DEFECT PRODUCT ESTIMATION USING CUSTOMER REVIEWS, AMAZON USE CASE

TARKAN EYERCİ

JANUARY 2022

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING
MASTER'S THESIS IN
COMPUTER ENGINEERING

DEFECT PRODUCT ESTIMATION USING CUSTOMER REVIEWS, AMAZON USE CASE

TARKAN EYERCİ

JANUARY 2022

ABSTRACT

DEFECT PRODUCT ESTIMATION USING CUSTOMER REVIEWS, AMAZON USE CASE

EYERCİ, Tarkan
Master of Science in Computer Engineering

Supervisor: Assist. Prof. Dr. Roya CHOUPANI
Co-Supervisor: Assoc. Prof. Dr. Kasım ÖZTOPRAK
January 2022, 54 pages

Commerce is highly affected by technological improvements, like every other field. Today, all businesses who want to reach end-users, such as manufacturers, retailers, or service providers, can reach their customers quickly over the internet through methods such as e-commerce sites and mobile applications. On the other side, customers now have the opportunity to choose from many options. While deciding, users generally benefit from the comments of other users who have shared the same experience before. In this respect, user comments contain very valuable information. However, on heavily used sites, so many comments accumulate that a person cannot review them one by one. In this study, we focused on a certain feature of the products, namely their defective features, and we propose a method to filter related comments from millions of comments containing the ones that include only defect information. A simple solution may be manually creating a word list related to defects and filtering the comments that contain these words. But manually creating the word list will not be efficient. For this, we trained our own word-embedding model using only the comments of the relevant product groups and created a more efficient list of defect words list by using word similarities using this model. We downloaded a ready-to-use pre-trained word-embedding model and compared it with our own model. We observed that the pre-trained model is more successful in general tasks, while our own model is more successful in creating a word list on a context-specific task.

Keywords: Artificial Intelligence, Machine Learning, Natural Language Processing, Sentiment Analysis, Defect Detection, Word Embeddings

ÖZ

AMAZON ÖRNEĞİ İLE MÜŞTERİ İNCELEMELERİNDEN KUSURLU ÜRÜN TAHMİNİ

EYERCİ, Tarkan
Bilgisayar Mühendisliği Yüksek Lisans

Danışman: Dr. Öğretim Üyesi Roya CHOUPANI
Ortak Danışman: Doç. Dr. Kasım ÖZTOPRAK
Ocak 2022, 54 sayfa

Teknoloji her alanı etkilediği gibi ticareti de çok etkiledi. Günümüzde artık, üreticiler, perakendeciler, hizmet sağlayıcılar gibi son kullanıcıya hitap eden tüm işletmeler e-ticaret siteleri ve mobil uygulamaları gibi yöntemlerle internet üzerinden müşterilerine hızlıca ulaşabiliyorlar. Diğer yandan, müşteriler ise artık birçok seçenek arasından seçim yapma şansına sahipler. Kullanıcılar genellikle seçimlerini yaparken daha önce aynı tecrübeyi paylaşmış diğer kullanıcıların yorumlarından faydalanırlar. Bu açıdan kullanıcı yorumları çok değerli bilgiler içerir. Fakat yoğun kullanılan sitelerde bir insanın tek tek inceleyemeyeceği kadar çok yorum birikir. Biz bu çalışmada, ürünlerin belli bir özelliğine, yani kusurlu özelliklerine odaklandık. Kusur bilgisi içeren milyonlarca yorum içinden ilgili yorumları filtre edebilmek için bir yöntem öneriyoruz. Kusur ile ilgili kelimeleri sözlük yardımı ile elle oluşturup bu kelimeler geçen yorumları filtrelemek bir çözüm önerisi olabilir. Fakat bu kelime listesini elle oluşturmak verimli olmayacaktır. Bunun için sadece ilgili ürün gruplarına ait yorumları kullanarak kendi kelime temsil modelimizi eğitip, bu modelle birlikte kelime yakınlıklarını kullanarak daha verimli bir kusur kelimeleri listesi oluşturduk. Kullanıma hazır önceden eğitilmiş bir kelime temsil modelini indirip, bu modelle kendi modelimizi kıyasladık. Genel konularda hazır modelin daha başarılı olurken, özel bir konuda kendi modelimizin kelime listesi oluşturmada daha başarılı olduğunu gördük.

Anahtar Kelimeler: Yapay Zekâ, Makine Öğrenmesi, Doğal Dil İşleme, Duygu Analizi, Kusur Tespiti, Kelime Temsilleri

ACKNOWLEDGEMENT

I wish to thank my supervisors Assist. Prof. Dr. Roya CHOUPANI and Assoc. Prof. Dr. Kasım ÖZTOPRAK for their motivation and guidance, and to my thesis committee members Assist. Prof. Dr. Abdül Kadir GÖRÜR and Assist. Prof. Dr. Erdal ERDAL for their valuable contributions.

I also would like to thank my colleagues Mr. Sedat AKEL, Mr. Engin TÜRELİ, and Mr. Y. Kürşat TUNCEL for their motivation and support.

Finally, special thanks to my family, my wife, and two daughters. It would not be possible without their understanding and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	: Artificial Intelligence
DL	: Deep Learning
ML	: Machine Learning
NLP	: Natural Language Processing
TF-IDF	: Term Frequency-Inverse Term Frequency
NB	: Naive Bayes
GB	: Gradient Boosting
LR	: Logistic Regression
SVM	: Support Vector Machine
RF	: Random Forest
DT	: Decision Tree
NN	: Neural Network
ANN	: Artificial Neural Network
KNN	: K-Nearest Neighbors
PCA	: Principal Component Analysis
LSTM	: Long Short-Term Memory
BiLSTM	: Bidirectional LSTM
CNN	: Convolutional Neural Network
LDA	: Latent Dirichlet Allocation
TP	: True Positive
TN	: True Negative
FP	: False Positive
FN	: False Negative
OOV	: Out-of-Vocabulary

CHAPTER I

INTRODUCTION

Customer ratings and reviews on e-commerce sites contain valuable information that helps to reach an impression about the product, and people usually consult user reviews and ratings before deciding to purchase a product. But, it is a time-consuming process and impossible for a human to digest all necessary information about products manually from a large size of unstructured data like the customer reviews. There are many challenges while extracting any specific information from text data. One challenge is, data size is big, unstructured, and in text format. Too much storage space, memory, and computing power are needed to handle such big data. Traditional rule-based algorithms are usually not efficient for most of the tasks, so smarter techniques are needed. One other challenge is, if data is in a specific context, general rules cannot be applied successfully. Or it may be the opposite, maybe, the data is not bound to any specific context, and it cannot be applied to a specific context-based task. One other challenge is about language. Each human language has different rules, uses cases, variations, and different ways of expressing meanings. A method for a language may not be applied to another language with the same success. One other challenge is data integrity. Generally, unstructured data is not clean and should be pre-processed, cleaned from outliers, classified, labeled, inspected to contain useful information, and verified to be used for a specific task reliably. Some of these challenges still need manual operations like the need of manual labeling. Google's reCAPTCHA and Amazon's MTurk are examples of smart solutions for such labeling tasks used by big companies. Most of the other challenges are welcomed by the data engineering, data science, AI, ML, and NLP skills.

1.1 PROBLEM DEFINITION

There is many useful information about products hidden in user reviews such as quality, usability, or durability. It may be possible to reach defect information or chronic problems of the products or product families from the customer reviews. There may be a relation between the products, that have similar defects or durability problems, that use the same components. Defected products may also have different relations like the seller or product version. But such information is lost in a large number of reviews. A solution is needed to filter reviews that contain hidden information about product defects.

1.2 RESEARCH QUESTION

Finding reviews of products that contain specific information such as weaknesses, poor quality, or chronic defects can be defined as an NLP problem. The simplest solution to this problem is manually constructing a defect-related words list like ‘defect’, ‘broken’, ‘nonfunctioning’, and searching reviews if they contain these words. But defect-related words are context-specific, a word for a product group may not be used in the same meaning for another product group, so the list cannot be successfully constructed manually. Our research question is; Is it possible to populate a defect-related words list automatically using NLP techniques, and specifically using word similarities in word embeddings? We also try to see if a self-trained word-embedding model using data in a specific context and with specific sentiment performs better than a pre-trained word-embedding model.

1.3 HYPOTHESIS

Our primary task is populating an effective defect-related words list specific to electronic products. This list can later be used to filter reviews. For this purpose, a simple solution is downloading a publicly available pre-trained word-embedding model and by using word similarities, populating defect-related words list. But we argue that, although this solution works, it does not perform very well. Since pre-trained models are trained with a wide range of data that is not context-specific. For our purpose, we suggest training our own model with the reviews of a specific product group and with only negative sentiments to increase the precision of defect-related words.

As a pre-task, before word-embedding training, we propose to test the reliability of our dataset, to see if the reviews contain enough information to judge the sentiments correctly or if the labels (ratings) are consistent with the reviews. For this purpose, we suggest training ML models for a simple sentiment classification task and evaluating the performances. This way we will be confident when using the dataset for word-embedding training.

CHAPTER II

BACKGROUND

2.1 ARTIFICIAL INTELLIGENCE (AI)

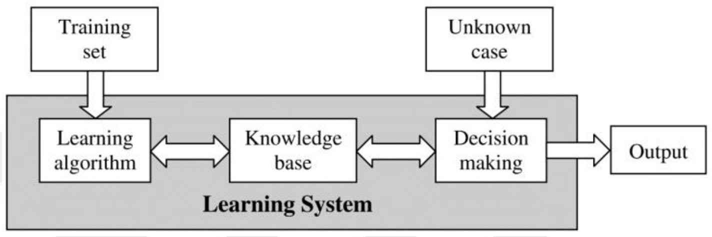
AI is a branch of computer science that deals with human intelligence requiring tasks being carried out by smart machines that mimic human intelligence with problem-solving and decision-making abilities. Alan Turing who is referred as the father of computer science published a paper named ‘Computing Machinery and Intelligence’ [1] and asked the question, ‘Can machines think?’. The term AI is defined as machines thinking and acting like humans since. In 1990, Kurzweil defined AI as ‘The art of creating machines that perform functions that require intelligence when performed by people’. In 1993, Luger and Stubblefield defined AI as ‘The branch of computer science that is concerned with the automation of intelligent behavior’. Russell & Norvig published ‘Artificial Intelligence: A Modern Approach’ in 2009, argued to this definition, and replaced the definition ‘thinking humanly’ with ‘acting rationally’. Acting rationally is possible with intelligent agents, which perceive through sensors and act in an environment while maximizing their performance according to predefined performance measures, adopting a goal and searching for the best path to reach that goal using their knowledge base.

It is believed that there were two AI Winters which were between 1973-1980 and 1980-2010s that led by unmet expectations and hypes. After the 2010s, with the aid of big data and increased computing power, training bigger networks became possible and AI spring began. As a result, concrete examples reached end-users like Google Translate and Google Image Search. At present, there are many use cases, successful, even astonishing examples of AI like speech-to-text applications, which decodes sound waves into text form, search engine ranking systems, which help retrieve the closest content according to the given query, autonomous, self-driving cars, image, sound and video generation, language translation and chatbots

2.2 MACHINE LEARNING (ML)

Machine Learning (ML) is a subset of Artificial Intelligence. In classical computer programs, algorithms are designed and developed by humans to get the expected outputs with the given inputs. In ML the case is quite the opposite, the algorithm is developed by the computer itself to get the expected result using data. This process is also called training the model with data. The parameters of the model are learned from data. A typical flowchart of a Machine Learning System is shown in Figure 2.1.

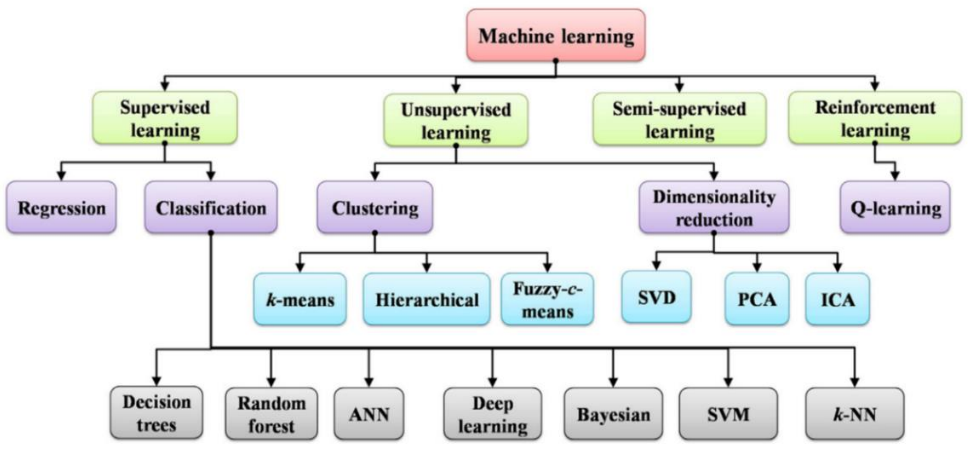
Figure 2.1: Machine Learning Flowchart



Du et al. [2]

In one type of ML which is called Supervised Learning, already existing inputs and outputs, which also are called features and labels, are used to develop the algorithm. In another type of ML which is Unsupervised Learning, no outputs (labels) are present and the ML model tries to extract the patterns out of data. In ML the quantity and quality of data used to train the model are critical for the performance of the model. Kumar et al. [3] ‘s figure demonstrates the taxonomy of common ML types as shown in Figure 2.2.

Figure 2.2: Machine Learning Taxonomy



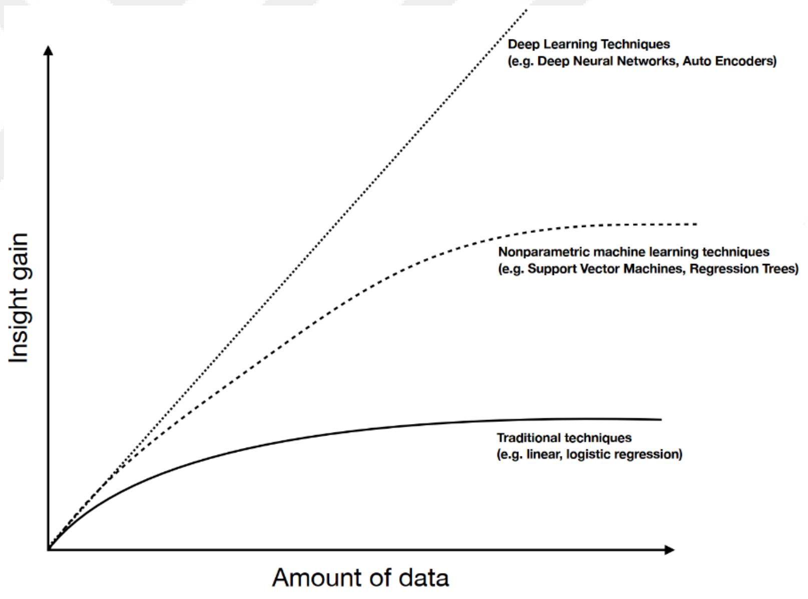
Kumar et al. [3]

2.3 DEEP LEARNING (DL)

An Artificial Neural Network (ANN) is one of the many Machine Learning methods. It is also called as Neural Networks (NN). It is inspired by the interconnected nature of neurons of human brains. An ANN is a collection of neurons connected to each other that can transmit signals. A unit transmits a signal processes it and sends the signal to another connected unit.

Deep Learning is an extension of ANNs which use very large networks. Traditional ML models' performance saturates at a point such that increasing the data quantity does not help anymore. In DL it is possible to increase the performance of the model as the quantity of the data increases. Hain et al. [4]'s work shows the relationship between data and performance of ML and DL techniques as seen in Figure 2.3.

Figure 2.3: Performance and Amount of Data



Hain et al. [4]

2.4 NATURAL LANGUAGE PROCESSING (NLP)

Natural Language Processing (NLP) is a branch of Artificial Intelligence trying to construct systems to understand human language which is in text or sound wave formats, manipulate it and respond in text or sound format like humans communicate.

NLP can be classified as a study area of Computer Science, Linguistics, and Artificial Intelligence. NLP uses a wide range of techniques including computational linguistics, statistics, machine learning, and deep learning.

Linguistics is the scientific study area of language which approaches the language in means of phonetics, phonology, morphology, syntax, semantics, and pragmatics [5]. Phonetics is the study of speech and sound in a physical aspect while Phonology studies sound in a mental aspect. Phonemes are the smallest pieces of sounds in a language. Morphology is the study of words. Morphemes are the smallest unit of words that has a meaning while lexemes are all words related to each other in meaning (like walk, walking, walked). Syntax studies phrases and sentences to define the ruleset to construct sentences from words. To represent the syntax of sentences parse tree approach can be used. While semantics studies the meaning of a word, phrase, or text, Pragmatics is the study of meaning in language focusing on the context.

The first studies over language were performed by linguists. They tried to define the rules and tried to model the languages, and they achieved continual success for more than 50 years. With the development of computers, the term ‘computational linguistics’ started to be used widely in the area of linguistics.

Later after the 1990s, with the growth of data and computer power, data-driven methods became possible. Statistical approaches started to be used instead of classical rule-based approaches with success. Still to this day, statistical methods are widely used in NLP.

After the 2010s Artificial Neural Networks became more popular in the field of NLP. One major advantage of deep learning techniques over statistical methods is feature engineering is less critical in deep learning methods.

Today NLP has also an important place in computer science. The need for interaction between humans and computers forced this area to grow and develop quickly. With the dizzying technological developments both in hardware and software means, NLP gains power and success. Today many previous challenging problems have successful solutions with the help of this rapid development.

2.4.1 Challenges in NLP

If we stop for a moment and think about how humans use the language, we realize that it is so natural and easy to speak for us forming words from sounds, words followed by words, forming sentences, absorbing meaning, emotions in a way like without thinking, and so smoothly. Even a child can express herself/himself easily. How about listening, we hear, decipher and understand the voices without any effort. Reading is one other hard task. To see the letters and digits, form words from letters, form sentences from words, and grasp the meaning. These are all kinds of processing tasks performed by humans casually and seamlessly like with near-zero effort. In reality, there are computational efforts performed by the brain and we realize this when the task is computationally difficult. Like trying to communicate in another language that we are not fluent in, or like reading a heavy text like a scientific paper. Reading the text and understanding the words is one thing, understanding the meaning is a much harder task. But in daily usage of language, the common tasks are so simple that we as humans can handle them very easily and quickly like real-time background jobs. Similar tasks must be accomplished by computers for interaction and communication with humans. NLP tries to solve these types of problems. But most of these tasks are too complicated and not easy for computers. NLP has many challenges which can be mainly grouped as syntactic and semantic problems.

2.4.1.1 Representation of Human Language

There are no easy rules to define a language. It is complicated and full of exceptions. There is no single human language on earth and every language has its own properties, representations, and structures. So a method used for a particular language may not be applicable to another language with equal success. Computers understand numbers and the input data are represented as numbers for the use of algorithms and programs. But how to represent a language, words, characters, sentences? More than that how to represent the meaning of a word or a sentence? A word may have different meanings according to the context. How to process idioms, proverbs? How about emotions, sarcasm, or irony? All these are challenges about representing the language.

2.4.1.2 Changing Nature of Language with Environment

Human language is not static, it grows, changes, new words come by, new forms of expressions appear, usage and style constantly evolve. There is no direct mapping between languages and languages also have variations according to the environment. A formal language is very different from a literature language or a language used in social media or any other internet environment. It is so that, the language used differs even in between to different social media platforms.

2.4.1.3 Uncertainty of Meaning

There are several situations that make it difficult to grasp the meaning of a text. Some cases are,

- A minor word change or a change of word order in a sentence may change the meaning completely.
- The use of language does not have standard procedures and humans do not follow strict rules.
- Different styles, dialects, inaccurate word selection.
- Usage of proverbs, idioms, humor, sarcasm, irony, symbolic or emotional elements, metaphors.

2.4.1.4 Common Knowledge

Humans have some knowledge that are assumed to be known by every human by default. So that they do not need to be learned or explained explicitly. Like ‘fish swims’ or ‘bird flies’. To tell computers all the common knowledge that they are lack of is another challenge.

2.4.2 Common NLP Tasks

NLP has many applications in action today. Some are visible, some are invisible to us like everything is happening behind the scenes. We knowingly or unknowingly use the results and products of NLP technologies. Common NLP tasks are;

- Speech-to-text (Speech recognition) and text-to-speech: Converting voice data into text form and text data into human voice. Examples are Amazon Alexa, Apple Siri, Microsoft Cortana.
- Classification Tasks: Classifying reviews, emails, tweets, or images like spam filtering, sentiment analysis, or benign/malignant tumor classification tasks.
- Optical Character Recognition (OCR) Tasks: Extracting text data from handwritten or hard-printed text.
- Text Segmentation: The process of dividing the text into smaller units like sentences (Sentence Boundary Disambiguation), words, or topics (Topic detection).
- Morphological Analysis: Techniques including tokenization, normalization, stemming, and lemmatization to process language data are usually used as a pre-stage before other NLP tasks.
- Part-of-Speech Tagging: Determining the use case of a word in a text based on the context.
- Syntactic Analysis: Parsing and analyzing the language using formal grammar rules.
- Named Entity Recognition: Identifying the words or phrases as entities. Like ‘Ankara’ as a city and, ‘cat’ as an animal in a text like ‘I live in Ankara. I have a cat’
- Word Sense Disambiguation: Determining the meaning of a word that has different meanings according to the context.
- Text Summarization: Generating a smaller text from a large text while capturing the meaning.
- Natural Language Generation: Generating human language from information like a context, idea, style, or any other entity.
- Recommendation Systems: Predicting user preferences to suggest relevant items. Nearly all of the well-known companies working with a business-to-customer (B2C) model like e-commerce companies and streaming companies like Amazon, eBay, Netflix, and Spotify use recommendation engines to predict the ratings and preferences of users.
- Chatbots and Dialogue Systems: Interacting and carrying forward conversations with humans and responding to user requests. Widely used as first-level call center

agents and digital assistants like Apple's Siri, in Google Assistant, and Amazon's Alexa.

- Word and Grammar Error Correction: Auto-correction or correction suggestion capabilities of document editors and more advanced grammatical correction services like Grammarly are possible with error correction systems.
- Machine Translation: Automatic translation from a language into another language. The examples are well-known public translation services from Google, Microsoft, and Amazon.

2.4.3 NLP Methods

Current NLP methods are mainly classified under 3 categories;

- Rule-Based Methods
- Statistical/Probabilistic Methods
- Machine Learning and Deep Learning Methods

While some tasks like spell checking, synonym suggestion, or keyword-search are possible with rule-based methods or mathematical calculations, some tasks like classifying documents, extracting information, predicting sentiments, or predicting the most likely word while making queries are possible with ML and DL methods. Even more complicated tasks like machine translation and chatbots are solved by more complicated DL models like CNNs, RNNs, and LSTMs.

It is possible to solve a task with different approaches. A method is preferred according to aspects of task complexity, performance expectations, and data quantity and quality. A chatbot may be developed by a rule-based model. But the task is so complicated that it is very hard to implement the task and the performance of the model will be low. On the other hand, for a simple spam classification task, statistical methods may give reasonably good performance near to ML or DL models. As a general rule, simple tasks may be solved with simpler models without high performance expectations. Similarly, with a small amount of data, simpler models should be used. Complicated or big models are useful when there is comparably big data.

2.5 DATA PREPROCESSING

Generally, language data is not useable directly as input for an NLP task without preprocessing it. The language text is not a standard text following strict rules. Data usually is not in a clean and standard format and it is pre-processed before use. Related steps are also called data cleaning or data normalization. Text data includes punctuations, it is in different letter case forms, there are abbreviations, emoji, emoticons and there are also occurrences of the same word in many different forms. There are some words called stop-words that occurred frequently in the text usually do not have much impact on the meaning. All these issues should be cared to increase the performance of the model. There is not any strict single correct method to preprocess the text. For each task, data type and quantity, or the complexity of the model, the pre-processing steps may differ. For example, for a sentiment analysis task, the emoji or emoticons should not be cleaned directly and the information should be kept in the text.

Each pre-processing task results in a loss of information. If the data is small and simple, preprocessing tasks will improve the performance but as a general rule, the preprocessing steps and the quantity of data are inversely proportional. More the quantity of data, less keen the preprocessing steps should be applied. One other issue is, heavy and more processing steps require more computing power and operations would be more time-consuming. More the size of the data, and heavier the processing tasks, more processing power and time are required. So there is a trade-off between the performance gain and processing time and computing power used.

2.5.1 Case Normalization

'Book', 'BOOK', 'book', 'BOok' or any other combinations of a word are all different words for the computer. By converting all text to lowercase (or uppercase), each different combination of a word becomes the same word. There may be rare occasions that case normalization process is avoided. Upper case letters may be abbreviations, resemble special meanings, or such forms may resemble emotions like frustration. These types of information are lost after case normalization.

2.5.2 Stop-Words Removal

All languages include words that are used frequently and most of the time stop words do not contribute to the meaning of the text. But because these words are frequent, they may have a higher weight in the input space that may mislead the models. The words in English like ‘he’, ‘she’, ‘our’, ‘that’, ‘in’, ‘up’, ‘to’, ‘the’, ‘and’, ‘but’ considered as stop-words. If the NLP task is a sentiment-analysis task, only a delicate stop-words removal process may help to increase the performance, otherwise, it may decrease the model performance. For example, removing the stop-word ‘not’ with all other stop-words, the text may lose the very important negative sentiment from the text. ‘I am not happy’ becomes ‘I am happy’ after removing ‘not’, which resembles the opposite sentiment. In more complex tasks like machine translation or chatbots, stop-words removal may not be helpful at any means. Each word resembles or refers to an entity, like ‘Janet plays volleyball, she is good at it’. After a strict stop-words removal, the reference information of ‘she’ to ‘Janet’, and ‘it’ to ‘volleyball’ are all lost for the model. Removing high frequency or low frequency words is a similar technique to stop-words removal that can be applied to normalize the text.

2.5.3 Removing Punctuations

Punctuations like ",-.!+:;?^$%_’”#&'()*/@[\]{|}" are heavily used in written texts. The usage of punctuations helps the grammar and the meaning but this increases the input space. If there is a need to decrease the input space for a less complex model and a task, the punctuations may be removed while some of them may be held purposefully in text like ‘!’, ‘?’, or the characters resembling emoticons or currency characters like ‘$’, ‘€’ or ‘₺’ to keep the relevant information.

2.5.4 Stemming

Stemming is the process of truncating words into their stems. In this way, all different forms of words transformed to their original stem. The words, ‘fly’, ‘flying’, ‘flied’ are all transformed to their stem which is ‘fli’. The disadvantage of this process is losing tense, grammar, and part of speech information.

2.5.5 Lemmatization

Lemmatization is a similar method to stemming but is more sophisticated and as a result, a more time-consuming approach. It requires the use of reference vocabulary. The words, ‘fly’, ‘flying’, ‘flied’ are all transformed to their lemma which is ‘fly’. Similar results are achieved from both stemming and lemmatization methods. The choice is made according to the nature of the data and the type of the NLP task.

2.5.6 Tokenization

Tokenization is the task of splitting texts into smaller pieces, commonly sentences into words. So that each word is an input for the NLP model. There are also character-based models, in this case, the text is split into the character level. Characters or words form a meaningful sentence when they are in a particular order. When the text is tokenized the original order of words may be preserved. If it is a complicated task like machine translation, the order is important. But for a simpler sentiment analysis task, even unordered tokens with simpler models may give decent performances.

2.5.7 Handling N-Grams

Some words are commonly used together to form a special meaning or a phrase like ‘New York City’, ‘United Kingdom’ or ‘Prime Minister’. Such words can be treated separately or these words can be concatenated to be treated as a single word. Bi-gram is a length of two words, tri-gram is a length of three words.

2.5.8 Further Normalizations

More preprocessing steps may be helpful according to the text used. Abbreviations like ‘Mr.’, ‘Mrs.’ or ‘FYI’ which is used for ‘For Your Information’ and ‘ASAP’ which is used for ‘As Soon as Possible’, numerical data, different usages of languages, and special uses of characters or character groups like URLs and Html Tags should be inspected, considered and processed accordingly.

All, some or none of the data processing tasks may be used, depending on the dataset and problem. Stemming, lemmatization, and lowercasing are helpful for small datasets but unhelpful for large datasets. Different meanings can be captured from different forms of words in a large corpus. Emotions can be captured with the help of punctuations. Negations may be important and this may be lost with removing stop-words. As in their work, Maas et al. [6] did not apply the stop word removal step not to lose negative sentiment indicating words, did not apply stemming, and did not remove some of the punctuations like ‘!’ and ‘:-)’ because of their sentiment indicating nature.

2.6 REPRESENTATION OF LANGUAGE

After necessary pre-processing steps, the text data is transformed and ready for the use of language models. But one more task should be accomplished before use, which is representing the data. The computers understand the numeric data, but the language is in the text form. The data cannot be used as it is and should be converted into a numeric format. There are several methods for representing the language. The first step is representing the tokens, and the second step is representing the sentences or sentence groups using token representations.

2.6.1 One-Hot Representation of Words (One-Hot-Encoding)

The first method is constructing a vocabulary from all the tokens in the corpus (the collection of all the documents), sorting and representing each word with its index value. For an example small corpus with two sentences; ‘The cat runs’ and ‘The dog and the cat sleep’. The vocabulary, which is built by all the unique words in the corpus is [‘and’, ‘cat’, ‘dog’, ‘runs’, ‘sleep’, ‘the’], and the words in the corpus may be represented numerically as their index values which are 0:’and’, 1:’cat’, 2:’dog’, 3:’runs’, 4:’sleep’, 5:’the’. This easy method has a problem. The index values have a greater-smaller relationship in between each other, which mislead the models. The inputs must be represented independently so that any unwanted relation should not be implied to the model.

These types of inputs are also called categorical inputs in many other machine learning problems. If we take a simple linear regression problem of predicting house prices as an example, the inputs such as the number of rooms and size of the house are among numeric features, and the inputs like the location, city, or district are among the categorical features. If the categorical features are treated just like the numerical features, the model will make wrong inferences. For representing the categorical features, the simplest method is one-hot-encoding.

The same approach can also be applied to NLP tasks. In one-hot-encoding, a word is represented with a vector of length equal to the vocabulary size, with all zeros and a single one at the index position same as the word's index in the vocabulary. These types of vectors representations are called one-hot-vectors as shown in Table 2.1.

Table 2.1: One-Hot Representations of Words

Corpus: [‘the cat runs’, ‘the dog and the cat sleep’]

Vocabulary: {0:’and’, 1:’cat’, 2:’dog’, 3:’runs’, 4:’sleep’, 5:’the’}

Vocabulary \ Index	0	1	2	3	4	5	One-Hot-Vectors
‘and’	1	0	0	0	0	0	**[1, 0, 0, 0, 0, 0]**
‘cat’	0	1	0	0	0	0	**[0, 1, 0, 0, 0, 0]**
‘dog’	0	0	1	0	0	0	**[0, 0, 1, 0, 0, 0]**
‘runs’	0	0	0	1	0	0	**[0, 0, 0, 1, 0, 0]**
‘sleep’	0	0	0	0	1	0	**[0, 0, 0, 0, 1, 0]**
‘the’	0	0	0	0	0	1	**[0, 0, 0, 0, 0, 1]**

2.6.2 Bag of Words Model (BOW)

This representation is also called count vectorization. The simplest form of representing collections of words or tokens, which are also called as documents or sentences, is the Bag-of-Words model (BOW). In BOW, the first document is split into words, and then converted into one-hot-vectors and finally, all words are summed up to get the vector representation of the document. It is called as bag of words because the word ordering or word's contribution to the meaning are ignored, and all words are treated independently like throwing them into a bag and counting at the end. So the BOW representations of the two sentences in the previous example are as in Table 2.2.

Table 2.2: BOW Representations of Documents

	BOW Representations
‘the cat runs’	**[0, 1, 0, 1, 0, 1]**
‘the dog and the cat sleep’	**[1, 1, 1, 0, 1, 2]**

BOW representation has two major problems. The first one is because no order is preserved, there is a loss in meaning. So while the model is effective for simple tasks and simple data with decent performance, this representation is not sufficient for complex tasks. The second problem is for a large vocabulary; the vector representations of documents will consist of large number of zeros which need too much storage space. These types of vectors are called sparse vectors. Programming language libraries use memory-efficient methods to handle the sparse vectors and sparse matrices to save space.

2.6.3 Term Frequency-Inverse Term Frequency (TF-IDF)

TF-IDF is a method widely used in information retrieval and text mining areas. This statistical method is used to determine the importance of a word for a document and for an entire corpus. If a word appears more frequently in a document, the word becomes more important. However, it becomes less important if it appears in more documents in the corpus. This method is used for scoring content in search engines and to match the query with the most relevant content.

The intuition behind TF-IDF is, some common words like ‘the’, ‘and’, ‘I’ appear frequently in most of the documents in a corpus and these types of common words are less distinctive between documents, so they should be represented less heavily. On the other hand, rare words like ‘football’, ‘dog’, ‘coffee’ across an entire corpus are more distinctive and these should be represented more heavily if they appear in specific documents more frequently than others.

TF-IDF method can be interpreted as an enhanced version of BOW. In BOW, frequent words in a document are represented more heavily according to their count. But the word vectors can be represented better with the TF-IDF method. TF-IDF is formed by two sub-components which are Term Frequency (TF) and Inverse Document Frequency (IDF).

Term Frequency (TF) is the count of a term in a document divided by the total number of words in that document just like the normalized version of the count vectorization/BOW model. Normalization is required because every document has a different length and some words may appear more times than the other words in long documents. For a specific document TF is calculated as,

$$\mathrm{Tf(t,d)}=\frac{\text{Term t count}}{\text{Total term count}} \tag{2.1}$$

Inverse Document Frequency (IDF) decreases the weight of terms that occur very frequently in all documents and increases the weight of terms that occur rarely. It is calculated by applying the logarithm function to the total number of documents divided by the count of documents in which the term appears. IDF is calculated as,

$$\text{Idf}(t,D)=\log\left(\frac{N}{\text{df}(t)}\right) \tag{2.2}$$

Where, N is the total number of documents, and df(t) is the count of documents containing the term across all the corpus. TF-IDF becomes as,

$$\text{Tf}(t,d).\log\left(\frac{N}{\text{df}(t)}\right) \tag{2.3}$$

where,

- Tf(t,d): number of occurrences of term t in document d,
- df(t): number of documents containing term t,
- N: total number of documents.

Using TfidfVectorizer class from the scikit-learn library, a simple python code as given in Table 2.3, helps us to calculate TF-IDF vector representations of the same example.

Table 2.3: Code for TF-IDF Vector Representations

```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = [‘the cat runs’, ‘the dog and the cat sleep’]
tf = TfidfVectorizer()
tfidf = tf.fit_transform(corpus)
print(tfidf.toarray())
```

The output of the given code is the vector representations of two sentences as shown in Table 2.4.

Table 2.4: TF-IDF Representations of Documents

	‘and’	‘cat’	‘dog’	‘runs’	‘sleep’	‘the’
‘the cat runs’	0.	0.50	0.	0.70	0.	0.50
‘the dog and the cat sleep’	0.43	0.30	0.43	0.	0.43	0.60

As seen, in the second sentence, the word ‘the’ appears more than others so has a larger weight. The word ‘cat’ has less weight than others because it appears in two documents while ‘and’, ‘dog’ and ‘sleep’ only appear in one document. In the first sentence ‘runs’ has a higher weight because it appears in only one document and ‘cat’ and ‘the’ have less weight because they appear in two documents. In TF-IDF, words weights are represented more precisely compared to BOW, but the order of words in a document is also ignored as in BOW, so this model also suffers from the problem of misrepresentation of the meaning.

2.6.4 Word Embeddings

Word Embeddings are also called Word Vectors or Vector-Space Embeddings. There are cases that two sentences having almost exact same words except for a minor difference, which may have different meanings. Like: ‘I am going to come’ and ‘I am going to go’. Or two sentences may have similar meanings while their almost every word are different. ‘Never mind’, ‘No harm done’, ‘That's all right’, ‘It's okay’, ‘Forget about it’, ‘No worries’, ‘No problem’ are all sentences for accepting apologies with different words but in similar meaning.

BOW and TF-IDF models may be ineffective to deal with these types of complex situations. Alternative methods should be used to represent the words with their meanings. This can be accomplished by examining the word groups which are used frequently together. As in a well-known saying: ‘Tell me who your friends are, and I will tell you who you are’, the same idea may be applied to derive relations between words by examining their co-occurrence in the corpus. Firth, J.R. (1957:11) approves this idea with his famous saying; ‘You shall know a word by the company it keeps’. The process of representing words with their relative meanings to other words can be achieved through using statistical or ML approaches, and these types of words representations are called word embeddings.

For intuition, an imaginary example is shown in Table 2.5. Assuming that the columns are embeddings of several properties, the word embeddings are manually created as word vectors of length 6. ‘Kitten’ is represented as the vector [0.96 0.93 0.87 0.02 0.12 0.95], and as seen, it is strongly represented to have the properties of Baby, Animal, Pet, and Four-Legs. ‘Cat’ is similar to ‘Kitten’ with one difference that, it is not strongly represented as Baby. On the other hand, words 'Ankara' and 'Turkey' have very light representations in embeddings related to the animal properties. And they are strongly represented with the embeddings of Place and Big.

Table 2.5: Word-Embeddings Example

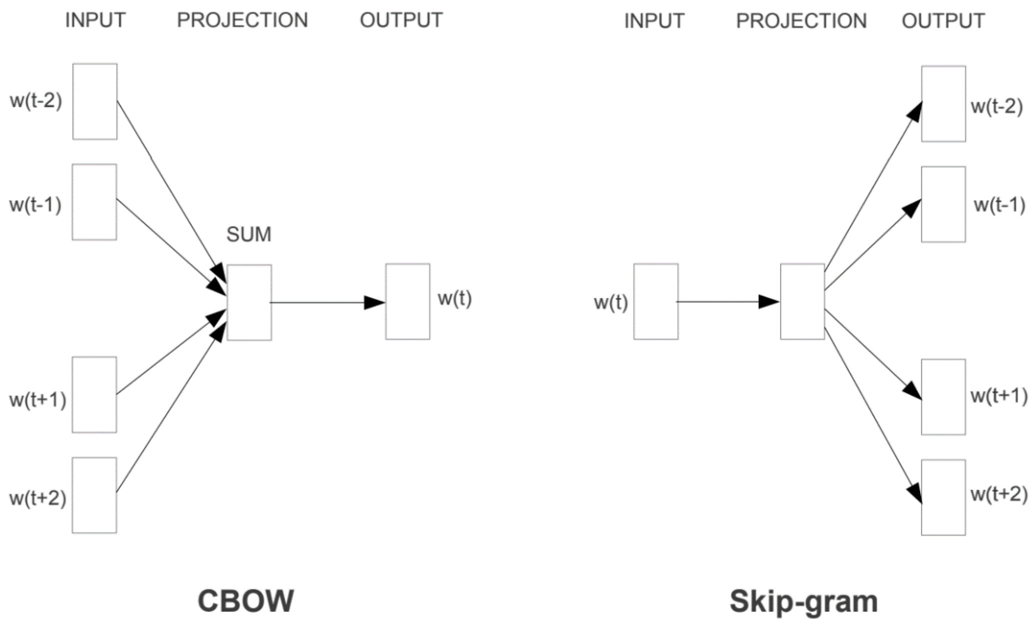
Vocabulary\Embeddings	Baby	Anima	Pet	Place	Big	Four-Legs
Kitten	**0.96**	**0.93**	**0.87**	0.02	0.12	**0.95**
Cat	0.45	**0.95**	**0.79**	0.01	0.54	**0.92**
Dog	0.56	**0.93**	**0.89**	0.05	0.65	**0.89**
Ankara	0.19	0.09	0.02	**0.85**	**0.79**	0.08
Turkey	0.14	0.04	0.01	**0.78**	**0.85**	0.12
Elephant	0.56	**0.94**	0.24	0.05	**0.96**	**0.87**

Creating word embeddings is not a manual process as in the example. Statistical or ML models are used to derive word relations automatically as word-embeddings which capture information about word relations from context. Compared to sparse vectors in one-hot-representation of words, word embeddings are dense representations. The lengths of the word embedding vectors are not as big as the vocabulary size. When training word embeddings, to represent all the words successfully, as one of the many hyperparameters, the size of the embedding vector is selected between 50 and 300 according to the size of the corpus. It is possible to capture more details using more data and bigger word-embedding sizes, but higher values need higher processing power. There are three widely used techniques for word embeddings which are, Word2vec, GloVe, and FastText. The training of word embeddings for a large corpus is a computationally expensive and time-consuming process. There are various pre-trained models that can be downloaded and used directly. Pre-trained word embeddings are useful for general purpose tasks, but self-trained word-embeddings give better results for special corpus types.

2.6.4.1 Word2vec

Since word2vec was proposed by Mikolov et al. [7] in 2013, it became very popular and widely used for its efficiency in representing words in vector space. There are two underlying model architectures for Word2vec as shown in Figure 2.4. In Continuous Bag of Words (CBOW), the model is trained to predict the target word from its surrounding (context) words. In Skip-Gram (SG), it is the opposite; the model is trained to predict the surrounding (context) words from a given target word. While CBOW model implies ‘Tell me who your friends are, and I will tell you who you are’, SG model implies ‘Tell me who you are, and I will tell you who your friends are’. For context words, a window size of ‘n’ is determined and, ‘n’ words before and ‘n’ words after are considered as context words.

Figure 2.4: CBOW and SG Architectures



Mikolov et al. [7]

If the syntax was cared for, the order of the words should be important, but the purpose is to represent a word with its meaning. So the context words can be treated like a bag-of-words model. Both Word2vec architectures produce similar results with slight differences. For a small corpus, SG performs better, and rare words can be represented well. CBOW is much more computationally efficient than SG, and so is better for a large corpus. Relatively to SG, CBOW represents frequent words better.

2.6.4.2 GloVe

Global Vectors for Word Representation (Glove) [8] is developed by Stanford in 2014. GloVe method takes into account the word co-occurrence statistics over the corpus. It is based on both the matrix factorization technique and the local context window.

2.6.4.3 FastText

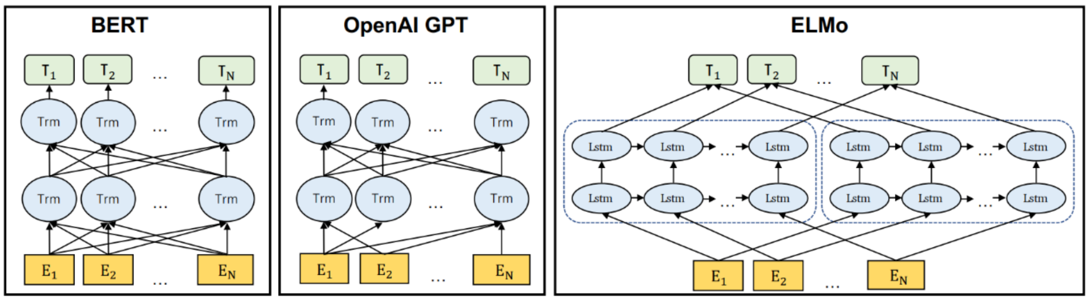
FastText [9] is created by Facebook in 2016. It is based on the skip-gram approach of the word2vec model. It operates on a sub-word level and supports words that do not exist in the vocabulary. Each word in the corpus is split into n-grams of characters and treated as bag-of-characters. A vector of an unknown word can be constructed with its n-gram vector representations. So, problems about infrequent words and out-of-vocabulary (OOV) words are addressed better. It is also faster compared to the word2vec model.

2.6.5 Advanced Word Embedding Methods

Instead of pre-training word-embeddings, in DL networks, Embedding Layers can be used as the first hidden layer for NLP tasks. After encoding the words in documents into integers, the lengths of the documents are normalized to be all in same length, and normalized documents are given as inputs to the DL model, Embedding Layer act as successfully as other common embedding methods, learning word vectors as the model trains.

There are also large-scale pre-trained language models using advanced DL networks. These pre-trained models can be used for NLP tasks as an alternative of self-training word embeddings. ELMo (Embeddings from Language Model) was developed by Allen Institute for AI in 2018 [10]. BERT (Bidirectional Encoder Representations from Transformers) was developed by Devlin et al. [11] from Google in 2018. MASS (Masked Sequence to Sequence pre-training) was proposed in 2009 by Song et al. [12] from Microsoft for encoder-decoder based language generation. They achieved state-of-the-art accuracy with a 37.5 BLEU score. In 2020 OpenAI developed GPT-3 (Generative Pre-trained Transformer) [13]. It was trained with the ‘Common Crawl’ dataset containing 500 billion words. It has 175 billion parameters which is 100 times bigger than the preceding GPT-2 model. Huang et al. [14] compared the architectures of the three models as in Figure 2.5.

Figure 2.5: BERT, OpenAI GPT, and ELMo Architectures



Huang et al. [14]

2.6.6 Similarity of Word-Embedding Vectors

The word ‘meaning’ is commonly used for word-embeddings. In reality, it is referred to the relation between words, not the real meaning of a word. The relation is the relative meaning and similarity between words. As ‘baby’ and ‘mother’ or ‘cat’ and ‘dog’, some words are similar/close to each other because they are usually used in the same context. Word-Embeddings inherit these types of closeness/similarity relationships and word vectors of close words are close to each other in vector space. To calculate the similarity or closeness of vectors there are two methods, which are, measuring the angle between vectors or calculating the distance between vectors.

2.6.6.1 Cosine Similarity

Cosine Similarity measure is the cosine of the angle (θ) between the vectors. It can be calculated by the Euclidean dot product formula. The value can be between -1 and 1. If it is close to -1, the vectors are dissimilar and if the cosine similarity value is close to 1, the vectors are close to each other and accepted as similar. Cosine similarity is generally used when the magnitudes of the vectors are not important.

$$\cos(\theta) = \frac{A \,.\, B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \text{ x } B_i}{\sqrt{\sum_{i=1}^{n} A_i^{\,2}} + \sqrt{\sum_{i=1}^{n} B_i^{\,2}}} \tag{2.4}$$

2.6.6.2 Euclidean Distance

Euclidean Distance is the length of a line between the two vectors. To find the distance between two points, the Euclidean distance formula is used.

$$d= \sqrt{\sum_{i=1}^{n}(A_i-B_i)^2} \tag{2.6}$$

2.6.7 Evaluation of Word-Embeddings

To evaluate the performance of the word-embedding training process, there are two methods, intrinsic evaluation and extrinsic evaluation.

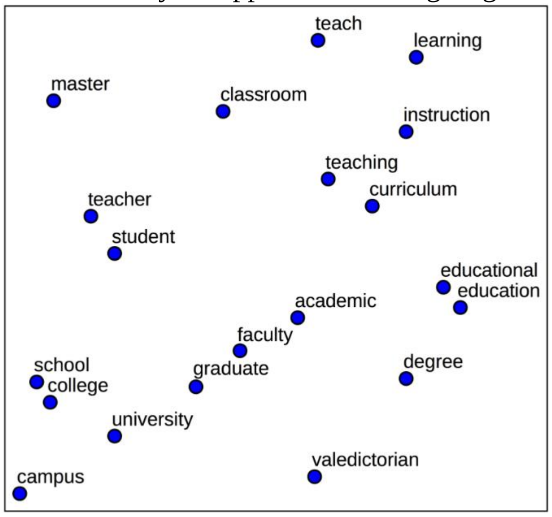
2.6.7.1 Intrinsic evaluation

Analogies can be tested to evaluate the performance of word embedding training. Like the vector operation of ‘Turkey - France + Ankara’ should give a vector very close to the vector of ‘Paris’.

Clustering is another approach that can be used for intrinsic evaluation. Similar vectors can be tested if they are in the same clusters.

Visualization methods also are helpful to demonstrate the similarity between vectors. But human brains are not capable to comprehend or visualize more than 3 dimensions. To visualize the word vectors in 2 or 3 dimensional spaces, dimensionality reduction methods like t-distributed Stochastic Neighbor Embeddings (t-SNE) and Principal Component Analysis (PCA). Butnaru et al. [15] generated 2D representation of 300-dimensional word embeddings by applying the PCA algorithm as seen in Figure 2.6.

Figure 2.6: Sentiment Analysis Approach Enabling Negation Identification



Butnaru et al. [15]

2.6.7.2 Extrinsic evaluation

Word embeddings can be tested on external tasks like sentiment analysis, named-entity-recognition, or parts-of-speech tagging and the performance is observed comparing the existing results. Extrinsic evaluation is more time-consuming and more difficult to troubleshoot compared to intrinsic evaluation.

2.7 EVALUATION OF CLASSIFICATION TASKS

Evaluation is the key step to judge the performance of any trained model. For a classification task, first, data is split into train and test sets. The model is trained with train set, and after training, test data is fed into the model, and predicted labels are compared with the actual labels of the test data.

Regardless of the model used, whether it is a rule-based model, a statistical model, an ML, or a DL model, the outputs of classification tasks are predictions of classes. While for a multi-class classification problem, there may be any number of possible classes, for a binary classification task, there may be only two possible classes (e.g., 1/0, Yes/No, Positive/Negative, Spam/Ham, Malignant/Benign, Good/Bad, Cat/Dog).

The most commonly used method for evaluating the performance of classification tasks is the Confusion Matrix (Table 2.6). After training the model with the train data, Confusion Matrix shows how well the model performs running the model with the test data. There are four possible values in the matrix for a two class (logistic) classification task.

Table 2.6: Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

where,

- TP: True Positive

- TN: True Negative
- FP: False Positive (Type I error)
- FN: False Negative (Type II error)

With the test data, the model predicts the class as Positive or Negative. If the prediction matches the actual class then it is a True prediction, else it is a False prediction. Correct predictions are predicting Positive and Negative classes 'True'ly (TP + TN). Incorrect predictions are predicting classes 'False'ly. (FP + FN). True predictions should be high for the model to perform well. There are several metrics derived from the Confusion Matrix to calculate the performance. Each metric has its own use cases and weaknesses according to the task and the dataset.

Accuracy is the basic metric that shows the ratio of true (correct) predictions to all predictions.

$$\text{Accuracy} = \frac{(\text{TP+TN})}{(\text{TP+TN+FP+FN})} \tag{2.8}$$

Accuracy is not a reliable metric when negative and positive classes are unbalanced in the dataset. For such cases, other metrics are used to evaluate the performance.

Precision is the ratio of correctly predicted positive observations (TP) to the total number of positive predictions (TP + FP).

$$\text{Precision} = \frac{\text{TP}}{(\text{TP+FP})} \tag{2.10}$$

Recall (Sensitivity) is the ratio of correctly predicted positive observations (TP) to the total number of actual positives. (TP + FN).

$$\text{Recall} = \frac{\text{TP}}{(\text{TP+FN})} \tag{2.12}$$

F1-Score (F-Measure) is the weighted average of Precision and Recall.

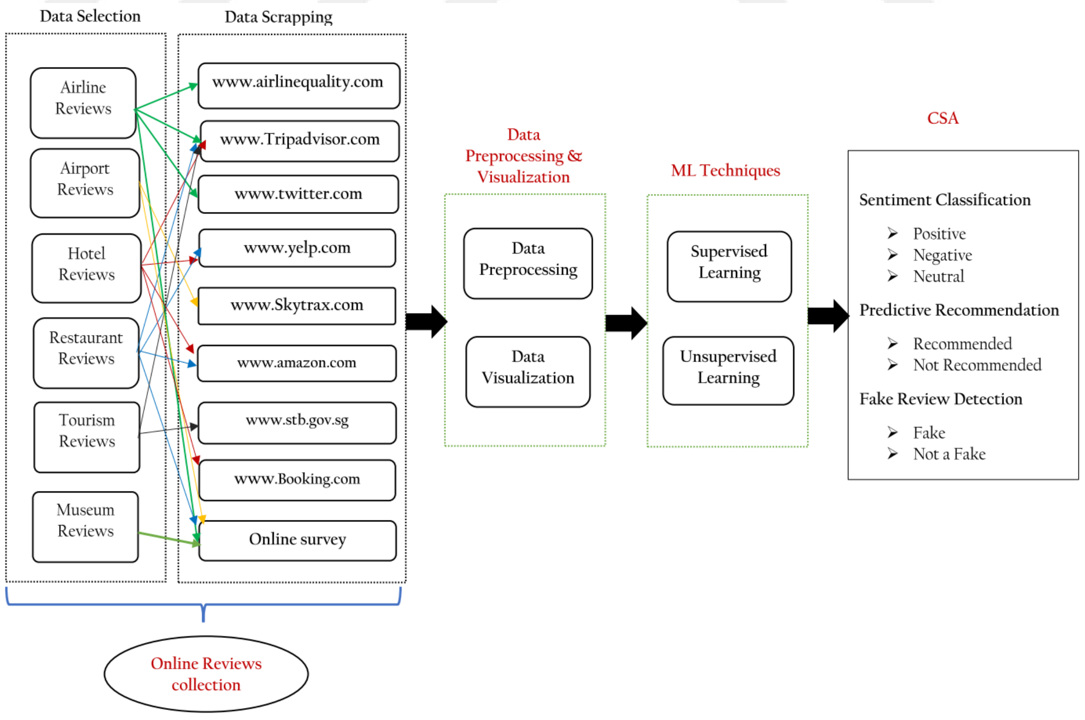
$$\text{F1-Score} = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} \tag{2.14}$$

2.8 LITERATURE REVIEW

Zhang et al. [16] used online reviews of air purifiers from t-mall.com for customer preferences extraction using aspect-level sentiment analysis. They used the Kano model to extract consumer demands from sentiment orientation. To identify multi-word product attributes they used the part-of-speech method and used k-means and word-embedding techniques to group synonyms of the product attributes.

Jain et al. [17] conducted a literature review on customer sentiment analysis in the tourism domain using ML techniques including Clustering, SVM, DT, LR, LSTM, BiLSTM, KNN, NB, Regression, CNN, RF, LDA, GB, and NN. The literature review includes 68 research papers from Jan 2017 to July 2020. They proposed a sentiment analysis with ML framework (Figure 2.7) with four phases which are, online reviews collection, data pre-processing and visualizations, ML techniques, and customer sentiment analysis phases.

Figure 2.7: Customer Sentiment Analysis Flowchart

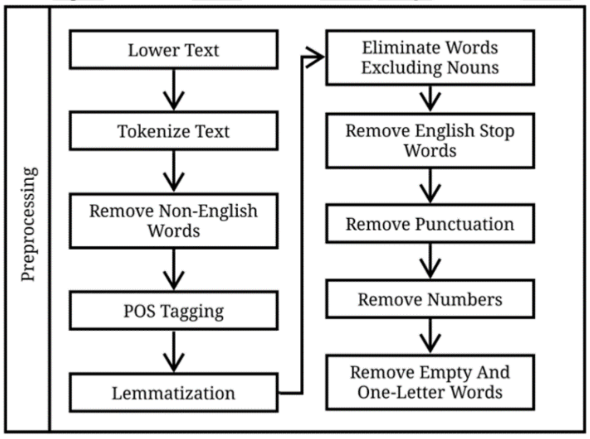


Jain et al. [17]

Rathor et. al [18] extracted Amazon online reviews with Amazon API and used both unigrams and weighted unigrams to train 3 supervised ML techniques which are SVM, NB, and ME for the sentiment classification task. It is a multi-class classifier, with three classes which are, positive, neutral, and negative. maximum accuracy score with weighted unigrams is achieved by SVM method with 81.20% while the accuracy of Maximum Entropy (ME) is 70.35% and Naive Bayes (NB) is 77.42%

Cataltas et al. [19] first clustered customer reviews with the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm to find common defects of a product. Then defects are used to find opinion words using Part-of-Speech patterns. They achieved to find the product defects using commonly reviews product features with negative opinion tendency. They argue that the proposed method may guide customers while making purchase decisions and also producers to improve their products. They also demonstrate a flowchart of the data pre-processing steps that they performed in their study as shown in Figure 2.8.

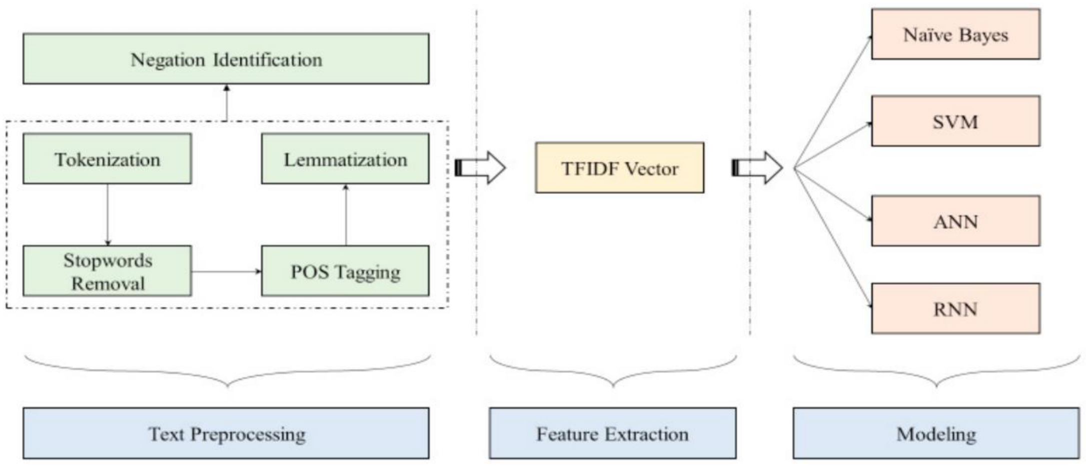
Figure 2.8: Data Pre-Processing Flowchart



Cataltas et al. [19]

Mukherjee et al. [20] worked on the sentiment analysis problem with including the negation (Figure 2.9). They trained Naive Bayes, Support Vector Machines, Artificial Neural Network (ANN), and Recurrent Neural Network (RNN) models on cell phone reviews data from Amazon. They prove that handling negations improve the performance of the models. DL models give the highest accuracy scores, where ANN with negation has an accuracy of 96.32% and RNN with 95.97%.

Figure 2.9: Sentiment Analysis Approach Enabling Negation Identification



Mukherjee et al. [20]

Zhang et al. [21] proposed a recurrent attention LSTM model for document-level sentiment analysis that is capable of locating regions more accurately about sentiments and demonstrated that the proposed method achieved better results than the state-of-the-art models.

Dey et al. [22] analyze the sentiments of Amazon product reviews by using two ML algorithms which are linear SVM and NB and compares the performance of the models. SVM model achieves better results which are, f1-score of 84% and accuracy of 84% for SVM, and f1-score of 82.67%, accuracy 82.88% for Naive Bayes.

Fry et al. [23] studied two unsupervised clustering algorithms which are K-means and Peak-searching, using Amazon product review data for a Samsung Galaxy smartphone. They used manually determined topics and they compared the results. K-means performed slightly better but neither of the models helped to achieve their goal.

Katić et al. [24] used Amazon review dataset to study sentiment analysis task to compare several document representation methods like bag-of-words, bag-of-ngrams, their tf-idf versions, pre-trained word embeddings; word2vec and GloVe, and several learning models like Logistic Regression, SVM, ConvNets, and LSTM. They concluded that DL models perform better than traditional models as the dataset size increases as expected. DL models need more data to generalize well. They proved that deep learning models perform better than traditional models on the large dataset. LSTM resulted in the best accuracy of 95.56%. Among traditional models, Linear SVM with bag-of-ngrams and TF-IDF has the highest accuracy score with 92.9%.

Dogan et al. [25] used Twitter data to detect influencers or possible speculators to evaluate their effect on stock markets. They focused on the largest companies on the NASDAQ stock exchange market. They used sentiment analysis dictionaries from Loughran and McDonald and trained several ML models to identify possible speculators or influencers who have too many positive or negative effects. They found that it was not possible to find a correlation of individual tweets with the stock market without noise reduction, and they achieved better results with RBF Kernel and SVM methods.

Mutlu et al. [26] used Twitter data to identify troll accounts using three supervised machine learning algorithms, which are KNN, NB, and C4.5 Decision Tree. They manually examined collected tweets from 2605 accounts and set rules like sending more than 50 tweets/day, follower/following rate less or equal to 0.4, retweet rate of 70%, and profile images to decide if a user is a troll or not. They got the best result with the C4.5 model with an accuracy score of 89%

Rao et al. [27] studied on detection of sarcasm on Amazon reviews using SVM, KNN, and Random Forest algorithms. The study uses tokenization, polarity identification, stemming, and lemmatization steps for feature extraction. They reached accuracy levels of 67.58% with SVM, 62.34% with Random Forest, and 61.08% with KNN models.

Uysal et al. [28] conducted a sentiment analysis task using tweets containing political keywords which are collected just before the Turkish General Elections held in 2015. They classified 93,653 tweets, and first created an emotion-specific dictionary in Turkish containing 1543 emotion words and 120 emoticons. They used Zemberek, which is a Turkish NLP framework, for word suggestions, suffix removal, and negative verbs detection, and SentiStrength library for sentiment analysis. They analyzed the polarity of tweets in three categories, which are part leaders, party names, and ideologies, and compared with the election results. They got better results with leaders and party names categorizations.

Ejaz et. [29] al proposed a lexicon dictionary-based approach with ngrams for sentiment analysis on Amazon product reviews and compared their model with three ML models which are, Random Forest with word vector, Decision Tree learner with document vector, and Random Forest with n-gram. The accuracy of each model is calculated by using the ROC curve and their proposed model outperformed all with an accuracy score of %89.5 while the next best score is %85 with Random Forest with n-grams model.

Haque et al. [30] compared CNN, LSTM, and LSTM-CNN architectures for sentiment analysis tasks using IMDB movie reviews. They reached the F-Score results of 91% for CNN, 86% for LSTM, and 88% for LSTM-CNN.

Almjawel et al. [31] worked on visualization of sentiments using Amazon book reviews. They presented four techniques, Interactive Packed bubbles, Linear chart, Stacked bars, and Word-cloud that enable users to explore relationships between different objectives.

Saranya et al. [32] proposed word cloud and word plot generation to find the most frequently used words in the YELP clothing reviews dataset. Latent Dirichlet Allocation (LDA) is used as a probabilistic generative model to generate topics from words in the corpus.

CHAPTER III

PROPOSED METHOD

3.1 PROPOSED SOLUTION

Pre-trained word-embedding models like Word2Vec and GloVe are available for public use, trained with big datasets in strong computing environments. It is not possible to accomplish similar large-scale training tasks on our local computers. Pre-trained models are trained with non-specific context data, and are very useful for a wide range of general-purpose tasks. But pre-trained models contain many unrelated words and unwanted relations between words that do not serve our purpose. For specific context and specific types of tasks like our case, training own word-embeddings should be more efficient. We propose a method of detecting 'defect' related words using our self-trained word-embedding model and using populated defect related words, to be able to classify reviews that contain information about defected products.

We train our word-embedding model using only the negative reviews as corpus, which is possible with our own modest computing environment. And after training using word vector similarities, we populate a set of words or phrases that are related with 'defect'. The set of defect words can be used later to judge if a review is complaining about a defect of a product or not.

3.2 DATASET

In this study, we use the Amazon Review Dataset by Ni et al. released in 2019 [33]. The dataset is high volume, shared, and ready to use. Full dataset includes total 233.1 million amazon reviews between May 1996 and October 2018. Raw data is 34GB in size. A subset of the full data was also prepared with the name '5-core' selecting only the reviews with users and items that have at least 5 reviews. The 5-core dataset includes 75.26 million reviews. The dataset is divided into 29 different product categories and served as per-category compressed files. We selected the Electronic category for our study. The full size of Electronic reviews consists of 20,994,353 reviews. We experimented on the 5-core subset of the same category that has 6,739,590 reviews with 1.2 GB compressed and 4 GB uncompressed size. An example review and related field information is shown in Table 3.1.

Table 3.1: An Example Review

```
{       "overall": 4.0,
        "vote": "5",
        "verified": true,
        "reviewTime": "05 18, 2000",
        "reviewerID": "A2BDENVIRDTXVP",
        "asin": "B00000JYVT",
        "reviewerName": "*****",
        "reviewText": "I use this recorder every day to make notes and record to-do items while driving, etc.  It works very well and the recording quality is decent enough for what I use it for.  There is one large design flaw, though.  The  record button is MUCH too easy to press.  At least 4 times a week, I'll try  to record a message only to find a 70+ minute recording of me walking  around with this in my pocket.  All I do is delete that message and I'm  back in business, but it's annoying.",
        "summary": "Great item with one flaw",
        "unixReviewTime": 958608000 }
```

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

3.3 METHODOLOGY

3.3.1 Feature Selection

The dataset includes many informative fields about reviews. For our case, we only used the three features that serve our purpose; ‘overall’ for the rating (sentiment) information, ‘reviewText’ and ‘summary’ for the text review heading and body.

- The text fields ‘summary’ (header) and ‘reviewText’ (body) are concatenated and obtained a single feature ‘text’ in text format.

- Reviews that have 1 and 2 ratings are filtered and labeled as negative (0). Reviews that have 5 ratings are filtered and labeled as positive (1). Reviews with 3 and 4 ratings are neglected that do not resemble strong sentiments.
- The character lengths and the word counts of the text fields are calculated, and this information is used for normalizing the text features.

3.3.2 Data Preprocessing

As the text parts of the reviews are unstructured data, basic NLP methods are applied and tested if they help the performance of the models.

- Text features are lowercased and all punctuations are cleared, so that they only contain characters [a-z] and digits [0-9].
- On the contrary, stop-words removal, stemming and lemmatization steps are not applied. The dataset is quite big, so these steps do not contribute to the performance. One other reason is; these steps are very time-consuming.
- The words with lengths smaller than 2 and larger than 20 characters are deleted.
- Too short reviews would not contain enough information to judge the review, so with reviews length less than 10-character and word count less than 3 are deleted.
- 1% of data is split for later manual tests.
- Negative and Positive reviews are not balanced. This may be misleading when evaluating the performance of the models. So they are balanced by reducing positive reviews to be 10% more of the negative reviews.
- Positive and negative reviews are split to train and test datasets with a ratio of 0.9 to 0.1

3.3.3 Testing Dataset Reliability

Before moving further, we wanted to test our dataset and confirm if the data includes enough information for a classification task and also see if labels (ratings) of the reviews are reliable or not. We trained and evaluated several ML models with several smaller sets of our cleaned and split data, observed the high accuracy and F1 scores, approved that the labels (ratings) truly represent negative and positive sentiments and the dataset includes enough information for a sentiment analysis task.

This step also helped us to observe the relationship between performance and the dataset sizes.

3.3.3.1 Data Representation

BOW and TFIDF representation methods are used to represent words for the base ML training task.

3.3.3.2 Model Selection

Several ML models (Logistic Regression, NB, and Random Forest) are trained for testing dataset reliability.

3.3.3.3 Evaluation

Since the problem is simplified to a classification task, we make use of Confusion Matrix and compare Accuracy, F1 scores to evaluate the performance of the trained models.

3.3.4 Training Word-Embeddings

Using only selected and cleaned negative reviews, the Word2Vec model is trained using python gensim library to obtain word-embeddings specific to negative sentiments. Bi-gram model is preferred instead of the unigram model to obtain also two word phrases like ‘already broken’ or ‘falling apart’. After training, defect-related words list is populated using vector similarities. Auto populated list contains some phrases that are not related to defects like ‘unique’, ‘fully functional’ or ‘innovative’ are cleared manually to prevent misleading inferences.

3.3.5 Re-labeling Dataset as Non-Defected/Defected

Non-defect/defect dataset which is a subset of positive-negative sentiment dataset is produced by;

- Negative reviews that contain any words from the defect-related words list are labeled as defected (0)
- Positive reviews that do not contain any words from the defect word list are labeled as non-defected (1)

- Test and train datasets are balanced again to contain non-defected labeled reviews to be not more than %10 of defected labeled reviews.

3.3.6 Training Models for Defect Detection Task

The final dataset which is labeled as non-defected/defected is used to train 2 ML models (LR, NB) and two DL models (CNN, RNN).

CHAPTER IV

EXPERIMENTS AND RESULTS

4.1 PROGRAMMING ENVIRONMENT AND LIBRARIES

- Python 3.9.5 is used as the programming language.
- Numpy, Pandas libraries for data handling.
- Mathplotlib, seaborn, WordCloud libraries for visualization.
- Gensim library for Word2Vec word-embedding training.
- Sklearn library for vectorization, train/test split, ML models, and evaluation tasks.
- Tensorflow, Keras libraries for DL models.

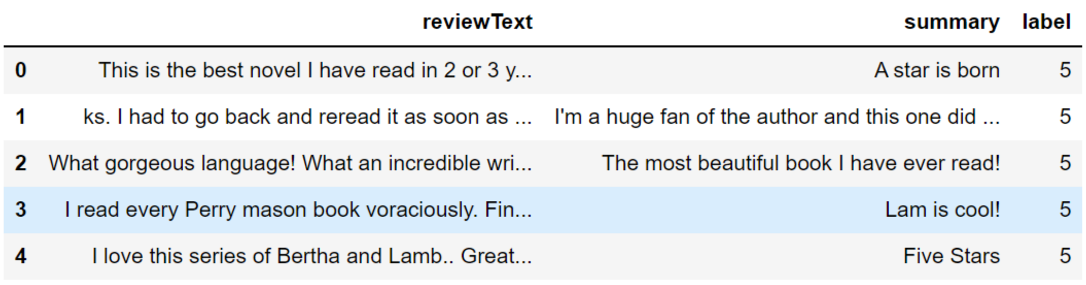
4.2 TRAINING ENVIRONMENT

Models are trained on a local laptop with 16 GB of RAM, 4th gen i7 CPU with 4 cores, equipped with Nvidia Geforce 850M GPU unit. Several DL models that use Keras and Tensorflow libraries are tested on Google Colab environment supplying Tesla-K80 GPU, which performs better than the local 850M GPU unit for DL models.

4.3 INSPECTING AND PRE-PROCESSING DATASET

To save memory, while loading gzip-compressed JSON formatted dataset (Electronics_5.json.gz) into pandas dataframe object, only the necessary fields are selected ('reviewText', 'summary', 'overall'), reviews with overall scores 1, 2 (for negative reviews) and 5 (for positive reviews) are filtered, and long reviews are truncated to 2000 characters. The resulting dataframe has 3 columns and 5.097.416 rows. The first 5 lines of the dataset are shown in Figure 4.1.

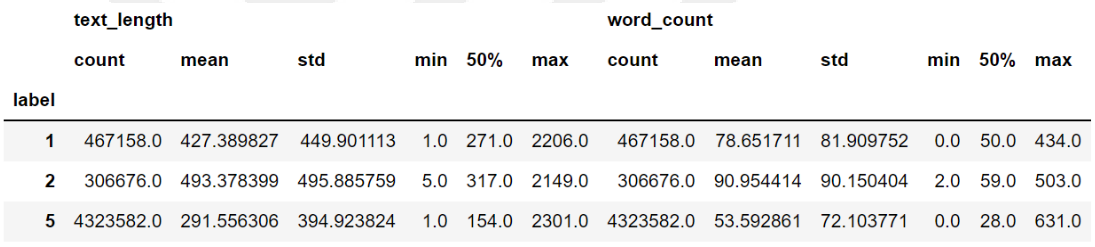
Figure 4.1: Dataset (First 5 Lines)



	reviewText	summary	label
0	This is the best novel I have read in 2 or 3 y...	A star is born	5
1	ks. I had to go back and reread it as soon as ...	I'm a huge fan of the author and this one did ...	5
2	What gorgeous language! What an incredible wri...	The most beautiful book I have ever read!	5
3	I read every Perry mason book voraciously. Fin...	Lam is cool!	5
4	I love this series of Bertha and Lamb.. Great...	Five Stars	5

'reviewText' and 'summary' columns are dropped after concatenated to create the new single ‘text’ column. 'text_length' and 'word_count' columns are calculated and inserted into the dataframe to get more information about the text column. Reviews with labels 1 and 2 (773.834) are much less than reviews with labels 5 (4.323.582). The statistics of the dataset grouped by labels are shown in Figure 4.2.

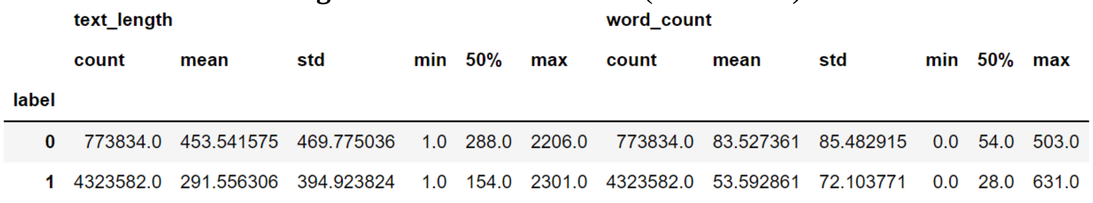
Figure 4.2: Dataset Statistics (Grouped By Labels)



	text_length						word_count					
	count	mean	std	min	50%	max	count	mean	std	min	50%	max
label												
1	467158.0	427.389827	449.901113	1.0	271.0	2206.0	467158.0	78.651711	81.909752	0.0	50.0	434.0
2	306676.0	493.378399	495.885759	5.0	317.0	2149.0	306676.0	90.954414	90.150404	2.0	59.0	503.0
5	4323582.0	291.556306	394.923824	1.0	154.0	2301.0	4323582.0	53.592861	72.103771	0.0	28.0	631.0

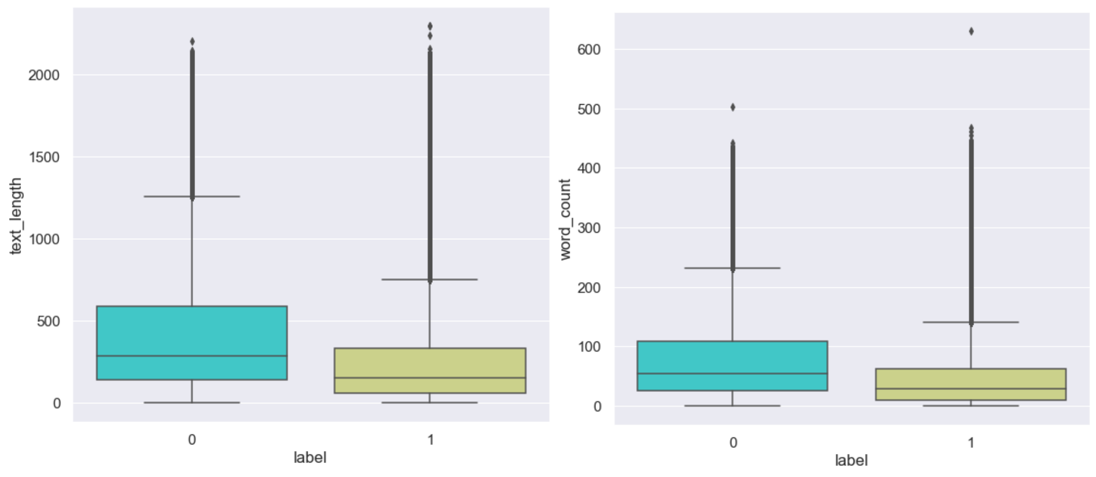
Reviews with ratings 1 and 2 are re-labeled as negative (0), and reviews with rating 5 are re-labeled as positive (1) as seen in Figure 4.3. Negative reviews are longer in size and also contain more words compared to positive reviews as shown in Figure 4.4.

Figure 4.3: Dataset Statistics (Labeled 0/1)



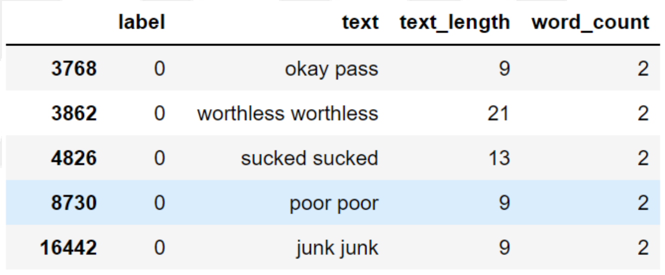
	text_length						word_count					
	count	mean	std	min	50%	max	count	mean	std	min	50%	max
label												
0	773834.0	453.541575	469.775036	1.0	288.0	2206.0	773834.0	83.527361	85.482915	0.0	54.0	503.0
1	4323582.0	291.556306	394.923824	1.0	154.0	2301.0	4323582.0	53.592861	72.103771	0.0	28.0	631.0

Figure 4.4: Text Lengths and Word Counts



There are 794 negative and 13873 positive reviews that have text_length smaller than 10 and word_count smaller than 3. These short reviews do not contain enough information as seen in Figure 4.5, so are deleted.

Figure 4.5: Short Reviews



	label	text	text_length	word_count
3768	0	okay pass	9	2
3862	0	worthless worthless	21	2
4826	0	sucked sucked	13	2
8730	0	poor poor	9	2
16442	0	junk junk	9	2

1% of the dataframe is split for later manual tests. And for the remaining part, the first positive reviews are reduced to 110% of negative reviews to balance the dataset. Later it is split into train and test sets with a ratio of 0.1 to 0.9. The resulting dataframes are of shape; Test Data (Negative/Positive): (76530, 4) (84182, 4), and Train Data (Negative/Positive): (688765, 4) (757642, 4).

4.4 TESTING DATASET WITH ML MODELS

In order to be confident about the dataset, we performed a base sentiment analysis task by training 3 ML models with two word-representations and with 5 different data sizes, as stated in 3.3.3. Accuracy and F1 scores are shown in Table 4.1, Figure 4.6, and Figure 4.7.

Table 4.1: Sentiment Analysis ML Performances

F1 Score	*480*	*2400*	*12K*	*60K*	*228K*
BOW_LR	0.8296	0.8774	**0.9382**	**0.9521**	0.9584
BOW_NB	0.8361	0.894	0.9243	0.9335	0.9358
BOW_RandomForest	0.8455	0.8383	0.8989	0.9187	0.9336
TFIDF_LR	0.8462	**0.8964**	0.9327	0.9502	**0.9601**
TFIDF_NB	**0.8489**	0.8911	0.9262	0.9326	0.9341
TFIDF_RandomForest	0.7846	0.8666	0.8961	0.917	0.9354
Accuracy	*480*	*2400*	*12K*	*60K*	*228K*
BOW_LR	0.8083	0.87	**0.935**	**0.9493**	0.9563
BOW_NB	0.8333	0.8917	0.9213	0.9302	0.9326
BOW_RandomForest	**0.8417**	0.8367	0.8967	0.9165	0.9315
TFIDF_LR	0.8333	**0.8933**	0.9304	0.9477	**0.9582**
TFIDF_NB	0.825	0.89	0.925	0.9307	0.9324
TFIDF_RandomForest	0.7667	0.865	0.8938	0.9147	0.9334

Figure 4.6: ML Models Performances - Accuracy

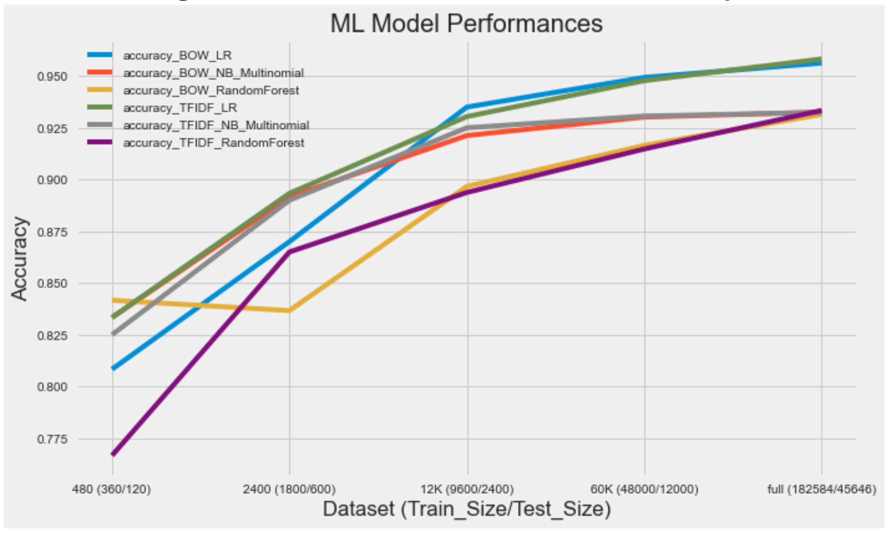
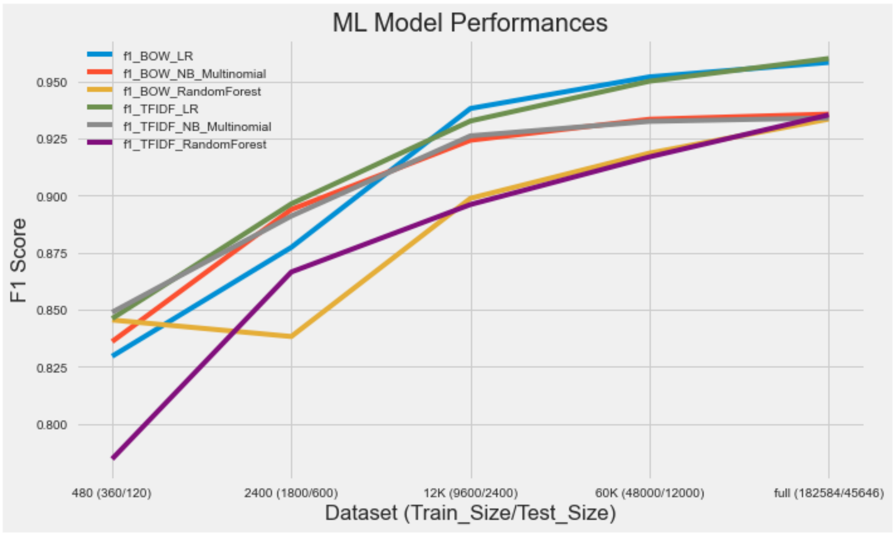


Figure 4.7: ML Models Performances - F1 Score



We reached the following judgements after performing this step;

- Performance of the models increases with the growth of the dataset.
- High F1 and Accuracy scores tell us that the labels are accurate for reflecting sentiments of the reviews. And also, reviews contain enough information so that, models can successfully predict the reviews as positive or negative.
- Both BOW and TFIDF models represent the words with similar success for the sentiment analysis task.
- Although all models performed well for the task, the LR model is slightly better than the others.

4.5 TRAINING AND EVALUATING WORD-EMBEDDINGS

We trained our own Word2Vec model using only the negative train set. To evaluate our self-trained model, we also downloaded a pre-trained model (glove-wiki-gigaword-300) which is trained by Wikipedia data, and used analogies and visualizations to compare the performances, strengths, and weaknesses of both models.

Pre-trained models can be used successfully for generic analogies like ‘Man’ to ‘King’ is ‘Woman’ to ‘Queen’ as shown in Table 4.2.

Table 4.2: Analogy Using Pre-Trained Model

```
> wiki.most_similar(positive=['woman', 'king'], negative=['man'], topn=1)
[('queen', 0.6713)]
```

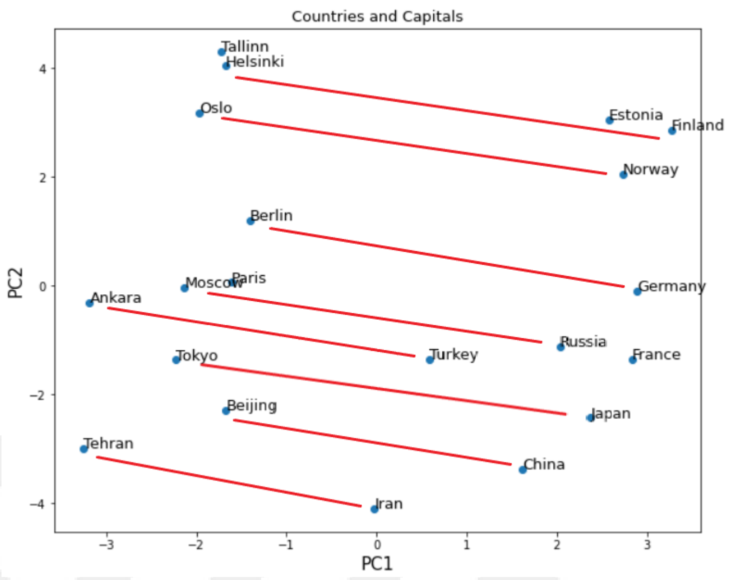
When we run the same analogy with our pre-trained model as shown in Table 4.3, the 3 most similar words are ‘vp’, ‘pr’, and ‘warrentech’, which are all unrelated words. This output shows that our self-trained model is performing badly for a different context which is an expected result.

Table 4.3: Analogy Using Self-Trained Model

```
> wv.most_similar(positive=['woman', 'king'], negative=['man'], topn=3)
[('vp', 0.5546),  ('pr', 0.5102),  ('warrentech', 0.4977)]
```

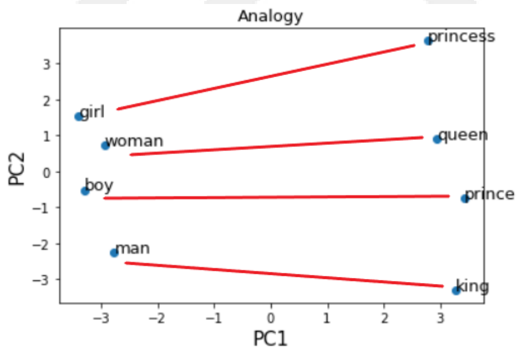
Visualization is a well know intrinsic evaluation method used for word-embeddings. To visualize how well a pre-trained model performs for generic tasks, we used the analogy of countries and capitals. Word vector dimensions are reduced into two dimensional space using PCA and the graphic is drawn as shown in Figure 4.8.

Figure 4.8: Countries and Capitals



We tested one more analogy as seen in Figure 4.9.

Figure 4.9: Man to King, Woman to Queen Analogy



These two examples show that a pre-trained model performs well for a generic task. This is a strength of a pre-train model over a self-trained model. The same analogies cannot be tested on the self-trained model because most of the words are not even in the vocabulary.

For our use case, which is a specific task and a specific context related to electronic product defects, we evaluated the performance of the self-trained model by using word similarities. The top 10 most similar words for three example words (‘broken’, ‘defect’, and ‘malfunctioning’) are shown in Table 4.4. As seen, words derived from the pre-trained model contain words from a different context like ‘bones’, ‘heater’ or ‘valve’. Our self-train model is trained with only negative reviews of a specific product group (Electronics), so related words are all in the same context.

Table 4.4: Top 10 Most Similar Words

Self-Trained Word2Vec Model	Pre-Trained GloVe Model
Most Similar Words to ‘**broken**’	
('busted', 0.7576), ('cracked', 0.7313), ('damaged', 0.7233), ('torn', 0.7211), ('bent', 0.6819), ('chipped', 0.6535), ('rusted', 0.64), ('broke', 0.6236), ('falling_apart', 0.6159), ('glued_together', 0.6048)	('broke', 0.647), ('fractured', 0.6322), ('breaking', 0.6321), ('shattered', 0.583), ('cracked', 0.5597), ('break', 0.5278), (*'bones'*, 0.5269), (*'collarbone'*, 0.5242), ('damaged', 0.5034), ('apart', 0.4933)
Most Similar Words to ‘**defect**’	
('manufacturing_defect', 0.8454), ('flaw', 0.7665), ('design_flaw', 0.7519), ('defects', 0.7431), ('factory_defect', 0.6787), ('issue', 0.6698), ('problem', 0.6375), ('faulty', 0.6286), ('manufacturing_flaw', 0.6281), ('defective', 0.6177)	('defects', 0.7401), (*'congenital'*, 0.5468), ('flaw', 0.4827), ('abnormality', 0.4824), ('defective', 0.4752), ('abnormalities', 0.47), (*'bifida'*, 0.4299), (*'deformity'*, 0.4208), ('defected', 0.4013), (*'anomalies'*, 0.398
Most Similar Words to ‘**malfunctioning**’	
('faulty', 0.7439), ('failing', 0.7099), ('acting_up', 0.685), ('defective', 0.6612), ('malfunction', 0.6402), ('malfunctioned', 0.6286), ('nonfunctional', 0.6274), ('operable', 0.6226), ('dying', 0.6105), ('inoperative', 0.5982)	('malfunctioned', 0.6457), ('faulty', 0.6185), ('malfunction', 0.6054), ('malfunctions', 0.5464), ('balky', 0.5113), ('defective', 0.5082), (*'heater'*, 0.4747), (*'valve'*, 0.4353), (*'transponder'*, 0.4318), ('glitch', 0.4292)
Most Similar Words to ‘**fall_apart**’ (bi-gram)	
('break', 0.8036), ('disintegrate', 0.7919), ('fray', 0.765), ('crumble', 0.7392), ('deteriorate', 0.7236), ('unravel', 0.7146), ('falling_apart', 0.7131), ('break_easily', 0.6981), ('fall_off', 0.698), ('falls_apart', 0.6779)	*Not in vocabulary*

On the other hand, if we test a context-specific analogy with our self-trained model, this time, for the analogy; ‘good’ – ‘bad’ + ‘malfunctioning’, 3 possible words are ‘operable’, ‘acting_up’ and ‘functioning’ (Table 4.5), which is expected since all three are opposites of ‘malfunctioning’.

Table 4.5: Analogy Using Self-Trained Model

```
> wv.most_similar(positive=['good','malfunctioning'],negative=['bad'],topn=3)
[('operable', 0.5438),  ('acting_up', 0.5084),  ('functioning', 0.4940)]
```

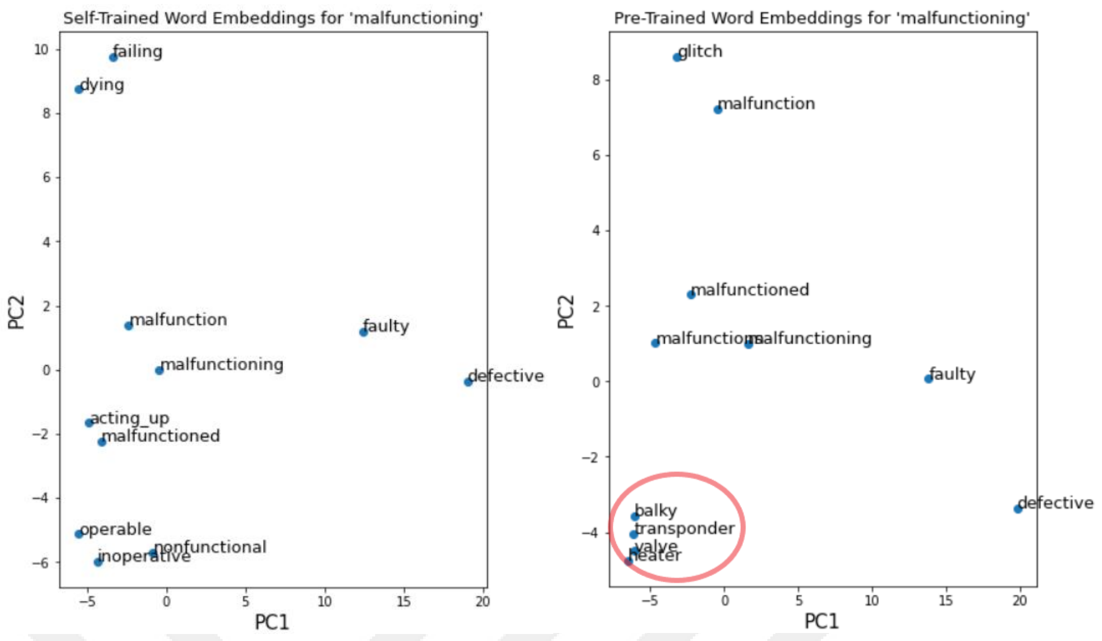
If the same test is performed on the pre-trained model the resulting three most related words are ‘malfunctioned’, ‘heater’, and ‘ventilation’ (Table 4.6), which shows the weakness of pre-trained models for context-specific tasks.

Table 4.6: Analogy Using Pre-Trained Model

```
> wiki.most_similar(positive=['good','malfunctioning'],negative=['bad'],topn=3)
[('malfunctioned', 0.4599),  ('heater', 0.4532),  ('ventilation', 0.3955)]
```

For the word ‘malfunctioning’, the graphics of the top 10 most similar words are visualized as shown in Figure 4.10 for both word embedding models. As seen, pre-trained model contains unrelated words while self-trained model is more precise. This shows for a specific context and for a specific task pre-trained models perform better. This is the strength of pre-trained models over self-trained models.

Figure 4.10: Top Most Similar Words to 'Malfunctioning'



4.6 POPULATING DEFECT RELATED WORDS LIST

Using own self-trained model which proved to be reliable for defect-specific context using analogies and word similarities experimented in 4.5, we populated the defect-related words list iteratively in three steps.

- In the first step, we constructed our base list with 10 defect related-words. ('broken', 'broke', 'cracked', 'fall_apart', 'malfunctioning', 'nonfunctional', 'defected', 'non_functioning', 'faulty', 'flawed').
- In the second step, for each word in the base list we find the top 20 words that have a similarity score greater than 0.6. Resulting list is checked manually and 11 words which are not directly related to defects are removed from the list ('last_long', 'questionable', 'inexcusable', 'unravel', 'acting_up', 'unique', 'fell', 'shotty', 'operable', 'innovative', 'top_notch'). This resulted in 90 new words.
- In the third step, the same process is repeated for the second words list, but this time words that have a similarity score greater than 0.75 are selected. Resulting list is checked manually and 9 words which are not directly related to defects are removed from the list again ('superb', 'unraveled', 'corners', 'covering', 'quit', 'outstanding', 'elegant', 'abysmal', 'protective_film'). This resulted in 63 new words.

After combining all three lists, we ended up with 163 words that are strongly related to defects as shown in Table 4.7.

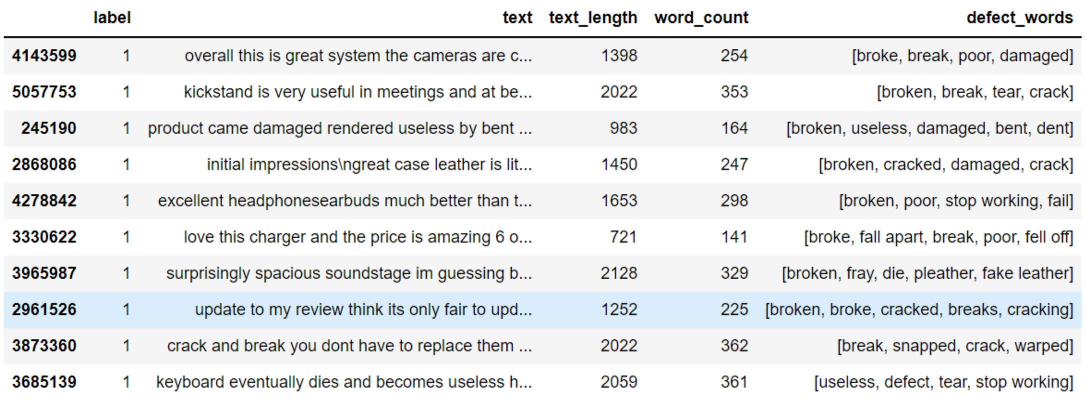
Table 4.7: Defect Related Words List

broken, broke, cracked, fall_apart, malfunctioning, nonfunctional, defected, non_functioning, faulty, flawed, utterly_useless, break, dreadful, disintegrate, unusable, stopped_working, bad_batch, disintegrated, useless, falling_apart, seriously_flawed, non_functional, inoperable, defect, snapped, inoperative, wears_out, shoddy, hairline_crack, fundamentally_flawed, lemon, poor, break_easily, completely_nonfunctional, shattered, coming_apart, deficient, hinge_broke, badly_designed, fell_apart, deffective, came_apart, peeling_off, unuseable, unglued, chipped, busted, halfbaked, fray, malfunctioned, defective, doa, crumbled, die, tear, screen_shattered, started_falling, stop_working, poorly_designed, snapped_off, unlucky, lackluster, torn, rusted, dented, poorly_engineered, fluke, come_undone, nonworking, fell_off, ripped, became_loose, deteriorate, peeled_off, nonfunctioning, cracks, fall_off, damaged, wore_out, manufacturing_defect, completely_useless, chipped_off, totally_useless, bent, glued_together, tore, started_tearing, substandard, breaks, malfunction, dud, popped_off, failing, dying, started_cracking, crumble, zipper_broke, falls_apart, dysfunctional, crack, peeling, pleather, fraying, died, slightly_bent, glued, brake, separated_from, flaking_off, started_chipping, crapped_out, stained, deformed, dismal, completely_unusable, design_flaw, basically_useless, stopped_functioning, stoped_working, flaw, unraveling, scuffed, separated, chipping_off, hairline_cracks, cracking, peels_off, crushed, already_fallen, poorly_constructed, split_open, twisted, started_fraying, half_baked, virtually_useless, fake_leather, rusty, warped, an_anomaly, worthless, horrendous, totally_unusable, faux_leather, fail, started_flaking, dents, came_unglued, subpar, literally_fell, peel_off, lousy, atrocious, horrific, dent, started_peeling, fractures, coming_loose, discolored, stains, sheared_off, quit_working, horrid, coming_undone

4.7 MAKING INFERENCES USING DEFECT RELATED WORDS LIST

Defect-related words can be used to filter reviews that contain defect-related information. Usually, reviews complaining about defects of products have negative reviews as expected. But positive reviews may also contain such words. We inspected and filtered such positive reviews which have more than 3 defect words in the same review as shown in Figure 4.11.

Figure 4.11: Positive Reviews Containing Defect Words



	label	text	text_length	word_count	defect_words
4143599	1	overall this is great system the cameras are c...	1398	254	[broke, break, poor, damaged]
5057753	1	kickstand is very useful in meetings and at be...	2022	353	[broken, break, tear, crack]
245190	1	product came damaged rendered useless by bent ...	983	164	[broken, useless, damaged, bent, dent]
2868086	1	initial impressions\ngreat case leather is lit...	1450	247	[broken, cracked, damaged, crack]
4278842	1	excellent headphonesearbuds much better than t...	1653	298	[broken, poor, stop working, fail]
3330622	1	love this charger and the price is amazing 6 o...	721	141	[broke, fall apart, break, poor, fell off]
3965987	1	surprisingly spacious soundstage im guessing b...	2128	329	[broken, fray, die, pleather, fake leather]
2961526	1	update to my review think its only fair to upd...	1252	225	[broken, broke, cracked, breaks, cracking]
3873360	1	crack and break you dont have to replace them ...	2022	362	[break, snapped, crack, warped]
3685139	1	keyboard eventually dies and becomes useless h...	2059	361	[useless, defect, tear, stop working]

Two examples of positive reviews that contain information about defective products are shown in Table 4.8.

Table 4.8: Two Positive Reviews with Defect Words

Defect words in the review : [broken, broke, cracked, breaks, cracking]
'update to my review think its only fair to update my review of the mushkin 32gb atom was disappointed that the little plastic part of the drive was **cracking** but mushkin has offered to replace them and these little flash drives are really great in every other respect so id like to give them five with minor point deductions for the plastic i was in love with these they are great but of the i bought the corner of the plastic cap not the removable cover where the lanyard attaches **broke** off on one and another is **cracked** across the same area and the caps just pulled completely off two others since receiving them in july\nwhen the first cap came off just put tiny dab of all purpose crazy glue in it and stuck it back on but the **cracked** and **broken** ones are disappointing\notherwise they are great wonderful fit for insertion great performance use them with linux for portable operating system and the debianubuntu versions mint zorin emmabuntus uberstudent lxle all boot in less then minute and perform nicely when run from these flash drives\nmaybe should give them four stars id like to give them five\nmushkin please make better cap plastic part **breaks** and pulls off but otherwise great'
Defect words in the review : [broke, fall apart, break, poor, fell off]
'love this charger and the price is amazing 6 or right around there when the same one at verizon is 40 never buy cheap off brand car charger as they **fall apart** and **break** so easy and can potentially harm your phone this is perfect charger durable charges fast and looks amazing the blue glowing logo in the middle looks great when it is plugged in and also lets you know it is being powered i bought 2nd one for my girlfriend as she **broke** her cheap one without even knowing it the tip just **fell off** as it was **poor** construction i bought her this one and she loves it i have had mine for over years now use it every day and it still works and looks like new the only charger you should buy'

As seen in the two examples above, defect-related information can be found even in positive reviews.

It is also possible to label reviews as defected/non-defected using the defect words list. After labeling any classification models can be trained.

As a final experimentation, we used a smaller defect word list containing only 17 words which are ['dead', 'died', 'broken', 'broke', 'cracked', 'fall_apart', 'malfunctioning', 'nonfunctional', 'defected', 'non_functioning', 'faulty', 'flawed', 'badly_designed', 'poorly_engineered', 'undependable', 'deficient', 'substandard'], and filtered negative reviews that contain defect words, re-labeled them as defected (0), filtered positive reviews that do not contain any of the defect-related words, and re-labeled them as non-defected (1). That resulted train and test datasets with sizes 8924/9816 (defect/non-defect) and 80292/88321 respectively. Later we trained several

ML and DL models with 4 different subsets of the same dataset and evaluated the performances for a defect estimation task (Figure 4.12, Figure 4.13).

Figure 4.12: Performances of ML Models for Defect Detection

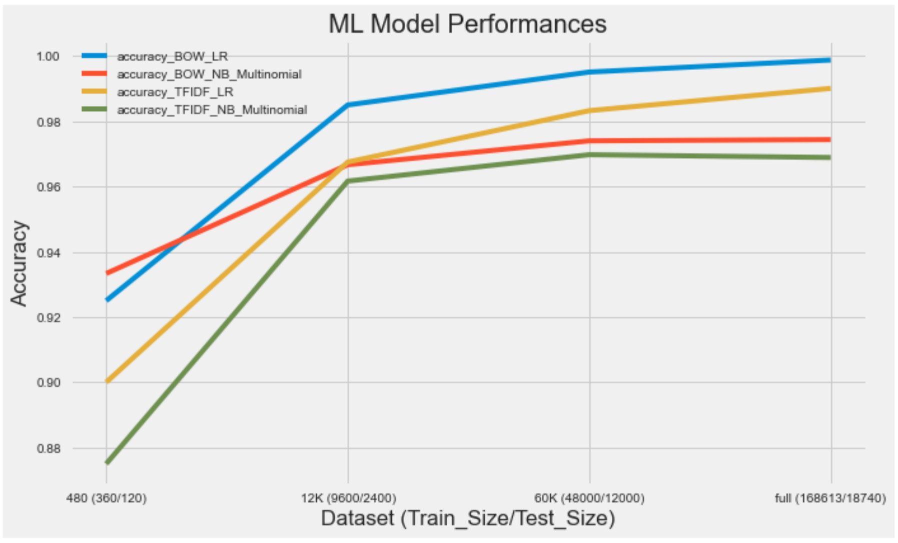
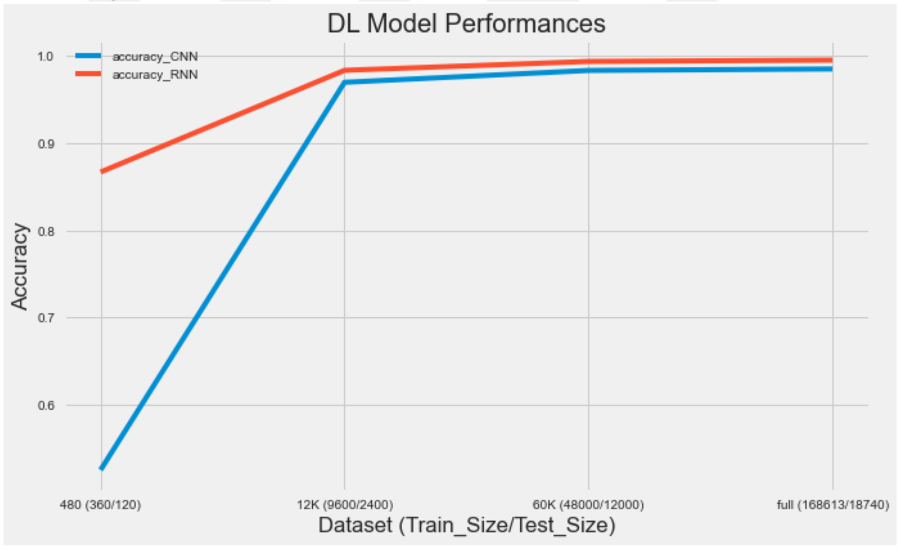


Figure 4.13: Performances of DL Models for Defect Detection



The accuracy and f1 scores are very high. Both reach almost 100% for the full dataset. But, this result must not be misleading. We only labeled reviews if they contain several words as the criteria. The trained models looked if such words exist in the review or not for prediction.

CHAPTER V

CONCLUSIONS

To filter reviews that contain any defect-related information we used word-embeddings and populated a word list that is related to defects. We observed that a self-trained word-embedding model performs better for specific data and a specific context. We also completed a sentiment analysis task before our main task to be sure of the reliability of our dataset and observed the effectiveness of this approach.

We believe that such context and sentiment-specific word lists trained with specific datasets can be useful for later tasks. For example, for a defect detection problem, using defect word lists, reviews can be labeled as ‘defected’ and ‘non-defected’ automatically. Or this approach can be used to ease manual labeling processes by filtering possible defect-related reviews and reducing the dataset for manual operations. After that, the problem becomes a simple text classification task with two possible labels.

As a future study, product groups or product versions can be examined that have reviews with similar defect information which may indicate chronic problems of a product. This task may serve as a defect risk alert application.

The same approach can also be applied to different datasets with different keywords to extract specific types of information.

Today’s world has many successful solutions to difficult NLP tasks like chatbots, machine translation, and text generation. NLP is still evolving rapidly parallel to other improvements in science and technology, so it will not be surprising to see many other successful solutions and applications in the near future.

REFERENCES

[1] P. H. Winston, “On Computing Machinery and Intelligence,” *Bost. Stud. Philos. Hist. Sci.*, vol. 324, pp. 265–278, 2017, doi: 10.1007/978-3-319-53280-6_11.

[2] C. J. Du and D. W. Sun, “Learning techniques used in computer vision for food quality evaluation: A review,” *J. Food Eng.*, vol. 72, no. 1, pp. 39–55, 2006, doi: 10.1016/j.jfoodeng.2004.11.017.

[3] D. Praveen Kumar, T. Amgoth, and C. S. R. Annavarapu, “Machine learning algorithms for wireless sensor networks: A survey,” *Inf. Fusion*, vol. 49, no. April 2018, pp. 1–25, 2019, doi: 10.1016/j.inffus.2018.09.013.

[4] D. S. Hain and R. Jurowetzki, “The Potentials of Machine Learning and Big Data in Entrepreneurship Research ∗ ,” 2018.

[5] A. Akmajian, R. A. Demers, A. K. Farmer, and R. M. Harnish, *LINGUISTICS An Introduction to Language and Communication*, Sixth. The MIT Press Cambridge, Massachusetts London, England, 2010.

[6] D. H. Andrew L. Maas, Raymond E. Daly, Peter T. Pham and and C. P. Andrew Y. Ng, “Learning Word Vectors for Sentiment Analysis,” 2011, doi: 10.11314/jisss.3.23.

[7] T. Mikolov, Q. V. Le, and I. Sutskever, “Exploiting Similarities among Languages for Machine Translation,” 2013, [Online]. Available: http://arxiv.org/abs/1309.4168.

[8] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global vectors for word representation,” *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, pp. 1532–1543, 2014, doi: 10.3115/v1/d14-1162.

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

[10] M. E. Peters *et al.*, “Deep contextualized word representations,” *NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, pp. 2227–2237, 2018, doi: 10.18653/v1/n18-1202.

[11] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[12] K. Song, X. Tan, T. Qin, J. Lu, and T. Y. Liu, “MASS: Masked sequence to sequence pre-training for language generation,” *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10384–10394, 2019.

[13] T. B. Brown *et al.*, “Language models are few-shot learners,” *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, 2020.

[14] Z. Huang *et al.*, “Recent Trends in Deep Learning Based Open-Domain Textual Question Answering Systems,” *IEEE Access*, vol. 8, pp. 94341–94356, 2020, doi: 10.1109/ACCESS.2020.2988903.

[15] A. M. Butnaru and R. T. Ionescu, “UnibucKernel: A kernel-based learning method for complex word identification,” *Proc. 13th Work. Innov. Use NLP Build. Educ. Appl. BEA 2018 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL-HTL 2018*, no. January, pp. 175–183, 2018, doi: 10.18653/v1/w18-0519.

[16] J. Zhang, A. Zhang, D. Liu, and Y. Bian, “Customer preferences extraction for air purifiers based on fine-grained sentiment analysis of online reviews,” *Knowledge-Based Syst.*, vol. 228, p. 107259, 2021, doi: 10.1016/j.knosys.2021.107259.

[17] P. K. Jain, R. Pamula, and G. Srivastava, “A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews,” *Comput. Sci. Rev.*, vol. 41, p. 100413, 2021, doi: 10.1016/j.cosrev.2021.100413.

[18] A. S. Rathor, A. Agarwal, and P. Dimri, “Comparative Study of Machine Learning Approaches for Amazon Reviews,” *Procedia Comput. Sci.*, vol. 132, pp. 1552–1561, 2018, doi: 10.1016/j.procs.2018.05.119.

[19] M. Cataltas, S. Dogramaci, S. Yumusak, and K. Oztoprak, “Extraction of Product Defects and Opinions from Customer Reviews by Using Text Clustering and Sentiment Analysis,” *Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020*, pp. 4529–4534, 2020, doi: 10.1109/BigData50022.2020.9377851.

[20] P. Mukherjee, Y. Badr, S. Doppalapudi, S. M. Srinivasan, R. S. Sangwan, and R. Sharma, “Effect of Negation in Sentences on Sentiment Analysis and Polarity

Detection,” *Procedia Comput. Sci.*, vol. 185, no. June, pp. 370–379, 2021, doi: 10.1016/j.procs.2021.05.038.

[21] Y. Zhang, J. Wang, and X. Zhang, “Conciseness is better: Recurrent attention LSTM model for document-level sentiment analysis,” *Neurocomputing*, vol. 462, pp. 101–112, 2021, doi: 10.1016/j.neucom.2021.07.072.

[22] S. Dey, S. Wasif, D. S. Tonmoy, S. Sultana, J. Sarkar, and M. Dey, “A Comparative Study of Support Vector Machine and Naive Bayes Classifier for Sentiment Analysis on Amazon Product Reviews,” *2020 Int. Conf. Contemp. Comput. Appl. IC3A 2020*, pp. 217–220, 2020, doi: 10.1109/IC3A48958.2020.233300.

[23] C. Fry and S. Manna, “Can We Group Similar Amazon Reviews: A Case Study with Different Clustering Algorithms,” *Proc. - 2016 IEEE 10th Int. Conf. Semant. Comput. ICSC 2016*, pp. 374–377, 2016, doi: 10.1109/ICSC.2016.71.

[24] K. Tamara and N. Milićević, “Comparing Sentiment Analysis and Document Representation Methods of Amazon Reviews,” *SISY 2018 - IEEE 16th Int. Symp. Intell. Syst. Informatics, Proc.*, pp. 283–288, 2018, doi: 10.1109/SISY.2018.8524814.

[25] M. Dogan, O. Metin, E. Tek, S. Yumusak, and K. Oztoprak, “Speculator and Influencer Evaluation in Stock Market by Using Social Media,” *Proc. - 2020 IEEE Int. Conf. Big Data, Big Data 2020*, pp. 4559–4566, 2020, doi: 10.1109/BigData50022.2020.9378170.

[26] B. Mutlu, M. Mutlu, K. Oztoprak, and E. Dogdu, “Identifying trolls and determining terror awareness level in social networks using a scalable framework,” *Proc. - 2016 IEEE Int. Conf. Big Data, Big Data 2016*, pp. 1792–1798, 2016, doi: 10.1109/BigData.2016.7840796.

[27] M. V. Rao and C. Sindhu, “Detection of Sarcasm on Amazon Product Reviews using Machine Learning Algorithms under Sentiment Analysis,” *2021 Int. Conf. Wirel. Commun. Signal Process. Networking, WiSPNET 2021*, pp. 196–199, 2021, doi: 10.1109/WiSPNET51692.2021.9419432.

[28] E. Uysal, S. Yumusak, K. Oztoprak, and E. Dogdu, “Sentiment Analysis for the Social Media : A Case Study for Turkish General Elections Categories and Subject Descriptors,” *Proc. SouthEast Conf. ACM*, pp. 215–218, 2017.

[29] A. Ejaz, Z. Turabee, M. Rahim, and S. Khoja, “Opinion mining approaches on Amazon product reviews: A comparative study,” *2017 Int. Conf. Inf. Commun.*

Technol. ICICT 2017, vol. 2017-Decem, pp. 173–179, 2018, doi: 10.1109/ICICT.2017.8320185.

[30] M. R. Haque, S. Akter Lima, and S. Z. Mishu, “Performance Analysis of Different Neural Networks for Sentiment Analysis on IMDb Movie Reviews,” *3rd Int. Conf. Electr. Comput. Telecommun. Eng. ICECTE 2019*, pp. 161–164, 2019, doi: 10.1109/ICECTE48615.2019.9303573.

[31] A. Almjawel, “Books ’ Reviews,” pp. 0–5, 2019.

[32] M. S. Saranya and P. Geetha, “Word Cloud Generation on Clothing Reviews using Topic Model,” *Proc. 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020*, pp. 177–180, 2020, doi: 10.1109/ICCSP48568.2020.9182111.

[33] J. Ni, J. Li, and J. McAuley, “Justifying recommendations using distantly-labeled reviews and fine-grained aspects,” *EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, pp. 188–197, 2020, doi: 10.18653/v1/d19-1018.