND", "OR' and "NOT".

We have defined our own functions to perform each operation. "AND" operations retrieves documents which contain both the word, "OR" operation retrieves documents containing either of the words and the "NOT" operation retrieves documents not having the word specified.

User defined function for "AND" operation :

```
def and_query(l1, l2):
  query_terms = [l1,l2]
  # Retrieve the postings lists for the query terms
  postings_lists = [inverted_index[term] for term in query_terms]

  # Compute the intersection of the postings lists
  doc_ids = set(postings_lists[0][i][0] for i in range(len(postings_lists[0])))
  for postings in postings_lists[1:]:
      doc_ids &= set(postings[i][0] for i in range(len(postings)))

  # Compute the TF-IDF weights for the matching documents
  matching_docs = []
  for doc_id in doc_ids:
      tfidf = sum(postings[i][1] for i, postings in enumerate(postings_lists)
                  if doc_id in set(p[0] for p in postings))
      matching_docs.append((doc_id, tfidf))

  # Sort the matching documents by their TF-IDF weights
  matching_docs.sort(key=lambda x: x[1], reverse=True)
  return matching_docs
```

User defined function for “OR” operation :

```
def or_query(l1,l2):
  query_terms = [l1,l2]
  # Retrieve the postings lists for the query terms
  postings_lists = [inverted_index[term] for term in query_terms]

  # Compute the union of the postings lists
  doc_ids = set(postings_lists[0][i][0] for i in range(len(postings_lists[0])))
  for postings in postings_lists[1:]:
      doc_ids |= set(postings[i][0] for i in range(len(postings)))

  # Compute the TF-IDF weights for the matching documents
  matching_docs = []
  for doc_id in doc_ids:
      tfidf = sum(postings[i][1] for i, postings in enumerate(postings_lists)
                  if doc_id in set(p[0] for p in postings))
      matching_docs.append((doc_id, tfidf))

  # Sort the matching documents by their TF-IDF weights
  matching_docs.sort(key=lambda x: x[1], reverse=True)
  return matching_docs
```

User defined function for “NOT” query :

```
def not_query(word):
  exclude_doc_ids = set(inverted_index[word][i][0] for i in range(len(inverted_index[word])))
  # Exclude documents containing the term
  all_ids = [x for x in range(0,len(df))]
  result = []
  for doc_ids in all_ids :
      if doc_ids not in exclude_doc_ids:
        result.append(doc_ids)

  # The resulting inverted index will not contain the excluded documents
  return result
```

Code to perform query on Boolean model :

```
# PERFORMING THE BOOLEAN QUERY

word1,operation,word2 = input("Enter the query : ").split()
if word1 in inverted_index and word2 in inverted_index:
  if operation=='and':
    matching_docs = and_query(word1,word2)
    print(f"Number of documents retrieved : {len(matching_docs)}")
    for doc_id, tfidf in matching_docs:
      print(f"Document {doc_id}: {df['Review'][doc_id]} (TF-IDF: {tfidf})")
  else:
    matching_docs = or_query(word1,word2)
    print(f"Number of documents retrieved : {len(matching_docs)}")
    for doc_id, tfidf in matching_docs:
      print(f"Document {doc_id}: {df['Review'][doc_id]} (TF-IDF: {tfidf})")
else:
  print("Entered word doesnt exist")
```

```
Enter the query :
```

In the above code, the system takes the input from the user and performs the operation specified in the query. It returns the resultant documents and if the queried word does’nt exist, it raises an error.

Output :

```
Enter the query : good and book
Number of documents retrieved : 513
Document 8192: this book was great is was well written and you got a feel for the characters. i would definitely recommend to those
Document 2: alan booth has the most wonderful way of making the reader feel his triumphs and sorrows. his writing style gives the b
Document 8196: i only read just a few pages of this book, and so far it is the same as all the others. captivating, and fun, positi
Document 8199: this is a great book for young children. there is no bad language just good, clean reading. it's funny and amusing a
Document 8200: revision 2 of lpi linux in a nutshell is a serious book to study for lpi linux and is a good linux reference guide.
Document 4110: i was a little disappointed in the book as it did not meet our needs. i was most interested in "inside" info for oly
Document 16: this is the best book i have ever read.i think this is a great book for kids.if i where you i would read this book.you
Document 6163: reads like a good novel. the first hand accounts woven into the narrative are well selected and perfectly integratec
Document 6166: though more loon research has been completed since 1985 this book is an excellent introduction to the common loon. r
Document 23: project management jump start is the best introduction to project management that i've seen. easy to follow, conversat
Document 2074: i bought this book because of the good reviews it had at amazon. the practice problems were very unrealistic. some c
Document 8220: i like this book, the way it show the posters and information about it. the posters and designers ares really good e
Document 2079: love monkey is a bad book by anyone's impartial standards or reviews. those who aren't objective have another agenda
Document 4127: so many "ezzo-ites" claim the only people who don't like this book are people who haven't read it and haven't tried
Document 6175: it's sad to me when a transparent christian and a fine writer dies and his writings completely disappear from public
Document 4132: good luck getting a workable plan out of this book. it is so chopped up and convaluted you will pull your hair out.
Document 4138: this book goes against all natural mother instincts within me. i was given this book by a friend when pregnant with
Document 4142: i read this book while i was pregnant and thought hey, i will give it a shot. i have talked with many moms who have
Document 8239: i am enjoying other des books, but "the english air" deals directly with politics and the start of wwii. it is impos
Document 53: i have a copy macromedia director 8 bible from the same authors, and thought i might learn new director capabilities l
```

In the above query, 513 reviews are retrieved which consist of both the words “good” and “book”.

```
Enter the query : good or book
Number of documents retrieved : 3792
Document 8192: this book was great is was well written and you got a feel for the characters. i would definitely recommend to those who lik
Document 2: alan booth has the most wonderful way of making the reader feel his triumphs and sorrows. his writing style gives the book a ve
Document 8196: i only read just a few pages of this book, and so far it is the same as all the others. captivating, and fun, positive and
Document 8199: this is a great book for young children. there is no bad language just good, clean reading. it's funny and amusing and will
Document 8200: revision 2 of lpi linux in a nutshell is a serious book to study for lpi linux and is a good linux reference guide. (TF-IDF
Document 16: this is the best book i have ever read.i think this is a great book for kids.if i where you i would read this book.you will lo
Document 23: project management jump start is the best introduction to project management that i've seen. easy to follow, conversational to
Document 8220: i like this book, the way it show the posters and information about it. the posters and designers ares really good example
Document 8239: i am enjoying other des books, but "the english air" deals directly with politics and the start of wwii. it is impossible to
Document 53: i have a copy macromedia director 8 bible from the same authors, and thought i might learn new director capabilities like fla
Document 8245: i used this book to study for the ap physics b exam on my own, and thanks to it, i was able to score a 5. good book for peo
Document 57: the format, color plates and essay about porter's life are all well done. however, it seems to me that despite his excellent
Document 68: say the boss drops dead and suddenly you're the acting boss. or the company reorganizes, everyone above you is fired and gues
Document 70: starts out promising and goes into a lot of detail but as it builds to a grande finale, we find out that the court case hadn't
Document 8270: marcia adams heartland is a beautfully done cookbook.ihave all her books and watch her tv shows.irecommend it to any one wh
Document 79: i came away from this book feeling bad for the children. i just don't think its right to have their early childhood and their
Document 8277: i used this book just prior to the exam -- it was a great refresher! questions are good, well written. this is the book to
Document 8278: this book was really good, it was like a gansta version of the coldest winter ever.. even dough i don't know why it was cal
Document 8279: i loved this book. it's really good. precious comes from the streets trying to support her and her mom who has a bad drug h
Document 8280: i think this is a very good book. it's keep you wanting more it's like when you read it you can actually see what's going o
Document 8291: i, like the previous reviewer, wanted to like the book. i like good whodunits, but this is no whodounit. this is a md turne
Document 100: this book is more about the careers and images of fred astaire and ginger rogers from 1934-49 than it is a study of their mo
Document 109: gotta admit, i was very disappointed with this book. if you don't own any astronomy books and want a good primer, this is ok
Document 8313: this is the only book i read on switching and i scored 956 out of 1000 on the 640-504. i was, however, fortunate enough to
Document 125: the incredible hulk may not be the darkest comic-to-big screen adaptation yet, but it may be the best mixture. it's a true p
Document 133: i found the book to be interesting. there was very little new material, however, especially for those of us who followed the
Document 8331: we used this book for our "bridge to abstract mathematics" class. this book falls short of what you would consider a book t
Document 147: i was so happy to find this! i read this book when i was a child in the 80s, re-read it so many times, i wore it out.it's th
Document 8346: if you need to decide how to rate an area, this book might be useful. however, if you want to know how to wire a motor from
```

Here is a query with “OR” operation for the same words. We can see that 3792 reviews have been retrieved.

Below is an example for a “NOT” query.

```
word = input("Enter a word : ")
result = not_query(word)
print(f"The number of documents which do not contain the term '{word}' : {len(result)}")
print("The document IDs are :  ")
print(result)
```

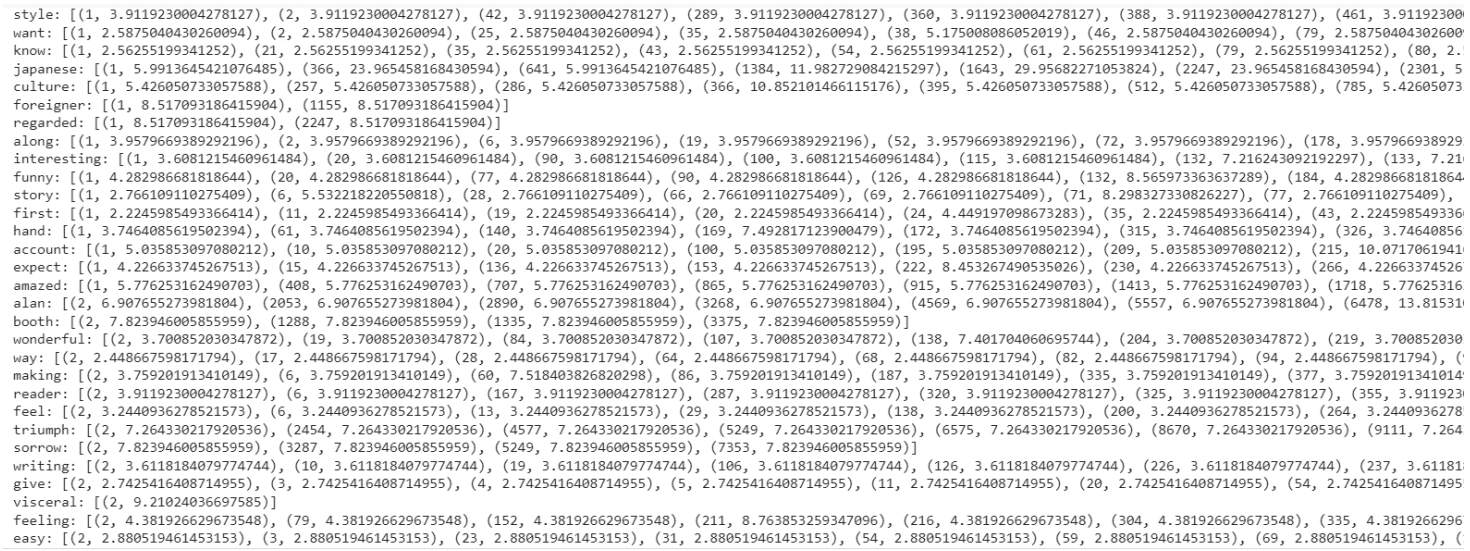
```
Enter a word : love
The number of documents which do not contain the term 'love' : 8987
The document IDs are :
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 17, 18, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
```

SECTION 3.2 : RESULT OF INVERTED INDEX ON FREE-TEXT QUERIES WITH RANK

Code for creation of inverted-index :

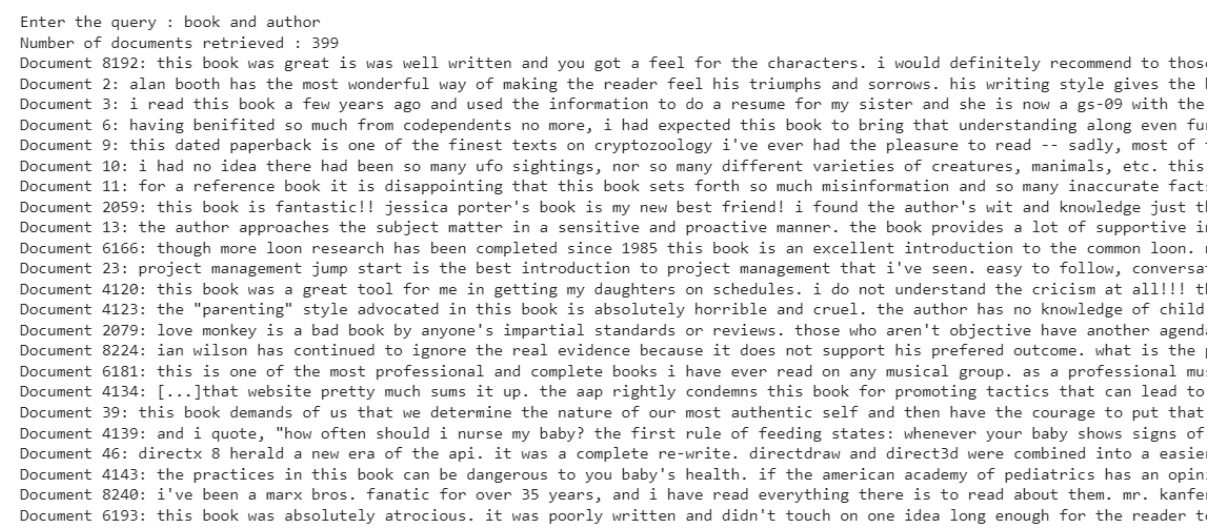
```
[111] import math
      from collections import defaultdict

      # Compute the document frequencies for each term
      dfs = defaultdict(int)
      for doc in final_train_lemma_word:
          for term in set(doc):
              dfs[term] += 1

      # Compute the inverse document frequencies
      N = len(df)
      idfs = {term: math.log(N / dfa) for term, dfa in dfs.items()}

      # Compute the TF-IDF weights for each term in each document
      tfidfs = []
      for doc in final_train_lemma_word:
          tfidf = defaultdict(float)
          for term in doc:
              tfidf[term] += 1
          tfidf = {term: tf * idfs[term] for term, tf in tfidf.items()}
          tfidfs.append(tfidf)

        # Construct the inverted index
      inverted_index = defaultdict(list)
      for i, doc in enumerate(tfidfs):
          for term, weight in doc.items():
              inverted_index[term].append((i, weight))

      # Print the inverted index
      for term, postings in inverted_index.items():
          print(f"{term}: {postings}")
```

Inverted-index :



Inverted index is a dictionary with terms as keys and the values are a list of tuples with document ID and its respective TF-IDF score.

Result of the query using inverted index :



The retrieved documents are ranked based on the tf-idf scores. In this way, the top documents are more relevant than the lower ones.

SECTION 3.3 : RESULT OF WILD-CARD QUERIES

Code:User defined function for wild-card query and taking input.

```
import re

def wild_card_query(word):
  wildcard_query = word
  # Compile a regular expression pattern from the wildcard query
  regex_pattern = re.compile(wildcard_query)

  # Find all terms in the inverted index that match the regular expression
  matching_terms = [term for term in inverted_index.keys() if regex_pattern.match(term)]

  # Retrieve the postings lists for the matching terms
  postings_lists = [inverted_index[term] for term in matching_terms]

  # Merge the postings lists into a single list of (document ID, TF-IDF weight) pairs
  merged_postings = []
  for postings in postings_lists:
      merged_postings += postings
  return merged_postings


#QUERY

word = input("Enter a wild card query (like 'poli.*') : ")
matching_docs = wild_card_query(word)
print(f"Number of documents retrieved : {len(matching_docs)}")
for doc_id, tfidf in matching_docs:
  print(f"Document {doc_id}: {df['Review'][doc_id]}")
```

```
Enter a wild card query (like 'poli.*') : lapto.
```

Output:

```
Enter a wild card query (like 'poli.*') : lapto.
Number of documents retrieved : 49
Document 129: i think that its awesome because the colors that it lights up in are cool {there red and blue}and also when you put your f
Document 279: while i think the price is a bit high considering what it is, the thickness of the material, the dog-bone shape and the wi
Document 317: unbelievably bad product. i received this item on june 8, 2010. by june 24 (16 days later) i knew there was something wron
Document 623: i purchased this laptop 3 years ago and have thoroughly used and abused it yet it still runs. i have dropped it (in case)
Document 754: i bought this laptop 2 years ago - the battery die 6months after. then the mother board died in 1 year. i got it fix - tha
Document 860: even though this power adapter says "120w", it cannot provide enough power for my dell d800 laptop which requires a 90w po
Document 1392: i just bought one of the starter systems with the controller and one wireless camera. all i really wanted was a camera at
Document 1706: i would give this a 1/2 a star, or less if i could. i tried to use this cable from the vga output on my 2007 laptop to th
Document 1711: i dont know !! what went wrong...i tried to connect laptop to tv, pc to tv, pc to projector but none worked....!!!!!
Document 2460: this bckrest has been the ultimate life saver for me. my doctor recently put me on bed rest due to back pain; with this s
Document 2678: i had lots of trouble with this mouse, especially the way it tracks. i normally use my laptop touch pad, but for making d
Document 2978: this dvd will not play on my dvd player. player is relatively new and should play any dvd. it will play on my laptop. thi
Document 2998: i bought this card to replace a belkin card of same model that died in my laptop. i thought i needed new card, until i tr
Document 3000: it made my wireless connection much slower than the built in wireless card in my laptop, to the point where it just disco
Document 3190: i chose this table because i wanted to use it as a laptop computer table while sitting in my recliner. i looked at a lot
Document 3632: the game simply does not work. we tried loading it on all five home computers (two desktops & three laptops), none worked
Document 4364: surely one disc for all 3 films unexextended should be included, or at least one disc per extended. changing out discs is
Document 4541: you couldn't want more!! it plays dvd's , cd's allows you to plug in all of your components (projector, laptop, vcr, comp
Document 4663: i am not sure anyone would still use 17" laptop. this is designed to carry 17" laptop. i bought it because i thought it w
Document 4702: i thing tigerdirect sent me a refurbished machine. bad packing, etc. i won't buy anything from them, ever. the laptop is
Document 5280: this router worked great for a couple of months. unfortunately my laptop stopped detecting the router. after 3 days of tr
Document 5282: this is a horrible product. i had to run the setup wizard about 10 times before the router would initiate an maintain a c
Document 5283: this is a good product, but the linksys 54g desktop card is not. i'd recommend this is you want to wireless network your
Document 5309: before buying this card, think about the laptop you're going to put it in. if it is a p3/800mhz or older machine, this ca
Document 5310: linksys obviously doesn't have all the bugs worked out. i followed the documentation that came with the card for windows
Document 5570: i burned a cd from my dell laptop and it will only play 10 songs or so. i thought one of the cds were bad so i tried agai
```

Wildcards are special characters that can stand in for unknown characters in a text value and are handy for locating multiple items with similar, but not identical data.

As you can see,49 relevant documents have been retrieved from the corpus for the wild card query”lapto.”.

SECTION 3.4 : RESULT OF PHRASE QUERIES

Phrase queries are performed with the help of positional index.

Code for creation of positional index :

```
pos_index = {}
file_map = {}
def generate_positional_index(data:list):
  fileno=0
  lineno=-1
  for line in data:
    lineno+=1;
    for pos, term in enumerate(line):
      if term in pos_index:
          if fileno in pos_index[term].keys():
            pos_index[term][0][lineno].append(pos)
          else:
            pos_index[term][lineno] = [pos]
      else:
          pos_index[term] = {}
          pos_index[term][lineno] = [pos]
      fileno += 1
  return pos_index

positional_index=generate_positional_index(l1)
count=0
for i in positional_index:
  count=count+1;
  if count<=20:
    print(i,positional_index[i])
  else:
    break;
```

Positional index :

```
love {0: [0], 12: [66], 16: [31], 19: [2], 22: [87], 44: [26], 52: [132], 74: [83], 76: [60], 77: [12], 90: [42], 91: [49], 115: [38], 117: [
this {0: [1], 1: [56], 2: [141], 3: [2], 5: [21], 6: [109], 8: [96], 9: [181], 10: [88], 11: [8], 12: [0], 13: [76], 15: [114], 16: [23], 17:
product {0: [2], 29: [5], 31: [100], 33: [8], 48: [6], 75: [1], 81: [15], 84: [61], 101: [5], 108: [49], 148: [6], 149: [30], 160: [25], 163:
. {0: [13], 1: [66], 2: [166], 3: [83], 4: [23], 5: [35], 6: [128], 7: [32], 8: [115], 9: [179], 10: [85], 11: [113], 12: [63], 13: [79], 14:
great {0: [4], 14: [0], 16: [13], 17: [27], 31: [4], 47: [57], 50: [68], 53: [38], 56: [81], 61: [132], 62: [40], 63: [2], 64: [140], 75: [7]
shine {0: [5], 198: [68], 380: [149], 419: [51], 1775: [26], 2537: [31], 5014: [154], 5252: [19], 6418: [25], 8082: [63], 8579: [73], 8731: [
even {0: [6], 6: [19], 15: [79], 22: [69], 36: [23], 38: [32], 43: [41], 51: [25], 60: [24], 64: [117], 69: [29], 70: [28], 76: [133], 91: [5
for {0: [7], 2: [97], 3: [16], 7: [19], 11: [61], 12: [44], 13: [71], 16: [15], 17: [39], 19: [55], 20: [93], 21: [16], 22: [13], 24: [48], 2
a {0: [25], 1: [58], 2: [163], 3: [73], 4: [11], 5: [25], 6: [103], 7: [29], 8: [92], 9: [71], 10: [55], 11: [101], 12: [45], 13: [16], 15: [
any {0: [9], 10: [61], 12: [56], 37: [62], 45: [53], 53: [90], 67: [49], 80: [99], 98: [31], 108: [17], 109: [17], 112: [27], 133: [71], 149:
nail {0: [10], 65: [20], 175: [17], 1858: [79], 3033: [61], 3138: [61], 3171: [72], 3327: [2], 4009: [46], 9353: [105], 9404: [43], 9991: [45
polish {0: [11], 224: [101], 382: [39], 2380: [19], 2383: [72], 3110: [85], 4388: [5], 5168: [3], 6117: [28], 7203: [37], 7689: [106], 8758:
color {0: [12], 52: [144], 56: [21], 57: [3], 279: [27], 447: [5], 556: [39], 637: [29], 725: [122], 799: [132], 926: [47], 957: [82], 998: [
dries {0: [14], 1825: [61], 2208: [29], 3582: [18], 3586: [44], 6535: [83], 9641: [72], 9921: [149]}
pretty {0: [15], 21: [25], 38: [133], 56: [190], 93: [33], 122: [11], 137: [1], 176: [14], 184: [4], 189: [127], 225: [187], 229: [24], 249:
quick {0: [16], 68: [184], 328: [47], 411: [7], 413: [18], 486: [74], 512: [24], 708: [3], 784: [42], 852: [33], 976: [29], 998: [119], 1153:
, {0: [17], 1: [55], 2: [150], 3: [62], 5: [17], 6: [23], 7: [24], 8: [111], 9: [148], 10: [101], 11: [109], 13: [62], 14: [21], 15: [100], 1
so {0: [18], 1: [14], 5: [18], 6: [2], 9: [46], 10: [13], 11: [16], 12: [49], 17: [32], 19: [83], 24: [33], 25: [37], 28: [13], 36: [77], 40:
i {0: [19], 1: [0], 2: [82], 3: [69], 4: [16], 5: [19], 6: [106], 7: [11], 9: [139], 10: [39], 12: [64], 15: [86], 16: [29], 17: [35], 18: [1
do {0: [20], 3: [13], 7: [12], 8: [112], 21: [36], 24: [58], 43: [70], 50: [71], 52: [79], 54: [45], 58: [77], 59: [39], 61: [157], 67: [46],
```

Code for performing phrase queries :

```
from IPython.display import clear_output

#FUNCTION TO PERFORM PHRASE QUERY
def phrase_query(phrase):
  query = phrase
  # Split the query into terms
  query_terms = query.split()

  # Initialize the result list
  result = []

  # Loop through the documents and find those that contain the phrase
  for doc_id in positional_index[query_terms[0]]:
      # Get the positions of the first query term in the document
      positions = positional_index[query_terms[0]][doc_id]
      # Loop through the positions and check if the rest of the query terms are present
      for pos in positions:
          match = True
          for i in range(1, len(query_terms)):
              if doc_id not in positional_index[query_terms[i]]:
                  match = False
                  break
              if pos+i not in positional_index[query_terms[i]][doc_id]:
                  match = False
                  break
          if match:
              # If all query terms are present in the correct order, add the document to the result list
              result.append(doc_id)
  return result
```

Results:

```
Enter a phrase :good product
Top 10 Documents containing the phrase 'good product':
```

- [ ] 751:Monster trucks
- [ ] 1117:WARNING: Made in China!
- [ ] 1369:Not good for ice cream :(
- [ ] 2273:6 PILLS A DAY??!
- [ ] 3798:It Happens.
- [ ] 4459:BANG! Ha! Ha! Ha! Ha!
- [ ] 5057:NEVER BUY A TOM TOM
- [ ] 5283:Good product
- [ ] 7185:Be AWARE the fastener i…
- [ ] 7657:Good product

The top 10 documents perceived as relevant by the system are retrieved according to the phrase query. A relevance feedback mechanism from the user is also implemented here which will be discussed in the next section.

SECTION 3.5 : RELEVANCE FEEDBACK

The system fetches top 10 document titles based on the user's query and asks the user for feedback.The user gives the feedback in the form of checks in the checkboxes as you can see from the screenshot. After the user checks, the system retrieves only document contents of those documents which are marked as relevant by the user. The system also calculates the precision based on this feedback.

Code:

```
data = []
print(f"Top 10 Documents containing the phrase '{phrase}':")
for i in range(10):
  temp = str(result[i])+ ":" +str(df['Title'][result[i]])
  data.append(temp)
checkboxes = [widgets.Checkbox(value=False, description=label) for label in data]
output = widgets.VBox(children=checkboxes)
display(output)
```

```
True_positive = 0
print("Here are the documents which you found relevant : ")
for i in range(0, len(checkboxes)):
    if checkboxes[i].value == True:
        True_positive += 1
        print(f"{result[i]} : {df['Review'][result[i]]}")

#Calculation of metrics for relevance
#Precision
precision = True_positive/10
print(f"The precision of the retrieved documents : {precision}")
```

Result:

```
Enter a phrase :good product
Top 10 Documents containing the phrase 'good product':
```

- [x] 751:Monster trucks
- [x] 1117:WARNING: Made in China!
- [x] 1369:Not good for ice cream :(
- [x] 2273:6 PILLS A DAY??!
- [ ] 3798:It Happens.
- [ ] 4459:BANG! Ha! Ha! Ha! Ha!
- [x] 5057:NEVER BUY A TOM TOM
- [ ] 5283:Good product
- [ ] 7185:Be AWARE the fastener i...
- [x] 7657:Good product

The user has checked 6 documents as relevant.

```
Here are the documents which you found relevant :
751 : my nephew loves monster trucks and he is only two. they are a little smaller than i hc
1117 : i was just about to give these to my dogs and realized that they are made in china. u
1369 : this product could be great if you just want to mix powdered drinks. however, i bough
2273 : i should have read the label closer. seems like a good product but keep in mind you w
5057 : if you want a good product stay away from tom tom products. i have used 3 different p
7657 : good product. fit perfectly and was easy to install. works great. just as advertised.
The precision of the retrieved documents : 0.6
```

After the feedback, document contents of those 6 relevant documents are retrieved and the precision of the system is calculated.