Springboard
Capstone Project 2

Building Recommender System Milestone Report

Outline

- Problem
- Data Sources
- Initial Data Exploration
- Data Wrangling
- Data Storytelling with Exploratory Data Analysis
- Creating Content-based Recommender System

Problem

Out of roughly 35000 decisions made in a day, let's keep personal decisions personal. Deciding on what to say to your boss tonight is not what is at stake here. Deciding how to raise your child is your decision to make.

We are asking the questions like, what are we eating for lunch tomorrow? Questions like, which movie should I watch next? Which type of sound speaker should a rock band guitarist purchase. And if the buyer was a construction worker?

It is very easy to go to a bookstore's 'horror' section and pick up the book next to the reader's favorite horror book. But the only connection between the first book and the second is that they are both classified as 'horror' and there is only a slim chance a reader will like all horror books . The probability that a reader will like the second horror book becomes pretty low.

So what are the predictors to keep track of? What sort of information is critical when an individual makes a purchase decision? Our goal is to gather these data and come up with a personalized list of items that a certain individual may have interests in.

Data Sources

All data to be used in this project has been provided by researchers in University of California San Diego. The dataset contains reviews and product metadata available on Amazon store. There are 29 total categories ranging from automotives to video games. The review data has been collected since 1994 to 2014. Original scope of the project was to analyze all category reviews to build the recommender system but due to limitation in resources,

Data Details

Product dataset format

- asin - ID of the product, e.g. 0000031852
- title - name of the product
- feature - bullet-point format features of the product
- description - description of the product
- price - price in US dollars (at time of crawl)
- imUrl - url of the product image
- related - related products (also bought, also viewed, bought together, buy after viewing)
- salesRank - sales rank information
- brand - brand name
- categories - list of categories the product belongs to
- tech1 - the first technical detail table of the product
- tech2 - the second technical detail table of the product
- similar - similar product table

Review dataset format

- reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- asin - ID of the product, e.g. 0000013714
- reviewerName - name of the reviewer
- vote - helpful votes of the review
- style - a dictionary of the product metadata, e.g., "Format" is "Hardcover"
- reviewText - text of the review
- overall - rating of the product
- summary - summary of the review
- unixReviewTime - time of the review (unix time)
- reviewTime - time of the review (raw)
- image - images that users post after they have received the product

Initial Data Exploration

Before starting the data preprocessing stage, it is important to understand exactly the types and various characteristics regarding the data. As an example, a single category was chosen for this section to learn about data. Category chosen : AMAZON_FASHION

In this section, each columns of the dataset will be analyzed and determined

Review Dataset

Contains the review data written by reviewers for specific products

Column ‘overall’

- Column 'overall' is a numeric rating from 1.0 to 5.0 given by the reviewer.
- This is one of the essential features to be used and it is important that this value is containing non-null variables.

Needs to be maintained.

Column ‘verified’

- Column 'verified' is a boolean field labeled by Amazon to verify that the reviewer has purchased this product at regular retail price and from Amazon shops.
- This field could give additional credit/weight to the review content and to the reviewer.

Initially, this column was selected to maintain if possible but as the data size became an issue, was decided to be removed.

Column ‘verified’

- Column 'reviewTime' contains the datetime when the reviewer recorded the review.
- As the products are generally unchanging, unless there is a need to analyze the data in time series, this field may not be useful.

To be removed.

Column ‘reviewerID’

- Column 'reviewerID' consists of the ID of different reviewers who wrote their review.
- It may assist when combined with column verified to identify users that provide consistently valid reviews and during collaborative filtering using user history.

To be maintained if possible.

Column ‘asin’

- Column 'asin' contains the Amazon Standard Identification Number (ASIN) that is assigned by Amazon and its partners used for product identification within their product catalog.
- This is a 10 alphanumeric unique identifiers and for books, asin is the ISBN number.
- This field is crucial in identifying the products in which the review is written for.

Needs to be maintained.

Column ‘reviewerName’

- Column 'reviewerName' contains the name of the reviewer who wrote the review.
- This field is of string data type and can have any name that the reviewer has set to be identified by.

To be removed.

Column ‘reviewText’

- Column 'reviewText' contains the actual text review in string from the reviewer.
- This data will be useful during sentiment analysis of the review and identifying keywords that describe the product.
- Data from this feature can assist in giving credit / weight to the overall ratings of the product.

Needs to be maintained.

Column ‘summary’

- Column 'summary' contains summary of the reviewer's review in string format.
- This feature is very similar to the column reviewText above but is a concise version of it and can be thought of as a supplement to reviewText and overall rating.
- This data will be useful during sentiment analysis of the review and identifying keywords that describe the product.

This field was decided to be combined with the column ‘reviewText’ to build a corpus with fewer documents but with each document having full review data of a review.

Column ‘unixReviewTime’

- Column 'unixReviewTime' contains the Unix timestamp of when the review was written.
- As the products are generally unchanging, unless there is a need to analyze the data in time series, this field may not be useful.

To be removed.

Column ‘vote’

- Column 'vote' contains the numeric value of votes in the datatype string that the review has received by other users.

To be removed.

Column ‘style’

- Column 'style' contains data regarding the specific style of the product purchased by the reviewer.
- Style data is stored in a dictionary format.
- However, approximately 65% of style data are null making this field not fit to be used.

To be removed.

Column ‘image’

- Column 'image' contains links to image files uploaded by the reviewers to add further details to written reviews.
- As we are not planning on analyzing the images in this project, this field is not needed.
- Moreover, there is less than 5% availability for the image field making it unfit to be used.

To be removed.

Product Dataset

Contains the product metadata available in Amazon store

Column ‘title’

- Column 'title' contains the full product title that is shown within Amazon shops.
- This field may not be needed during analysis or modelling but is needed in post-modelling output data formatting. This is because we will mainly be working with the asin to identify the products. However in the final output, we would need to provide to the users the actual title of the asin product as well.

Needs to be maintained.

Column ‘image’

- Similar to column 'image' in review dataset, this column contains links to images that showcase the product.
- Unlike review dataset's image field, this field contains the official images uploaded by the seller.
- As we are not planning on analyzing the images in this project, this field is not needed.

To be removed.

Column ‘brand’

- Column 'brand' contains the company brand that the product is marked by.
- Similar to the title column, this field is also not needed during analysis and modelling.
- However, this field will come in handy in the final output formatting by giving additional information about the product.

Needs to be maintained.

Column ‘rank’

- Column 'rank' contains information regarding the rank of the product within different categories.
- This field may be useful in determining the popular products and serve as a supplement to the overall rating of the product.
- If this field can be wrangled into some feature that we can use, but the rank categories do not seem to follow the traditional category format like how the dataset is categorized.
- If it is the case that these rank categories are difficult to manage, we will focus on using other features for our model.

Initially, this column was selected to maintain if possible but as the data size became an issue, was decided to be removed.

Column ‘feature’

- Column 'feature' contains information regarding product dimension and weight.
- As we are not analyzing any dimension or weight data in this project, this field is not needed.

To be removed.

Column ‘date’

- Unlike the expectation, this field does not seem to contain the date information but various text fields that cannot be categorized into a theme.
- Possibly, this field contains the secondary information that did not make it into brand names or product names.

To be removed.

Column ‘asin’

- Column 'asin' contains the Amazon Standard Identification Number (ASIN) that is assigned by Amazon and its partners used for product identification within their product catalog.
- This is a 10 alphanumeric unique identifiers and for books, asin is the ISBN number.
- This field is crucial in identifying the products in which the review is written for.

Needs to be maintained.

Column ‘description’

- Column 'description' contains string in lists data information regarding description of the products.
- This field contains too many null values for this to be used in this project.

To be removed.

Column ‘price’

- Column 'price' contains the price information of the product.
- This field has about 10% data availability which makes it not fit to be used. Too many missing variables to be used.

To be removed.

Column ‘also_view’

- Column 'also_view' contains a list of different ASINs that is associated with the record.
- These linked products would be the products that are closely related or searched together by users.
- If there is enough data to be used after preprocessing, this field can be used.

To be maintained if possible.

Column ‘also_buy’

- Column 'also_buy', similar to 'also_view' above, contains a list of different ASINs that is associated with the record.
- These linked products would be the products that are closely related and bought together by users.
- If there is enough data to be used after preprocessing, this field can be used.

To be maintained if possible.

Column ‘fit’

- Column 'fit' contains information regarding products' variation and fits.
- The values in the field seem to be html/css language attempting to visualize the product.

To be removed.

Column ‘details’

- Column 'details' contains additional information regarding the products.
- Very similar to the column 'fit' above, this field contains html scripts that show various details like different colors, links to product details, reviews etc.

To be removed.

Column ‘similar_item’

- Column 'similar_item' holds information regarding items that are similar to the product mentioned in the record.
- This field is similar to the also_view and also_buy but this field is focused on showing the similar items rather than things that are usually bought together.
- If there is enough data to be used after preprocessing, this field can be used.

To be maintained if possible.

Column ‘tech1’

- Column 'tech1' consists of additional technical details about the products.
- This field can include dimensions, weight, brand names etc.

To be removed.

Column ‘tech2’

- Column 'tech1' consists of additional technical details about the products.
- This field can include dimensions, weight, brand names etc.

To be removed.

Column ‘main_cat’

- Some product datasets also contain a field called main_cat which contains information about categories the media is in.
- This field will be useful when it comes to identifying and grouping different products together.

To be maintained if possible.

Column ‘category’

- Similar to above, we also have a category field available for some media related products where it provides details as to which group media belongs.
- This field will be useful when it comes to identifying and grouping different products together.
- Since we already have the main category field defined and this field provides additional data which we don't need, this field can be considered redundant.

To be maintained if possible.

Data Wrangling

After successful initial data exploration, data cleaning and preprocessing began. However, the sheer amount of data was too large to be handled all together. Many attempts were made to reduce the data size in a way that does not take away too much of the required data but was ultimately unsuccessful.

It was decided afterwards to focus on one category : Books.

**Failed attempts are kept under Jupyter Notebook titled 'ARCHIVED_<NOTEBOOK_NAME>"

Even bringing in the whole review dataset regarding Books was troublesome and the master review dataset text file had to be splitted to 14 files of 1GB sizes and was imported and cleaned in chunks.

It was imperative to reduce noisy records and trim down the data size to reasonable work.

1, Low Review Count Trimming

All products with less than 50 reviews were removed so the dataset is left with products that have been reviewed at least 50 times.

2, Column 'reviewText' and 'summary' Combining

Because the two columns already are very similar in context, there is no need to keep two different columns of text. These columns were merged to single column called 'review'

3, Drop Unnecessary Columns

As decided in the Initial Data Exploration, columns ['price','reviewerName','helpfulness','timestamp'] were removed. Other columns may be removed in a later section after preprocessing depending on data availability.

4, Null Check

Any records with NaN or empty string were reviewed and dealt with.

Data Storytelling with EDA

After starting initial check on data and cleaning the large dataset in chunks, it is time to combine them together for further data cleaning and exploratory data analysis.

Data Cleaning

14 chunks of review data was concatenated into a single dataframe.

In further efforts of trimming down the dataset to essentials and key data points, it was decided to also have a minimum review written per reviewer of 30. Meaning the only reviews kept will be the ones written by reviewers who have written at least 30 or more reviews.

1, Drop Duplicates

As the dataset was divided into 14 chunks, there was no point in performing the duplicate check at that time. Now that we have a single dataframe to work with, a duplicate check was performed.

2, Combine Multiple Reviews Written by Same Reviewer

Drop duplicates above take care of any identical reviews made. However, it cannot distinguish the different reviews written by the same reviewer multiple times as the text column would have different values.

All reviews for a product written by the same reviewer were combined to one record each.

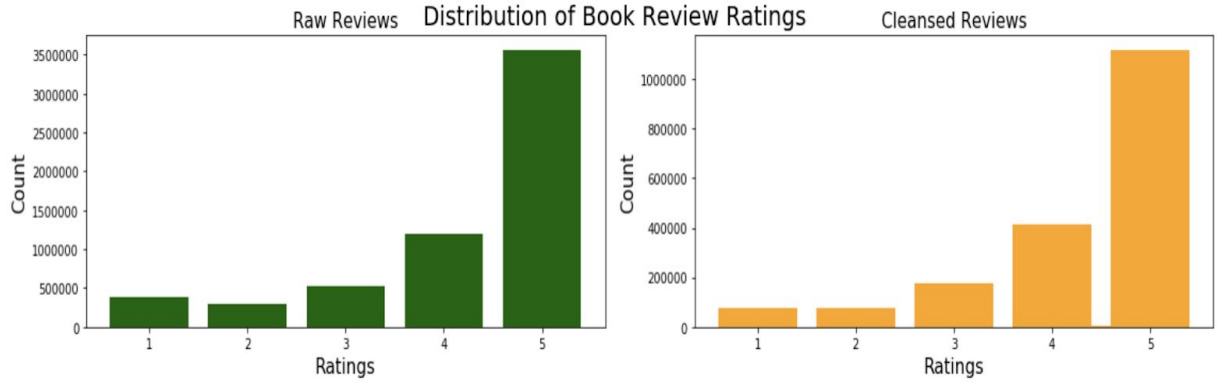
3, Updated Minimum Reviews Trimming

Now that all duplicates of different forms were taken care of, the dataset was trimmed down to remove any reviews written by reviewers with less than 30 reviews.

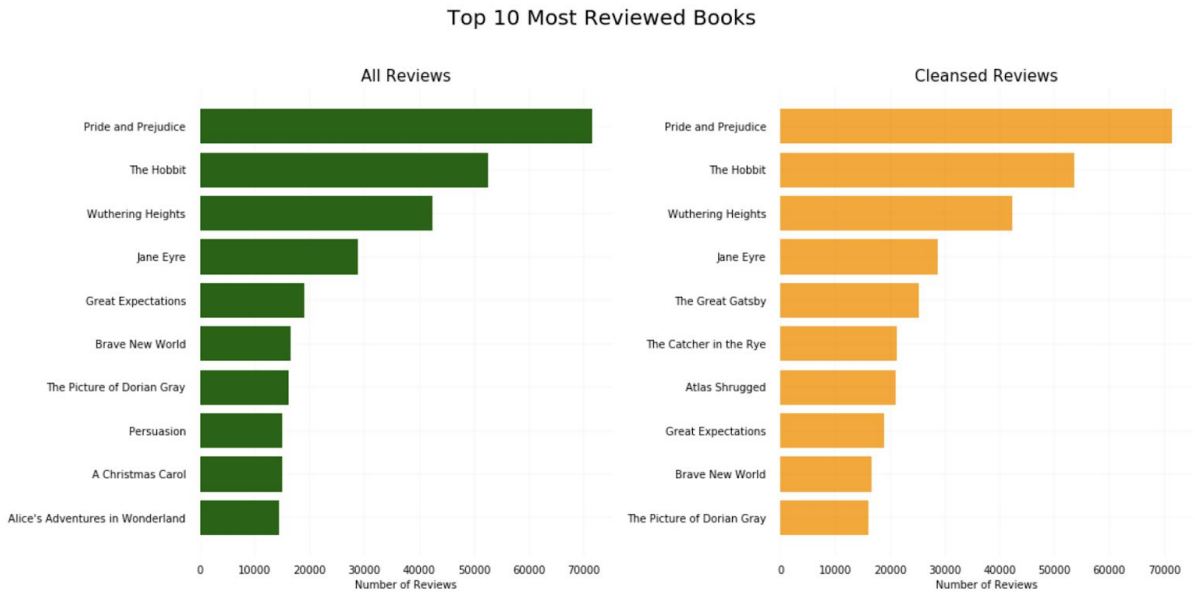
Data Visualization

At this point, the review dataset is free of any data records that are deemed unnecessary or irrelevant. This trimmed down dataset was compared with the original review dataset to confirm the new dataset still holds true for the initial data characteristics found.

Understanding the Review Data

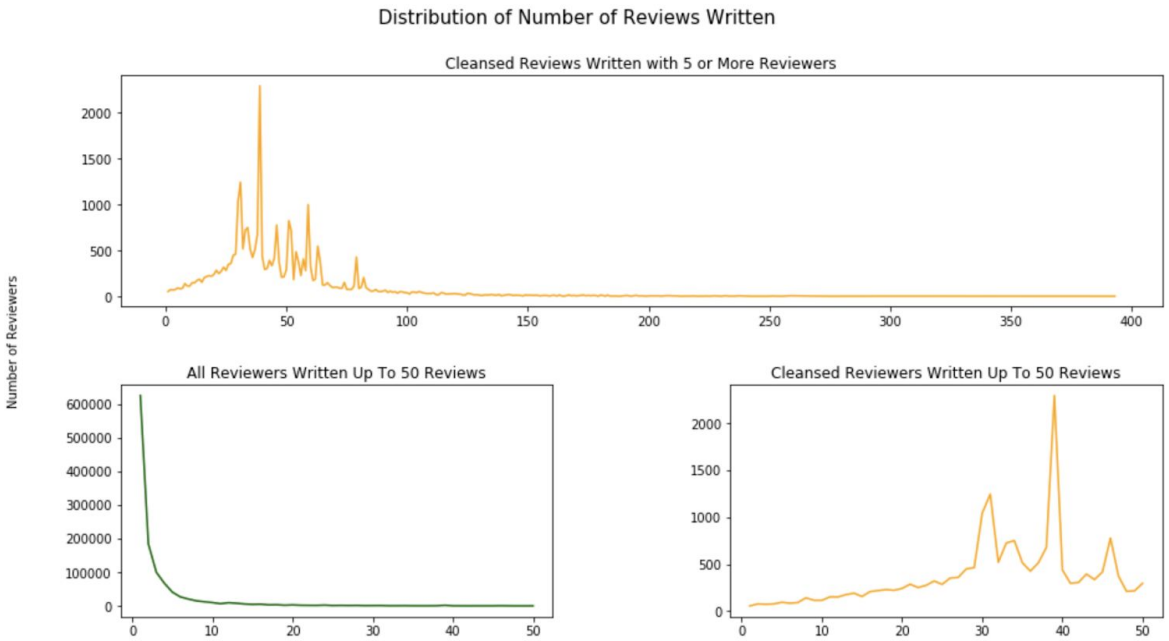


The distribution of overall ratings is heavily skewed. Sheer number of 5-star reviews looks to be higher than the sum of all remaining four star ratings. Also, the before and after trimming seemed to carry over the overall distribution.



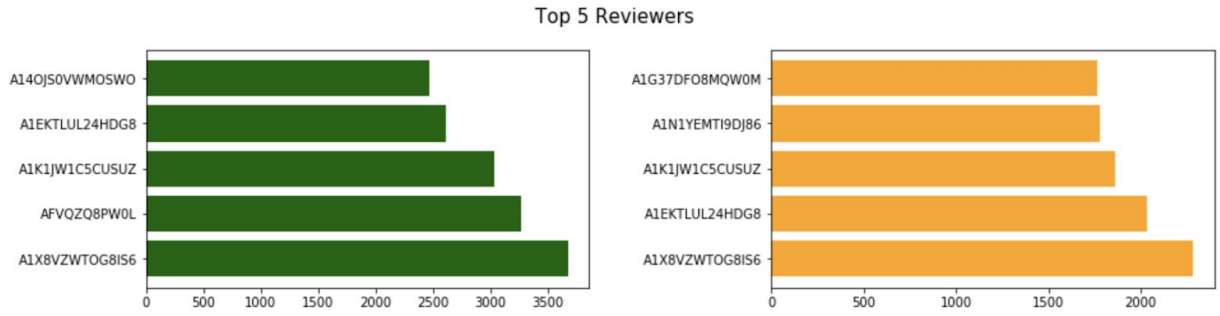
Interestingly, even with all the data trimming, the top 10 books with most reviews have not changed a lot as well as the number of actual reviews.

This means our cleansed dataset is maintaining the data characteristics from the total dataset.



Top most plot shows the distribution of the number of reviews written after low count data trimming. Although we set the low count to 30, we can see there are still reviewers with less than 50 reviews as noted in the plot below. This is due to our additional ASIN trimming after trimming the number of reviewers down. This decision was made as it is acceptable to have reviewers write less than 30 reviews but each product must have at least 50 reviews.

Second-row's left plot shows the distribution for all reviewers written up to 50 reviews. Important thing to note from this plot is how many reviews have been written by users who have never reviewed or only reviewed a few products as the steeply decreasing tail represents. The left-tail eventually trickles down to 1 (at max, 1 reviewer wrote 3680 reviews).



Final row shows the top 5 reviewers from both all reviewers and cleansed reviewers. We can see that for the most part the ranking did not change meaning our cleansed data set holds a good representation of population data.

Creating Content-based Recommender System

Review Dataset

Text Preprocessing

Prior to building the model, dataset needed to be process for the following:

- Further trim down dataset
- Aggregate text data to a single column
- Data does not contain punctuation
- Data does not contain null words
- Tokenize each words
- Lemmatize words to their root word

Custom methods were created to deal with the above preprocessing.

```
#clean up review
df_re['review'] = df_re['review'].apply(lambda x: cleanText(x))

#build stopwords
eng_stopwords = set(stopwords.words('english') + ['book','books','author','authors'])


#tokenize and remove stopwords
df_re['review'] = df_re['review'].apply(lambda x: removeStopwords(x, eng_stopwords))

#lemmatize each work token to root word given we know the pos_tag of the word
df_re['review_lemmatized'] = df_re['review'].apply(lambda x: lemmatize_words(x))
```

Vectorizer Parameter Optimization

This lemmatized review text was then used to build the count vectorizer to review the vocabulary selected.

```
#instantiate CountVectorizer with default settings: no lower casing, preprocessing or tokenizing.
vectorizer = CountVectorizer(lowercase=False,preprocessor=dummy, tokenizer=dummy)

#tokenize and build vocab fitting the corpora then tranform to make count vector
count_vector = vectorizer.fit_transform(corpora)
```

After testing through multiple iterations of vectorizers, following hyperparameters were chosen.

- Min_df : 5
    - We have at least 30 reviews per product, 5 was chosen to reduce any words counts that are one-off.
- Max_df : 0.5
    - Since we are dealing with people's opinions instead of the product descriptions, words that appear in more than half of the corpora are considered unneeded.
- Max_features : 50000

Throughout this process, few issues were noticed.

- tokens that start with _

```
df_re.loc[df_re.review.str.contains('_good')].head()
```

asin	reviewerID	overall	review
b000mooajg	ak81wlvd5kgux	2.0	['atlas', 'shrugged', 'undoubtedly', 'interest...
b000mwc3fq	ak81wlvd5kgux	2.0	['quot', 'positive', 'uphold', 'defend', 'atta...
b000pgi7qi	ak81wlvd5kgux	2.0	['quot', 'need', 'destroy', 'objectivism', 're...
b000pwmt1g	ak81wlvd5kgux	2.0	['quot', 'need', 'destroy', 'objectivism', 're...
b0000cki8j	ak81wlvd5kgux	2.0	['quot', 'need', 'destroy', 'objectivism', 're...

- tokens with numeric variables

```
x = df_re.loc[df_re.review.str.contains('1660')]
print('number of reviews: ', x.shape[0])
x.head()
```

```
ber of reviews:  145
```

	asin	reviewerID	overall	review
45	b00007k45c	a1cqjzsi3lk1vs	4.5	['enjoyable', 'eloquent', 'thoroughly', 'enjoy...
48	b0006iu7c2	a1cqjzsi3lk1vs	4.5	['enjoyable', 'eloquent', 'thoroughly', 'enjoy...
41	0006513204	a1cqjzsi3lk1vs	4.5	['enjoyable', 'eloquent', 'thoroughly', 'enjoy...
73	1565115430	a319kyeiaz3son	5.0	['vivid', 'simple', 'poetic', 'girl', 'hyacint...
68	1417627530	a39abkrs1mkftw	5.0	['picked', 'year', 'wonders', 'geraldine', 'br...

- tokens that start with numbers then string

```
df_re.loc[df_re.review.str.contains('_weakness_')].head()
```

	asin	reviewerID	overall	review
517	b000nowyr0	a2b0xo8btprx7r	5.0	['love', 'loss', 'sacrifice', 'gain', 'tender'...
771	b000kaavli	a2b0xo8btprx7r	5.0	['love', 'loss', 'sacrifice', 'gain', 'tender'...
066	b0006amffm	a2b0xo8btprx7r	5.0	['love', 'loss', 'sacrifice', 'gain', 'tender'...
250	0808514601	a2b0xo8btprx7r	5.0	['love', 'loss', 'sacrifice', 'gain', 'tender'...
465	b000pen42c	a2b0xo8btprx7r	5.0	['love', 'loss', 'sacrifice', 'gain', 'tender'...

- tokens that includes noise

Some of the invalid tokens were saved as unique words that will interfere with our final tf idf vector.

SPAM Filtering

In the prior data cleaning, we have checked the duplicates for the entire rows but further processing revealed there were reviewers making the same reviews to different books passing the null check.

To account for this, the original dataset was imported again and cleaned.

```
#find spam index
spam = findSpam(df_re.review_lemmatized)
```

```
#make it into index list
spam_ind = list(set(spam))
```

```
#remove spam
df_re = df_re.loc[~df_re.index.isin(spam_ind)]
```

Then was put through the updated cleaning process to account for the above issues mentioned and filtered for SPAM.

Cosine Similarity

After parameters for the vectorizer were chosen and all tokens cleaned, we build the cosine similarity matrix for the input data with our tf-idf vector.

Count vectorizer shape : (578449, 50000)

```
#instantiate tfidf transformer
transformer = TfidfTransformer(smooth_idf=True,use_idf=True)
tfidf_vector = transformer.fit_transform(count_vector)
```

Simple code was written to create the input search text as a vectorizer on the fly and make comparison. This method was chosen due to our data size. We could create the cosine similarity matrix for all possible documents however we lacked the firepower to do so. Therefore, we have kept all the components in memory and made recommendations on the fly.

```
'''
Takes in book title to search, tfidf of corpus and number of recommendations

Outputs recommendations
'''

def recommend(book, titles, df, cv, tt, tf, num_rec=5):

    book_text = cleanText(book)
    book_text = removeStopwords(book_text)
    book_text = lemmatize_words(book_text)

    book_vector = cv.transform([book_text])
    book_tfidf = tt.transform(book_vector)

    cosine_matrix = linear_kernel(book_tfidf,tf).flatten()
    related_docs_indices = cosine_matrix.argsort()[:-num_rec:-1]


    print('recommending for ',book,'\n')

    print('recommendations: ')

    for i in related_docs_indices:

        print(titles.loc[titles.asin == df_re.asin.iloc[i]].title)
```

Review based recommendation test looks like this:

```
recommending for  HTML Complete Reference

recommendations:
0    HTML: The Complete Reference
Name: title, dtype: object
0    HTML: The Complete Reference
Name: title, dtype: object
0    HTML: The Complete Reference
Name: title, dtype: object
4833    HTML Goodies
Name: title, dtype: object
0    HTML: The Complete Reference
Name: title, dtype: object
0    HTML: The Complete Reference
Name: title, dtype: object
0    HTML: The Complete Reference
Name: title, dtype: object
0    HTML: The Complete Reference
Name: title, dtype: object
25377    HTML 4 for Dummies, Fourth Edition
Name: title, dtype: object
```

```
recommending for  Lord of the Rings: Two Towers

recommendations:
22646    The Dark Tide: Book One of the Iron Tower Trilogy
Name: title, dtype: object
35736    The Lord of the Rings Weapons and Warfare
Name: title, dtype: object
23926    The Iron Tower Omnibus (Mithgar)
Name: title, dtype: object
35790    Dark Tower Boxed Set: v. 1-1v
Name: title, dtype: object
23926    The Iron Tower Omnibus (Mithgar)
Name: title, dtype: object
23926    The Iron Tower Omnibus (Mithgar)
Name: title, dtype: object
13907    The Dark Tower, Books 1-3: The Gunslinger, The...
Name: title, dtype: object
31607    The Fifth Ring
Name: title, dtype: object
23926    The Iron Tower Omnibus (Mithgar)
Name: title, dtype: object
```

```
recommending for  statistics application

recommendations:
12273    The Lady Tasting Tea: How Statistics Revolutio...
Name: title, dtype: object
35574    Statistics For Dummies (For Dummies (Math & Sc...
Name: title, dtype: object
33939    Statistics for the Utterly Confused (Schaum's ...
Name: title, dtype: object
35574    Statistics For Dummies (For Dummies (Math & Sc...
Name: title, dtype: object
35574    Statistics For Dummies (For Dummies (Math & Sc...
Name: title, dtype: object
12273    The Lady Tasting Tea: How Statistics Revolutio...
Name: title, dtype: object
33939    Statistics for the Utterly Confused (Schaum's ...
Name: title, dtype: object
35574    Statistics For Dummies (For Dummies (Math & Sc...
Name: title, dtype: object
17863    History: Fiction or Science?
Name: title, dtype: object
```

Our review-based recommender is providing the same recommendation multiple times. This is due to multiple reviews being available for each book. We could further expand our recommender to skip the asin if they were selected to be recommended in the future. However, we are seeing the limitation of review content based recommender systems as the keywords are determined by the user reviews. Although our current recommender system is doing a good job in recommending similar (mostly the same for now) books, there were many manual data trimming involved to keep the corpora manageable. This means that the corpora itself was built by the decision of the developer to keep certain reviews or not introducing bias. This may not work best in production situations.

Content based filtering is usually applied for situations where we analyze text for titles, descriptions or even transcripts, but not usually a someone's opinion as there are not many variations to describe someone's emotion towards a product as opposed to the product's direct descriptions.

As part of building a recommender system for this project, we will take two steps back at this time to create a basic content based recommender system with 1) title of the books, 2) descriptions and genre of the book.