An Algorithmic Framework to Compute Sentiment Intensity Scores and Enable Information Extraction

Akash Gupta

A project report submitted in
partial fulfilment of the requirement for the award of the

POST GRADUATE DIPLOMA IN MANAGEMENT
IN
RESEARCH AND BUSINESS ANALYTICS

MADRAS SCHOOL OF ECONOMICS
MSE BUSINESS SCHOOL
Chennai-600025
May 2021

Degree: Post Graduate Diploma in Management

Branch: Research and Busines Analytics

Month/Year of Submission: May 2021

Title of Project: An Algorithmic Framework to Compute Sentiment Intensity Scores and Enable Information Extraction

Name of Student: Akash Gupta

Roll Number: 2019DMB02

Supervisor: Dr. Sowmya Dhanaraj

Bonafide Certificate

This is to certify that the project report titled, **An Algorithmic Framework to Compute Sentiment Intensity Scores and Enable Information Extraction** is the bonafide work of **Akash Gupta**, a student of the Madras School of Economics, who implemented this analysis under my supervision and guidance. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. Sowmya Dhanaraj
Madras School of Economics
Chennai: 600025

ACKNOWLEDGEMENT

First and foremost, I would like to thank Dr. Sowmya Dhanaraj and Dr. Rakesh Nigam for their continued guidance in my academic learning. The implementation of this analysis became a possibility only due to the efforts of their teaching, particularly with regard to the vital technicals in the field of data science like - Linear Algebra, Calculus, Algorithms and Statistics.

Dr. Srikant's support has also proved invaluable with regard to the programmatic implementation of various algorithms we have developed in this report. His emphasis on programming with an object oriented framework has helped immensely. I am thankful for the guidance of Mr. Guru as well, for providing detailed explanations on various Machine Learning libraries across Python and R frameworks.

My batchmates at the Madras School of Economics (PGDM) have been extremely supportive throughout various phases of this analysis - productive discussions with them indeed brought out interesting ideas and solutions. Finally, I'd like to thank my family for their continued motivation throughout.

- Akash Gupta

Contents

Abstract

We present an analysis on three broad objectives - The first is to present an algorithm or an automated process that performs the task of computing sentiment intensity scores for product reviews and consequently, label the reviews as per their broad sentiment class. In order for such an algorithm to function effectively and be generalizable - we train it on hand labeled datasets. A point to note here is that the scope of this algorithm is within the confines of 'laptop product reviews in the E-commerce markets' - however, with the general training methodology outlined, we suggest that this algorithm could be trained and deployed for other product categories as well. Our key finding here is that this algorithm outperformed the accuracy of sentiment intensity labels derived from VADER computations of intensity scores (accuracy was judged with respect to the gold class labels). Both the algorithms were judged on the same threshold values and the same data as well. It was also seen that our algorithm outperforms VADER on completely new and unseen datasets as well (within the confines of our analysis subjects).
The second is to present a general series of algorithms and steps - that could be used as a procedural template for analyzing unstructured textual data and consequently obtaining insightful information from it. The broad outline is to first use a trained algorithm to compute sentiment intensity scores and label the reviews accordingly - we then create certain variables or metrics that effectively characterize and encode information about each review. Regression analysis is then used to establish relationships between these metrics and the sentiment intensity scores - the next step is to compute the statistical properties of the distribution of sentiment intensities of the reviews and also to conduct statistical tests that establish significance in difference between means and variances of the scores across different product categories - lastly, we deploy various unsupervised learning methods to extract information from similar groups of reviews and characterize them, so as to obtain insights that could potentially augment strategic decision making.
The third is to actually analyze and present the findings of our analysis in the context of the data being effectively used to establish the other two objectives. The data consists of manually parsed reviews from Amazon and Flipkart, with respect to top laptop brands operating in the Indian market. We use the above techniques to essentially arrive at inferences that inform us about the positioning of the brands within the market environment, in relation to their competitors.

Chapter 1

Introduction

1.1 Motivating the Analysis Objective

Since the beginning of last year, significant changes have been noticed in the retail landscape, globally - with more emphasis being given to the online retail space and electronic products like Laptops and Smartphones[1]. This is perhaps the effect of pandemic induced Lockdowns, whereby activities of professional work, learning and entertainment have been limited to the home environment. This is the point where the motivation for this analysis stems from. With the intuition that with people, on a rather large scale, being confined to their homes would develop an increased dependence on electronic gadgets like Laptops - it would then imply a higher demand for such products, which in turn suggests an increasingly competitive market environment for the manufacturers. In such a scenario, it becomes important for the Laptop brands to be able to gather informative insights from consumer data - to address problems arising from consumer satisfaction and consequently, to expand market share. It is our view that customers leave significant traces of information and their sentiment toward a product or service, in the form of these reviews. We present our analysis as a method for such brands to be able to extract information embedded in a large number of customer reviews - so as to develop marketing strategies, supported by analytics. Our work contributes to establishing a deeper understanding of the lexical qualities of product reviews, that characterize brands over various metrics.

1.2 Data Collection

The review data for all four brands under our consideration, has been manually collected from Amazon and Flipkart. A total of 500 reviews were collected for each brand - Lenovo, HP, Mac, Dell - resulting in a cumulative dataset of 2000 customer reviews, across all the brands. Furthermore, the reviews have been manually labeled as **positive** or **negative** - these categorical assignments reflect a manual judgement of the overall sentiment of each review. Such a manual labeling of the data serves many purposes - one of them being, the labeled

[1]Business-Standard (Arnab Dutta): Work from Home drives Laptop Sales amid Covid-19

dataset becomes eligible for classification algorithms to be applied on it, which would in turn help us extract useful features about product reviews in both categories - secondly, since our aim is to develop a general Machine Learning tagging process that extracts information from reviews and attempts to perform a sentiment labeling, the gold standard labels provide us with a benchmark against which we establish the accuracy of the automated labeling process. Prior work on analyzing product reviews has invovled obtaining publicly available datasets [2], while others have straight away adopted a machine learning tagging approach before subjecting review data to classification algorithms [3]. This is where our approach differs slightly, in the sense that we have hand labeled the entire review dataset, so as to firmly establish a numerical degree of effectiveness of our general sentiment labeleing algorithm, as well to establish the correctness of extracted features, pertaining to each category. With this level of validation, our aim is to design a robust sentiment extraction and labeling algorithm that could be deployed over large, unlabeled and unstructured review datasets.

1.3 Texts as Computational Units

After going through the data collection process, what we essentially have - are collections of unstructured data in the form of text. Before we think about developing automated procedures that extract insightful information from such data, we must first transform it into units over which computation could be performed [4]. At this point, we present some principles and concepts from the NLP that help us structure the textual data.

- In our context, one unit of data is a customer review, which is known as a **text** or a **document**. Each text is composed of **sentences**, which are in turn composed of **words** - which are nothing but strings, or a collection of **symbols**.
- In order to computationally utilize this unit of data for the purpose of analysis, we break it down into more fundamental constituent units. Here, we refer to these fundamental constituent units as **tokens** - which are words, or a collection of words.
- Each review then, is thought of as an **array of tokens**. We note that when a review is broken down into this array of tokens such that each token represents one word - then what we have are essentially, **unigrams**.
- If however, we break the text into tokens, such that each token is a **tuple** of a pair of consecutive words - then what we have are essentially, **bigrams**. [5]

[2] Shanshan Yi, Xiaofang Liu: Machine Learning based customer sentiment analysis for recommending shoppers, shops based on customers' review

[3] Xing Fang, Justin Zhan: Sentiment analysis using product review data.

[4] Schubert Lenhart: Computional Linguistics, The Stanford encyclopedia of philosophy

[5] Daniel Jurafsky, James H. Martin: Speech and Language Processing.

- We then compute the **vocabulary** of our datasets. The vocabulary of a dataset of reviews is an array of all distinct tokens contained in all the reviews, taken together.
- For our analysis, we would be converting our dataset of reviews into the above three forms of fundamental computational units - **unigrams**, **bigrams** and **vocabularies**.

Chapter 2

Automated Labeling Algorithm for Product Reviews

We present the set theoretic relations between custom-constructed lexicons that play a vital role in the process of designating sentiment scores for textual data, in particular, product reviews.

2.1 The Backdrop

Our aim is to develop an algorithm - or an automated process - that takes as input, textual data like product reviews and outputs the **sentiment intensity score** of the review and its **sentiment class label**. Note that **sentiment intensity score** is a numerical measure of the intensity of positive/negative sentiment expressed by a customer, in a particular review. Also, **sentiment class label** is a binary class variable that represents whether the particular review is ”positive” in its overall sentiment or ”negative” in its overall sentiment. [1]

2.2 Review Dataset with Gold Class Labels

- In order to train an algorithm to learn features such that it is able to identify the degree of positive or negative sentiment in a review - the idea is to first and foremost, extract important features that are good discriminators of positive and negative sentiments [2]. Naturally - to be able to do us, we require a dataset of reviews that is hand labeled - such that each review in the dataset has an associated gold class label [3] indicating positive/negative sentiment.
- The importance of these gold class labels is that - they provide us with a benchmark against which the efficacy of our automated labeling mechanism can be judged. The other importance of these gold class labels is that - we can fit a classification algorithm on the labeled data - to essentially extract important features or discriminators, which we can then exploit, using our algorithm.

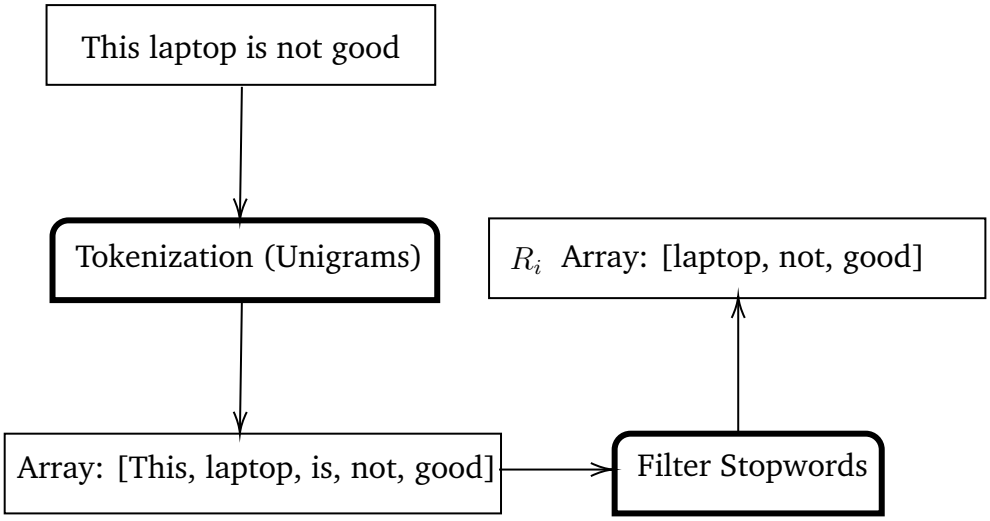
2.2.1 Data Preparation

- In order to extract informative features and to make sure that our initial application of the classification model is free of **noisy features** [4] - we implement a removal of **stopwords** [5]. These words are considered irrelevant for such application since they do not encode much characteristic information about a review or a sentence. Here is a list of custom stopwords that we created for the purpose of this application:

”to”, ”it”, ”with”, ”a”, ”that”
”an”, ”so”, ”in”, ”the”, ”he”, ”for”, ”my”
”if”, ”too”, ”of”, ”such”, ”as”, ”can”, ”is”
”and”, ”i”, ”any”, ”do”, ”for”, ”what”, ”when”
”this”, ”its”, ”at”, ”by”, ”ish”, ”it’s”, ”on”
”are”, ”from”, ”was”, ”has”, ”were”, ”or”, ”we”
”you”, ”g”, ”am”, ”e”, ”your”, ”had”, ”be”, ”me”
”his”, ”r”, ”u”, ”soo”, ”now”, ”very”, ”but”

- What we have done here is to create our own custom-set of stopwords. The reasons for doing this are - the already provided list of stopwords in Python and R [6] libaries tend to remove various informative words and - a simple filtering of stopwords of a certain length (say 2) ends up discarding certain important words that might be of length 2. Hence, we have carefully curated a custom list of stop words.

- Each review in the review dataset is first **tokenized** into **unigrams** - and then the stopwords are filtered out of each review. Through an iterative process - the entire dataset of reviews has been filtered to remove the stopwords. Below is a schematic that shows the basic architecture of this process and the initial preview of the dataset, so prepared.



0	using last days laptop main usage development ...	neg
1	got detective laptop didn't turn worst custome...	neg
2	guys bought dell laptop week ago amazon which ...	pos
3	after using about month good specs performance...	pos
4	using product days no issues all laptop boot t...	pos
...	...	...
1992	not satisfied battery life	neg
1993	decent product price	pos
1994	meeting expectations	pos
1995	value money good speed	pos
1996	good laptop	pos

Figure 2.1: A small snippet of the dataset containing 1997 reviews such that the stopwords are filtered out from the reviews. Associated gold class labels are also visible.

2.2.2 Model Fitting

- After the appropriate data prepation involving filtering out the stopwords - we attempt to fit a **Naive Bayes Classification Model** on the training data [7] (70% of the entire data) and then measure its accuracy on the test data.
- After fitting the classification model - we obtain a reasonably good accuracy of 86.55% on the test data.
- With this framework in place - we can then extract the **most informative features** [8] that tend to discriminate well between the two sentiment classes. The appropriate manner to interpret the figure below is - "if a review contains the word "xyz" then it is associated with an odds of p to 1 of belonging to class Y". Such that p is a real valued number and Y is an indicator variable such that 0 indicates "negative sentiment" and 1 indicates "positive sentiment". Below is a figure that shows a snapshot of such a list of most informative features.
- We note that the above figure is merely a snapshot - we have actually isolated all features that exhibit a likelihood of positive/negative discrimination of at least 2.5 to 1.0. Another crucial point to note is that this list gives out some very general words like - *"seems, put, minutes"* - we essentially ignore these to avoid **overfitting** in our algorithm.

```
Most Informative Features
         contains(worst) = True              0 : 1      =     28.6 : 1.0
         contains(waste) = True              0 : 1      =     25.9 : 1.0
        contains(return) = True              0 : 1      =     16.7 : 1.0
         contains(hangs) = True              0 : 1      =     13.7 : 1.0
     contains(excellent) = True              1 : 0      =     13.4 : 1.0
      contains(horrible) = True              0 : 1      =     12.7 : 1.0
          contains(easy) = True              1 : 0      =     12.5 : 1.0
  contains(disappointed) = True              0 : 1      =     12.1 : 1.0
         contains(seems) = True              0 : 1      =     11.8 : 1.0
          contains(nice) = True              1 : 0      =     10.7 : 1.0
          contains(best) = True              1 : 0      =     10.5 : 1.0
       contains(amazing) = True              1 : 0      =      9.8 : 1.0
          contains(love) = True              1 : 0      =      9.8 : 1.0
       contains(premium) = True              1 : 0      =      9.5 : 1.0
       contains(awesome) = True              1 : 0      =      9.2 : 1.0
     contains(defective) = True              0 : 1      =      9.1 : 1.0
       contains(minutes) = True              0 : 1      =      9.0 : 1.0
      contains(customer) = True              0 : 1      =      8.6 : 1.0
        contains(weight) = True              1 : 0      =      8.3 : 1.0
      contains(pathetic) = True              0 : 1      =      8.2 : 1.0
          contains(poor) = True              0 : 1      =      8.1 : 1.0
     contains(customers) = True              0 : 1      =      8.0 : 1.0
         contains(fault) = True              0 : 1      =      8.0 : 1.0
           contains(put) = True              0 : 1      =      8.0 : 1.0
 contains(replacement) = True              0 : 1      =      8.0 : 1.0
         contains(request) = True              0 : 1      =      8.0 : 1.0
```

Figure 2.2: Example to interpret the first line - (1) contains(worst) $=$ True - this means that if it is TRUE that a review contains the word "worst". (2) $0:1 = 28.6:1.0$ - this means that if point 1 is TRUE, then the likelihood of the review being negative (to positive) is 28.6 to 1.

2.3 Setting the Stage

- In order to develop an algorithm that takes as input textual data like reviews and performs various operations on it - we must first convert the raw textual data into fundamental units of computation. [9]
- Utilizing principles from **computational linguistics** and **set theory** [10] - we decompose each review into two sets - an array of **unigrams** - and an array of **bigram tuples**. [11]
- As explained in prior sections, a unigram is nothing but a string, or a collection of symbols. A typical review represented as a set of unigrams can be denoted as:

$$R_i = \{w_j | \forall w_j = (a_1 a_2 \cdots a_n), a_k \in \Sigma_{english}\} \tag{2.1}$$

$$U = \{R_1, \cdots, R_n\} \implies \textit{Set of all unigram review sets} \tag{2.2}$$

Equation 1.1 simple states that - A unigram set review R_i is nothing but a set of words w_j such that each word is a string of symbols a_k such that each symbol belongs to the **english alphabet**. Equation 1.2 states that the set U represents the set of all reviews $R_1, \cdots, R_n$ that are decomposed into unigram sets.

- In a similar manner, we define a review as a set of bigram tuples as well. A bigram tuple is nothing but a tuple or a pair of consecutive unigrams. The notation of such a review decomposition is shown below:

$$R_i = \{(x_i, y_i) | \forall x_i = (a_1, \cdots, a_n) \ \& \ \forall y_i = (b_1, \cdots, b_n), a_k, b_k \in \Sigma_{english}\} \quad (2.3)$$

$$B = \{R_1, \cdots, R_n\} \implies \textit{set of bigram review sets} \quad (2.4)$$

Equation 1.3 simply states that - A bigram set review R_i is nothing but a set of word tuples (x_i, y_i) such that each word of the tuple is a string of symbols a_k and b_k respectively, such that both sets of symbols belong to the **english alphabet**. Equation 1.4 states that the set B represents the set of reviews that are decomposed into bigram tuple sets.

- A key idea to note before we move on - is that of a **negation bigram**. This is a bigram that contains a sentiment possessing word as the second element of the bigram tuple, while the first element is said to be a **negation word** - essentially a word that negates the sentiment implied by the second word. For instance:

$$(\underbrace{not}_{negation}, \underbrace{good}_{sentiment\ (pos)}) \implies \underbrace{bad}_{sentiment\ (neg)} \quad (2.5)$$

$$(\underbrace{not}_{negation}, \underbrace{bad}_{sentiment\ (neg)}) \implies \underbrace{good}_{sentiment\ (pos)} \quad (2.6)$$

We note in equation 1.5 that - a positive sentiment second element of a bigram tuple accompanied by a negation word results in an opposite sentiment of 'negative'. Similarly, in equation 6 we notice that - a negative sentiment second element of a bigram tuple accompanied by a negation word results in an opposite sentiment of 'positive'.

2.4 Lexicon Design

We now discuss an essential feature that would enable our algorithm to assign sentiment intensity scores to reviews and subsequently, their sentiment class labels. This essential feature is that of a **sentiment lexicon** [12]. Our attempt is to utilize existing sentiment lexicons like the **Bing lexicon (R)** [13] in conjunction with our own custom mapped sentiment lexicons for this purpose. A **sentiment lexicon** is nothing but a dictionary of words mapped to whether they exhibit a 'positive' or 'negative' sentiment. Essentially each word in such a dictionary is mapped with its associated sentiment class.

- The central idea is to iterate through the review arrays (unigram and bigram sets) - and for each element of the review array - we check if that element belongs to a particular lexicon. If it does - we increment the intensity score of that review by a certain value. In this manner, as we keep incrementing the intensity score over each element of the review array - we obtain a cumulative intensity score for that particular review. The precise process for implementing this will be shown later on. In the next few points - we illustrate the various lexicons we have constructed, essentially as sets.

- **L1 Negative**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 10 and 30. Since these words are associated with a very high discriminative likelihood - for every word in a review that contains a word in this set would get incremented by a value of -3.

$$L_1^- = \{\textit{worst, waste, return, hangs, replacement, } \cdots\} \tag{2.7}$$

Algorithm 1: L1 Negative score increment

inputs; $U = [R_1, \cdots, R_n]$, $L_1^- = \{w | L(w) \in [10, 30]\}$
for *i* $\in (0, |U|)$ **do**
  $T_i \leftarrow 0$
  for *j* $\in (0, |R_i|)$ **do**
    $w_j \leftarrow R_i[j]$
    if $w_j \in L_1^-$ **then**
      $T_i := T_i + (-3.0)$
    end
  end
  return T_i
end

- **L1 Positive**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 10 and 30. Since these words are associated with a very high discriminative likelihood - for every word in a review that contains a word in this set would get incremented by a value of $+3$.

$$L_1^+ = \{\textit{excellent, best, nice, } \cdots\} \tag{2.8}$$

Algorithm 2: L1 Positive score increment

inputs; $U = [R_1, \cdots, R_n]$, $L_1^+ = \{w | L(w) \in [10, 30]\}$
for *i* $\in (0, |U|)$ **do**
  $T_i \leftarrow 0$
  for *j* $\in (0, |R_i|)$ **do**
    $w_j \leftarrow R_i[j]$
    if $w_j \in L_1^-$ **then**
      $T_i := T_i + (+3.0)$
    end
  end
  return T_i
end

- **L2 Negative**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 7 and 10. For every word in a review that contains a word in this set would get incremented by a value of -2.5. The algorithm works the same way as outlined in algorithm 1.

$$L_2^- = \{\textit{defective, customer, pathetic, fault, } \cdots\} \tag{2.9}$$

- **L2 Positive**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 7 and 10. For every word in a review that contains a word in this set would get incremented by a value of $+2.5$. The algorithm works the same way as outlined in algorithm 2.

$$L_2^+ = \{\textit{amazing, love, premium, awesome, } \cdots\} \tag{2.10}$$

- **L3 Negative**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 5 and 7. For every word in a review that contains a word in this set would get incremented by a value of -2.0. The algorithm works the same way as outlined in algorithm 1.

$$L_3^- = \{\textit{help, repair, defect, browser, } \cdots\} \tag{2.11}$$

- **L3 Positive**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 5 and 7. For every word in a review that contains a word in this set would get incremented by a value of $+2.0$. The algorithm works the same way as outlined in algorithm 2.

$$L_3^+ = \{\textit{budget, light, design, value, } \cdots\} \tag{2.12}$$

- **L4 Negative**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between

2.5 and 5. For every word in a review that contains a word in this set would get incremented by a value of -1.5. The algorithm works the same way as outlined in algorithm 1.

$$L_4^- = \{\textit{error, replaced, warranty, bad, } \cdots\} \tag{2.13}$$

- **L4 Positive**: Referring to figure 1.2 - we consider all the **most informative feature words** that exhibit a class discriminative likelihood score between 2.5 and 5. For every word in a review that contains a word in this set would get incremented by a value of $+1.5$. The algorithm works the same way as outlined in algorithm 2.

$$L_4^- = \{\textit{sleek, perfect, fast, wonderful, } \cdots\} \tag{2.14}$$

2.4.1 Lexicons Excluding Discriminative Likelihood Terms

- **Bigram Negative**: This is a bigram lexicon unlike the previous unigram lexicons. This is a dictionary or an array containing bigram tuples - in particular, negation bigram tuples. For this, we essentially iterate through the bigram set review array and for every instance of a 'negative' resulting negation bigram, we increment the review sentiment intensity score by -1. Below is a glimpse of the lexicon followed by its algorithmic implementation.

$$Bi_{neg} = \{\textit{(not, happy), (not, great), (didn't, like), (not, worth), } \cdots\} \tag{2.15}$$

Algorithm 3: Bigram Negative score increment

```
inputs; B = [R_1, ··· , R_n],
  Bi_neg = {(x_i, y_i)|x_i ∈ [negation words] & y_i ∈ [positive words]}
for i ∈ (0, |B|) do
    T_i ← 0
    for j ∈ (0, |R_i|) do
        (x_j, y_j) ← R_i[j]
        if (x_j, y_j) ∈ Bi_neg then
            T_i := T_i + (−1.0)
        end
    end
    return T_i
end
```

- **Bigram Positive**: This is a bigram lexicon unlike the previous unigram lexicons. This is a dictionary or an array containing bigram tuples - in particular, negation bigram tuples. For this, we essentially iterate through the bigram set review array and for every instance of a 'positive' resulting negation bigram, we increment the review sentiment intensity score by $+1$. Below is a glimpse of the lexicon followed by its algorithmic implementation.

$$Bi_{pos} = \{\textit{(no, issues), (no, lags), (no, problems), (not, slow), } \cdots\} \tag{2.16}$$

Algorithm 4: Bigram Positive score increment

inputs; $B = [R_1, \cdots, R_n]$,
$Bi_{pos} = \{(x_i, y_i) | x_i \in [\textit{negation words}]\ \&\ y_i \in [\textit{negative words}]\}$
for *i* $\in (0, |B|)$ **do**
  $T_i \leftarrow 0$
  for *j* $\in (0, |R_i|)$ **do**
    $(x_j, y_j) \leftarrow R_i[j]$
    if $(x_j, y_j) \in Bi_{neg}$ **then**
      $T_i := T_i + (+1.0)$
    end
  end
  return T_i
end

2.4.2 Obtaining Unigram Sentiments of Bigram Sets

The previous section showed how we can deploy negation bigram lexicons to increment and account for sentiment intensity scores for bigram set reviews. However at this juncture, we also note that the bigram set reviews have till now, only been accounted for the intensity score contributed by the negation bigrams - we must also account for the unigram intensity scores contained in the bigram set reviews. For instance, if we have a review as follows:

The laptop is ***not good*** *but it has a* ***nice*** *keyboard* (2.17)

When we subject such a review to the negation bigram lexicon - we would only obtain a sentiment score of -1 due to the presence of the negation bigram - **not good**. However, this sentiment score does not fully account for the degree of sentiment information contained in the given review. We must also then, formulate a unigram set of the above review and pass it through the unigram sentiment lexicons to account for the word - **nice** - which would add $+1$ to the overall score - resulting in a cumulative score of 0 for the given review. We also note here that such a lexicon must **not have the positive word of the negation bigram - so as to not double count and nullify its effect**. With this backdrop, Here are some more lexicons that augment our scoring mechanism.

$$A^+ = \{\textit{amaze, commendable, consistent, } \cdots\} \implies \textit{set of pos words} \tag{2.18}$$

$$A^+ := A^+ - \{A^+ \cap L_1^+\} - \{A^+ \cap L_2^+\} - \{A^+ \cap L_3^+\} - \{A^+ \cap L_4^+\} \tag{2.19}$$

Note that we **subtract** the discriminative likelihood sets from the above set since the 'L' sets have differential score weighting - whereas the set of positive words A^+ is simply associated with a $+1$ score weighting for each positive word encountered in the review arrays. Similarly we have the negative version of this as well.

$$A^- = \{\textit{abrupt, ailing, beset, betray, } \cdots\} \implies \textit{set of neg words} \tag{2.20}$$

$$A^- := A^- - \{A^- \cap L_1^-\} - \{A^- \cap L_2^-\} - \{A^- \cap L_3^-\} - \{A^- \cap L_4^-\} \tag{2.21}$$

Finally we present the unigram sentiment lexicons in which we also subtract out the positive/negative words contained in the negation bigram lexicons:

$$B^{+} := A^{+} - \{y_i | \forall (x_i, y_i) \in Bi_{pos}\} \tag{2.22}$$

$$B^{-} := A^{-} - \{y_i | \forall (x_i, y_i) \in Bi_{neg}\} \tag{2.23}$$

2.5 STEP 1 - Dividing the Dataset into Components

- As a first step of building our algorithm, we divide the dataset of reviews into two essential components - the **first component** contains all those reviews that contain **negation bigrams** - the **second component** contains all those reviews that do not contain negation bigrams. We present this contituent algorithm below:

Algorithm 5: Dataset Division

inputs; $B_N = []$, $B_Y = []$
for $i \in (0, |U|)$ **do**
  $B_i \leftarrow \{(x_i, y_i) | \forall x_i \in R_i \ \& \ \forall y_i \in R_i\}$
  if $B_i \cap Bi_{neg} \bigcup B_i \cap Bi_{pos} > 0$ **then**
    $B_Y[i] \leftarrow R_i$
    if $B_i \cap Bi_{neg} \bigcup B_i \cap Bi_{pos} == 0$ **then**
      $B_N[i] \leftarrow R_i$
    end
  end
end
return B_N, B_Y

- In the above algorithm we note that Bi_{pos} and Bi_{neg} denote the positive and negative, negation bigram lexicons, B_i represents the bigram set for review R_i, B_N is the array in which we iteratively append reviews R_i that **do not** contain negation bigrams that B_Y is the array in which we iteratively append reviews R_i that contain negation bigrams. Below is a schematic that shows this as well.

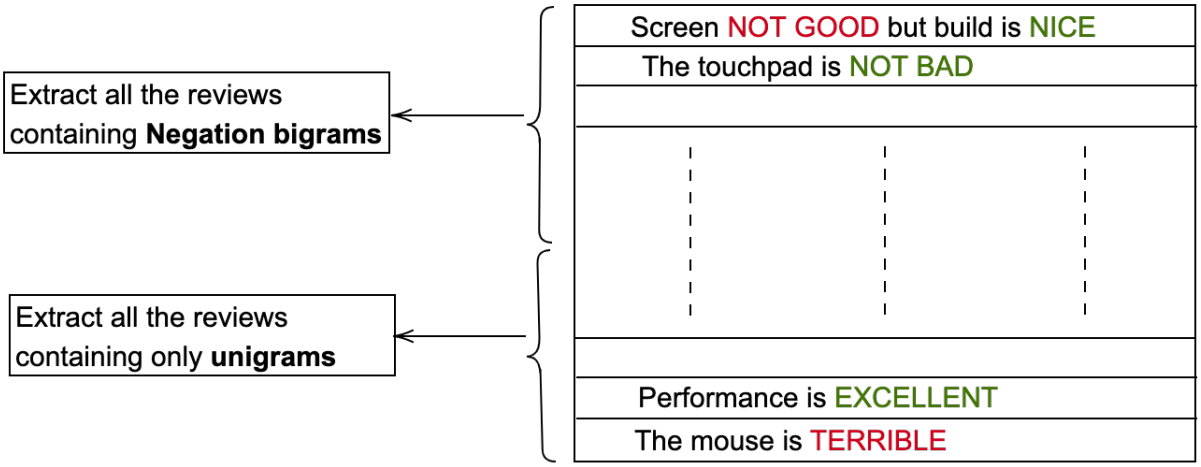
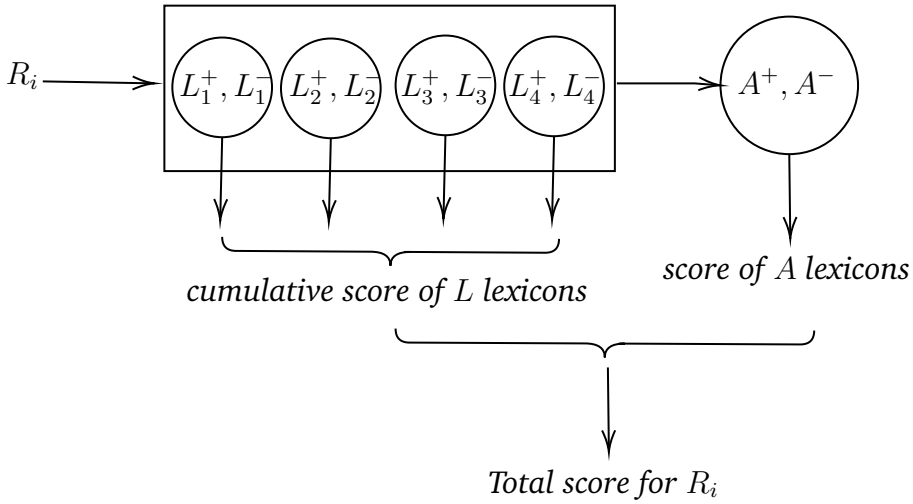


Figure 2.3: Dividing the dataset into B_Y and B_N sets.

- After dividing the dataset into these two sets - we then pass all the reviews in these through a chain of lexicons that we had described above. This is shown below for the unigram set. The only difference for the bigram set would be that an additional set would be added to get the bigram reviews total score.



2.6 STEP 2 - Assigning Labels

- In the previous section we saw how the sentiment intensity scores are computed by cumulatively adding the scores of each review such that each review is parsed through all the lexicons we had defined.
- Now we get to the task of assigning class labels based on these intensity scores. The general rule we have adopted for this task is to assign a label of 'negative' if the sentiment intensity score is below or equal to 0 and 'positive' if the sentiment intensity score is above 0. In the notation below $s(R_i)$ denotes the score of review R_i. ch

$$label = \begin{cases} negative, \ s(R_i) \leq 0 \\ positive, \ s(R_i) > 0 \end{cases} \tag{2.24}$$

- Below is a glimpse of what the dataset looks like after we have implemented this process of scoring and labeling. Note that 0 indicates negative and 1 indicates positive sentiment. [14]

REVIEW	score	LABELS
key quality not good board	-3.5	0
hence looks user boot look other handy conveni...	20.5	1
supports android camera memory all user no lag...	2.0	1
android premiere rendering no lag laptop worki...	5.0	1
large complaints difficult scroll smooth handy...	10.0	1
...	...	...
learning great thing flawlessly attending hdmi...	6.5	1
came even touchpad laptop working pickup refun...	-8.0	0
product really cute excellent liked	5.5	1
properly sensor not working	3.0	1
buddy can't even wastage i'm don't money plz f...	1.0	1

Figure 2.4: Dataset post scoring and labeling

- **NOTE** - that the labels assigned in the above figure are the result of our algorithm. We must now ascertain the accuracy of these labels by comparing them with the gold class labels. To do this, we simply count the number of instances where the gold class labels match the predicted labels by our algorithm and divide this by the total number of reviews - to get a

percentage measure of accuracy [15] of the predicted labels with respect to the gold class labels. This is done by the following function.

Algorithm 6: Accuracy Computation

inputs; counter $= 0$, Gold Labels G_L, Predicted Labels P_L
for $i \in (0, |Dataset|)$ **do**
  if $G_L[i] == P_L[i]$ **then**
    counter = counter $+1$
  end
end
return counter$/|Dataset|$

2.7 STEP 3 - Computing VADER scores

- VADER is an ML algorithm widely utilized for the purpose of computing sentiment intensity scores. [16] It is available as part of the NLTK library in Python [17] - which is the standard library for NLP related computation. Our aim is to compare the sentiment labeling accuracy of our algorithm with that of VADER, with respect to the benchmark Gold class labels.
- Essentially VADER is a powerful algorithm that has been trained to perform a multitude of operations like - removing stopwords and applying various processes - to assign sentiment intensity scores to textual data. [18]
- We have passed the contents of our dataset fo this algorithm as well - computed the VADER scores for each review and consequently, labeled each review with respect to the same 0 score threshold as we had done for our algorithm. **This ensures a level playing field and an unbiased comparison between the two mechanisms**.

2.8 Results on the Training Data

We have found that our algorithm **consistently outperforms** the VADER sentiment labeling mechanism - by giving a higher accuracy for all the **brand level** datasets, as well as the **comprehensive dataset** - containing reviews for all the brands put together. Here are the accuracies so obtained:

Datasets	**Algorithm**	**VADER**
MAC	89%	86%
DELL	85%	83%
LENOVO	87%	80%
HP	88%	86%
Complete	87%	86%

2.9 Results on Unseen Data

In order to further validate the efficacy of our algorithm - we have compiled an extra set of 200 reviews, parsed in a random manner. This set of reviews contains reviews from various brands including - *ACER, ASUS, HP, LENONO, DELL, MAC, MI*.

$$\textit{Unseen Dataset} \implies \textit{Algorithm} : 91\%, \textit{VADER} : 80\% \tag{2.25}$$

We see from the above results that our algorithm has, rather significantly outperformed the VADER mechanism on previously unseen data as well - albeit pertaining to only a small subset of product reviews - those concerning Laptop brands in the Indian market.

Chapter 3

Statistical Insights

In this chapter, we test hypotheses with respect to the sentimenet intensity scores computed in the first section - and consequently attempt to make statistical inferences about the laptop brands.

3.1 The Backdrop

Given the sentiment intensity scores for reviews across the four brands - LENOVO, HP, MAC, DELL - what we essentially have are **data distributions** of a measure that provides information about the **perceived quality** of each brand. We must note that a distribution of various degrees of sentiment intensity can be considered as a distribution of various degrees of 'perceived quality' as well - by establishing such an equivalence between the variables - **sentiment intensity** and **perceived quality** - we can then study the statistical properties of these data distributions to essentially make inferences on the perceived quality of various brands, in comparison. Comparisons on such measures can give us further insight on the **positioning** of the brands in the market environment, with respect to their competitors [19]. The following analytical methods are followed for this purpose:

- **First**: We look at the fundamental **summary statistics** and **histogram plots** with respect to the sentiment intensity scores of each brand.
- **Second**: Compare the means among pairs of brands to see if their theoretical/population means differ - by using a **t-test** [20]. Before implementing such a statistical test - we examine the **QQ-plot** [21] for each brand to essentially validate our **assumption of normally distributed data** - which is a prerequisite for performing a statistical t-test.
- **Third**: We perform a one way t-test to confirm if the population mean of a certain brand is indeed greater in value than the population mean of another brand. By establishing the statistical significance of the average sentiment intensity score for a particular brand - as being greater than that of the other - we can state that on average it has a higher **perceived quality** in comparison [22]. This measure would then directly relate to the market positioning of the pair of brands.

3.2 LENOVO

- Given below are the relevant summary statistics for the data distribution of sentiment intensity scores for the LENOVO brand:

$$\begin{bmatrix} \textbf{mean} & \boldsymbol{\sigma} & \textbf{min} & \textbf{25\%} & \textbf{median} & \textbf{75\%} & \textbf{max} \\ 0.15 & 5.88 & -16 & -3 & -0.5 & 4 & 27 \end{bmatrix} \tag{3.1}$$

- The histogram plot and QQ-plot below validate our normality assumptions to be correct.

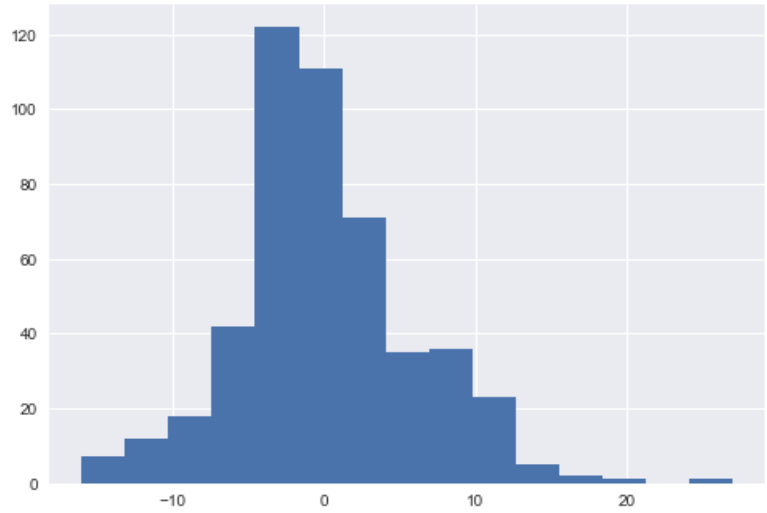


Figure 3.1: Histogram plot - data distribution of sentiment intensity scores for LENOVO

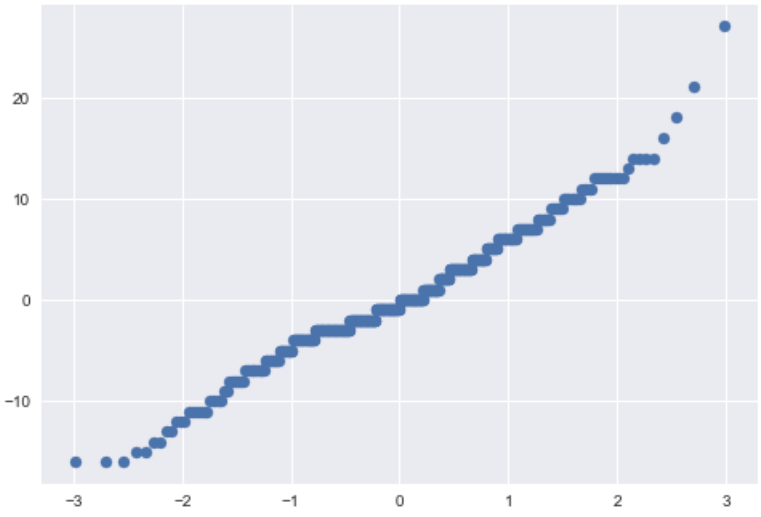


Figure 3.2: QQ-plot for LENOVO - validating normality assumption (appoximately along 45 degree line)

3.3 HP

- Given below are the relevant summary statistics for the data distribution of sentiment intensity scores for the HP brand:

$$\begin{bmatrix} \textbf{mean} & \sigma & \textbf{min} & \textbf{25\%} & \textbf{median} & \textbf{75\%} & \textbf{max} \\ 2.34 & 5.62 & -16 & -1 & 2 & 5 & 30 \end{bmatrix} \tag{3.2}$$

- The histogram plot and QQ-plot below validate our normality assumptions to be correct.

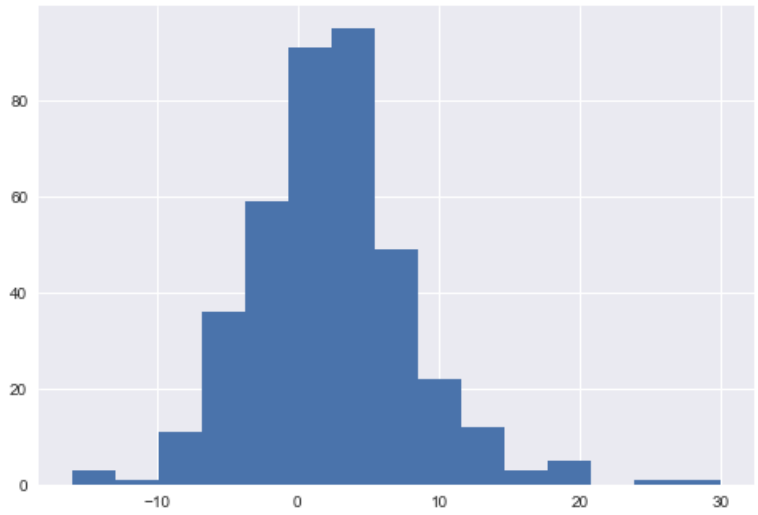


Figure 3.3: Histogram plot - data distribution of sentiment intensity scores for HP

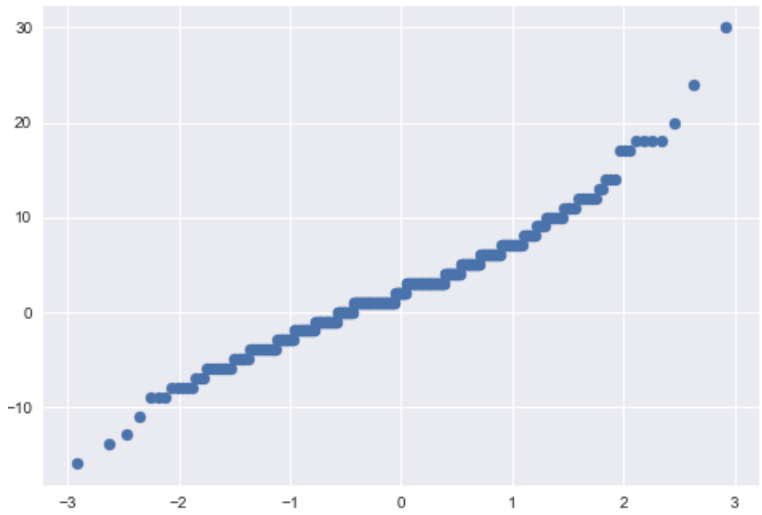


Figure 3.4: QQ-plot for HP - validating normality assumption (appoximately along 45 degree line)

3.4 DELL

- Given below are the relevant summary statistics for the data distribution of sentiment intensity scores for the DELL brand:

$$\begin{bmatrix} \textbf{mean} & \sigma & \textbf{min} & \textbf{25\%} & \textbf{median} & \textbf{75\%} & \textbf{max} \\ 3.09 & 7.12 & -27 & -1 & 3 & 7 & 31 \end{bmatrix} \tag{3.3}$$

- The histogram plot and QQ-plot below validate our normality assumptions to be correct.

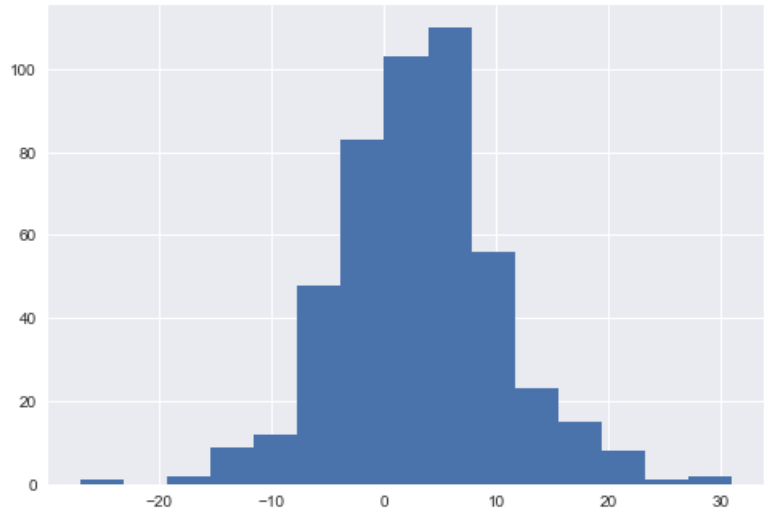


Figure 3.5: Histogram plot - data distribution of sentiment intensity scores for DELL

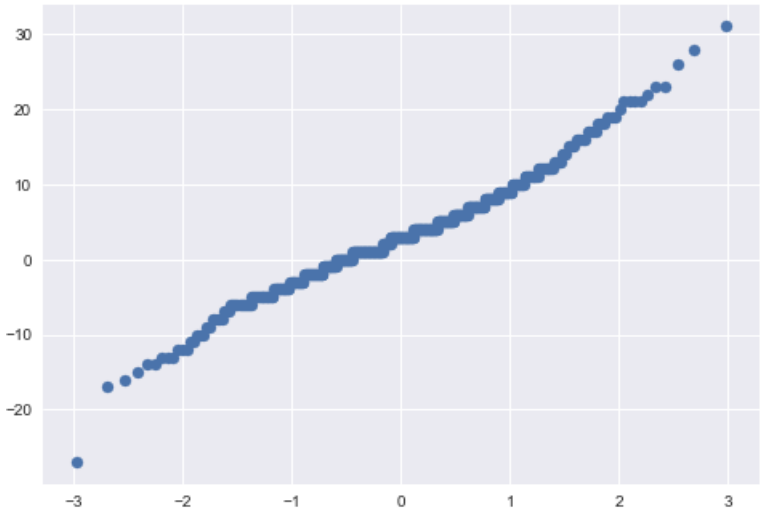


Figure 3.6: QQ-plot for DELL - validating normality assumption (appoximately along 45 degree line)

3.5 MAC

- Given below are the relevant summary statistics for the data distribution of sentiment intensity scores for the MAC brand:

$$\begin{bmatrix} \textbf{mean} & \sigma & \textbf{min} & \textbf{25\%} & \textbf{median} & \textbf{75\%} & \textbf{max} \\ 2.57 & 8.96 & -33 & -2 & 3 & 7 & 51 \end{bmatrix} \tag{3.4}$$

- The histogram plot and QQ-plot below validate our normality assumptions to be correct.

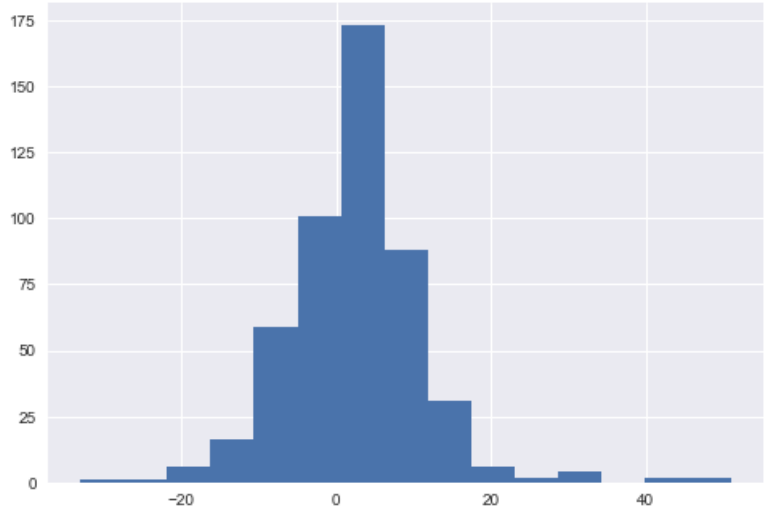


Figure 3.7: Histogram plot - data distribution of sentiment intensity scores for MAC

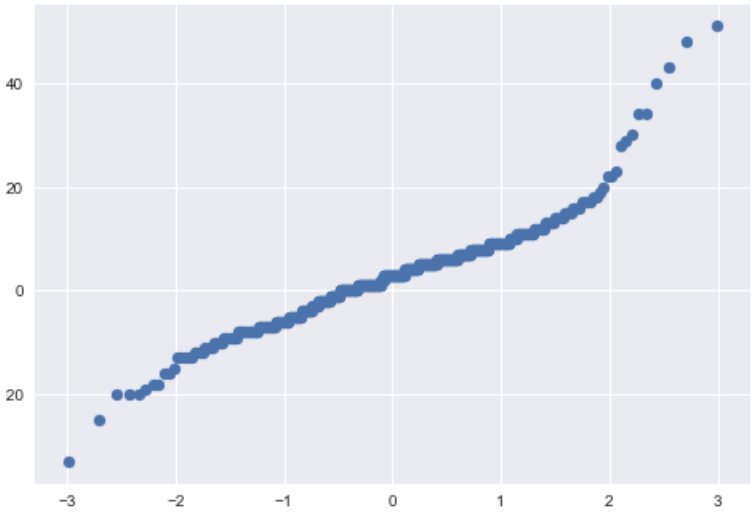


Figure 3.8: QQ-plot for MAC - validating normality assumption (appoximately along 45 degree line)

3.6 Results for Statistical Tests

As explained in the earlier section - we will be using the statistical **t-test** to compare the mean sentiment intensity scores across various brands. We indeed find ourselves on fertile ground to conduct such a test on account of our reasonable assumption of **normality of data** - as was validated through the data distributions and QQ-plots. **Note** that since HP and LENOVO are essentially considered as **budget brands** in this analysis, likewise MAC and DELL are considered as **high bracket brands** - we will make a **constant variance assumption** while comparing brands belonging to the same bracket - and we will not utilize this assumption while comparing brands across brackets.

- We first construct the t-test formulation for brands HP and LENOVO - since these are brands belonging to the same bracket - we assume equal variances [23]. Note that x and y encode observations for sentiment intenstiy scores for the two brands, respectively.

$$\textbf{assume} \implies x_1, \cdots, x_n \sim N(\mu_1, \sigma^2) \implies \textit{HP} \tag{3.5}$$

$$\textbf{assume} \implies y_1, \cdots, y_m \sim N(\mu_2, \sigma^2) \implies \textit{LENOVO} \tag{3.6}$$

$$\textit{null hypothesis} \implies H_0 : \mu_1 - \mu_2 = 0 \tag{3.7}$$

$$\textit{alternate hypothesis} \implies H_A : \underbrace{\mu_1}_{HP} - \underbrace{\mu_2}_{LEN} > 0 \tag{3.8}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \ \bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i \tag{3.9}$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{m-1}\sum_{i=1}^{m}(y_i - \bar{y})^2 \tag{3.10}$$

$$t = \frac{\bar{x} - \bar{y}}{s} \tag{3.11}$$

  Under a significance level of 0.05 - we observe a statistically significant t-value of 5.60 and a p-value of 2.86×10^{-8} - hence we can **reject the null hypothesis** and state that **the true average sentiment intensity score for HP is higher than that of LENOVO, thus implying a better perceived quality**.

- We then construct the t-test formulation for brands MAC and DELL - since these are brands belonging to the same bracket - we assume equal variances. Note that x and y encode observations for sentiment intenstiy scores for the two brands, respectively.

$$\textbf{assume} \implies x_1, \cdots, x_n \sim N(\mu_1, \sigma^2) \implies \textit{MAC} \tag{3.12}$$

$$\textbf{assume} \implies y_1, \cdots, y_m \sim N(\mu_2, \sigma^2) \implies \textit{DELL} \tag{3.13}$$

$$\textit{null hypothesis} \implies H_0 : \mu_1 - \mu_2 = 0 \tag{3.14}$$

$$\textit{alternate hypothesis} \implies H_A : \underbrace{\mu_1}_{MAC} - \underbrace{\mu_2}_{DELL} > 0 \tag{3.15}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i \tag{3.16}$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{m-1}\sum_{i=1}^{m}(y_i - \bar{y})^2 \tag{3.17}$$

$$t = \frac{\bar{x} - \bar{y}}{s} \tag{3.18}$$

Under a significance level of 0.05 - we do not observe a statistically significant t-value of -0.99 and a p-value of 0.32 - hence we **do not reject the null hypothesis** and state that **the true average sentiment intensity score for MAC and DELL is the same - implying a similar perceived quality**.

- We first construct the t-test formulation for brands DELL and LENOVO - since these are brands belonging to different brackets - we do not assume equal variances. Note that x and y encode observations for sentiment intenstiy scores for the two brands, respectively.

$$\textbf{assume} \implies x_1, \cdots, x_n \sim N(\mu_1, \sigma^2) \implies \textit{DELL} \tag{3.19}$$

$$\textbf{assume} \implies y_1, \cdots, y_m \sim N(\mu_2, \sigma^2) \implies \textit{LENOVO} \tag{3.20}$$

$$\textit{null hypothesis} \implies H_0 : \mu_1 - \mu_2 = 0 \tag{3.21}$$

$$\textit{alternate hypothesis} \implies H_A : \underbrace{\mu_1}_{DELL} - \underbrace{\mu_2}_{LEN} > 0 \tag{3.22}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i,\ \bar{y} = \frac{1}{m}\sum_{i=1}^{m} y_i \tag{3.23}$$

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \neq \frac{1}{m-1}\sum_{i=1}^{m}(y_i - \bar{y})^2 \tag{3.24}$$

$$\textit{pooled variance} \implies s_P = \frac{(n-1)s_x^2 + (m-1)s_y^2}{n+m-2}\left(\frac{1}{n} + \frac{1}{m}\right) \tag{3.25}$$

$$t = \frac{\bar{x} - \bar{y}}{s_P} \tag{3.26}$$

Under a significance level of 0.05 - we observe a statistically significant t-value of 6.97 and a p-value of 5.98×10^{-12} - hence we **reject the null hypothesis** and state that **the true average sentiment intensity score for DELL is higher than that of LENOVO - implying a similar perceived quality**.

- After conducting similar tests for the other pairs we find the following results:

Comparison	**T-value**	**P-value**
HP vs. LEN	5.600	0.0000000286
MAC vs. DELL	-0.994	0.32
MAC vs. LEN	5.010	0.000000662
MAC vs. HP	0.459	0.64
DELL vs. LEN	6.97	0.0000000000598

NOTE - we noticed a very small margin of accepting the null hypothesis with respect to DELL vs. HP. A p-value of 0.08 was observed - essentially this would lead to a rejection under a 10% significance level.

- **To summarize** - we note a higher perceived quality of HP, DELL and MAC over LENOVO and similar perceived quality for MAC and DELL.

Chapter 4

Establishing Metrics affecting Sentiment Intensity Scores

Based on certain metrics defined by indicator variables, we establish the significance of these variables in relation to the sentiment intensity score.

4.1 Obtaining Relevant Metrics

In the previous notes, we had explained in detail as to how the sentiment intensity scores were computed by our algorithm, in addition to establishing the accuracy of its sentiment labeling process with respect to the gold class labels. Now, the attempt will be to further drill down into the contents of the reviews - and establish whether the presence or absence of certain **metrics** or variables - would indeed significantly effect the sentiment intensity scores. The motivation for such an analysis is that it would help us identify **classes of keywords** and other factors affecting the sentiment scores - **such that we can derive more insight into factors that tend to trigger various degrees of sentiment amongst customers**.

4.1.1 Unsupervised Learning - LDA

Since the premise of our analysis is anchored in the aim of providing a generalized framework for extracting insightful information from product reviews, through an automated process - we would refrain from applying manual inspection of this particular dataset to extract our explanatory metrics. Rather, we deploy the **LDA algorithm** - a topic modeling approach - to extract **topics** or **classes of keywords**, inherent in the dataset of reviews [24]. This approach validates our aim of presenting a generalized method to extract information, from practically any textual dataset.

- Setting the **topic components** to 3 - we obtain the following classes of keywords as given out by the LDA algorithm. [25]

- We show 3 **wordcloud plots** [26] below - such that each wordcloud simply plots the words constituting the 3 topics. Based on these words, we have termed these 3 topics as - **performance related**, **customer service related** and **price/looks related**.

Figure 4.1: Performance related keywords - words that define usage and performance capabilities of the laptop brands, as perceived by consumers

Figure 4.2: Customer service related keywords - words that define customer service related issues

Figure 4.3: Customer service related keywords - words that define price related and laptop looks related concerns

- Based on the lists of words given out - we essentially obtain our **metrics** - which in this case, are sets of keywords. We utilize a numerical encoding of these metrics by developing **indicator variables** that indicate the presence or absence of these sets of keywords, in the reviews.
- For instance, if we have a review of the form - *"this laptop has issues with software and customer service"* - then the indicator metric for the **performance related** set would take on a value of 1 due to the presence of the keyword **"software"** and - the indicator metric for the **customer service related** set would also take on a value of 1 due to the presence of the keywords **"customer service"**. We note that the indicator metric for **price/looks related** in this case would be 0 since no term of this category is present in this particular example.

4.2 Setting the Stage for Regression

With the general framework in place, we would now extend our analysis from the previous notes on sentiment intensity scores - to factor in effects of our newly extracted metrics. The idea is to perform a **dummy variable regression using interaction terms** [27] - to establish statistically significant relationships between the explanatory indicator variable metrics and the sentiment intensity score response variable [28]. By establishing such a relationship - we can gain insight into factors that could potentially help us dictate the prevailing sentiment amongst customers, with respect to certain products. The process for setting up a dataset suitable for regression analysis is given below:

- We first compute the indicator metric values for each review in our dataset, according to the 'topics' extracted above. Below is an algorithm for this purpose. Before moving on to the algorithm, here are some important notations:

$$P = \{w_p | w_p \in \textit{Performance related words}\} \tag{4.1}$$

$$C = \{w_c | w_c \in \textit{Customer related words}\} \tag{4.2}$$

$$L = \{w_l | w_l \in \textit{Price/looks related words}\} \tag{4.3}$$

$$R_i = \{w_i | w_i \in \textit{unigram set of review } R_i\} \tag{4.4}$$

$$I_P = \begin{cases} 1, R_i \cap P \neq \phi \\ 0, R_i \cap P = \phi \end{cases} \tag{4.5}$$

$$I_C = \begin{cases} 1, R_i \cap C \neq \phi \\ 0, R_i \cap C = \phi \end{cases} \tag{4.6}$$

$$I_L = \begin{cases} 1, R_i \cap L \neq \phi \\ 0, R_i \cap L = \phi \end{cases} \tag{4.7}$$

Algorithm 7: Dummy Variable Encoding

inputs; $I_P = []$, $I_C = []$, $I_L = []$
for $i \in (0, |U|)$ **do**
  if $|R_i \cap P| > 0$ **then**
    $I_P[i] \leftarrow 1$
  end
  if $|R_i \cap C| > 0$ **then**
    $I_C[i] \leftarrow 1$
  end
  if $|R_i \cap L| > 0$ **then**
    $I_L[i] \leftarrow 1$
  end
end

- In the next step, each review is also labaled as per its **brand**. This would then form another dummy variable, taking on $0/1$ values for various brands in the entire dataset. Finally, we append this variable, along with the indicator metrics [29] - to the dataset containing reviews and their sentiment intensity scores, as computed by our algorithm. Here is a snapshot of the dataset. **NOTE** - the score values have been standardized. We also note that in the subsequent regression analysis - the sentiment intensity scores would be the **response variable** and the indicator metrics, along with the brand indicator - would be the **explanatory variables**.

reviews	brands	scores	performance	customers	pricelooks
no problem working almost yet months well	HP	-0.146039	0	1	0
used not clearly frequently dull become sleek ...	HP	-0.074592	1	0	1
lasts ms quite use no fast battery heating ava...	HP	-0.217486	1	1	0
not bad ok	HP	-0.360381	0	0	0
grabbing eye good fan display performance not ...	HP	-0.146039	1	0	1
...	...	...	...	...	...
no subscription bad mentioned they dell not ho...	DELL	-0.074592	1	1	0
laptop descriptiom ssd included drive part not...	DELL	-0.074592	1	0	0
no saying title because they mentioned care fa...	DELL	-0.931957	1	1	0
processing fast look premium sleek	DELL	0.496984	1	0	1
would year bought amazon initial october warra...	DELL	-1.217745	0	1	0

Figure 4.4: Dataset for Regression analysis. To interpret the second row - this review belongs to the **HP brand**, has a sentiment intensity score of -0.07 (standardized) and shows a presence of **performance** and **price/looks** related metrics

4.3 Regression Analysis

We have fit 6 models using the above framework - each time trying out various combinations of **interaction terms**. Each model is also compared with the others in a pairwise manner - to establish a superior explanatory power of the models via an **ANOVA F test** [31]. For this presentation however, we only present the baseline model and the final model selected. Below we present the model equations and their individual summaries, before moving on to a model-wise comparison.

- **NOTE**: The coefficients indicate the average sentiment intensity score for the corresponding indicator variable metric . Further we must note that these are standardized scores [30], so a value between $[-5.0, -0.57]$ corresponds to an absolute score range of $[-33.0, -1.5]$ - a value between $[-0.57, -0.07]$ corresponds to an absolute score range of $[-1.5, 2.0]$ - a value between $[-0.07, 0.49]$ corresponds to an absolute score range of $[2.0, 5.5]$ - a value between $[0.49, 7]$ corresponds to an absolute score range of $[5.5, 51.0]$.

- **Model 1** - This model features the basic explanatory variables with no interaction terms.

$$Y_{score}^{(i)} = \beta_0 + \beta_1 X_P^{(i)} + \beta_2 X_C^{(i)} + \beta_3 X_L^{(i)} + \beta_4 B^{(i)} + \epsilon^{(i)} \tag{4.8}$$

  With this, the **customers variable** corresponds to an average sentiment intensity of -0.7 - the **performance variable** has an average sentiment intensity of 3.7 - the **price/looks variable** has an average sentiment

intensity of 6.3 - the **HP brand** has an average sentiment intensity of 1.1 - the **LENOVO brand** has an average sentiment intensity of -0.5.

```
Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -0.181873   0.056386  -3.226  0.00128 **
df$customers1   -0.418040   0.044544  -9.385  < 2e-16 ***
df$performance1  0.249313   0.045457   5.485 4.72e-08 ***
df$pricelooks1   0.623432   0.043186  14.436  < 2e-16 ***
df$brandsHP     -0.131195   0.062629  -2.095  0.03633 *
df$brandsLEN    -0.385964   0.058638  -6.582 6.04e-11 ***
df$brandsMAC     0.003003   0.058863   0.051  0.95932
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9056 on 1833 degrees of freedom
Multiple R-squared:  0.1826,    Adjusted R-squared:   0.18
F-statistic: 68.26 on 6 and 1833 DF,  p-value: < 2.2e-16
```

Figure 4.5: Model 1: We see that all the variables are individually, statistically significant based on the **p-values** and **t-stats** (as per a significance level of 0.05) - except the indicator variable for MAC. The model's overall **F-value** also indicates that the variables together, exhibit a relation with the response variable.

- **Model 6**: This is our final model that has turned out to have the best explanatory power as compared to the rest. Here are the details:

$$Y_{score}^{(i)} = \beta_0 + \beta_1 X_P^{(i)} + \beta_2 X_C^{(i)} + \beta_3 X_L^{(i)} + \beta_4 B^{(i)} + \beta_5 X_P^{(i)} X_C^{(i)} + \beta_6 X_P^{(i)} X6(i)_L + \beta_7 X_C^{(i)} X_L^{(i)} + \beta_8 X_C^{(i)} B^{(i)} + \beta_9 X_L^{(i)} B^{(i)} + \epsilon^{(i)} \quad (4.9)$$

  With this, the **customers variable** corresponds to an average sentiment intensity of -1.7 - the **price/looks variable** has an average sentiment intensity of 5.1 - the **performance-customer interaction** has an average sentiment intensity of 3.4 - the **performance-price interaction** has an average sentiment intensity of 4.6 - the **customer-price interaction** has an average sentiment intensity of 3.4 - the **customer-LENOVO interaction** has an average sentiment intensity of 0.2 - the **price-HP interaction** has an average sentiment intensity of -0.5 - the **price-LENOVO interaction** has an average sentiment intensity of 0.2.

```
Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -0.108320   0.083433  -1.298   0.1944
df$performance1                -0.007131   0.071645  -0.100   0.9207
df$customers1                  -0.556981   0.110563  -5.038 5.18e-07 ***
df$pricelooks1                  0.433361   0.107009   4.050 5.34e-05 ***
df$brandsHP                     0.025556   0.100975   0.253   0.8002
df$brandsLEN                   -0.141324   0.095508  -1.480   0.1391
df$brandsMAC                    0.063679   0.104202   0.611   0.5412
df$performance1:df$customers1   0.204336   0.092431   2.211   0.0272 *
df$performance1:df$pricelooks1  0.373958   0.090099   4.151 3.47e-05 ***
df$customers1:df$pricelooks1    0.201138   0.089700   2.242   0.0251 *
df$customers1:df$brandsHP       0.054221   0.133521   0.406   0.6847
df$customers1:df$brandsLEN     -0.266959   0.118686  -2.249   0.0246 *
df$customers1:df$brandsMAC     -0.155634   0.117160  -1.328   0.1842
df$pricelooks1:df$brandsHP     -0.310884   0.123995  -2.507   0.0123 *
df$pricelooks1:df$brandsLEN    -0.268405   0.115883  -2.316   0.0207 *
df$pricelooks1:df$brandsMAC     0.008912   0.116659   0.076   0.9391
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8931 on 1824 degrees of freedom
Multiple R-squared:  0.2089,    Adjusted R-squared:  0.2024
F-statistic: 32.12 on 15 and 1824 DF,  p-value: < 2.2e-16
```

Figure 4.6: Model 6: We see that the variables - $X_C, X_L, (X_P X_C), (X_P X_L), (X_C X_L), (X_C B_{len}), (X_L B_{hp}), (X_L B_{len})$ - are individually, statistically significant based on the **p-values** and **t-stats** (as per a significance level of 0.05). The model's overall **F-value** also indicates that the variables together, exhibit a relation with the response variable.

- Finally, we show that the one way ANOVA test [32] shows that our final model indeed shows a significant reduction in residual sum of squares and is endowed with a better predictive power with an R^2 of around 20%. Here is the ANOVA result.

```
Analysis of Variance Table

Model 1: df$scores ~ df$customers + df$performance + df$pricelooks + df$brands
Model 2: df$scores ~ df$performance + df$customers + df$pricelooks + df$brands +
    df$performance * df$customers + df$performance * df$pricelooks +
    df$customers * df$pricelooks + df$customers * df$brands +
    df$pricelooks * df$brands
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1   1833 1503.1
2   1824 1454.8  9    48.353 6.7361 1.496e-09 ***
---
```

Figure 4.7: ANOVA F-test proving the one way significance of the final model

- Below is a table that summarizes the results and significant parameters for both the models we have analyzed so far.

Model	**Signigicant Parameters**	**Coefficient (average intensity)**
Model 1 $R^2 = 18\%$ $p_F = 2.2 \times 10^{-16}$	- Customers - Performance - Price/Looks - HP - LENOVO	$\rightarrow -0.7$ $\rightarrow 3.7$ $\rightarrow 6.3$ $\rightarrow 1.1$ $\rightarrow -0.5$
Model 6 $R^2 = 20.2\%$ $p_F = 2.34 \times 10^{-16}$	- Customers - Price/Looks - Per * Cust - Per * Price - Cust * Price - Cust * LENOVO - Price * HP - Price * LENOVO	$\rightarrow -1.7$ $\rightarrow 5.1$ $\rightarrow 3.4$ $\rightarrow 4.6$ $\rightarrow 3.4$ $\rightarrow 0.2$ $\rightarrow -0.5$ $\rightarrow 0.2$

Figure 4.8: Table for Model 1 and Model 6 parameter summaries.

4.4 Points of Interpretation

- Reviews that contain **customer related** keywords tend to be associated with a **negative sentiment**. We might suggest customer service as an area of improvement.
- Reviews that contain **price/looks related** keywords tend to be associated with a **high positive sentiment**. We could say that customers place a strong emphasis on price and appearance of laptops - consequently, they are more likely to be satisfied with laptops that are priced optimally and have an attractive appearance.
- Reviews that contain both **performance** and **customer** related keywords tend to have a decent **positive sentiment**.
- Reviews that contain both **performance** and **price/looks** related keywords tend to have a **high positive sentiment**.
- The **LENOVO customer** based reviews are more likely to be associated with **negative sentiment**.

Chapter 5

Unsupervised Analysis - Extracting Characteristic Information

Using principles from decomposition of document-term matrices and clustering mechanisms - in this note we shall decipher defining characteristics of product review clusters

---

5.1 Vectorizing Documents

Before we delve into the technicals of utilizing matrix decomposition and clustering techniques on document term matrices - we must first understand as to how documents are **vectorized**. Vectorization [33] refers to the process of transforming documents and words into numeric-valued vectors belonging to some n dimensional vector space. Out of the various methods at hand, we will, for our purpose, be using the **count vectorization technique** [34] - whereby we first create a **vocabulary** of all the distinct terms across our corpus - and then denote each document or review as a vector whose elements are the **counts** of each vocabulary word in that particular document [35]. Essentially then, what we have is a $m \times n$ matrix where m represents the number of rows or **the documents** and n represents the number of distinct words in **vocabulary**. With this framework in place - each document is then essentially a $(1 \times n)$ vector in an n dimensional space. We also note that the $(i, j)^{th}$ element of this document-term matrix denotes - *the count of word j in document i* [36]. The technicals of this process for our case are given below:

- Since the vocabulary contains various kinds of **tokens**, involving numerical values, punctuation marks and single-letter words - we filter these out by essentially specifying a **regular expression** [37] that generates the following language:

$$L(R) = (\Sigma_{english})^2 \cup (\Sigma_{english})^3 \cup \cdots \cup (\Sigma_{english})^{20} \tag{5.1}$$

  This regular expression filters and considers only those words to form the vocabulary that - contain only alphabets (and not numbers) and are between the length 2 and 20.

- Using the following Python functionality we use the **count vectorizer** technique to form the document-term sparse matrix [38]. The following code signifies the document-term matrix for all the HP and LENOVO reviews taken together.

```
regex = r'[a-zA-Z]{2,20}'
tfidf = CountVectorizer(token_pattern=regex)
TDM = tfidf.fit_transform(DD['reviews'])
matrix = pd.DataFrame(TDM.toarray(), columns=tfidf.get_feature_names())
```

```
TDM
```

```
<875x2441 sparse matrix of type '<class 'numpy.int64'>'
        with 13864 stored elements in Compressed Sparse Row format>
```

Figure 5.1: A document term sparse matrix denoting 875 reviews and 2441 vocabulary terms.

- We note clustering algorithms such as the **KMeans algorithm** - do not function efficiently with extremely high dimensional sparse data matrices such that this document-term matrix. Instead - it is generally advised to perform a **dimensionality reduction** [39] using techniques like the **singular value decomposition (SVD)** - and then consequently perform clustering based on these reduced form document-term matrices.

5.2 Singular Value Decomposition

- We denote C as an $m \times n$ document term matrix. Then, its SVD is given by the following equation [40]:

$$C = U\Sigma V^T \tag{5.2}$$

  Where Σ is a diagonal matrix containing at most r nonzero **singular values** such that r is the rank of the matrix. Essentially the i^{th} diagonal element of this matrix is of the form $\sigma_i = \sqrt{\lambda_i}$ - such that λ_i represents the i^{th} **eigenvalue** of the square matrix CC^T. We also note that these singular values are typically arranged in descending order along the principal diagonal. [41]

- Our next idea is to then form a **truncated matrix** C_k out of the original matrix C by utilizing this decomposition - where the subscript k denotes the **effective rank** of the matrix. By minimizing the **Frobenius norm** [42] of the **discrepancy matrix** $X = C - C_k$ by appropriately choosing C_k - we find this optimal truncated form. Note that:

$$||X||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}^2} \tag{5.3}$$

- With this we get the truncated form of the document term matrix as follows by omitting all rows and columns except the first k.

$$C_k = U_k \Sigma_k V_k^T \tag{5.4}$$

  With this transformation - we essentially reduce the extremely high dimensionality of the document-terms matrices to a few dimensions that capture a significant amount of variability in the original data.

- It has been suggested in prior works that selecting components in the order of a few hundreds, for high dimensional document-term matrices - is generally advisable. With this, here is the code implementation of this algorithm wherein we have represented each document/review as a 150 element vector, reduced from the erstwhile 2441 component vector.

```
svd = TruncatedSVD(n_components=150, n_iter=9, random_state=42)
svd.fit(TDM)
result = svd.transform(TDM)
svd.explained_variance_ratio_.sum()
```

```
0.7220105960151609
```

Figure 5.2: We note that the explained variance by these 150 components is almost 72% - a significantly large value considering our analysis

5.3 KMeans Clustering

- The primary aim of K-means clustering is to allocate documents to a pre-specified number of clusters K such that the average **Euclidean distance** between the documents and their corresponding cluster centroids is minimized [43]. We denote $\boldsymbol{x}$ as a document vector and $\boldsymbol{\mu}(\omega_k)$ as the centroid vector with respect to cluster k. Note that the centroid is nothing but a component-wise mean vector of all the document vectors in a particular cluster and its computation is given as follows:

$$\boldsymbol{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\boldsymbol{x} \in \omega} \boldsymbol{x} \tag{5.5}$$

- A measure of how well the centroids represent their corresponding cluster is often taken to be the **residual sum of squares (RSS)** - which is the squared distance of each vector from its centroid, summed across all such vectors. [44]

$$RSS_k = \sum_{\boldsymbol{x} \in \omega_k} |\boldsymbol{x} - \boldsymbol{\mu}(\omega_k)|^2 \tag{5.6}$$

$$RSS = \sum_{k=1}^{K} RSS_k \tag{5.7}$$

- While deciding the optimal number of clusters k for our purpose - we attempt to choose a value k such that the subsequent decrease in RSS after increasing the number of clusters beyond k - is much lesser compared to the decrease in RSS till k. This is denoted by a **screeplot** [45] and is shown below for the HP and LENOVO dataset. In the screeplot shown below - the x axis denotes the number of clusters and the y axis denotes the RSS values.

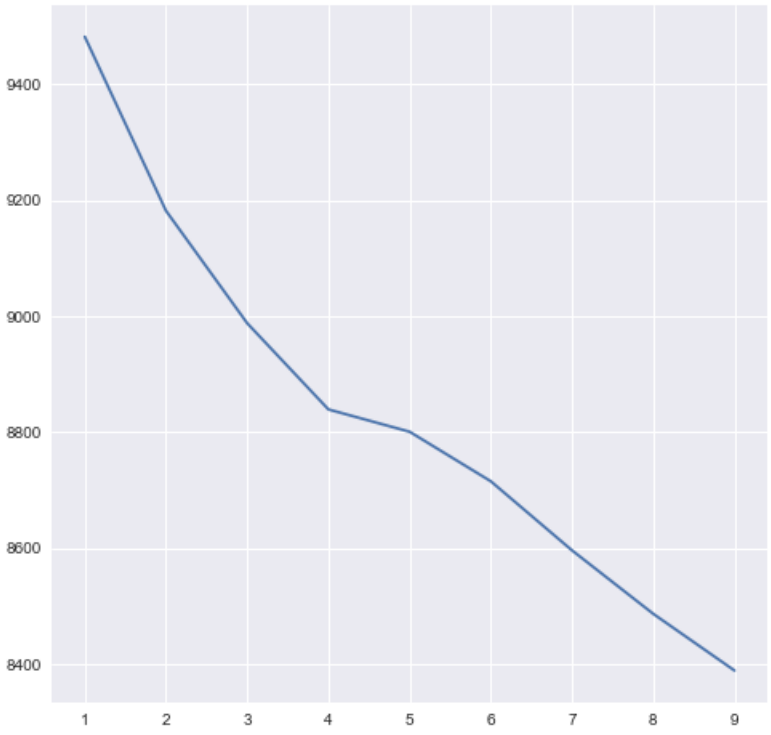


Figure 5.3: We look at the 'knee' in this plot and select the optimal number of clusters to be 4.

- Upon running the K-means algorithm with a pre-specified $k = 4$ on the HP and LENOVO review dataset - we obtain the following cluster assingments:

$$\textit{cluster number } 0 \implies 151 \textit{ reviews} \tag{5.8}$$

$$\textit{cluster number } 1 \implies 426 \textit{ reviews} \tag{5.9}$$

$$\textit{cluster number } 2 \implies 45 \textit{ reviews} \tag{5.10}$$

$$\textit{cluster number } 3 \implies 253 \textit{ reviews} \tag{5.11}$$

- After this cluster assignment - we have essentially separated the data into subsets - in order to analyze the characteristics of reviews in each such cluster.

5.4 Cluster Characteristics

After having separated out the data values into subsets as per their respective cluster numbers - we utilize features of the NLTK Python library to compute various features of the clusters and attempt to gain further insight about the brands.

- We begin with computing the **average word count** in each data subset. Here are the results:

$$\begin{bmatrix} \textbf{cluster 0} & \textbf{cluster 1} & \textbf{cluster 2} & \textbf{cluster 3} \\ 26.82 & 9.23 & 60.67 & 13.02 \end{bmatrix} \tag{5.12}$$

  We note that cluster 1 contains reviews of the shortest average length of 9.23 and cluster 2 contains reviews of the longest average length of 60.67. We further compute the **mean sentiment intensity scores** for each cluster:

$$\begin{bmatrix} \textbf{cluster 0} & \textbf{cluster 1} & \textbf{cluster 2} & \textbf{cluster 3} \\ -3.43 & 0.68 & 8.8 & 3.23 \end{bmatrix} \tag{5.13}$$

  This suggests that cluster 0 is associated largely with negative reviews whereas cluster 2 is associated with highly positive reviews. This further suggests that **longer reviews on average tend to have a high degree of positive sentiment**. However as a note of caution we also observe that cluster 2 and cluster 0 are associated with high values of **standard deviation** in sentiment intensity scores of 8.74 and 6.06 respectively. In order to get a more holistic sense of the **sentiment score characterizations** amongst the clusters, the boxplots [46] presented for each cluster in figure 4.4 below.

- Due to the sentiment intensity departure from an average neutrality for clusters 0 and 2 - we will explore the characteristics of these two clusters in more detail.

5.4.1 Cluster 0 - HP and LENOVO

- This cluster contains 111 reviews for LENOVO and 40 reviews for HP. The distribution of reviews across the three metrics is shown as per the **Venn diagrams** shown in figure 4.5.

- Some of the most frequently occurring words in cluster 0 reviews are as follows:

Lenovo, slow, amazon, battery, worst, money, return, poor, life, service, months, processor, experience, customer, warranty, problem, waste, camera, display

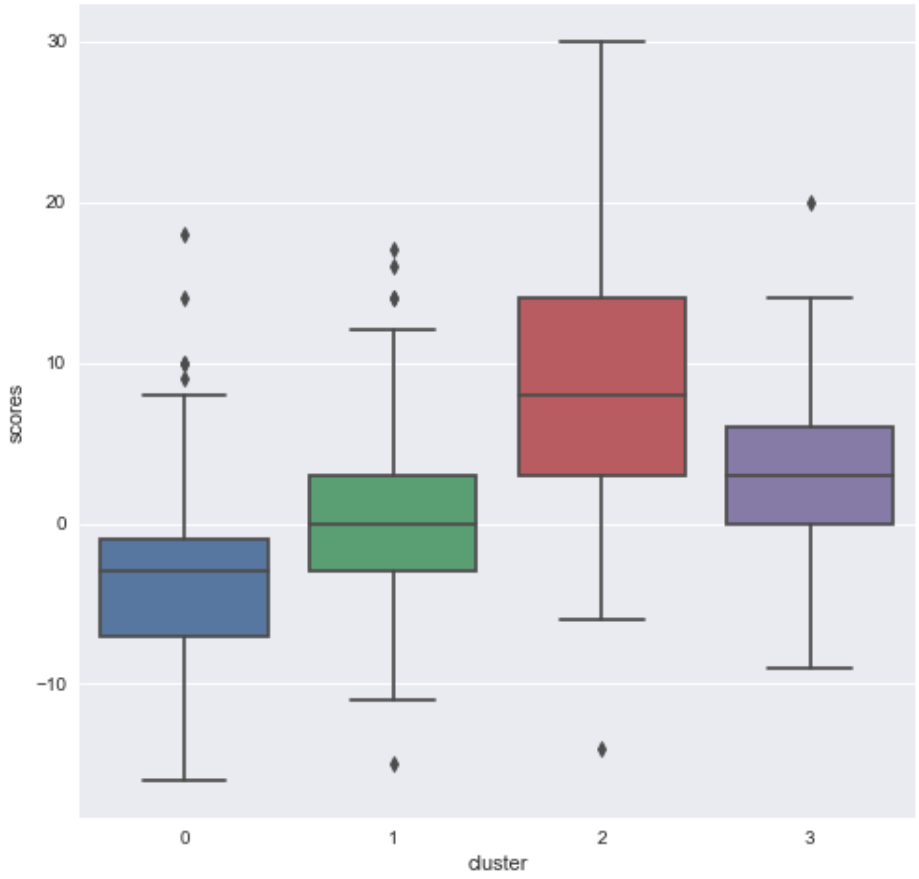


Figure 5.4: Cluster boxplots for sentiment intensity distribution. **Cluster 0** indeed showcases the most negative sentiment intensity with positive outliers. **Cluster 1** and **Cluster 3** are more evenly distributed across negative and positive sentiments with near zero means and more symmetric distributions. **Cluster 4** is by far the most positive in terms of sentiment intensity with just one highly negative outlier.

It is interesting to note the presence of the word "Lenovo" amongst a lot of negative sentiment words as well. This might indicate that cluster 0 tends to cluster the LENOVO related reviews with a largely negative sentiment. The words highlighted in 'red' indicate negative sentiment words as tagged by our lexicons.

- At this point we shall refer to the SVD transformation of the document-term matrix. Since we have essentially computed the most frequent words, we can also compute a measure how **correlated** these words are - or a measure of **co-occurrence** of certain words in this cluster. This is given by the **cosine distance** between pairs of **word vectors** [47]:

$$CD = \cos(\theta) = \frac{\boldsymbol{w}_1 \cdot \boldsymbol{w}_2}{||\boldsymbol{w}_1||_2 \times ||\boldsymbol{w}_2||_2} \tag{5.14}$$

Observations - we note that the cosine distance between the words slow and Lenovo is almost 10% whereas the corresponding value for HP is only

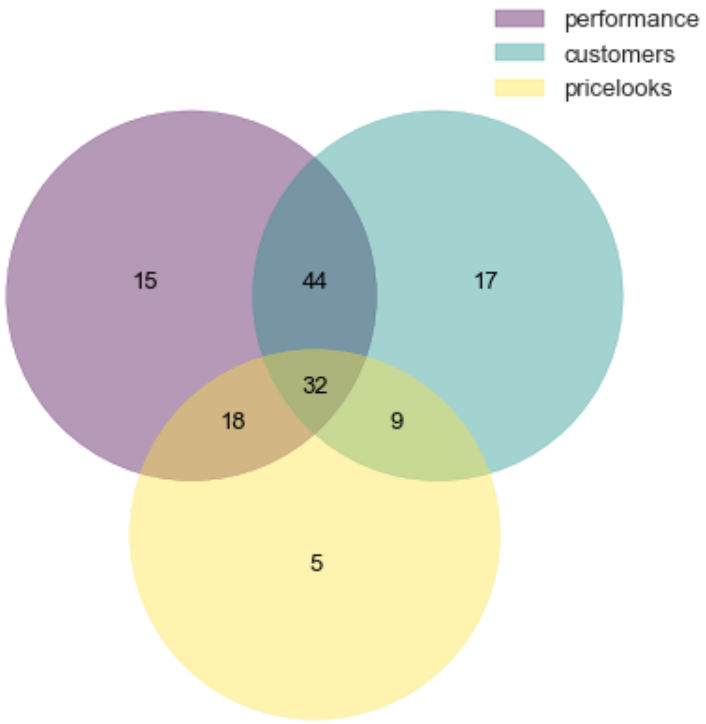


Figure 5.5: Venn diagram indicating the distribution of reviews across the three metrics. We see that almost 55% of the reviews in this cluster contain both **performance** and **customer** related keywords. 22% of the data contains all three metric keywords almost 35% of reviews contain **performance** and **price/looks** related keywords. A very few percentage of reviews in this cluster contains only single metrics.

3%. Cosine distance between the words battery and Lenovo is almost 14% whereas the corresponding value for HP is only 8%. The cosine distance for the word customer is similar for Lenovo and HP at 26% and 21% respectively. Interestingly, the negative word warranty is more closely related to hp at 19% as compared to Lenovo at 16%. This kind of an analysis then lets us essentially tag these words into 'negative/positive' sentiment classes.

- After establishing the most frequent words that characterize these clusters, we would like to get an idea of the **nouns** or **subjects** and certain adjectives in the context of which the reviews are being discussed. Here are the nouns and adjectives for cluster 0:

  jack, internet, laggy, adapter, compact, centre, mousepad, desktop, junk, department, consumer, malfunction, wireless, fix, wastage, corrupt, microphone, lightweight, emulator, multitask, battery, lying, escalation

- Upon computing the cosine distance values of some of these characteristic terms with **positive** and **negative** words, we discover the following

observations - the word jack has a cosine score of 12% with the word **bad** and a score of 0 with the word **good**. It also associates with an 8% cosine distance with respect to the negative word slow. Hence we state that the word **jack** is associated with a negative sentiment - We also note that the word centre is associated with the negative word battery at a 7% cosine distance and is also associated with the word **Lenovo** at a 16% cosine score. This also indicates a negative sentiment for this word - the sentiment trigger might have been battery related service issues with respect to the Lenovo laptop.

5.5 Cluster 2 - HP and LENOVO

- This cluster contains 22 reviews for LENOVO and 23 reviews for HP. The distribution of reviews across the three metrics is shown as per the **Venn diagrams** shown below:
- Some of the most frequently occurring words in cluster 2 reviews are as follows:

  good, quality, battery, performnace, ssd, light, weight, fast, processor, backup, build, gaming, display, design, value, camera, system

- With a cosine distance comparison among words - we find that the word quality has a 35% cosine assoication with the positive word **good** and a 20% association with the negative word **bad**. We might suggest that the overall sentiment for this word is positive. Additionally, it is associated at 15% with HP and at 9% with lenovo. Moreover, since the word bad is more highly correlated with lenovo at 7% as compared to 3% for hp, we might conclude that the word **quality** is being used in a negtive connotation in Lenovo reviews - in a simialar manner, the word display is almost 21% associated with the positive word **good** as opposed to 16% with the negative word **bad**, we might suggest a positive likelihood for this word as well. Additionally, it is also highly correlated with HP at 14% as compared to with lenovo at only 4%.
- Given the above insights derived from word correlations within certain clusters - we can expand the scope and accuracy of our algorithm by adding these terms to our positive and negative lexicons by assigning them likelihood weights that signify sentiment intensity contribution.
- **NOTE** - An analysis along similar lines can be conducted for the other brands as well. What we presented here are techniques and analytical procedures that could be followed - in order to derive meaningful insights from the data.

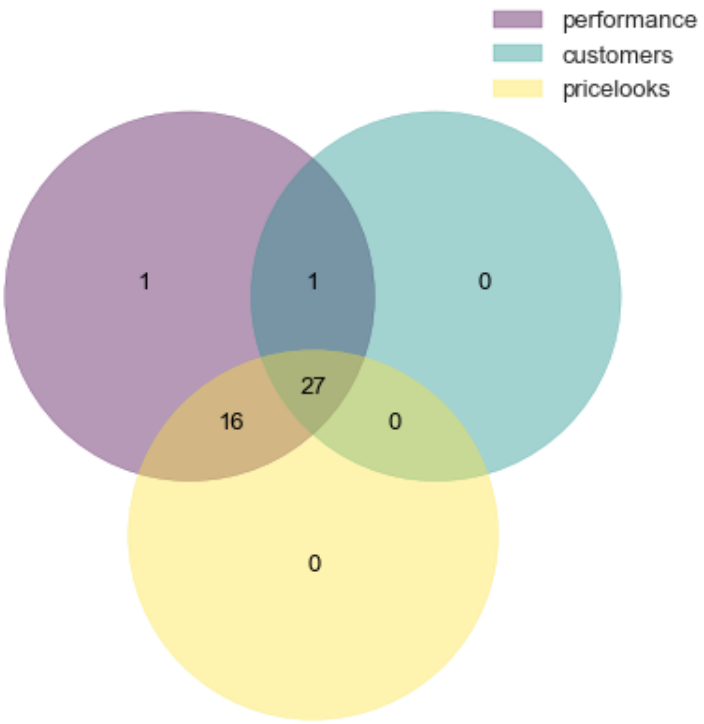


Figure 5.6: Venn diagram indicating the distribution of reviews across the three metrics. The distribution of reviews across the three metrics here is markedly different than what was observed for cluster 0. Here we have a 0 occurrence of reviews that contain **customer** and **price/looks** metrics. There are almost no reviews that contain only single metrics. Almost the entire dataset is composed of reviews that contain **performance** and **price/looks** metrics. Since this cluster has a highly positive sentiment intensity value - we might suggest that reviews in this cluster are characterized by positive reviews and those having the 'performance' and 'price/looks' metrics.

Chapter 6

Summary

To summarize the underlying analysis pertaining to our initially outlined, three broad objectives - here are the main points:

- Having parsed data from Amazon an Flipkart reviews about certain top-selling Laptop brands in the Indian-online-retail space, we manually labeled each review as having a positive or negative sentiment - these served as our gold class labels, providing us with a benchmark against which our sentiment labeling algorithm could be judged. After having trained our algorithm to compute sentiment intensity scores and consequently, sentiment class labels - we found that its accuracy was consistently better than that of the VADER algorithm, for the purpose of this analysis. With these encouraging results, we can state further that given the availability of more data, along with the flexibility in tuning the various hyperparameters of this algorithm - we can scale it up such that it can be deployed to analyze various other kinds of textual datasets.

- In each subsequent chapter - we have also outlined a general series of steps and processes that can be followed as a template for the purpose of extracting insightful information from unstructured data such as product reviews. The algorithm proceeds in the order of - computing sentiment intensity scores (and labels) - providing statistical inferences on scores - establishing significant relationships among scores and key metrics using regression analysis - using unsupervised machine learning, alongside standard NLP tools to extract characteristic and differentiating information about clusters of textual data.

- Finally - in the process of outlining the above procedures, we have also analyzed the market environment with respect to the E-commerce Laptop space in India. The outcomes and interpretations are listed out in the various chapters. **Note** - Apart from using standard algorithms like - KMeans, LDA, SVD, Linear Regression and Naive Bayes Classification - all the algorithms that have been presented throughout various chapters are original formulations. The following Github repository contains all the code that been written to make this analysis possible - https://github.com/akash3101/NLP-programs-and-analyses.

References

[1] Sentiment Classification and Polarity Shifting: Shoushan Li, Sophia Yat Mei Lee, Ying Chen, Chu-Ren Huang, Guodong Zhou.

[2] Hastie, Trevor, Robert Tibshirani, and Jerome H. Friedman. 2001. The Elements of Sta- tistical Learning: Data Mining, Inference, and Prediction. Springer.

[3] Ground Truth Gold — Intelligent data labeling and annotation: Mohan Reddy, CRO, The Hive

[4] A taxonomy on impact of label noise and feature noise using machine learning techniques: A. Shanthini, G. Vinodhini, R. M. Chandrasekaran, P. Supraja.

[5] Rajaraman, A.; Ullman, J. D. (2011). "Data Mining". Mining of Massive Datasets.

[6] Loughran, T. and McDonald, B. (2016). Textual analysis in accounting and finance: A survey. Journal of Accounting Research 54(4), 1187-1230. doi: 10.2139/ssrn.2504147.

[7] Speech and Language Processing: Daniel Jurafsky, James H. Martin.

[8] Natural Language Processing with Python: Steven Bird, Ewan Klein, and Edward Loper.

[9] Schubert, Lenhart, "Computational Linguistics", The Stanford Encyclopedia of Philosophy (Spring 2020 Edition), Edward N. Zalta (ed.).

[10] Discrete Mathematics and Its Applications: Kenneth H. Rosen.

[11] Text Mining with R-A Tidy Approach: Julia Silge and David Robinson.

[12] Sentiment lexicons and non-English languages-a survey: Mohammed Kaity and Vimala Balakrishnan.

[13] This dataset was first published in Minqing Hu and Bing Liu, "Mining and summarizing customer reviews.", Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), 2004.

[14] Python Data Science Handbook-Essential Tools for Working with Data: Jake VanderPlas

[15] An Introduction to Statistical Learning: Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani.

[16] VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text: C.J. Hutto, Eric Gilbert.

[17] The Natural Language Toolkit (NLTK) is a Python package for natural language processing: Steven Bird.

[18] StopWords and Lexicon Normalization for Sentiment Analysis: Min Zhou.

[19] Marketing Management: Philip Kotler, Kevin lane Keller.

[20] Introduction to the Theory of Statistics: Mood, Alexander McFarlane.

[21] Understanding Q-Q Plots: Clay Ford, Statistical Research Consultant, University of Virginia Library.

[22] OpenIntro Statistics: David Diez, Mine Cetinkaya-Rundel, Christopher D Barr.

[23] MIT OCW Statistics: Jeremy Orloff and Jonathan Bloom.

[24] Latent Dirichlet Allocation: David M. Blei, Andrew Y. Ng, Michael I. Jordan.

[25] Linear Discriminant Analysis With Python: Jason Brownlee, Machine Learning Mastery Pty. Ltd.

[26] ”Are tag clouds useful for navigation? A network-theoretic analysis”: Helic, Denis; Trattner, Christoph; Strohmaier, Markus; Andrews, Keith (2011) - International Journal of Social Computing and Cyber-Physical Systems.

[27] Econometric Methods: Jack Johnston, John DiNardo.

[28] Econometrics: Fumio Hayashi.

[29] Bruin, J. 2006. newtest: command to compute new test. UCLA: Statistical Consulting Group. https://stats.idre.ucla.edu/stata/ado/analysis/.

[30] Data Consolidation and Integration: David Loshin, in Master Data Management, 2009.

[31] Lomax, Richard G. (2007). Statistical Concepts: A Second Course. p. 10. ISBN 0-8058-5850-4.

[32] Data Analysis in R: Steve Midway.

[33] Applied Text Analysis with Python: Benjamin Bengfort, Rebecca Bilbro, Tony Ojeda.

[34] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[35] Introduction to Machine Learning with Python-A Guide for Data Scientists: Andreas C. Müller and Sarah Guido.

[36] Introduction to Information Retrieval: Christopher D. Manning, Prabhakar Raghavan, Hinrich Schutze.

[37] Introduction to the Theory of Computation: Michael Sipser.

[38] Yan, Di; Wu, Tao; Liu, Ying; Gao, Yang (2017). An efficient sparse-dense matrix multiplication on a multicore system. IEEE. doi:10.1109/icct.2017.8359956. ISBN 978-1-5090-3944-9.

[39] Dimensionality Reduction for k-Means Clustering and Low Rank Approximation: Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, Madalina Persu.

[40] Introduction to Linear Algebra: Gilbert Strang, Massachusetts Institute of Technology.

[41] Latent semantic analysis: Nicholas E. Evangelopoulos. WIREs Cogn Sci 2013. doi: 10.1002/wcs.1254.

[42] Weisstein, Eric W. ”Frobenius Norm.” From MathWorld–A Wolfram Web Resource. https://mathworld.wolfram.com/FrobeniusNorm.html.

[43] Applied Multivariate Statistical Analysis: RICHARD A. JOHNSON, DEAN W. WICHERN.

[44] Pattern Recognition and Machine Learning: Christopher M. Bishop.

[45] How to get the optimal k-means cluster solution: Anna Makles Schumpeter School of Business and Economics.

[46] John W. Tukey (1977). Exploratory Data Analysis. Addison-Wesley.

[47] Getting to Know Your Data: Jiawei Han, Jian Pei, in Data Mining (Third Edition), 2012.