**IST 664**

**Natural Language Processing**

**Homework 1**

Dhwani Rekhang Gandhi

[dgandhi@syr.edu](mailto:dgandhi@syr.edu)

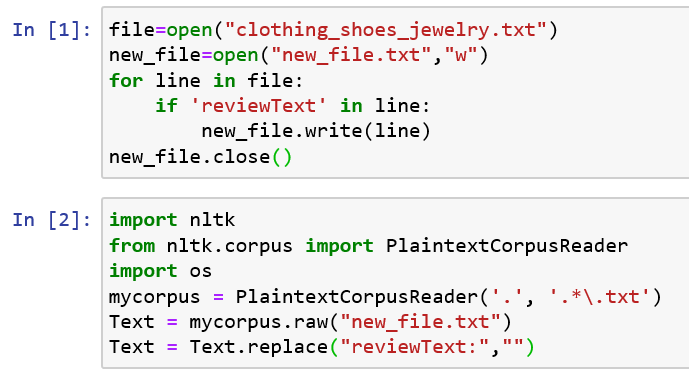
Table of Contents

1. Data Pre-Processing

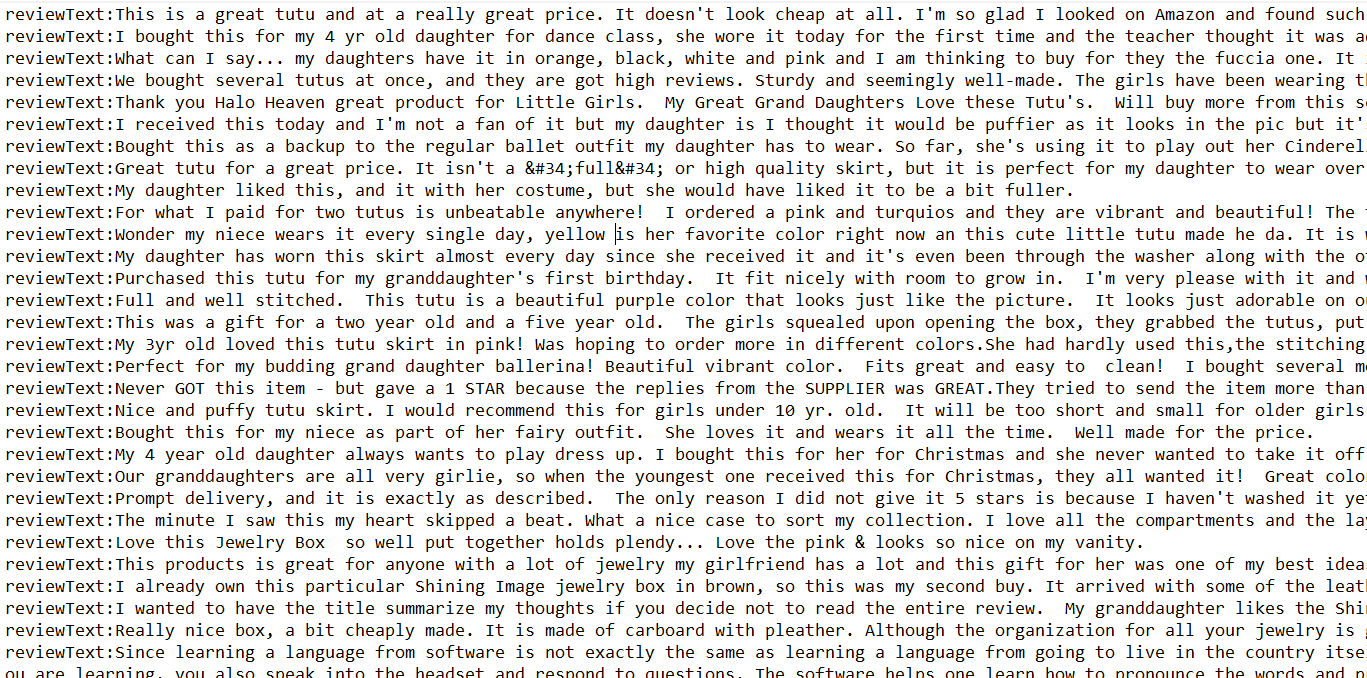
1. Analysis
   1. Task 1
   2. Task 2
   3. Task 3
2. Interpretation of Results
3. Additional Analysis
4. **Data Pre-Processing**

We were given a dataset which consisted of product 142.8 million reviews and metadata from Amazon. We have used only the 5-core subsets of the category “Clothing, Shoes and Jewellery.” It was converted into a text file that included reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

From the above text file, we extracted only the reviewText part and documented that into a new file. It was extracted as given below:

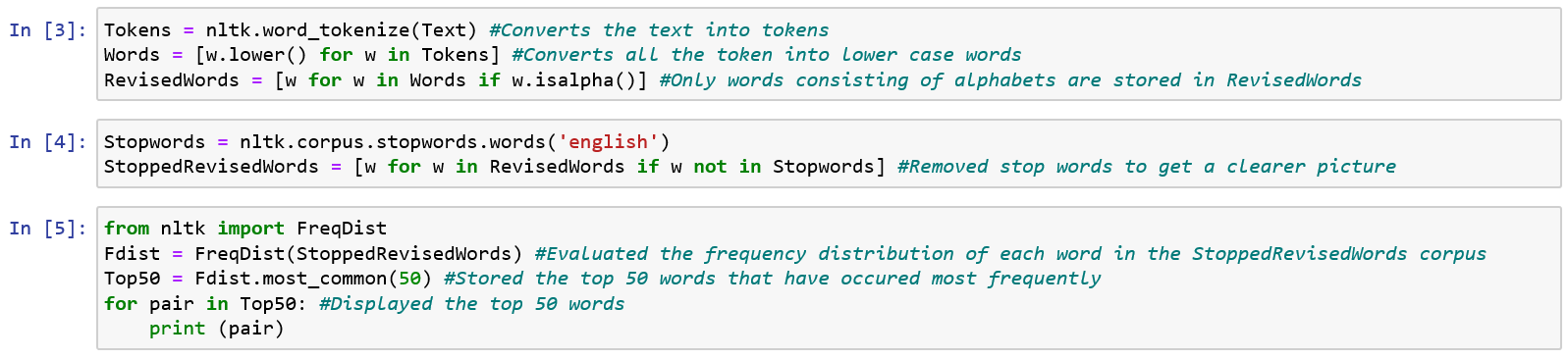


In the first block I have first opened that original file to read and then in the second line I have opened a new empty file which is in the write mode. The for loop is then used to extract all the lines consisting of the word reviewText and those lines are then written down in the new\_file. The new\_file.txt appears as below before it is converted into a corpus and the word reviewText is removed:



In the second block I have used the PlaintextCorpusReader to access the new\_file.txt text file. This file is now treated as any regular corpus. The new\_file\_txt has the word reviewText before each review as shown above so I have used the replace function to replace all the “reviewText:” with no value/character.

1. **Analysis**
   1. **Task 1: Top 50 words by frequency**



Firstly, I tokenized the entire text, after which I converted all the words into lower-case. I did not want the upper-case words and the lower-case words to be counted differently. I wanted the totality of each word, so I converted all of them into lower-case. I also filtered all the special characters and the stop words since those are mainly for giving a sense to the text grammatically. They do not hold much importance when we want a count of words to understand the text briefly. I did this to get a general idea on which words are most frequently used in the entire text. I realized the result as below:

('like', 90212)

('size', 85325)

('fit', 80248)

('wear', 79775)

('great', 78392)

('would', 67419)

('good', 60382)

('love', 60148)

('well', 58211)

('comfortable', 57492)

('one', 56475)

('shoes', 50444)

('really', 46108)

('nice', 46101)

('little', 42750)

('look', 42677)

('bought', 42611)

('get', 38875)

('price', 34828)

('color', 33823)

('pair', 33641)

('quality', 33293)

('small', 32928)

('time', 32872)

('ordered', 32734)

('perfect', 31904)

('shoe', 31523)

('also', 30148)

('much', 29976)

('buy', 29284)

('got', 29009)

('made', 28195)

('looks', 27625)

('watch', 26975)

('even', 26081)

('bit', 25904)

('fits', 25884)

('long', 25654)

('feet', 25123)

('still', 23872)

('recommend', 23301)

('back', 22956)

('big', 22913)

('work', 22482)

('wearing', 22323)

('could', 21558)

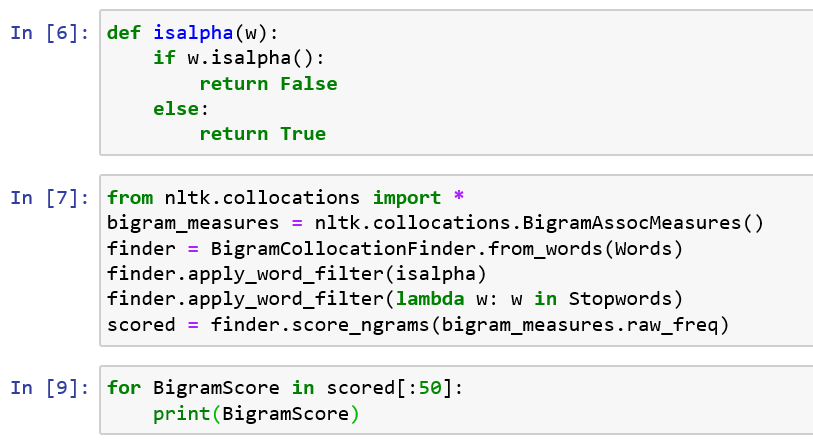
('pretty', 21510)

('looking', 21485)

('cute', 21380)

('material', 20985)

* 1. **Task 2: Top 50 bigrams by frequency**



In the first block, a function isalpha is defined which is used as a special character filter for the bigrams. In the second block the NLTK package is used to create a finder consisting of all the bigrams in the original lower case, tokenized words. I did not want to differentiate between the upper case and the lower-case bigrams and hence used the previous lower-case words. Using a similar approach as before, the bigrams where if either of them is a special character or a stop word is removed. Only those bigrams consisting of lower-case alphabets are used for the analysis. After the filtering is done a raw frequency score is calculated for each bigram present and are then stored. The top 50 bigrams with the highest score are then displayed. These are those bigrams that have appeared together the greatest number of times. The results achieved are as below:

(('well', 'made'), 0.0004659809938034675)

(('would', 'recommend'), 0.0003272013946439088)

(('good', 'quality'), 0.0003203846989221607)

(('highly', 'recommend'), 0.00028099356410259346)

(('really', 'like'), 0.0002634574842687835)

(('fit', 'perfectly'), 0.00023234004883668143)

(('fit', 'well'), 0.0002272925718518756)

(('look', 'like'), 0.0002227654533191116)

(('looks', 'great'), 0.00021282660688511245)

(('another', 'pair'), 0.00020044207572651667)

(('look', 'great'), 0.00019601902888416104)

(('looks', 'like'), 0.00018826568794873762)

(('feel', 'like'), 0.00018228156575025647)

(('year', 'old'), 0.00018176120729821463)

(('great', 'price'), 0.0001750485832668749)

(('even', 'though'), 0.0001697929629012523)

(('fit', 'great'), 0.00016609841789175524)

(('usually', 'wear'), 0.00016177944273980797)

(('light', 'weight'), 0.00015324556412632181)

(('one', 'size'), 0.0001490306606647829)

(('normally', 'wear'), 0.00014845826636753688)

(('long', 'time'), 0.00014616868917855278)

(('would', 'buy'), 0.0001457003665717151)

(('fits', 'well'), 0.00014424336290599799)

(('every', 'day'), 0.0001399243877540507)

(('arch', 'support'), 0.00013883163500476284)

(('size', 'larger'), 0.00013534523337608252)

(('look', 'good'), 0.00013456469569801975)

(('little', 'bit'), 0.00012941314702280554)

(('first', 'time'), 0.00012806021504749675)

(('half', 'size'), 0.00012639506800096286)

(('fits', 'perfectly'), 0.00012623896046535033)

(('really', 'nice'), 0.00012587470954892102)

(('much', 'better'), 0.0001243136341927955)

(('great', 'quality'), 0.00011895394213676456)

(('looks', 'good'), 0.00011817340445870181)

(('high', 'quality'), 0.00011796526107788507)

(('perfect', 'fit'), 0.00011603993480533026)

(('different', 'colors'), 0.00011343814254512106)

(('long', 'enough'), 0.00011182503134379135)

(('fits', 'great'), 0.00011052413521368677)

(('would', 'definitely'), 0.00010901509570276543)

(('flip', 'flops'), 0.00010823455802470267)

(('definitely', 'recommend'), 0.00010110564723172946)

(('second', 'pair'), 0.00010084546800570854)

(('really', 'cute'), 9.959660772080812e-05)

(('super', 'cute'), 9.652649285376127e-05)

(('size', 'smaller'), 9.605817024692361e-05)

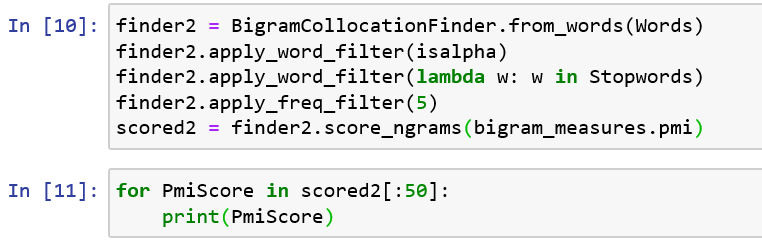
(('make', 'sure'), 9.600613440171942e-05)

(('good', 'price'), 9.569391933049433e-05)

**Reason for using this approach:**

I have not used the lemmatization method since it cuts of many words unnecessarily. Even though similar meaning words like look great and looks great and fit well and fits well are displayed as different bigrams, it still gives me a better picture as to how the reviews are really. In the lemmatization method not only the ‘s’ for looks and fits were removed but several other words were altered which were not necessary for the analyzation of the text.

* 1. **Task 3: Top 50 bigrams by their mutual information scores (using minimum frequency 5)**



The same approach is used for calculating the mutual information score like calculating the raw frequency. The lower-case tokenized words are used to create a finder consisting of bigrams using the NLTK package. Two filters are applied to remove the stop words and the special character. In addition to these two filters another filter is applied which defines that the min frequency of the bigram in the text should be 5. All these bigrams are then stored and displayed. The top 50 bigrams by their mutual information score are displayed. The score displayed is the measure of mutual dependence of the two words on each other. Once the top 50 bigrams are printed the results obtained are as below:

(('badgley', 'mischka'), 21.873990889369754)

(('salvatore', 'exte'), 21.873990889369754)

(('spatestruck', 'lenders'), 21.873990889369754)

(('tessuto', 'vela'), 21.873990889369754)

(('krav', 'maga'), 21.610956483535958)

(('pepto', 'bismol'), 21.610956483535958)

(('herman', 'munster'), 21.38856406219951)

(('hypo', 'allergenic'), 21.38856406219951)

(('birko', 'flor'), 21.195918984257112)

(('myia', 'passiello'), 21.195918984257112)

(('norman', 'reedus'), 21.195918984257112)

(('hola', 'gente'), 21.166171640863062)

(('saudi', 'arabia'), 21.166171640863062)

(('charlotte', 'russe'), 20.873990889369754)

(('giorgio', 'brutini'), 20.87399088936975)

(('grady', 'harp'), 20.736487365619816)

(('sherpani', 'soleil'), 20.710492157086872)

(('laurel', 'burch'), 20.58448427217477)

(('fecha', 'indicada'), 20.514094944283368)

(('caslynn', 'lizzie'), 20.511420809985044)

(('carolyn', 'pollack'), 20.441031482093646)

(('vince', 'camuto'), 20.441031482093642)

(('buenas', 'tardes'), 20.38856406219951)

(('muk', 'luks'), 20.369948384032163)

(('liz', 'claiborne'), 20.32144986634097)

(('juanita', 'wilson'), 20.289028388648592)

(('strawberry', 'shortcake'), 20.195918984257116)

(('hanky', 'panky'), 20.195918984257112)

(('yak', 'trax'), 20.177997076259853)

(('bon', 'bebe'), 20.17355117122866)

(('audrey', 'hepburn'), 20.147555962695716)

(('muay', 'thai'), 20.0259939828148)

(('darth', 'vader'), 20.010052438945777)

(('nether', 'regions'), 19.999521771453608)

(('hallux', 'limitus'), 19.973526562920668)

(('alt', 'alt'), 19.964964549417243)

(('gloria', 'vanderbilt'), 19.947991470813527)

(('pom', 'poms'), 19.947991470813527)

(('aurora', 'borealis'), 19.87399088936975)

(('puerto', 'rico'), 19.87399088936975)

(('tai', 'chi'), 19.87399088936975)

(('ros', 'hommerson'), 19.803601561478356)

(('buzz', 'lightyear'), 19.803601561478352)

(('viet', 'nam'), 19.791528729177777)

(('tuckable', 'lengthprior'), 19.73648736561982)

(('prima', 'donna'), 19.688124344058416)

(('haute', 'couture'), 19.663423903430093)

(('koh', 'koh'), 19.610956483535958)

(('onitsuka', 'tigers'), 19.610956483535958)

(('libby', 'sue'), 19.558489063641822)

1. **Interpretation of results**

From Task1, words such as like, love, great, love, well, and comfortable are few words that occupy the top 10 most used words in the text. They suggest that the major amount of reviews are all positive about the products sold by amazon. We can also infer that people might have bought more shoes or might have written more reviews about shoes since the word ‘shoes’ and ‘shoe’ is mentioned in almost the top 30 words. After this product, the product ‘watch’ is displayed the maximum number of times suggesting that it might be the next highest reviewed product. In terms of product features that the customer gives importance to is most importantly the size, then the comfort, then the price and color interchangeably and then the delivery time the product takes. This is also an inference that can be made from the frequency of these words occurring.

From Task2, the first 10 bigrams are sufficiently enough evidence that suggests that the product reviews are very optimistic. Most of the customers have recommended the product majorly due to the quality of the product and the perfect size of the product. Many reviews also state the price of the product is great. All these inferences are made looking at the top 50 bigrams.

In Task3, the bigrams are displayed along with the mutual information score. It tells us the extent of how the words are dependent on each other. It tells us the amount of information the two words share. This task tells us about such top 50 bigrams used in the reviews.

1. **Additional Analysis**

For further analysis we can even create trigrams that can give a much better insight than the unigrams or the bigrams. For better visualization we can even create word clouds. These word clouds are not only appealing to the eye, but it gives us a general picture about the text within seconds.