**AIWR ASSIGNMENT-1**

Team members :

1. Sehag A - PES2UG20CS457
2. Setti Durga Poojitha - PES2UG20CS458
3. Shreyas Sai Raman - PES2UG20CS461
4. Varshini Jayasankar - PES2UG20CS567

**SECTION 1: CORPUS DETAILS AND SOURCE**

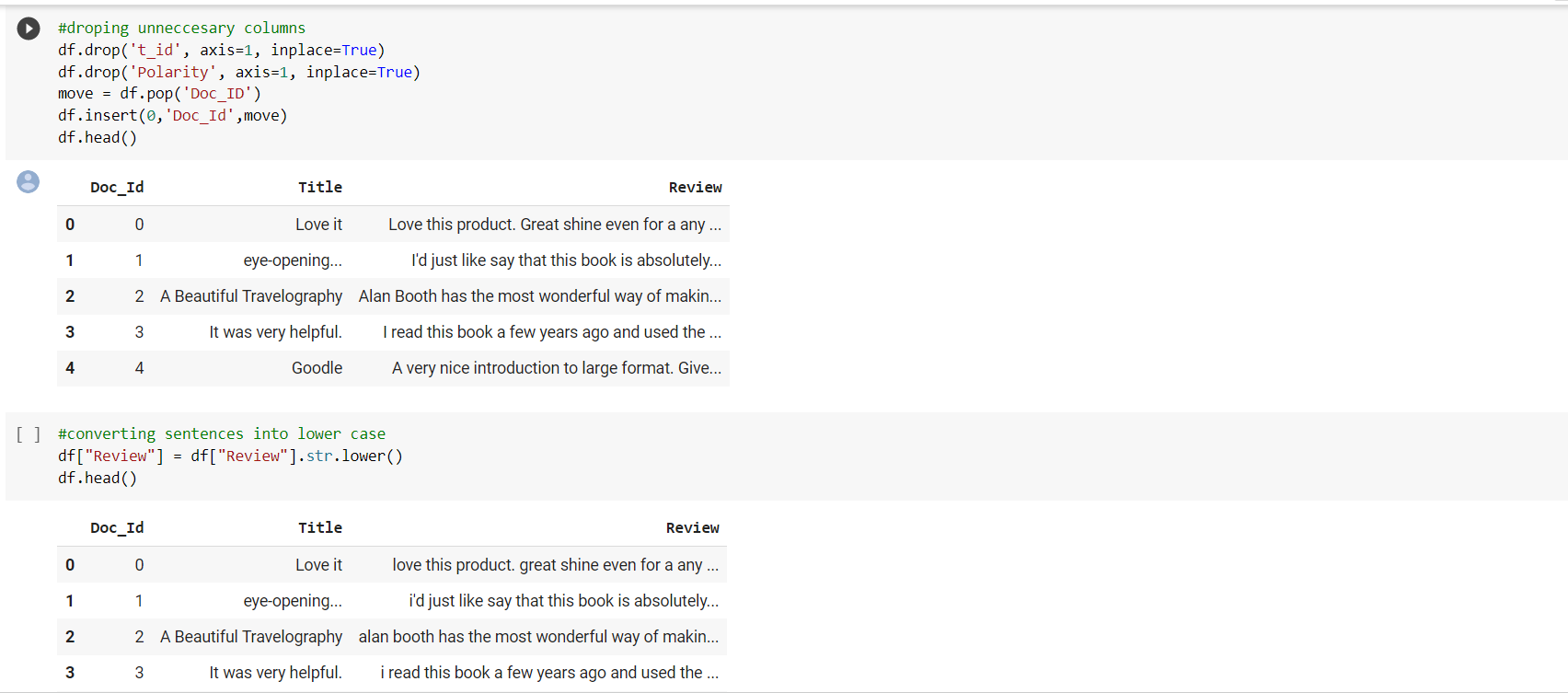
Title : Amazon review polarity dataset

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.

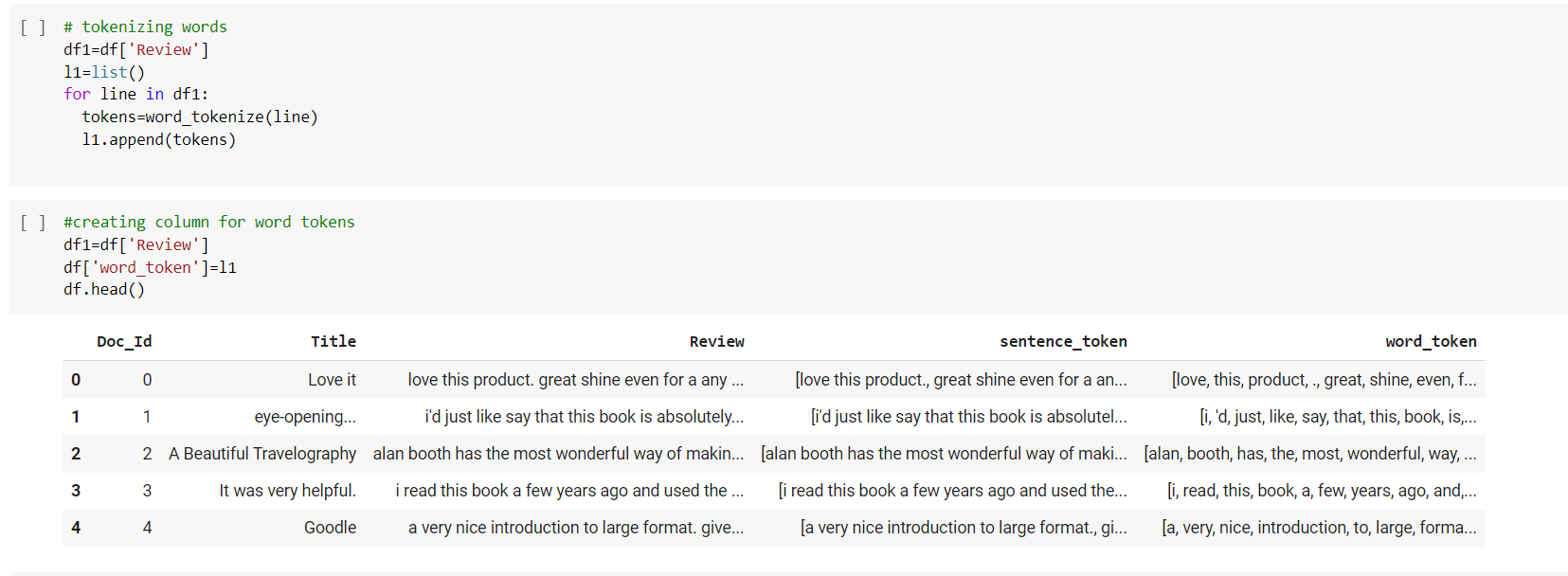
We have used 10000 samples from the training data for building our corpus.We obtained the dataset from kaggle.

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

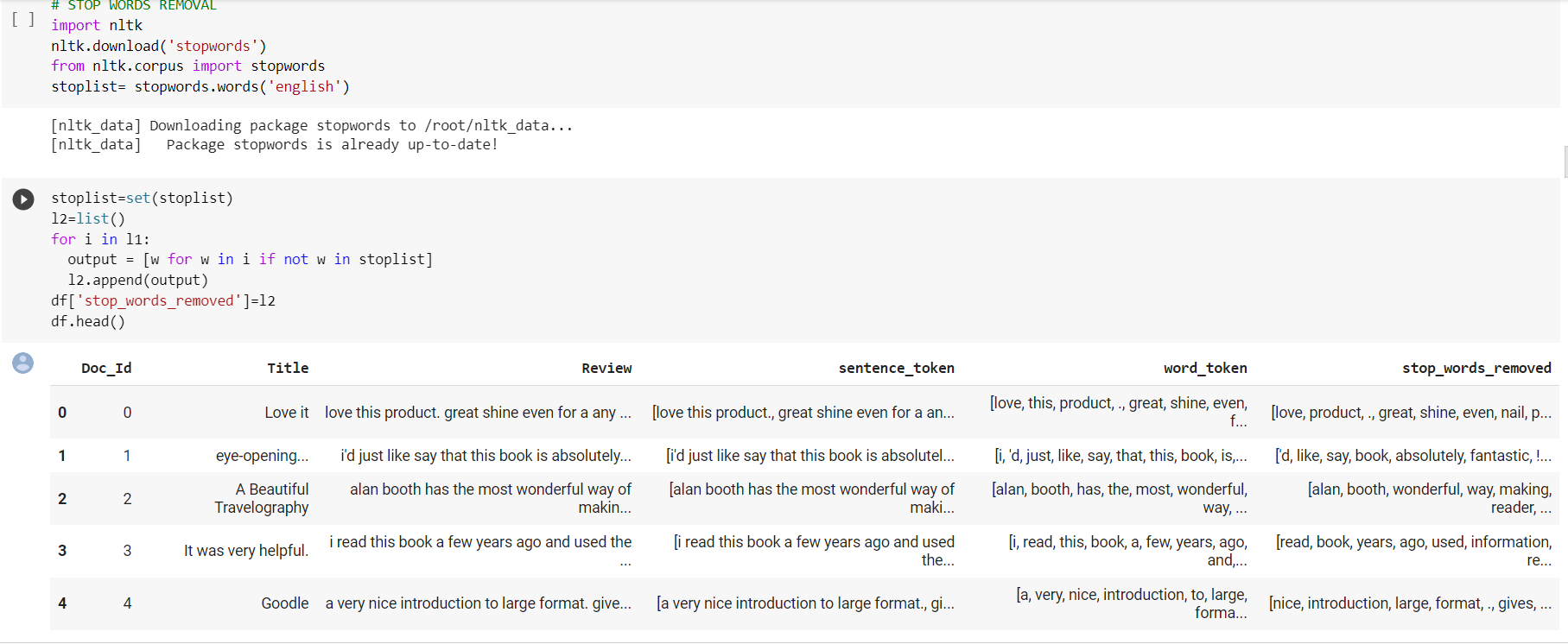
1.Removing unnecessary columns



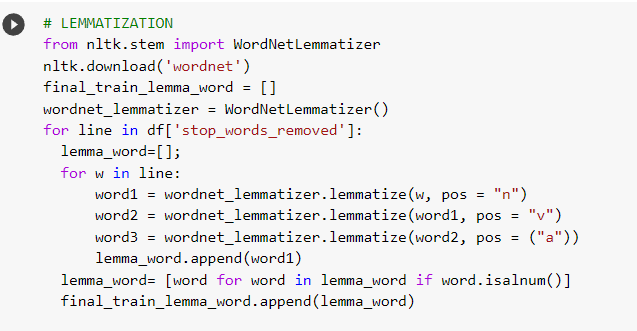
2.Tokenization:



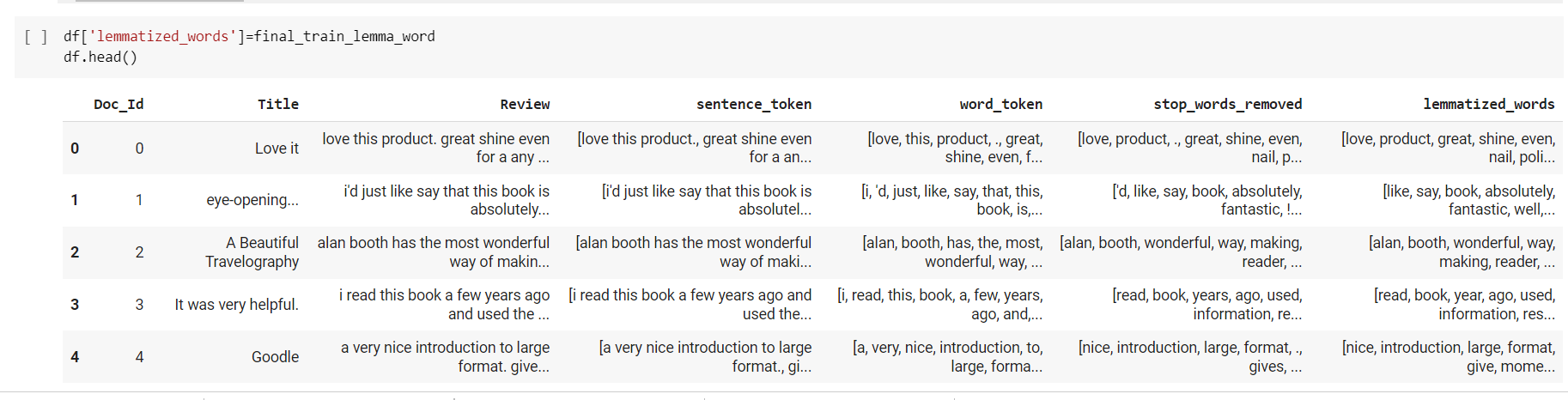
3.Stop words removal



4.Lemmatization



Results of lemmatized words :



**SECTION-2 : DATA STRUCTURES USED**

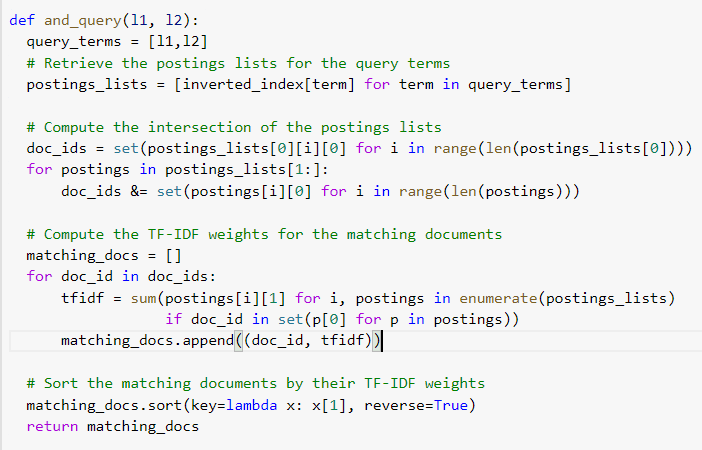
* All the tokens generated after lemmatization are stored in a list. The data structure of ‘lemmatized\_words’ is a list of lists, where each primary list represents a document and the secondary list represents the tokens of that document.
* For creating an inverted index, we have used a dictionary where 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**

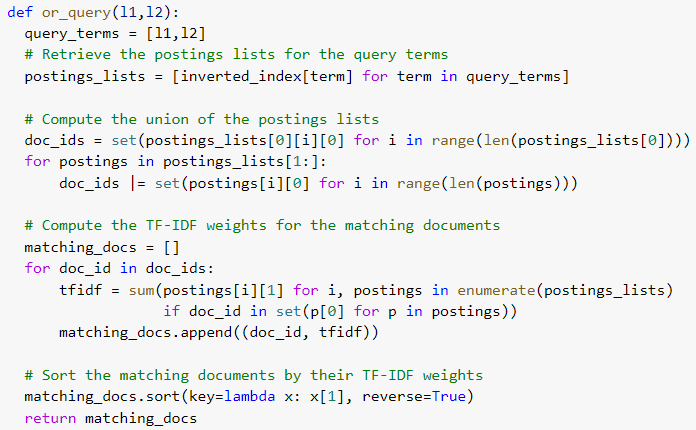
The Boolean retrieval model can perform 3 operations namely, “AND”, “OR’ and “NOT”.

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.

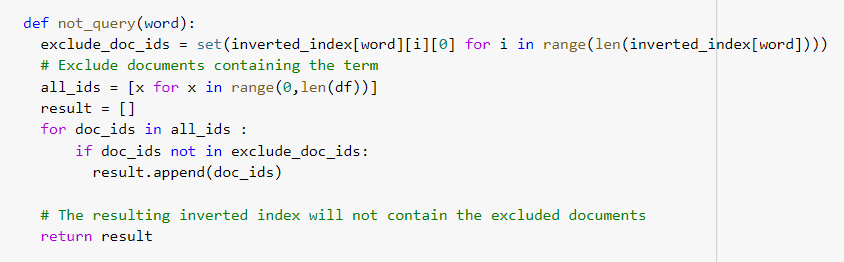
User defined function for “AND” operation :



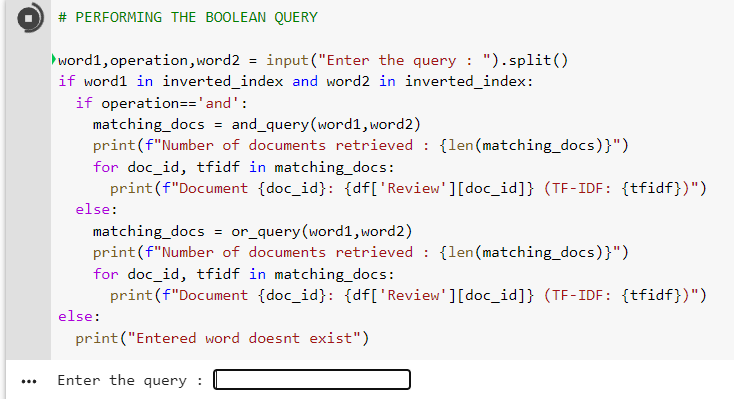
User defined function for “OR” operation :



User defined function for “NOT” query :

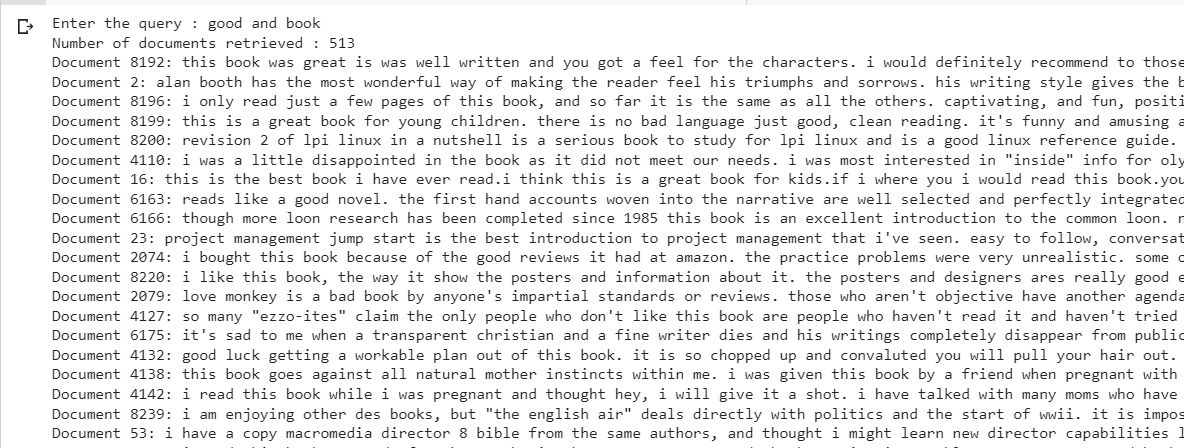


Code to perform query on Boolean model :

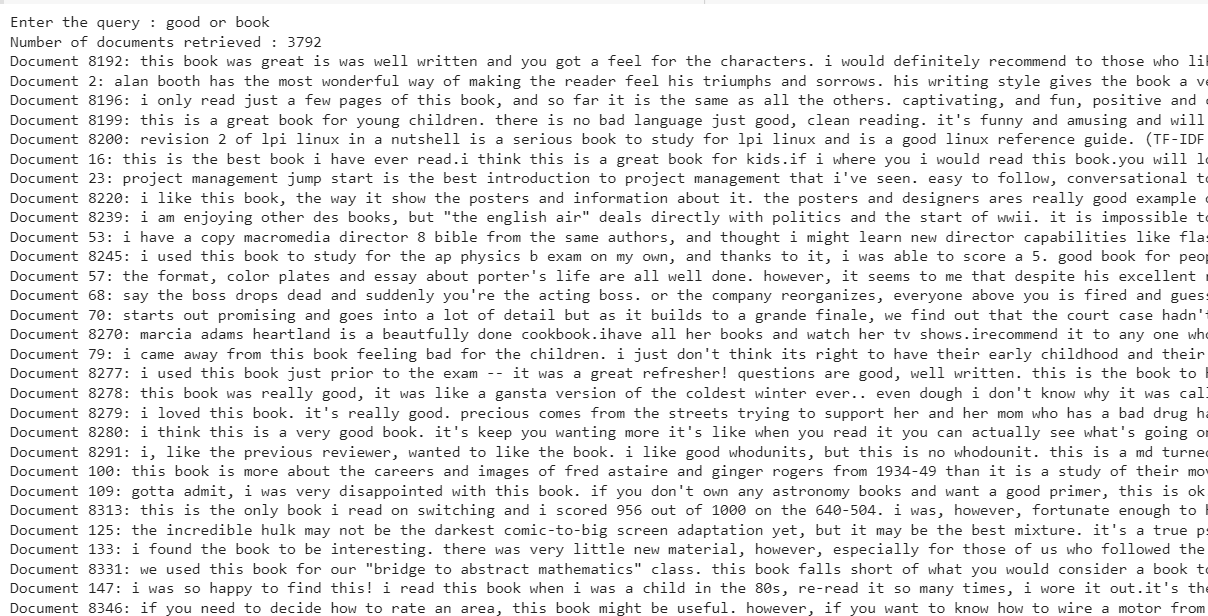


In the above code, the system takes the input from the user and performs the operation specified in the query. It returns the resultant documents and if the queried word does’nt exist, it raises an error.

Output :

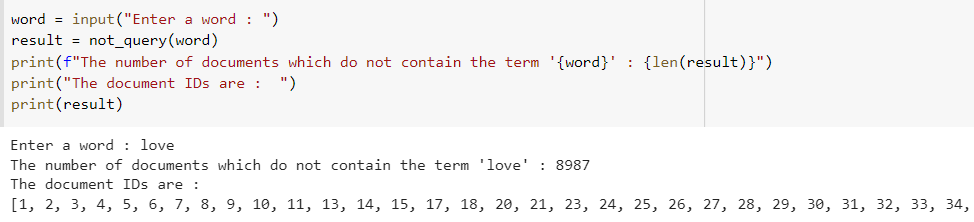


In the above query, 513 reviews are retrieved which consist of both the words “good” and “book”.

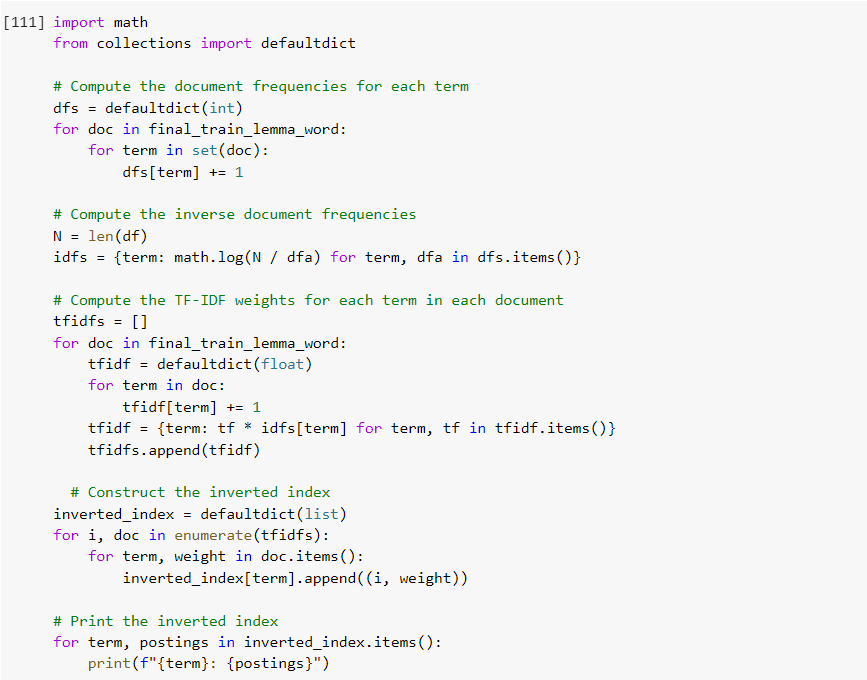


Here is a query with “OR” operation for the same words. We can see that 3792 reviews have been retrieved.

Below is an example for a “NOT” query.



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

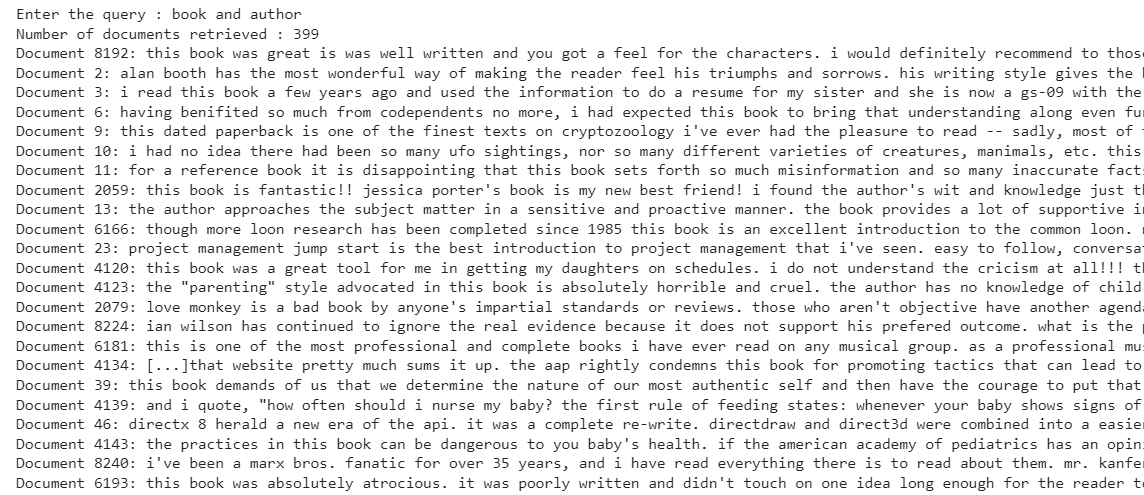
Code for creation of inverted-index : 

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.

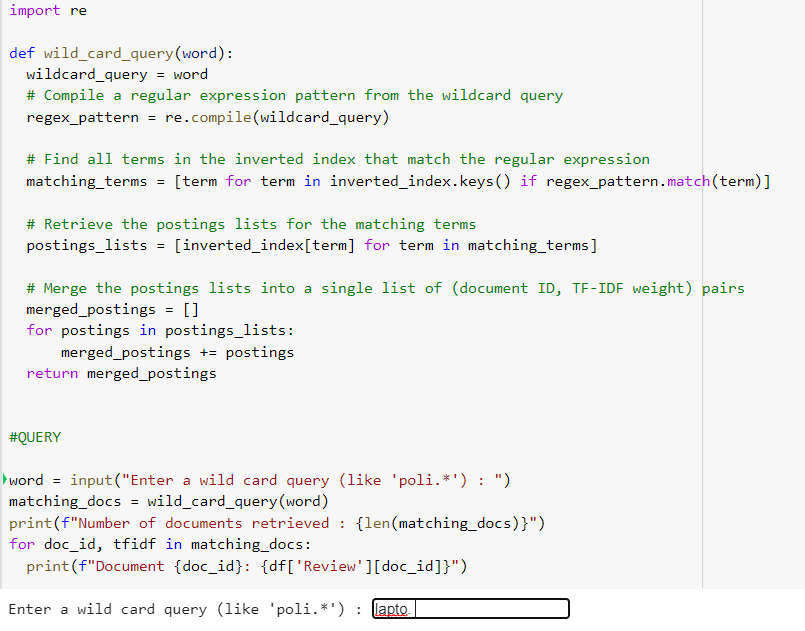
Result of the query using inverted index :



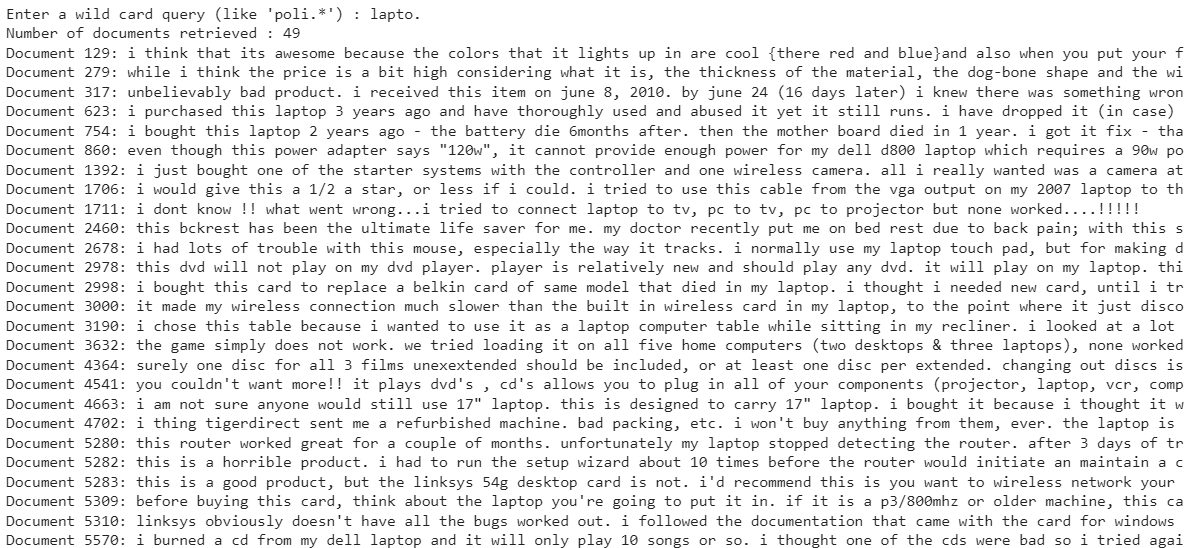
The retrieved documents are ranked based on the tf-idf scores. In this way, the top documents are more relevant than the lower ones.

**SECTION 3.3 : RESULT OF WILD-CARD QUERIES**

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



Output:



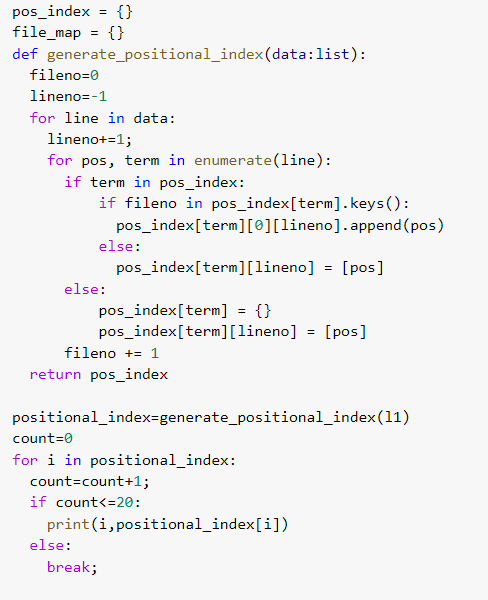
Wildcards are special characters that can stand in for unknown characters in a text value and are handy for locating multiple items with similar, but not identical data.

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

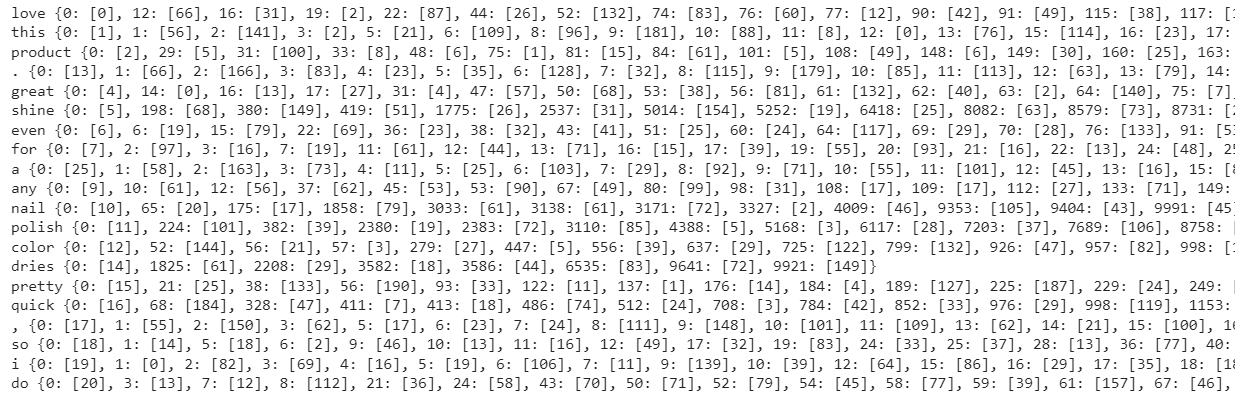
**SECTION 3.4 : RESULT OF PHRASE QUERIES**

Phrase queries are performed with the help of positional index.

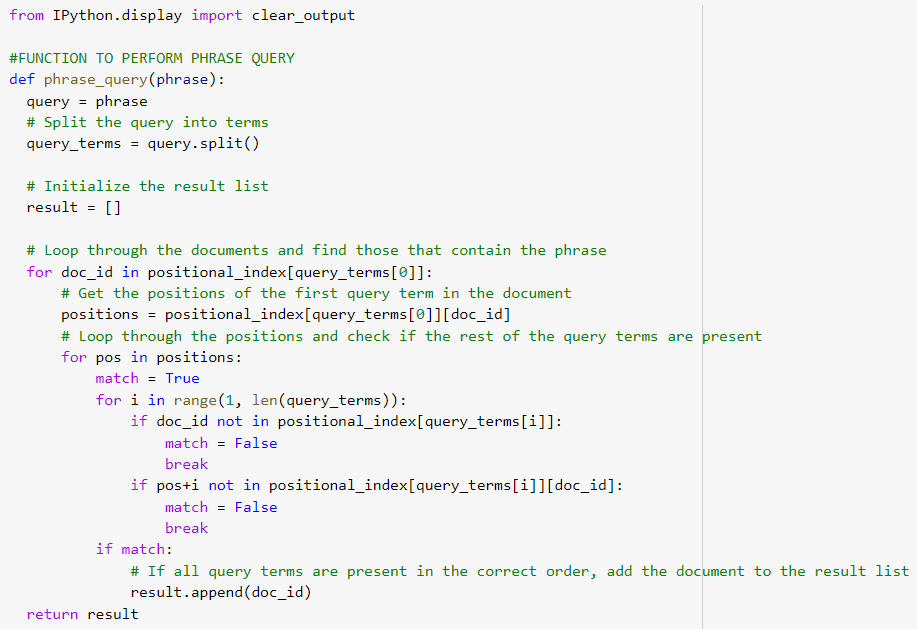
Code for creation of positional index :



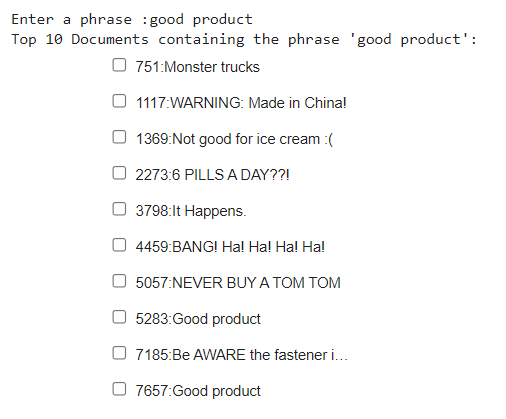
Positional index :



Code for performing phrase queries :



Results:

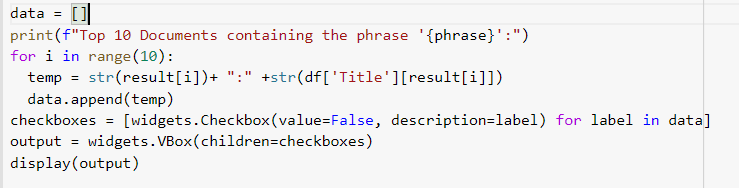


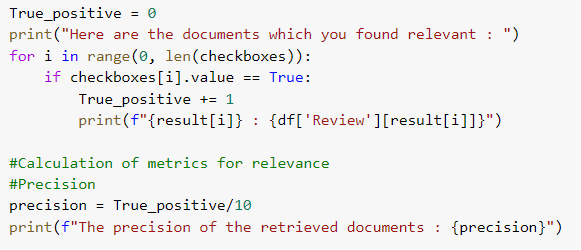
The top 10 documents perceived as relevant by the system are retrieved according to the phrase query. A relevance feedback mechanism from the user is also implemented here which will be discussed in the next section.

**SECTION 3.5 : RELEVANCE FEEDBACK**

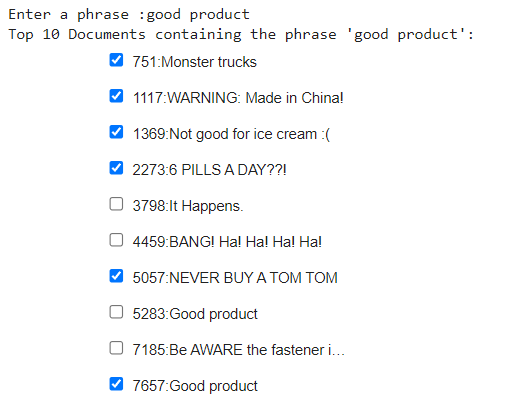
The system fetches top 10 document titles based on the user’s query and asks the user for feedback.The user gives the feedback in the form of checks in the checkboxes as you can see from the screenshot. After the user checks, the system retrieves only document contents of those documents which are marked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

Code:

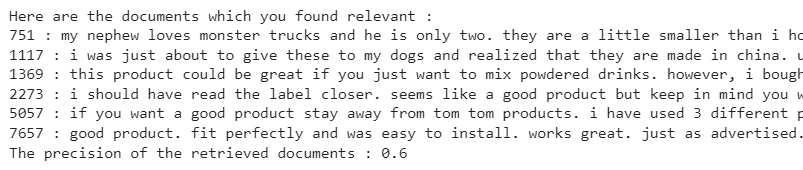




Result:



The user has checked 6 documents as relevant.



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