Model Building Notes:

1. Tools Used: Python programming using Anaconda IDE
2. Python Packages: Os, Pandas, Sklearn, Numpy, Matlplotlib, Seaborn, NLTK, Statsmodels.api, Pylab, SpaCy, Gensim, PyLDAvis
3. Key assumption is to reflect the overall sentiment using Information extraction on reviews dictionaries. We are not attempting at single product aspect entity relations (Single product feature extraction, given the time constraint and the requirement to build a large pre-trained data set)
4. Models not attempted as part of solutions, considering them not required as part of our problem case: Sentiment Classifier, Ratings based analysis - Classifying products based on ratings, calculating mean rating scores of products

Data Modelling Approach -Assumptions and comments mentioned in Ipython NoteBook

Data-set source link: https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products

*Problem Statement: Understanding insights from Customer Reviews - identify key product terms in the sentiments (Topic Modelling) - do sentiment analysis (classify sentiments - for future use cases) - understand product wise sentiments (Data Modeller Understanding of the problem)*

Different types of NLP analysis which can be employed for our use: Given that we are dealing with Reviews, (Textual Data - Applications of NLP Techniques), following sub analysis -

*1. Entity Recognition (Named Entity Recognition), 2. Identifying Emerging Trends 3.Sentiment Analysis 4. Text Summarization 5. Topic Modelling*

*and solutions deduced can be applied as a universal attempt for exploring the data.*

We will restrict ourselves to Sentiment Analysis & Topic Modelling and explore advanced enhancements at later when we deploy real time sentiment monitors - probably by streaming data analysis using Databricks/ Apache Spark

Observational Notes from Model - 1 :

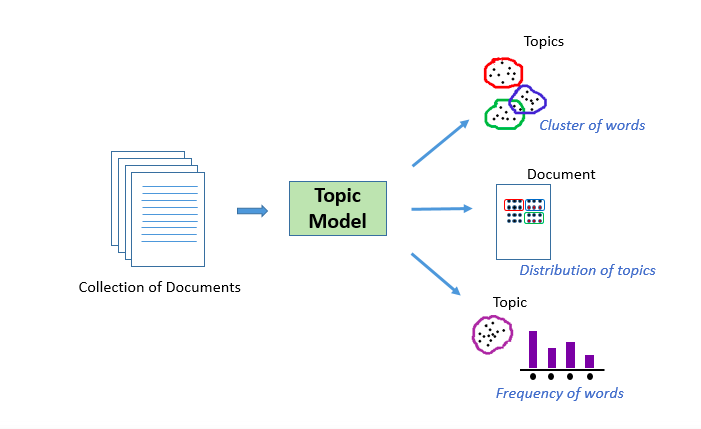
Exploratory Data Analysis:

* Data Matrix Vector Length: 21 Columns -Features, 34660 Rows - Sentiments for initial data set
* Identified 42 Unique products using id column as unique identifier for initial data set
* Frequencies for ratings are skewed for positive values at 4, 5 - evident from distribution plot - From these the data set is class imbalance in nature .
* Given the nature of the problem, we need to employ solution measures for class imbalance - sampling equal distribution of ratings by populating data or under/over sampling data with equal frequencies of ratings
* As Class Imbalance problems persist in the initial sample, we employ other two parts of the data as well and take the necessary steps, despite that, the nature of the dataset remains the same.

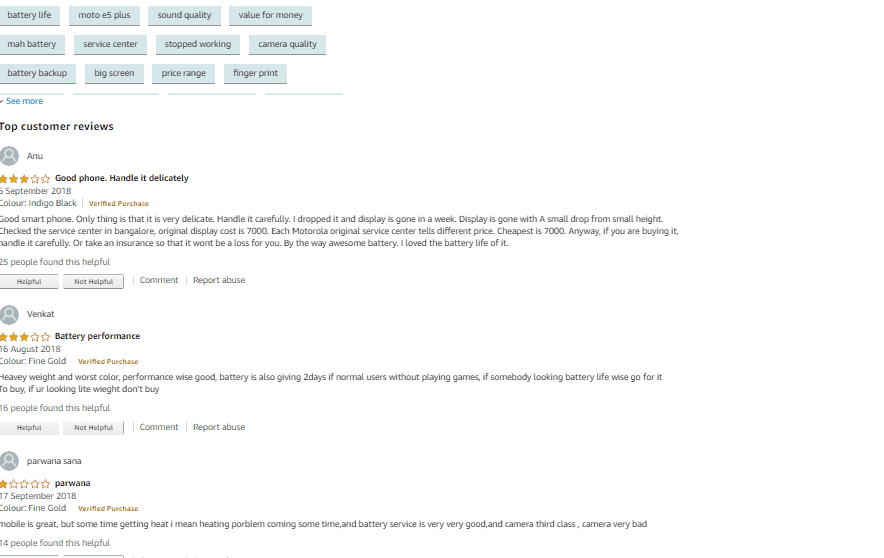
Problem statement Scope and Topic Modelling:

By the nature of unstructured Textual data, *we want to derive insights about sentiments of consumers in terms of product features - authenticity of the product, quality of the product and from consumer standpoint, our support engine should be able to redflag fake reviews*. Topic Modelling is a process to automatically identify topics present in a text object and to derive hidden patterns exhibited by a text corpus, thus assisting in better decision making.

Topic Modelling :



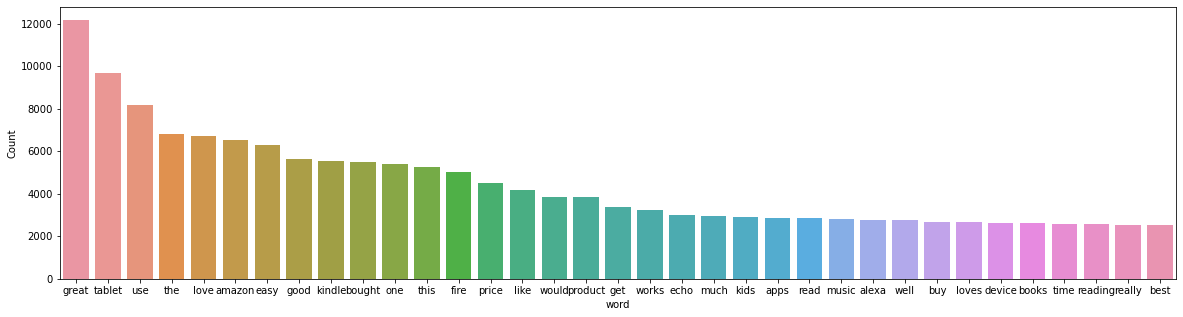
Intended OutCome:



Data Preprocessing Step:

1. Standard NLP flow : Case Correction - Tokenization - Stemming - Lemmatization and Stop Word Removal

In frequently used words among sentiments, following words appear more, before pre-processing:

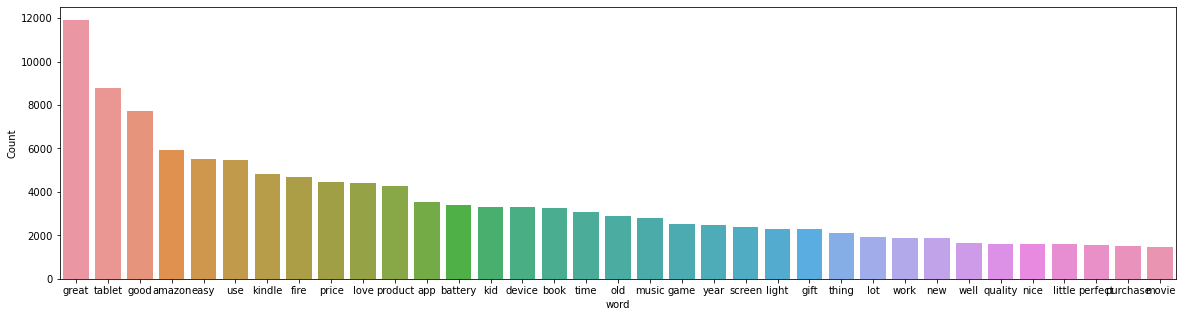


Modelling words using Spacy package - Models: <https://spacy.io/models>

Spacy Modelling Notes:

1. Applying fast sentence segmentation without dependency parses and employing the senter component instead:
2. Lemmatization (using defined function for Nouns and Adj in POS tagging) is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization does morphological analysis of the words.
3. Tokenizing involves splitting sentences and words from the body of the text.

After Pre-processing:, frequently appearing tokens - impactful words -



* 8245 unique tokens identified as part of Dictionary Corpus
* Our next attempt is to build a model identifying important words and topics
* A document-term matrix is a mathematical matrix that describes the frequency of terms that occur in a collection of documents.(in our case : frequency of 8245 tokens in reviews)

Topic Modelling using Latent Dirichlet Allocation(LDA):

High level overview of LDA:

There are many approaches for obtaining topics from a text such as – Term Frequency and Inverse Document Frequency. NonNegative Matrix Factorization techniques. Latent Dirichlet Allocation is the most popular topic modeling technique. In our problem setting, LDA assumes reviews are produced from a mixture of topics. Those topics then generate words based on their probability distribution. Given a dataset of reviews, LDA backtracks and tries to figure out what topics would create those reviews in the first place.LDA is a matrix factorization technique. In vector space, any corpus (collection of sentiments via reviews) can be represented as a document-term matrix.

Parameters of LDA: (In our Case)

Alpha and Beta Hyperparameters – alpha represents review-topic density and Beta represents topic-word density. Higher the value of alpha, reviews are composed of more topics - (in our case consumer sentiment about products & product features) and lower the value of alpha, sentiments contain fewer topics. On the other hand, higher the beta, topics are composed of a large number of words in the corpus, and with the lower value of beta, they are composed of few words. Number of Topics – Number of topics to be extracted from the corpus. Researchers have developed approaches to obtain an optimal number of topics by using Kullback Leibler Divergence Score. Number of Topic Terms – Number of terms composed in a single topic. It is generally decided according to the requirement. If the problem statement talks about extracting themes or concepts, it is recommended to choose a higher number, if the problem statement talks about extracting features or terms, a low number is recommended.Number of Iterations / passes – Maximum number of iterations allowed to LDA algorithm for convergence.

LDA Model Output:

[(0,

'0.055\*"good" + 0.043\*"great" + 0.024\*"tablet" + 0.024\*"quality" + 0.021\*"price" + 0.020\*"amazon" + 0.020\*"speaker" + 0.019\*"voice" + 0.018\*"sound" + 0.017\*"streaming"'),

(1,

'0.116\*"battery" + 0.057\*"time" + 0.032\*"box" + 0.029\*"last" + 0.020\*"week" + 0.019\*"long" + 0.017\*"life" + 0.017\*"month" + 0.015\*"issue" + 0.015\*"day"'),

(2,

'0.055\*"app" + 0.035\*"movie" + 0.032\*"game" + 0.030\*"great" + 0.025\*"kid" + 0.022\*"many" + 0.022\*"love" + 0.022\*"tablet" + 0.021\*"thing" + 0.018\*"fun"'),

(3,

'0.067\*"amazon" + 0.036\*"music" + 0.033\*"device" + 0.027\*"prime" + 0.025\*"alexa" + 0.024\*"product" + 0.024\*"home" + 0.024\*"remote" + 0.016\*"question" + 0.016\*"smart"'),

(4,

'0.133\*"fire" + 0.064\*"kindle" + 0.048\*"love" + 0.027\*"screen" + 0.025\*"light" + 0.024\*"amazon" + 0.023\*"new" + 0.022\*"money" + 0.021\*"book" + 0.015\*"worth"'),

(5,

'0.130\*"use" + 0.115\*"easy" + 0.106\*"great" + 0.091\*"good" + 0.087\*"product" + 0.028\*"price" + 0.028\*"tablet" + 0.023\*"gift" + 0.019\*"purchase" + 0.016\*"happy"'),

(6,

'0.078\*"year" + 0.076\*"cable" + 0.076\*"old" + 0.030\*"room" + 0.029\*"roku" + 0.028\*"daughter" + 0.027\*"item" + 0.027\*"setup" + 0.023\*"love" + 0.023\*"perfect"')]

*The above LDA model is built with 7 different topics where each topic is a combination of keywords and each keyword contributes a certain weightage to the topic. We can see the keywords for each topic and the weightage (importance) of each keyword.*

[(0,

'0.077\*"voice" + 0.061\*"echo" + 0.054\*"nice" + 0.052\*"small" + 0.040\*"size" + 0.033\*"list" + 0.030\*"tap" + 0.030\*"light" + 0.030\*"news" + 0.027\*"great"'),

(1,

'0.169\*"kindle" + 0.084\*"light" + 0.083\*"book" + 0.043\*"new" + 0.032\*"first" + 0.027\*"time" + 0.027\*"read" + 0.023\*"button" + 0.020\*"turn" + 0.019\*"husband"'),

(2,

'0.152\*"app" + 0.107\*"amazon" + 0.057\*"many" + 0.046\*"store" + 0.036\*"apple" + 0.033\*"free" + 0.029\*"available" + 0.023\*"device" + 0.021\*"download" + 0.021\*"name"'),

(3,

'0.141\*"thing" + 0.094\*"alexa" + 0.079\*"item" + 0.061\*"smart" + 0.040\*"way" + 0.037\*"one" + 0.037\*"music" + 0.036\*"home" + 0.028\*"show" + 0.028\*"device"'),

(4,

'0.083\*"screen" + 0.075\*"fast" + 0.056\*"awesome" + 0.039\*"reader" + 0.038\*"ipad" + 0.037\*"good" + 0.036\*"need" + 0.035\*"internet" + 0.033\*"love" + 0.032\*"connection"'),

(5,

'0.238\*"product" + 0.141\*"great" + 0.092\*"movie" + 0.054\*"music" + 0.050\*"good" + 0.048\*"purchase" + 0.048\*"love" + 0.041\*"weather" + 0.038\*"game" + 0.029\*"friend"'),

(6,

'0.279\*"battery" + 0.080\*"time" + 0.057\*"remote" + 0.047\*"week" + 0.047\*"long" + 0.041\*"life" + 0.032\*"charge" + 0.032\*"month" + 0.029\*"hour" + 0.026\*"user"'),

(7,

'0.312\*"use" + 0.277\*"easy" + 0.154\*"love" + 0.032\*"set" + 0.026\*"phone" + 0.017\*"much" + 0.016\*"convenient" + 0.016\*"fun" + 0.012\*"computer" + 0.011\*"mom"'),

(8,

'0.150\*"good" + 0.102\*"tablet" + 0.098\*"price" + 0.066\*"great" + 0.050\*"money" + 0.031\*"brand" + 0.029\*"worth" + 0.027\*"buy" + 0.021\*"cheap" + 0.019\*"sale"'),

(9,

'0.128\*"fire" + 0.070\*"amazon" + 0.045\*"box" + 0.043\*"cable" + 0.041\*"last" + 0.026\*"day" + 0.025\*"stick" + 0.024\*"well" + 0.019\*"new" + 0.018\*"device"'),

(10,

'0.107\*"kid" + 0.056\*"great" + 0.056\*"prime" + 0.048\*"question" + 0.042\*"control" + 0.042\*"amazon" + 0.041\*"lot" + 0.034\*"fun" + 0.030\*"music" + 0.029\*"family"'),

(11,

'0.131\*"work" + 0.078\*"device" + 0.052\*"home" + 0.041\*"slow" + 0.038\*"little" + 0.032\*"bit" + 0.030\*"version" + 0.029\*"different" + 0.028\*"stream" + 0.027\*"fine"'),

(12,

'0.060\*"great" + 0.058\*"quality" + 0.054\*"speaker" + 0.049\*"good" + 0.049\*"sound" + 0.047\*"streaming" + 0.031\*"amazon" + 0.028\*"room" + 0.026\*"amazing" + 0.026\*"device"'),

(13,

'0.059\*"unit" + 0.058\*"setup" + 0.049\*"simple" + 0.049\*"wife" + 0.048\*"great" + 0.045\*"good" + 0.042\*"service" + 0.042\*"reading" + 0.038\*"port" + 0.033\*"quick"'),

(14,

'0.147\*"year" + 0.143\*"old" + 0.094\*"gift" + 0.057\*"son" + 0.055\*"tablet" + 0.053\*"daughter" + 0.036\*"perfect" + 0.028\*"great" + 0.026\*"game" + 0.025\*"disappointed"')]

While the initial model is to generate word features, further LDA models are built extending topics. Convergence is subjective, given discrepancies in accepting a standard norm such as Coherence value or using other family of Matrix factorization models. Hence, we reiterate further extending the topic count in model.

Scientifically, this can be iterated until no further development can be made (Limitation of having to use TPU) or probably build other models based out of these. Currently, our model serves the purpose of the problem in our context.

For Coherence value usage, Hyperparameter tuning and further extension of models: <https://towardsdatascience.com/unsupervised-nlp-topic-models-as-a-supervised-learning-input-cf8ee9e5cf28>

Model -2: Aspect Based Sentiment Analysis *also known as Feature Based Sentiment Analysis is a technique to find out various features, attributes, or aspects from a given text and their respective sentiments.*

Model is built on initial data set owing to following constraints:

1. Computing power for large Spacy Models
2. Merging and Processing missing values - excess data columns and time frame for assignment.

The model is being built, assuming that we are extracting the products- product features and descriptors for one of the Amazon Tablets and hence the competitor list for comparisons.

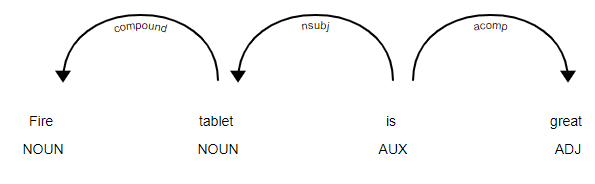
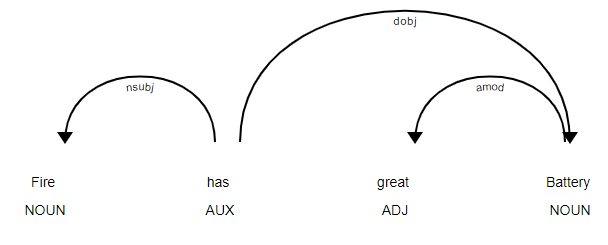
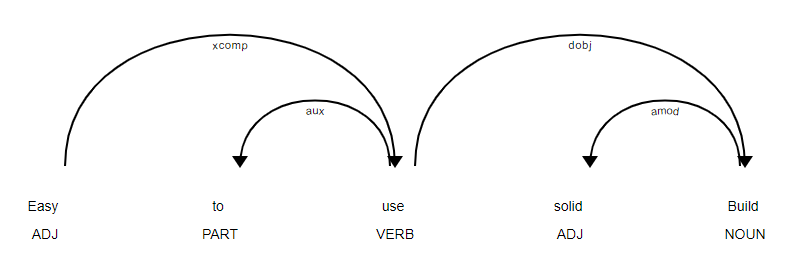
Custom built function for Aspect Based Entity Recognition and feature extraction is drawn from Kaggle notes, Github.

Custom Built function does the following:

1. *Stanford Core NLP Package - Pre-processing*
2. *Modelling the dependency relations between different words using Spacy package*

Following are the modifiers used in the function for language modelling :

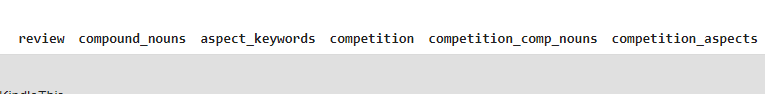
1. An adjectival modifier of a Noun is any adjectival phrase that serves to modify the meaning of the Noun
2. An adverb modifier of a word is a (non-clausal) adverb or adverb-headed phrase that serves to modify the meaning of the word
3. An open clausal complement (xcomp) of a verb or an adjective is a predicative or clausal complement without its own subject
4. Dependency Modelling Visualization examples:

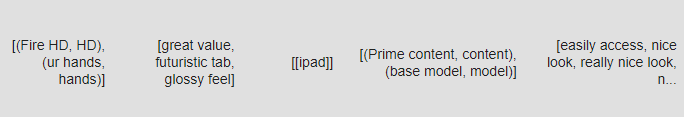


1. Generally from a review standpoint, compound words often do not offer us sentiments, hence the code looks for possible compound word pairs and then checks with the aspect words extracted if it can add more detail to the extracted aspects - ex Outstanding Battery for a Tablet gives \*more context\* than Outstanding Tablet, while the compound word search will identify passenger van as a compound word

We appended Competitor list to achieve better classification and derive comparative implied semantic relations.

*Final appending the terms as information extracted from above modifiers and mapping them as product descriptors, matching the strings of competitors and generating key feature list among the reviews in separate columns as below: (Model output clipped*





1. Analysing sentiment using Vander sentiment analysis and generating polarity strength for classifying reviews as positive, negative and neutral categories.
2. Performance Metrics:

accuracy

0.9111197019751177

f1 score

0.9530475465532927

recall

0.9668148259755914

precision

0.9396668497162731

Additional Coding references - websites index:

Github - Stackexchange - Paperswithcode - Medium and towardsdatascience

* Analytics Vidhya