# **COMPLAINT CLASSIFICATION USING ML BASED APPROACHES**

*Final Report*

*M.Sc. IN MATHEMATICS*

*Submitted By:*

Aripra Kar (2020MAS7362)

Annu Goyal (2020MAS7367)

*Under The Guidance of:*

Prof. Niladri Chatterjee



Department of Mathematics

Indian Institute of Technology Delhi

# **Certification**

This is to certify that the thesis entitled **COMPLAINT CLASSIFICATION USING ML BASED APPROACHES**, being submitted by **Annu Goyal, 2020MAS7367** and **Aripra Kar, 2020MAS7362** to the Department of Mathematics, Indian Institute of Technology Delhi, India, in partial fulfilment of requirements for the award of the degree of **M.Sc. in Mathematics**, is a bona fide record of research work carried out by them under my guidance and supervision.

To the best of my knowledge, the results embodied in this project have not been submitted in part or full to other University or Institute for the award of any degree or diploma.

**Prof. Niladri Chatterjee**

**Department of Mathematics**

**Indian Institute of Technology Delhi**

# **Acknowledgement**

We would like to extend our sincere thanks to the people without whose kind support and help, it would not have been possible to complete this project. We would also like to express our gratitude towards **Prof. Niladri Chatterjee** for his guidance and constant supervision as well as for the continuous encouragement which helped us intellectually and motivated us throughout the duration of the project, and hence helped immensely in the successful completion. In addition, we would like to express special gratitude to the Department of Mathematics for constantly supporting quality research work in the realm of abstract and applied mathematics.

# **Abstract**

# Computers today almost invariably deal with unstructured text. It is available all over the internet in abundance. However, it is very hard to process such raw data to understand, interpret, and analyze it. In recent times processing unstructured text has become the backbone of various applications. The current project focuses on the classification of complaints into several classes. Each class contains complaints of similar nature. The major challenge comes from two different angles.

# On the one hand, the complaints sent by users through email/social media are often full ­­of unnecessary text data. Moreover, often the language used is ungrammatical, making it difficult to parse by standard English parsers. On the other hand, the type and number of complaint classes are not known a priori, making straightforward classification of complaints a challenging task. In this work, we have used NLP and ML-based techniques for designing a classification system. The dataset used comprises 1000 complaints collected from a well-known courier service.

# **Contents**

TABLE OF CONTENTS

introduction 8

objective 8

Dataset 8

proposed approach 9

Survey of NLP Tools 10

text preprocessing 10

Process 10

Challenges Encountered 12

semantics 18

WordNet 18

SentiWordNet 19

feature selection 20

Feature Extraction Methods 20

clustering 21

Flat Clustering 21

Hierarchical Clustering 22

Feature extraction and analysis 23

n-grams 23

TF-IDF 25

RAKE 28

Sentiment Analysis 31

textblob library 31

RAKE results on textblob 32

RAKE results on sentiwordnet 33

TF-IDF results on textblob 34

RAKE results on sentiwordnet 35

common keyphrases of RAKE 36

common keyphrases of TF-IDF 37

union of RAKE and TF-IDF results 38

Clustering 39

introduction 39

types of clustering 40

K-Means Clustering 41

Hierarchical Clustering 42

DBSCAN 44

clustering in the problem statement 45

results 47

cluster analysis 50

multilabel clustering 54

conclusion 56

further work 56

references 57

# 

# **List of Figures**

Figure 1.1 Sample Classification of an email

Figure 2.1 Pos Tagging

Figure 2.2 Removal of Proper Nouns

Figure 2.3 Stemming and Lemmatization

Figure 2.4 WordNet Example

Figure 2.5 SentiWordNet Score of Keywords

Figure 3.1 Top 20 Bigrams

Figure 3.2 Top 20 Trigrams

Figure 3.3 Top 10 Tetragrams

Figure 3.4 Top Keyphrases from TF-IDF

Figure 3.5 RAKE Scores of Some Keyphrases

Figure 4.1 RAKE Textblob

Figure 4.2 RAKE SentiWordNet

Figure 4.3 TF-IDF Textblob

Figure 4.4 TF-IDF SentiWordNet

Figure 4.5 RAKE Final List

Figure 4.6 TF-IDF Final List

Figure 4.7 Union of RAKE and TF-IDF Lists

Figure 5.1 Clustering Methods

Figure 5.2 Hierarchical Clustering Flowchart

Figure 5.3 Linkages of Agglomerative Clustering

# **Chapter 1**

# **Introduction**

## **Objective**

To categorize each email under a specific Complaint aspect and classify the emails of the Courier company to determine customer satisfaction levels and identify the problem/improvement areas, such as non-delivery of goods or misplaced packages along with perceiving customer experiences.

Here is a sample classification: -

|  |  |  |
| --- | --- | --- |
| Consignment No M09404103 sent from Bhilai (C.H) TO Bengaluru On 31.07.14 was received on 04.08.14. To my surprise, Instead of actual Sosdexo Coupons Worth Rs 3900 sent, I received a 10 Rs paper Dairy inside the envelope  Why would anybody send those 10 rs paper bits paying 100 RS as Courier Fee ??  Kindly address my complaint and do the needful actions in this regard. i had a big loss of 3900 Rs because of their irresponsible behaviour.XYZ India is not picking up calls and Not addressing my greivance on email as well.  Courier was sent from the XYZ Bhilai Supela branch RF-138. | lodged complaint | missing consignment |

### **Figure 1.1**

### **Dataset**

The dataset consists of 1000 emails extracted from the complaints section of XYZ Courier Services. The emails are constituted of dates, names, and delivery locations of the parcels along with tracking details and addresses. They show a diverse range of language, experiences, and dissatisfaction expressed by customers from all over the country. Some emails are follow-ups of their original complaint due to neglect of the company. Hence they also vary in size and grammatical interpretation and opinions.

## 

## **Proposed Approach**

All 1000 emails were cross-validated 5-folds to find the optimum train-test pair in 80 : 20 ratio. 800 emails out of the 1000 were used for training the model and it was tested on the remaining 200 test data. Class labels were then assigned to each test instance by the system and on manual runs. Both the results were compared to identify the set of one or more classes for each test email.

A brief description of the proposed model is given below.

#### **Text - Preprocessing**

The objective is to extract negative sentiments from the unstructured text of the emails. The raw data consists of unnecessary details like contact numbers, email ids, customers’ names, and addresses that are not relevant to our problem statement. Hence the following noise reduction steps were performed - numerics removal, email id removal, punctuation removal, stray character removal (single letters between whitespaces), and stop-word identification and removal.

#### **Feature Extraction**

All possible phrases (depending on the sentiment) that can potentially be keywords for our classification were extracted. The feature selection methods like n-grams, RAKE, and TF-IDF were used to split the sentences into keyphrases of varying lengths. Further, the sentiments of the extracted keyphrases were checked according to their sentiment scores, and those with negative sentiments were shortlisted.

#### **Clustering**

The extracted features were fed into different clustering algorithms, including Flat and Hierarchical Clustering, which divided the features into (in particular with 6, 9, 12, 15) classes that served as labels to categorize the emails. The system-generated labels to our test set were compared with our manual labeling to assign one or more classes to each email. Thus the emails were identified under multiple complaint categories. Since the labels are unknown initially, this falls under the category of unsupervised learning, and hence clustering algorithms will serve the purpose.

# **Chapter 2**

# **Survey of NLP Tools**

## 

## 

## **Text Preprocessing**

It is the process of removing unnecessary details from the data to make it clean for further analysis.

The steps to cleaning are mentioned here:-

#### ***Numerics and Email-Id removal***

The text had a lot of Numeric Data in the form of Consignment Numbers, Tracking Ids, dates, and monetary amounts. Also, the customers often mentioned their email id for reference of prior complaints or tracking parcels. Both of these were handled using a combination of simple regular expressions. For numeric removal, a pattern of “/d+” was replaced, while for email Ids, a pattern of “/S+@/S” was used, where /d stands for all numeric digits and /S stands for non-whitespace characters.

#### ***Removal of Custom Punctuations***

#### The standard list of punctuations consists of *!"#$%&\'()\*+,-./:;<=>?@[\\]^\_`{|}~*

All punctuations except ‘!’, ’?’ were removed from the data, which can be done with the help of the ‘string’ module. A sentence ending with an exclamation mark ‘!’ expresses strong emotion, majorly negative in sentiment. Similarly, a question mark ‘?’ is indicative of an interrogative or demanding sentiment. These punctuations were kept as their presence would affect the sentiments of the customers and hence would affect our keyphrase selection later.

#### 

#### ***Removal of Stopwords***

After the removal of numerics and punctuations, a lot of stray single-letter characters were left in the text. For e.g., A consignment number, namely N3245670, was mentioned by a customer. After removing the numerics from it, the single character ‘N’ was left in the data, which served no purpose. Hence such characters were cleaned using the regular expression pattern of ‘\b\w\b’. All single characters were replaced with white space.

A **stop word** is a commonly used word (example: “the”, “a”, “in”, “an”) that search engines have been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words to take up space in our database or take up valuable processing time. For this, we can remove them easily by storing a list of words that you consider to stop words. The basic NLTK list of stopwords were considered along with a few custom additions (“sir”, “madam”, “hi” and many more) and these words were removed from the text for further feature extraction.

### Following is the list of Stopwords:

["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now", "would", "dear", "madamsir" , "sirmadam" , "hi", "rd" , "th" , "rs" ]

### 

### ***Challenges Encountered***

#### ***Part-of-speech tagging***

**POS tagging** is popular Natural Language Processing process that refers to categorizing words in a text in correspondence with a particular part of speech, depending on the definition of the word and its context.

In NLTK, the tagging was done with the ‘punkt’ module and ‘averaged\_perceptron\_tagger’. punkt module. This ‘punkt’ module tokenizer divides the given document into a list of sentences with the help of an unsupervised approach and builds a model for abbreviations, collocations, and sentence starting words.

The averaged\_perceptron\_tagger contains the pre-trained English [Part-of-Speech (POS)] <https://en.wikipedia.org/wiki/Part_of_speech> tagger in NLTK.

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

def pos\_tagging(sentence):

words = nltk.word\_tokenize(sentence)

return nltk.pos\_tag(words)

In SpaCy, the tagging was done with the help of ‘en\_core\_web\_sm’ model, which tokenizes the text and checks the part of speech of each token. The tokenize attribute has an accuracy of 1.00 and the POS tagging has an accuracy of 0.97, according to the library documentation.

d=dict()

def pos\_tagging(sentence):

nlp = spacy.load('en\_core\_web\_sm')

doc = nlp(sentence)

for word in doc:

if word.pos\_ == "PROPN":

d[word.text] = True

df['POS\_tags'] = df['Email'].apply(pos\_tagging)

|  |  |  |  |
| --- | --- | --- | --- |
| TERM | TAG | TERM | TAG |
| Noun | NN | Proper Noun | NNP |
| Adjective | JJ | Pronoun | PRP |
| Verb | VB | Verb(Past) | VBD |

**Figure 2.1**

Here is the glimpse of some words extracted from the email and their corresponding POS tags:

[(Email\_words, POS\_tags)]

[(Consignment, JJ), (No, DT), (:, :), (Z342116...

[(HORRIBLE, NNP), (EXPERIENCE, NNP), (WITH, NN...

[(Hi, NNP), (,, ,), (My, NNP), (Brother, NNP),...

[(I, PRP), (had, VBD), (booked, VBN), (the, DT

***Removal of proper nouns***

We wanted to remove the proper nouns, including Customer Names and Places, from our dataset as they impart no contribution to detecting the Sentiment of the complaint. We used the NLTK library with WordNet Dictionaries and SpaCy library. After the tagging, a dictionary of only the names were extracted from the extended dictionary. Their frequency was checked. The words with a frequency lower than 20 were separated into another dictionary.

However, the problem was, according to the corpora, Indian proper names were not identified correctly. It was tagging some non-proper-noun words under NNP, like “Horrible : NNP ”, which would affect the outcome of our problem statement. Moreover, it removes extra words than needed while not identifying some target words

We also tried to remove the proper nouns from the dataset by manually constructing a list of all the names in our test dataset and removing them from each email.

Following is the manually extracted list of proper names from the training dataset:

['Sumana', 'Mukherjee', 'Mr.', 'Saanjeev', 'Das', 'Ghosh', 'Mr.Ujjal', 'Siddhartha', 'Mr', 'Deshmukh', 'Sagar', 'Shilimkar', 'Dhiren', 'Nirbhay', 'Jaiswal', 'Shivaji', 'Nagar', 'Krish', 'Dewakar', 'Awasthi', 'RS', 'Parthoguchit', 'Upama', 'Mr.Prakash', 'Ghatbande', 'Yogesh', 'Ranga', 'Rahul', 'Avinash', 'Ankush', 'Aman', 'Rastogi', 'Jitender', 'setthi', 'sanjay', 'Nitin', 'mahesh', 'Rana', 'Lalit', 'Sandeep', 'Singh', 'Sarita', 'Yadav', 'Sibayan', 'Chakraborty', 'addressshubham', 'khare', 'CH.', 'Jalandhar', 'Ravi', 'Srinivas', 'Prashanti', 'A.R. Khan', 'Sowmik', 'CA.', 'Dharam', 'Pal', 'Singla', 'sir', 'Sir', 'Madam', 'madam', 'Bir', 'inder', 'singh', 'Anjan', 'Bhattacharya', 'Samir', 'Kanani', 'Ballia', 'Pawan', 'Dubey', 'Ravinder', 'Saptarshi', 'Raha', 'Gupta', 'Puneet', 'Panwar', 'Subhasish', 'chakraborty', 'VINAY', 'MS',]

However, the problem faced using this approach is that there can be multiple spellings for a single Indian name, and most of the names were not occurring in the training dataset, which was there in the test dataset. Hence the algorithm was not working well on the test dataset. So, even though the logic was implemented correctly, the algorithm still had some loopholes. Hence, we had to look for other options to aid our Feature Extraction process.

Figure 2.2 shows the comparison between one of the sample emails along with their cleaned versions. The proper noun removal algorithm was supposed to remove words like “Mr. Sanjeev Das” completely. However, it only removed “Mr. Sanjeev” and kept the word surname “Das” intact after running. Similarly, other proper nouns like Delhi, Jogomaya etc. were supposed to be removed ideally. However, most corporas failed to identify these words and hence were not producing any results on them.

On experimentation with cleaning without removing the proper nouns and performing the other steps of numeric, email ids, punctuation, and stopword removal, the emails produced much better results, as demonstrated in the rightmost column of Fig. 2.2. Thus, these were the final steps that we went ahead with.

|  |  |  |
| --- | --- | --- |
| Original Email | Preprocessing With Proper Noun Removed | Preprocessing Without Proper Noun Removed |
| Consignment No: Z34211660 from Sumana Mukherjee 3rd floor, Xezal Park, Opp: Baba Guest House, Near TPDDL, Panchvati, booked at XYZ Azadpur Delhi-110033 Office destination to Mr. Saanjeev Das, Jogomaya Memorial Institute, Singur Ratanpur, Hooghly PIN-712409, M: XXXXXXXXXX, XXXXXXXXXX. Since 23/10/2014 this Packet send to wrong address to Kalna, Burdwan by operation team at XYZ Serampur, Neel Jheel Apartment, Hoohly, Contact Person Mr. S. Ghosh M: XXXXXXXXXX, & Mr.Ujjal M: XXXXXXXXXX. Total Expected Loss for this consignment not delivered to proper destination timely is Rs. 15000/-(Rupees Fisteen thousand only). Contact to Siddhartha Mukherjee, M: XXXXXXXXXX, XXXXXXXXXX, XXXXXXXXXX, eaml: XXXXXXXX@gmail.com | Consignment No rd floor Xezal Park Opp Baba Guest House Near TPDDL Panchvati booked XYZ Azadpur Delhi Office destination Das Jogomaya Memorial Institute Singur Ratanpur Hooghly PIN Since Packet send wrong address Kalna Burdwan operation team XYZ Serampur Neel Jheel Apartment Hoohly Contact Person Total Expected Loss consignment not delivered proper destination timely Rupees Fisteen thousand only. Contact Mukherjee eaml | Consignment No Sumana Mukherjee floor Xezal Park Opp Baba Guest House Near TPDDL Panchvati booked XYZ Azadpur Delhi Office destination Mr Saanjeev Das Jogomaya Memorial Institute Singur Ratanpur Hooghly PIN Since Packet send wrong address Kalna Burdwan operation team XYZ Serampur Neel Jheel Apartment Hoohly Contact Person Mr Ghosh Mr Ujjal Total Expected Loss consignment not delivered proper destination timely Rupees Fisteen thousand Contact Siddhartha Mukherjee eaml |

**Figure 2.2**

#### ***Stemming and Lemmatization***

**Stemming** is the text preprocessing algorithm which slices the word from prefix or suffix as per the need without checking the meaning of the base word. On the other hand, **Lemmatization** is the technique which is used for converting the word to its base form by linking meaning to the word. Basically, in this step, the different forms of the word can be grouped together, which makes the analysis easier.

|  |  |  |
| --- | --- | --- |
| **Word** | **Stemming** | **Lemmatization** |
| flies | Fli | Fly |
| wolves | Wolve | Wolf |
| booked | Book | Book |

**Figure 2.3**

In NLTK, this can be done using *WordNetLemmatizer*which is embedded inside module *nltk.stem*

import nltk

nltk.download('wordnet')

from nltk.stem.wordnet import WordNetLemmatizer

obj\_lemmatizer = WordNetLemmatizer()

obj\_lemmatizer.lemmatize(word)

We expected Lemmatization to improve our feature extraction procedure. However, the problem with it was that it was changing words like “worst” to “bad”, which was impacting the intensity of a negative review. Moreover, words which were not identified in the NLTK WordNet Corpora were being changed to a close English word, which was affecting our results.

#### ***SpellCheck and Normalization***

Normalization is an attempt to reduce its randomness, bringing it closer to a predefined “standard”. This helps us to reduce the amount of different information that the computer has to deal with, and therefore improves efficiency. Stemming and Lemmatization can be considered as Normalization techniques. However, for really unstructured data like our sample reviews, these techniques do not come into much use. Hence, we tried to make use of duplicate letter reduction techniques and spell checkers to aid our process.

*Method 1:*

The TextBlob library in Python, with its inbuilt NLTK library along with the WordNet dictionary, was used for this purpose. The code handles misspellings by testing each token in the text against the spellcheck() function, which returns a confidence score in relation to whether the word is present in the WordNet database or not. If the Confidence score is not 1.0, implying the word does not match any word in WordNet, the word is returned after stripping it off of unnecessary duplicate characters, else the word is lemmatized to its base form and returned.

def misspelling(word):

w = Word(word)

c = w.spellcheck()

if c[0][1] != 1.0:

word = removeDuplicates(word)

else:

word = w.lemmatize()

return word

This is working fine for most words that match in WordNet, but it is removing duplicate letters from Indian words like, “Jheel” becomes “Jhel”, “Mukherjee” becomes “Mukherje”, “Rupees” becomes “Rupes”. However, these words are not related to our problem statement, so they can be ignored. But, it is not correcting an actual English word error, like, “Fisteen” which should be “Fifteen” is neither corrected nor modified by it.

*Method 2 :*

We also did it using pyenchant spell checker which was providing a better result for English words in terms of misspellings. However, it was modifying all non-english words to its closest English words that matches the spelling and returning the text. This was converting many important words to something different. This resulted in loss of information.

nlp = spacy.load("en\_core\_web\_md")

dictionary = enchant.Dict("en\_US")

def misspelling(word):

if dictionary.check(word) == False:

new\_word ="".join(OrderedDict.fromkeys(word))

list = dictionary.suggest(new\_word)

score = [(nlp(i)).similarity(nlp(new\_Word)) for i in list]

if list == []:

return new\_word

pos = score.index(max(score))

new\_word = lemmatizer.lemmatize(l[pos])

else:

new\_word = word

return new\_word

This, on the other hand, is working for only English words, and in the case of Indian words or names, it is giving a completely undesired output. “Kewal” becomes “Cakewalk”, “Baba” becomes “Bob”, and many more. These are proper nouns so that they can be ignored. And for the actual misspelled word “Fisteen”, it is converting it to “Listen” rather than “Fifteen”.

## **Semantics**

Natural Language Processing in the context of linguistic study has four major steps that include Lexical Analysis, Syntactic Analysis, Semantic Analysis and Pragmatic Analysis.

Lexical Analysis is dividing the whole text document into paragraphs, sentences and words. Whereas, Syntactic analysis, commonly known as parsing, is for checking relationships between strings of words or sentences, separated by delimiters, which can be used in punctuation correction or dialogue systems. Semantic analysis, on the other hand, is a deeper dive into the study of language.

Structure of a sentence does not clearly convey the sense in which it is used. They might have the same syntactic structure but their meaning changes with constituent words. Thus, Semantics refers to the logical meaning of a sentence or phrase in language. This approach assigns sentiment values to words and this can be done by looking into the synonyms and antonyms of those words.

However, the most challenging issue that stands between a syntactic utterance and its logical form is ambiguity. Pragmatic analysis comes into play here such that it takes the results of Semantic analysis and identifies its specific context. Moreover, to tackle this, various Python libraries like NLTK, TextBlob and online knowledge bases namely WordNet, SentiWordNet come into use.

### **WordNet**

WordNet 3.1 is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with ‘<http://wordnetweb.princeton.edu/perl/webwn>’.

WordNet groups words having similar meaning together, thus it’s mechanism is similar to that of a Thesaurus. However, it does not just club similar meaning words together, but in fact checks the relationship between words and in what sense they are just before merging them. WordNet even labels the semantic relationship among words.

A simple WordNet Search of the word “Consignment” returns results as follows:

[S:](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=consignment&i=0&h=000#c) (n) [cargo](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=cargo), [lading](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=lading), [freight](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=freight), [load](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=load), [loading](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=loading), [payload](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=payload), [shipment](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=shipment), **consignment** (goods carried by a large vehicle)

[S:](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=consignment&i=1&h=000#c) (n) [commitment](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=commitment), [committal](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=committal), **consignment** (the official act of consigning a person to confinement (as in a prison or mental hospital))

[S:](http://wordnetweb.princeton.edu/perl/webwn?o2=&o0=1&o8=1&o1=1&o7=&o5=&o9=&o6=&o3=&o4=&s=consignment&i=2&h=000#c) (n) **consignment** (the delivery of goods for sale or disposal)

**Figure 2.4**

Each keyphrase is provided as input to WordNet and relations like synonym, hyponym, hypernym, meronym and antonym are applied. Each relation gives an output called synset. A synset is a set of context-relevant words.

### **SentiWordNet**

SentiWordNet is lexical resource for opinion mining which is available publicly. It associates sentiment information to each WordNet synset.

SentiWordNet works on the subjectivity of words. It is able to distinguish between positive, negative and neutral sentiments by assigning a score against each word or phrase.

For our Courier Complaint problem Statement, we are looking for customer phrases with negative sentiment. Purely frequency-based methods are unable to extract such keyphrases, which is where SentiWordnet comes into play. It helps in eliminating positive and neutral sentiment phrases. This in effect highlights the negative sentiment associated with customer complaints which should help us in identifying the nature of the comments better.

Here are some important phrases from our dataset along with their positive and negative score as per SentiWordNet Databases:

|  |  |  |
| --- | --- | --- |
| Phrase | Positive Score | Negative Score |
| Consignment Number | 0.0 | 0.375 |
| Not Delivered | 0.0 | 0.625 |
| Able track | 0.125 | 0.0 |
| Irresponsible Behaviour | 0.5 | 0.0 |
| Consignment Not Yet Delivered | 0.25 | 0.625 |

**Figure 2.5**

It is clear from the above table that the phrases giving Negative Score higher than positive score are of more significance to our problem statement than the others and hence these can be extracted as keyphrases for further processing. Also, for certain phrases it is not giving the desired score, so we cannot fully rely on this form of Sentiment Analysis.

## **Feature Selection**

The process of selecting the most important features to feed into machine learning algorithms is called Feature Selection. It is one of the main components of Feature Engineering. This eliminates redundant or irrelevant variables and narrows down the input variables to the set of most relevant features for the machine learning model.

It is advisable to perform feature selection before inputting variables to the machine learning models for the mentioned benefits:-

* We get simpler models that are not too complex and easy to explain.
* It takes shorter training times i.e. a more precise subset of features decreases the amount of time needed to train a model.
* It also leads to variance reduction i.e. increase the precision of the estimates that can be obtained for a given simulation
* It also helps reduce the dimensionality, which is highly beneficial because as dimensionality and the number of features increases, the volume of space increases so fast that the available data become limited. PCA or Principal Component Analysis feature selection may be used to reduce dimensionality.

### **Feature Selection Methods**

Once the data is cleaned, the remaining text is grouped into key phrases and their priority with respect to the problem statement (in our case, complaints) is checked and they are allotted weights in order to classify them into relevant groups. This can be done in following ways:-

* ***N- grams*** - The sentences can be grouped in unigrams, bigrams, trigrams and so on and the frequency of the keyphrases are checked to identify the top keyphrases for classification.
* ***TF-IDF*** - TF-IDF stands for term frequency — inverse document frequency. It is a scoring measure widely used in information retrieval or summarization. It is intended to reflect how relevant a term is in a given document.
* ***RAKE (Rapid Automatic Keyword Extraction)*** – RAKE splits the text into sentences and generates candidate phrases based on user- defined parameters. RAKE focuses on the phrases with non-stop words more than those with stop words and assigns them scores accordingly, thus resulting in more valid classes. This overcomes the drawback of the TF-IDF technique.

## **Clustering**

Clustering is the process of grouping data in such a way that objects falling into the cluster would have similar properties and those lying-in different clusters would differ from each other in some aspect or the other. Clustering algorithms use some form of a similarity metric to form these groups. Distance based metrics like the Euclidean Distance is one of the most common forms of metric used in determining clusters.

### **Types of Clustering**

For grouping keywords, we need to attain two major goals: a similarity between one data point with another and a distinction of those similar data points with others which differ heuristically from those points. Now, we know that our data is not limited in terms of dimensions, we have data that is multidimensional in nature. The clustering algorithm that we intend to use should successfully cross this hurdle.

### **Flat Clustering**

#### ***k - Means Clustering:*** - k-Means is one of the most commonly used unsupervised clustering approach that is indeed very simple to use. Using this algorithm, we perform classification on a given dataset through a predetermined number of clusters say “k”. Each cluster is assigned a designated cluster center and they are placed as far from each other as possible. Subsequently, each point belonging to that input gets associated to its nearest centroid till no point is left unassigned. Once it is done, the centers are re-calculated and the above steps are repeated. The algorithm converges at a point where the centroids cannot move any further. This algorithm targets to minimize an objective function called the squared error function F(V) :

Where, |||| is the distance between and . is the count of data in a cluster. C is the number of cluster centroids.

However, this reflects that this form of clustering fails for non-linear as well as categorical data and it cannot handle outliers well.

### **Hierarchical Clustering**

Divisive and Agglomerative clustering are the main two approaches that falls under Hierarical Clustering. The implementation family contains two algorithms respectively, the divisive Analysis and AGNES (Agglomerative Nesting) for each of the approaches**.**

* ***Divisive Analysis*** *-* The divisive clustering algorithm is a top-down approach which begins with one single cluster where all the data points belong. Further it is split into multiple clusters and the data points get reassigned to each of the clusters on the basis of the nearest distance measure of the pairwise distance between the data points. These distance measures can be Ward’s Distance, Centroid Distance, average linkage, complete linkage or single linkage. Ideally, the algorithm continues until each data has its own cluster.

#### ***Agglomerative Nesting*** *-* Agglomerative clustering algorithm is a bottom to up approach which considers the fact that each data point has its own cluster, i.e., if there are n data rows, then the algorithm begins with n clusters initially. Then, iteratively, clusters that are most similar – again based on the distances as measured in Divisive Analysis – are now combined to form a larger cluster. The iterations are performed until we are left with one huge cluster that contains all the data-points.

### **Dynamic Clustering**

This particular type of clustering embraces different scenarios: dynamic features, dynamic data objects and dynamic clusters. Many challenges can be envisioned with each of these clustering scenarios especially in the presence of non-stationarity, data drift and shifts that take various change rates (slow, medium, fast, random, gradual, abrupt and cyclic).

*Dynamic clustering* as a form of unsupervised online/incremental machine learning considers two concepts: (1) incrementality of the learning methods to devise the clustering model and (2) self-adaptation of the learned model (parameters and structure). Hence, incrementality tackles the problem of time-intensive re-training and memory constraints. As a consequence, dynamic aspects (e.g., behavior, structural elements) of the model to be learned can be captured via adaptation of the current model.

# **Chapter 3**

# **Feature Extraction and Analysis**

## **N-Grams**

N-Gram is a continuous sequence of *n*-words from a given sample. Basically, the sentences can be grouped in bigrams, trigrams, tetragrams and so on, and the frequency of the keyphrases are checked to identify the list of keyphrases for classification. Since higher order n-grams are sparsely populated, we decide to trim these NGrams having frequency one in the training corpus, because chances are that these N-Grams are not good keyphrases.

This algorithm includes splitting of the text into words which are referred to as tokens, and grouping these tokens into keyphrases. Further, the frequency of the keyphrases is calculated to extract the top keyphrases for classification.

def find\_ngrams(input\_sequence):

n , ngrams = 2, [] # n = 2 for bigram,

tokens = input\_sequence.split()

for i in range(0 , len(tokens) , n):

ngram = tokens[i:i+n]

ngram = ' '.join(ngram)

ngrams.append(ngram)

return ngrams

Following are the Top 20 Ngrams extracted using NLTK packages in Python. Inbuilt functions found in the package were used.

|  |  |  |  |
| --- | --- | --- | --- |
| **KeyPhrases** | **Frequency** | **KeyPhrases** | **Frequency** |
| not delivered | 63 | NOT DELIVERED | 17 |
| customer care | 45 | sent parcel | 17 |
| courier service | 44 | courier not | 17 |
| not received | 32 | not yet | 17 |
| no one | 28 | not reached | 16 |
| sent courier | 27 | courier company | 15 |
| still not | 24 | courier services | 15 |
| customer service | 24 | also not | 14 |
| till date | 22 | Thanks Regards | 14 |
| consignment no | 22 | not able | 13 |

## **Figure 3.1 Bigrams**

|  |  |  |  |
| --- | --- | --- | --- |
| **KeyPhrases** | **Frequency** | **KeyPhrases** | **Frequency** |
| XYZ courier service | 11 | XYZ worst courier | 4 |
| Subject DAYS PASSED | 10 | DAYS PASSED COURIER | 5 |
| COURIER NOT DELIVERED | 10 | Pallavi Pawar Sent | 5 |
| not delivered till | 9 | called many times | 4 |
| not reached destination | 8 | sent one courier | 4 |
| XYZ Courier Cargo | 6 | Received damaged condition | 4 |
| not yet delivered | 6 | Courier Cargo Ltd | 4 |
| courier not delivered | 5 | Received via Manifest | 4 |
| XYZ courier services | 5 | worst customer service | 4 |
| customer care number | 5 | not yet received | 4 |

## **Figure 3.2 Trigrams**

|  |  |
| --- | --- |
| **Keyphrases** | **Frequency** |
| PASSED COURIER NOT DELIVERED | 5 |
| not delivered till date | 4 |
| commitment reach destination positively | 3 |
| Consignment Details Shipment No | 3 |
| XYZ worst courier service | 3 |
| encountered regard consignment regret | 3 |
| service standards fallen short | 3 |
| may legally privileged not | 3 |
| intended recipient distribute copy | 3 |
| immediately destroy copies message | 3 |

**Figure 3.3 Tetragrams**

A glimpse of the extracted n-grams show that bigrams and trigrams produce much superior keyphrases than tetragrams. This is possibly because length of tetragrams is more which is why it is failing to extract meaningful keyphrases to the problem statement, since it is extracting unrelated words along with the relevant ones. So, the bigrams and trigrams extracted are fed into frequency-based algorithms, namely TF-IDF and RAKE. These approaches extract phrases based on a score and this score can be used to identify the topmost keyphrases relevant to the problem statement.

## **TF-IDF**

TF-IDF or (Term Frequency (TF) — Inverse Document Frequency (IDF) ) is a technique which is used to find the meaning of sentences consisting of words. It’s a score which the

machine keeps where it evaluates the words used in a sentence and measures it’s usage compared to words used in the entire document. In information retrieval, TF-IDF is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

### **How is TF-IDF calculated?**

TF-IDF for a word in a document is calculated by multiplying two different metrics:

* The **term frequency** (TF) of a word in a document. The simplest way of calculating this is getting a raw count of instances of a word in a document. Then the frequency can be adjusted by length of a document, or by that raw frequency of the most frequent word in it.
* The **inverse document frequency** (IDF) of the word across a set of documents. This means how common or rare a word is in the entire document set.
* So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

This algorithmic approach consists of the following steps:

1. **Tokenize the text**: First step is to tokenize the text into sentences, and further sentences will be tokenized into key-phrases. During tokenization, it might be possible that words like ‘the’, ‘my’ etc. will become the part of keyphrases which are certainly irrelevant to the problem statement, these words are commonly occurring stopwords. For this process, one can use Tfidfvectorizer from sklearn library to vectorize the documents into bigrams and trigrams and then they are fit into the model using fit\_transform to assign a score to each of the keyphrases in the documents. Also, a list of english stopwords has been passed to make sure that these words will be removed during tokenization.

This would generate a sparse matrix, with a token number given to each extracted candidate according to the documents, along with a score.

1. **Calculation of TF-IDF Score**: In the next step , each candidate token’s rank would be calculated by adding up the respective TF-IDF scores of each candidate token in all of the documents.

Mathematically, the TF-IDF score is calculated as follows:

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidfconverter = TfidfVectorizer(min\_df = 3, ngram\_range=(2,3), stop\_words='english', use\_idf=True)

tfIdf = tfidfconverter.fit\_transform(df['letter\_removal'])

features = (tfidfconverter.get\_feature\_names())

print(features)

sums = tfIdf.sum(axis = 0)

data1 = []

for col, term in enumerate(features):

    data1.append( (term, sums[0, col] ))

ranking = pd.DataFrame(data1, columns = ['term','rank'])

words = (ranking.sort\_values('rank', ascending = False))

Following table shows the list of top 20 keyphrases extracted using tfidf and their scores

|  |  |  |  |
| --- | --- | --- | --- |
| **Keyphrases** | **Score** | **Keyphrases** | **Score** |
| XYZ courier | 4.863693 | parcel delivered | 1.895223 |
| customer care | 4.139959 | service XYZ | 1.814006 |
| courier service | 3.638945 | XYZ courier service | 1.704512 |
| till date | 2.753451 | courier delivered | 1.702043 |
| XYZ office | 2.715193 | booked parcel | 1.61057 |
| consignment number | 2.396765 | delivery boy | 1.595938 |
| tracking number | 2.16037 | new delhi | 1.548012 |
| sent courier | 2.126402 | soon possible | 1.433081 |
| customer service | 1.906643 | till today | 1.418858 |
| courier company | 1.903459 | send courier | 1.385483 |

**Figure 3.4 Top Keyphrases from TFIDF**

Fig. 3.4 depicts that TF-IDF scores alone cannot be used to selection of Candidate keyphrases relevant to the problem statement.

For e.g., phrases like “till date” has a higher score than “Courier delivered”.

This is based on the frequency of the occurring keywords in the documents which determines its score and hence it would not be a correct indication of the sentiment it bears. Thus, for a better understanding of the problem statement and for optimum selection of the Candidate keyphrases, one needs to look into the sentiments of these keyphrases in the documents and the context in which they are used.

## 

## **RAKE**

*Rapid Automatic Keyword Extraction* (RAKE) is a keyword extraction method which uses a list of stop-words and phrase delimiters to detect the most relevant words or phrases in a piece of text. This method splits the text into a list of words and removes stop-words from that list. This returns a list of *content words*. Once the text has been split, the algorithm creates a matrix of word co-occurrences. Each row shows the major metrics to calculate RAKE Score are: -

1. *Degree of a word in the matrix* that favors words occurring frequently and in longer candidates.
2. *Word Frequency,* that is the number of times the word appears in the text regardless of the words with which they co occur in the text.
3. *Degree of the word divided by its frequency* that favors words occurring in longer candidates.

The final score of each candidate is calculated as the sum of each of its member keywords.

### **Steps to be Approached**

1. Stopwords: - NLTK natural English Language corpus of stopwords, excluding the words like “no”, “not”, “nor” and adding a few custom stopwords to it like “would”, “dear”, “madam”, “sir”, “rs”, “hi” would be used.

2. Punctuation: - The punctuations had already been specified in the pre-processing part, wherein all punctuation from the default NLTK list other than “!”, “?” were removed (because they affect negative sentiment of the text)

3. The maximum length and minimum length of the candidate keywords would be chosen to be 4 and 2 respectively.

4. The scores given to the extracted keywords were calculated as the *degree of the word divided by its frequency.*

A keyword or keyphrase is chosen if its score belongs to the top T scores where T is the number of keywords you want to extract. According to research, T defaults to one third of the content words in the document.

from RAKE\_nltk import Metric, RAKE

from collections import OrderedDict

r = RAKE(stopwords = stop\_words, max\_length=4 , min\_length=2)

li=[]

for index,row in df.iterrows():

    plot = row['letter\_removal']

    r.extract\_keywords\_from\_text(plot)

    ra = r.get\_ranked\_phrases\_with\_scores()

    for k in ra:

      li.append(k)

List of stopwords used :

["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now", "would", "dear", "madamsir" , "sirmadam" , "hi" ]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bigrams** | | **Trigrams** | | **Tetragrams** | |
| **KeyPhrase** | **Score** | **KeyPhrase** | **Score** | **KeyPhrase** | **Score** |
| Not received | 5 | Yet no delivered | 8.5 | Highly irresponsible idiot employee | 16 |
| Consignment no | 5 | Charged high price | 9 | Shipment id tracking no | 16 |
| Delivery boy | 4.667 | Past five days | 9 | average courier service must | 15.25 |

**Figure 3.5 RAKE Scores of some Keyphrases**

Fig. 3.5 shows one phrase each possessing a negative sentiment, a neutral sentiment and a positive sentiment chosen manually from the bigrams, trigrams and tetragrams extracted, for demonstration. In the first column of bigrams, “not received” is a negative phrase which is more relevant to the problem than “consignment no” or “Delivery boy” as a bigram itself. However, their RAKE scores are almost same and thus do not reflect their relevance clearly. Had the phrase been something in lines of “Delivery boy was rude” would it have made sense to the problem, however as a bigram alone it doesn’t have much significance. Similarly, it can be noticed that the tetragrams have received scores in the range of 15-16 irrespective of their relevance. This is due to their longer length. This biasness of RAKE algorithm of assigning higher scores to the longer n-grams can cause incorrect results.

On extracting keyphrases using RAKE Model, we get keyphrases with negative, neutral and positive sentiments, all of which are not relevant to the problem Statement. RAKE is clearly giving higher score to longer phrases wherein the score of tetragrams > trigrams > bigrams.  Clearly, this frequency-based method is slightly biased towards longer phrases, giving them a higher importance in the document. This is where Sentiment-based extractions come into play, to further shortlist the KeyPhrases.

**Chapter 4**

**Sentiment Analysis**

The aim of the experiment is to identify Complaint classes from the emails. To identify such Complaint classes, we need to identify phrases having negative implication. From TFIDF and RAKE result, we can see the frequency-based approaches extract keyphrases of positive, negative and neutral sentiments, all of which are not relevant to problem statement. Hence, we need to go for Sentiment analysis in NLP which will help us to identify relevant keyphrases to the problem statement. This is why there are corporas like WordNet and SentiWordnet which helps us understand the context and sentiment of text data, about which we discussed in chapter 2.

Here, in this chapter, we are mainly focusing on how SentiWordNet Corporas can be used for our purpose. Along with this, we will also see how sentimental analysis can be performed using python library TextBlob.

**TextBlob**: It is a python library for Natural Language Processing (NLP). It uses NLTK (Natural Language Tool Kit) to achieve the tasks.

It returns **polarity** and **subjectivity** of a sentence.

* **Subjectivity: It quantifies the amount of personal opinion and factual information contained in the text.**

**Higher subjectivity means that the text contains personal opinion rather than factual information**.

Subjectivity lies between [0,1]

* **Polarity**: It defines the sentiment of the sentence.

Polarity lies between [-1,1], -1 defines a negative sentiment and 1 defines a positive sentiment.

For the experiment, the keyphrases extracted from n-grams, particularly bigrams and tetragrams were fed into the RAKE and TFIDF algorithms to test their sentiments.

**RAKE Results :-**

A keyword or keyphrase is chosen if its score belongs to the top T scores where T is the number of keywords you want to extract. According to research, T defaults to one third of the content words in the document. So, first a list of candidate keyphrases with score >12 was selected. A few of the top phrases looked like this,

sir madam good evening: 16.0

sent one complementary gift: 16.0

one else doesn say: 14.25

delhi customer care executives: 15.0

customer care email ids: 14.666666666666666

book speed delivering consignment: 13.5

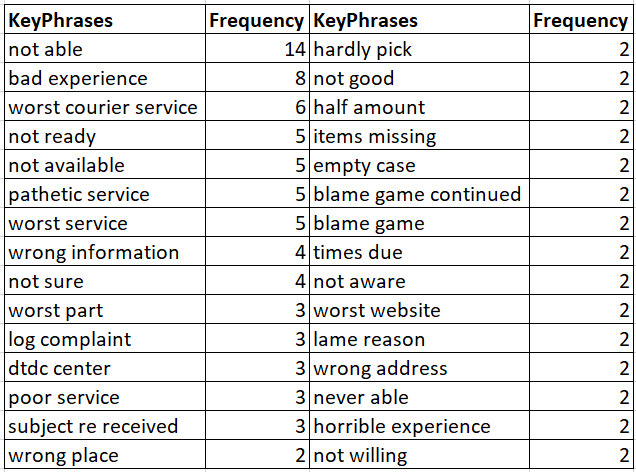
no service area requested: 13.75

This clearly shows that longer phrases were assigned higher scores and as a result a lot of phrases which were extracted were not aiding the purpose of the problem statement.

**TextBlob:**

RAKE extracted 237 keyphrases from the dataset of 300 emails and 430 keyphrases from 500 emails and **1025 from 800 emails** Train set.

The keyphrases were filtered by frequency count to extract phrases that occurred at least twice and it was reduced to **95 phrases** from 1025.



**Fig.4.1 RAKE TextBlob**

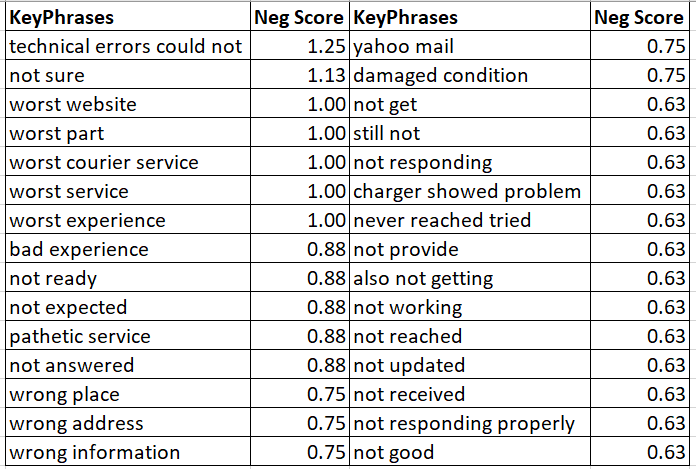
**SentiWordNet:**

SentiWordnet gives the output of positive, negative and neutral scores of the keyphrases according to their sentiment.

From the extracted list of keyphrases, two experiments were done. The phrases having a negative score greater than zero, that is phrases having some negative implications were extracted, which are **360** in number.

Further, to reduce the phrases, a filtering was done by extracting the phrases having a negative score greater than it’s respective positive score, which comes out to be **306** in number, which shows **15%** reduction.

Following table shows the list of KeyPhrases which are having negative score more than positive.



**Fig. 4.2 RAKE SentiWordNet**

**TFIDF Results:-**

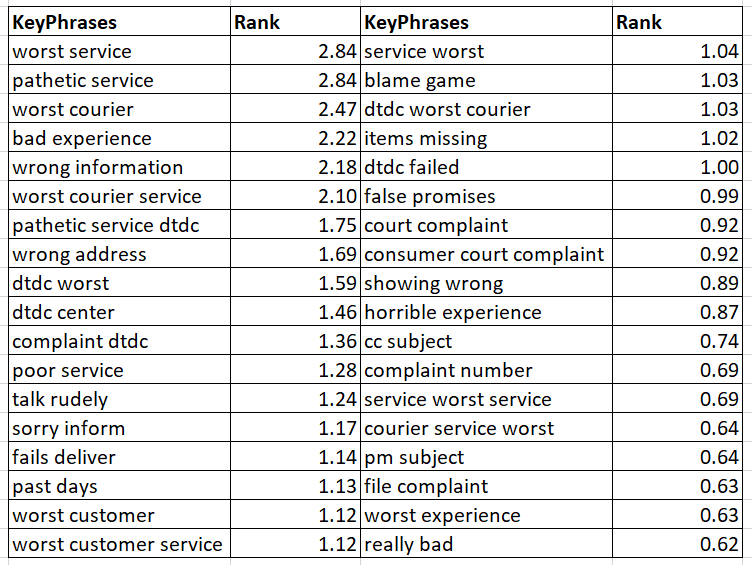
Each of the 800 complaints is represented by 2454 features. This is a segment of the list of the extracted features.

['action courier', 'bangalore branch', 'branch near', 'checked xyz', 'consignment reached', 'courier delivered', 'courier th', 'days delivered', 'deliver parcel', 'delivered received', 'delivery th', 'don know', 'xyz employees', 'xyz services', 'false promises', 'hi booked', 'kind service', 'local office', 'nd time', 'office returned', 'parcel xyz', 'picking phone', 'reach days', 'regular customer', 'said courier', 'sent th', 'shipment dispatched', 'south delhi', 'super franchisee', 'th october', 'today parcel', 'tried calling', 'waiting courier']

**TextBlob**:

After that, all candidates were checked for sentiment using the TextBlob.Sentiment command and the ones with negative sentiment > 0.0 were extracted along with their TF-IDF rank.

Out of the 2454 features, about **106** of them were of negative sentiment.



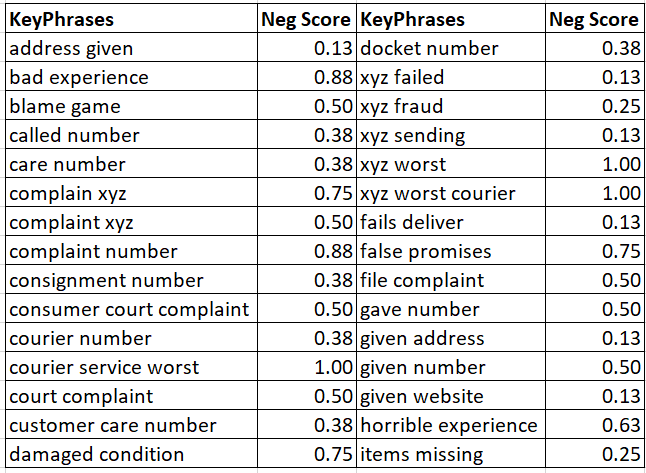
**Fig. 4.3 TextBlob Analysis of TF-IDF Keyphrases**

**SentiWordNet** :

From the extracted list of keyphrases, a similar experiment was performed as in RAKE. The phrases having a negative score greater than zero, that is phrases having some negative implications were extracted, which are **328** in number.

Further, to reduce the phrases, a filtering was done by extracting the phrases having a negative score greater than its respective positive score, which comes out to be **249** in number, which shows **24.08%** reduction.

Following table shows the list of keyphrases which are having negative score more than positive.

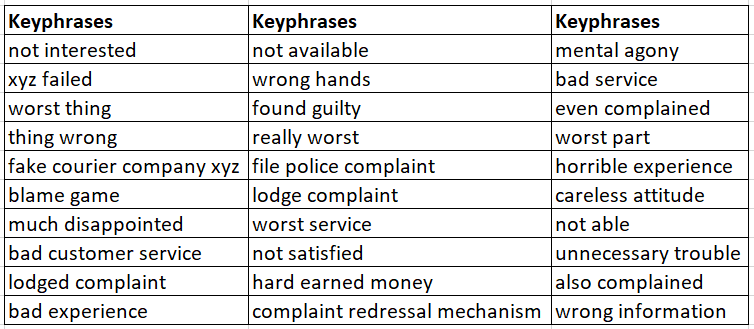


**Fig. 4.4 SentiWordNet Analysis of TF-IDF Keyphrases**

Now since the list of extracted keyphrases are quite huge and some of the unrelevant keyphrases are termed as negative by TextBlob, but not by SentiWordNet, and conversely. Hence, to identify the most relevant ones to the problem statement. The following steps were conducted to achieve this :-

* One way it can be done is by obtaining those keyphrases which falls at the intersection of both the sentiment analysing tools used by us.
* Thus, we extract the common keyphrases of TextBlob and SentiWordNet from RAKE and TFIDF individually to obtain two lists of keyphrases.

**Common KeyPhrases obtained using RAKE:**



**Fig. 4.5 RAKE Final List**

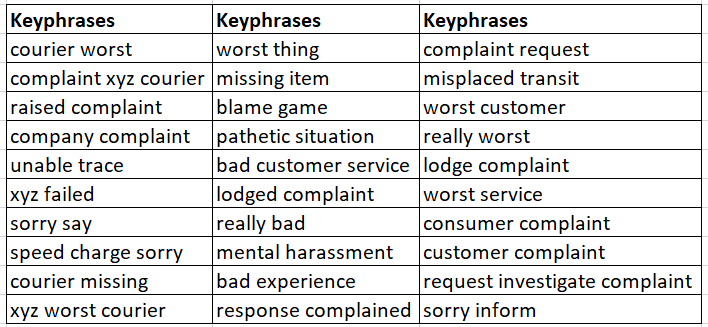
The accuracy of the RAKE and TFIDF was calculated manually using a simple formula: -

Accuracy = (No. of relevant negative phrases / No. of Negative phrases)

From the list of RAKE, the phrases like *[‘thing wrong’, ‘hard earned money’, ‘not able’, etc.]* were not relevant to our problem.

**Accuracy of RAKE**: - 50/54 = 92.6% (No. of relevant negative phrases / No. of Negative phrases)

**Common KeyPhrases obtained using TF-IDF:**

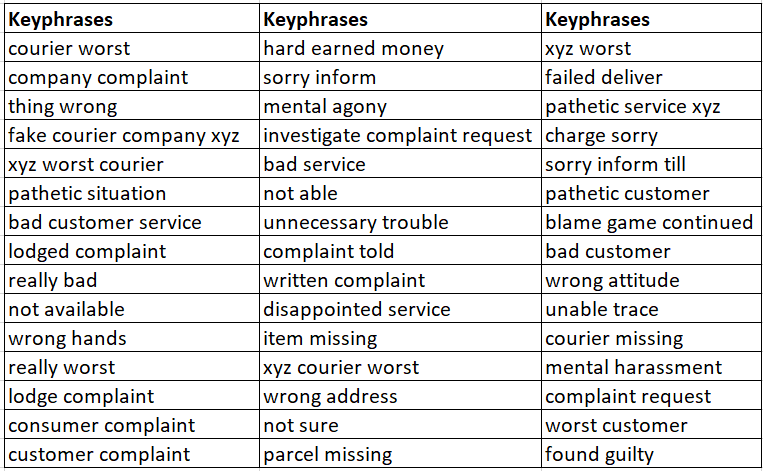


**Fig. 4.6 TF-IDF Final List**

From the list of TF-IDF, the phrases like *[ ‘sorry say’, ‘speed charge sorry’, ‘sorry inform’, etc.]* were not relevant to our problem.

**Accuracy of TFIDF**: - 68 / 76 = 89.40 % (No. of relevant negative phrases / No. of Negative phrases)

Next step is to take the union of these two lists is taken obtain our final lists of keyphrases which serve as the candidate keyphrases. The table below shows the list of the extracted candidate keyphrases. This list depicts the most relevant keyphrases out of the emails dataset that has negative sentiment and can serve in the construction of labels into which the emails are to be classified later.



**Fig. 4.7 Union of RAKE and TFIDF Lists**

**Accuracy of Final List: -** 98/108 = 90.7 % (No. of relevant negative phrases / No. of Negative phrases)

**Summary of the Steps: -**

The common keyphrases obtained using TextBlob analysis and SentiWordNet analysis were extracted to reduce the list further.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TextBlob** | **SentiWordNet** | **Common to Both** |
| **RAKE** | 95 | 306 | 54 |
| **TFIDF** | 106 | 249 | 76 |

Then the **union** of the 54 **RAKE** and 76 **SentiWordNet** keyphrases were taken to obtain **108 final candidate keyphrases**.

**Chapter 5**

**Clustering**

**Supervised Learning**: In Supervised Learning, the computer is taught by example. It learns from past data and applies the learning to present data to predict future events. In this case, both input and desired output data provide help to the prediction of future events.

**Unsupervised Learning**: In Unsupervised learning, machine is trained to use data that is neither classified nor labelled. It means machine is trained only on the inputs, without their labels. The model classifies the input data into classes that have similar features.

**Reinforcement Learning:**  In **reinforcement learning**, the machine tries to take the best possible action in a given situation to maximize the total profit. Basically, the machine learns by getting feedback on its past outcomes.

In case of Unsupervised Learning, one needs to use the concept of clustering the group the similar data into one cluster and dissimilar data into different clusters.

**Clustering:** It is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data. It groups data instances that are like (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.

In clustering, we don’t have a target to predict. We try to club similar observations and form different groups.

Clustering of unlabelled data can be performed with the module **sklearn.cluster**

The algorithms implemented in this module can take different kinds of matrix as input. All the methods accept standard data matrices of shape (n\_samples, n\_features). These can be obtained from the classes in the sklearn.feature\_extraction module.

Each clustering algorithm comes in two variants: a class, that implements the fit method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the labels\_ attribute.

**Types of Clustering: - ­­**

Clustering algorithm can widely be classified into two classes:

1. **Flat clustering**: In case of flat clustering, one has to specify to the machine how many categories the data should be clustered into.

2. **Hierarchical Clustering**: In case of hierarchical clustering, the machine is allowed to decide how many clusters to create based on its own algorithms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Name** | **Parameters** | **Scalability** | **Use Case** | **Geometry**  **(metric)** |
| **K - Means** | number of clusters | Very large n\_samples, medium n\_clusters with MiniBatch code | General-purpose, even cluster size, flat geometry,  not too many clusters, inductive | Distances between points |
| **Affinity Propagation** | damping, sample preference | Not scalable with n\_samples | Many clusters, uneven cluster size, non-flat geometry, inductive | Graph distance (e.g. nearest-neighbor graph) |
| **Agglomerative Clustering** | number of clusters or distance threshold, linkage type, distance | Large n\_samples and n\_clusters | Many clusters, possibly connectivity constraints, non-Euclidean  distances, transductive | Any pairwise distance |
| **DBSCAN** | neighborhood size | Very large n\_samples, medium n\_clusters | Non-flat geometry, uneven cluster sizes, transductive | Distances between nearest points |

A brief comparison of the popular Clustering Methods available in sklearn:

**Fig. 5.1 Clustering Method**

The most popular and widely used methods out of these include K-Means and Hierarchical Clustering, particularly Agglomerative Clustering.

**K-Means Clustering: -**

K means algorithm falls under the category of flat clustering. It is a centroid-based algorithm, where we calculate the distances to assign a data-point to nearest cluster.

Here, our motive is to minimize the intracluster distance and maximize the intercluster distance.

In short, for a successful grouping, we need to attain two major goals:

1. First, a similarity between data points in same cluster.
2. Second, a distinction of those similar data points from other data points in another cluster.

**Algorithm:**

1. This algorithm starts with picking random ‘k’ cluster centers.
2. Step 2 is to assign every item to its nearest cluster center.
3. Step 3 is to calculate the mean of the formed clusters.

Repeat Step 2 and Step 3 repeatedly until either our goal is achieved, or we met the stopping criterion.

**Stopping Criteria:**

Main stopping criteria that can be adopted to stop k-means algorithm are:

1. Centroid of newly formed clusters in Step 3 is same as the centroid of old cluster.
2. Re-assigning data points according to mean of newly formed clusters don’t change the clusters, i.e. points remain in the same cluster.
3. Reached to maximum limit of iterations, by default, it is 300 in sklearn.

First and second point basically shows that tolerance value is achieved, by default, it is 1e-4 in sklearn.

The K-Means Algorithm is simple and easy to implement and is efficient with linear time complexity. However, it is applicable only when mean or some centroid finding criteria is explicitly defined and it is also sensitive to outliers, which might be some of its disadvantages.

**Hierarchical Clustering: -**

Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample.

The two major approaches of Hierarchical clustering are:

* Agglomerative Clustering (Bottom Up)
* Divisive Clustering (Top Down)

Diagram

Description automatically generated

**Fig. 5.2 Hierarchical Clustering Flowchart**

**Agglomerative Clustering: -**

The Agglomerative Clustering object in sklearn performs a hierarchical clustering using a bottom-up approach. Each observation starts in its own cluster, and clusters are successively merged. The linkage criteria determine the metric used for the merge strategy:

* **Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.
* **Maximum** or **complete linkage** minimizes the maximum distance between observations of pairs of clusters.
* **Average linkage** minimizes the average of the distances between all observations of pairs of clusters.
* **Single linkage** minimizes the distance between the closest observations of pairs of clusters.

**Diagram

Description automatically generated**

**Fig. 5.3 Linkages of Agglomerative Clustering**

**Algorithm:**

* Each data point is assigned as a single cluster.
* Determine the distance measurement and calculate the distance matrix.
* Determine the linkage criteria to merge the clusters.
* Update the distance matrix.
* Repeat the process until every data point become one cluster.

No information about the number of clusters is required beforehand. Hence, it is easy to implement and gives best results in some cases.

However, it does have some disadvantages. It has large storage requirements, and they can be computationally intensive. This is especially true for big data. Also, merging can’t be reversed, which can create a problem if you have noisy, high-dimensional data. Outliers can cause less-than-optimal merging, that is, drawback is that groups with close pairs can merge sooner than is optimal, even if those groups have overall dissimilarity.

**Divisive Clustering: -**

Divisive Clustering can be done by initially grouping all the observations into one cluster, and then successively splitting these clusters.

Divisive is done in opposite way of agglomerative but it becomes cumbersome. One way is by k-means, so at the very first level it is easy but after that which way it must be split depends upon conditions which makes it computationally expensive.

**DBSCAN: -**

DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise. It finds *core samples* of high density and expands clusters from them. It is good for data which contains clusters of similar density. The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. Therefore, clustering by DBSCAN can be of any shape as opposed to K-Means which assumes clusters are of convex-shape.

Some important terms in context of DBSCAN include:

* Eps = specified radius
* Density = number of points within an Eps (specified radius)
* Core point = A point is a core point if it has more than a specified number of points within Eps
* Border point = A point is a border point if it has less than a specified number of points but is in the neighborhood of core points.
* Noise point = A point that is not a core point or border point is a noise point

**Algorithm:**

* Label all points as core, border, or noise points.
* Eliminate noise points.
* Put an edge between all core points that are within specified radius of each other.
* Make each group of connected core points into a separate cluster.
* Assign each border point to one of the clusters of its associated core points.

DBSCAN can easily handle noise or outliers and it can be used for clusters of any shape. However, it fails when data has clusters of varying densities. It is also difficult to handle large datasets with DBSCAN.

**Clustering in the problem statement:**

After extracting the list of candidate keyphrases from the given emails which has negative sentiments attached to it, the next task is to come up with certain complaint categories which can serve as labels to the customer emails. For this, we must identify these few labels out of the extracted keyphrases list. The keyphrases having close or similar implications can be grouped into one cluster and that cluster can serve as a label later.

So, we can resort to clustering at this stage.

**Targeted Approach:**

1. Vectorize the final list of keyphrases to create the sparse matrix, and hence the similarity matrix, say B using it.
2. Fitting matrix B using some clustering algorithm and finding the centroid of each cluster.
3. Fitting emails to the classes and predicted the result of test data.

Explanation of each step:

**Creating Similarity Matrix:**

A cosine similarity matrix was created out of the extracted list of keyphrases to be fed into sklearn clustering algorithm.

**Clustering Model:**

The Cosine Similarity matrix was fed into the Clustering model using sklearn.cluster module to obtain6, 9, 12 and 15 classes. The classes were predicted using K-Means Clustering (Flat clustering by calculating mean of n clusters) and Agglomerative Clustering (Hierarchical clustering starting from n clusters).

Out of the two algorithms, Agglomerative produced better clusters, further experiments were carried out using the results obtained from Agglomerative Clustering.

Here is how clustering performed using Agglomerative:

cluster = AgglomerativeClustering(n\_clusters = n, linkage='complete')

labels\_ward = cluster.fit(B).labels\_

val = labels\_ward

d = {}

  for i in range(len(final\_keyphrases)):

    if d.get(val[i],False)==False:

      d[val[i]] = [[final\_keyphrases[i]]]

    else:

      d[val[i]] = d[val[i]] + [[final\_keyphrases[i]]]

  Y = [0]\*len(d)

  for i in sorted(d.keys()):

    d[i] = [item for sublist in d[i] for item in sublist]

    print(i, d[i])

    Y[i] = d[i][0]

  return d

**Finding Centroid of Each Class:**

For finding the centroid, start by vectorizing each of the class phrases and calculating their mutual TFIDF scores. Next step is to find the top feature of the class which has the median TFIDF score. This serves as the centroid of that class. These centroid of each class in the array will serve as Class Representatives.

# Function for finding Centroid of each cluster

from sklearn.feature\_extraction.text import TfidfVectorizer

def get\_tfidf\_top\_features(i = [], n\_top = 1):

  vect = TfidfVectorizer(ngram\_range=(2,3))

  tfidf = vect.fit\_transform(i)

  importance = np.argsort(np.asarray(tfidf.sum(axis=0)).ravel())[::-1]

  feature = np.array(vect.get\_feature\_names())

return feature[importance[:n\_top]]

**Fitting each Email to a Class:**

For fitting the emails into classes, the cosine similarity of each email with respect to all the Class Representatives would be calculated. The class having maximum similarity with the email is to be chosen. That email would be fit to that class.

#Function for cosine similarity

def co\_sim(X , Y):

  s = []

  for i in Y:

    l1 =[];l2 =[]

    X\_set = {w for w in word\_tokenize(X)}

    Y\_set = {w for w in word\_tokenize(i) }

    rvector = X\_set.union(Y\_set)

for w in rvector:

       if w in X\_set: l1.append(1) # create a vector

       else: l1.append(0)

       if w in Y\_set: l2.append(1)

       else: l2.append(0)

     c = 0

    for i in range(len(rvector)):

      c+= l1[i]\*l2[i]

    cosine = c / float((sum(l1)\*sum(l2))\*\*0.5)

    s.append(cosine)

  return s

def max\_co\_sim(X, Y):

  max\_index = np.argmax(co\_sim(X,Y))

  return(Y[max\_index])

Lastly, the count of the number of emails in each class would be calculated.

**Results**

A detailed representation of the clusters is shown with the Class names given manually to each cluster, which can serve as labels for the complaint categories.

**Train\_data Analysis: (800 emails)**

The train data consisted of 800 emails chosen randomly out of the complete dataset of 1000 emails

If distributed to 15 classes: -

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Keyphrases** | **Centroid** | **Count** |
| Pathetic Service | ['service bad', 'pathetic service', 'disappointed service', 'bad service', 'pathetic service XYZ', 'bad customer service'] | 'pathetic service' | 309 |
| Worst Courier | ['courier service worst', 'courier worst', 'worst courier service', 'XYZ courier worst', 'XYZ worst', 'XYZ worst courier', 'worst courier'] | 'worst courier' | 254 |
| Not Willing To Cooperate | ['not good', 'not sure', 'not available', 'not aware', 'not able', 'not ready', 'not satisfied', 'not willing', 'not interested'] | 'not willing' | 73 |
| Missing Packet | ['missing consignment', 'missing packet', 'missing items', 'consignment missing', 'courier missing', 'missing item', 'items missing', 'box missing'] | 'missing packet' | 31 |
| Pathetic Courier Service | ['pathetic customer', 'XYZ failed', 'XYZ pathetic', 'kind pathetic', 'courier pathetic'] | 'pathetic customer' | 23 |
| Lodging and Investigate Complaint | ['file complaint', 'like lodge complaint', 'consider complaint', 'customer complaint', 'consumer complaint', 'company complaint', 'registered complaint', 'complaint consumer', 'investigate complaint', 'lodged complaint', 'complaint redressal mechanism', 'complaint told', 'raise complaint', 'subject complaint', 'register complaint', 'written complaint', 'lodge complaint', 'raised complaint', 'request investigate complaint', 'complaint request', 'investigate complaint request'] | 'investigate complaint' | 22 |
| Complaint XYZ | ['complaint XYZ courier', 'complaint courier', 'complaint XYZ'] | 'complaint XYZ' | 22 |
| Wrong Address / Information | ['wrong address', 'giving wrong', 'wrong hands', 'wrong information', 'showing wrong'] | 'wrong information' | 15 |
| Bad Experience | ['horrible experience', 'bad experience', 'bad experiences', 'bad customer', 'really bad', 'bad experience XYZ'] | 'bad experience' | 15 |
| Worst Experience/ Website | ['really worst', 'worst part', 'worst customer', 'worst website', 'worst thing', 'worst experience'] | 'worst website' | 11 |
| Unprofessional Services | ['worst service', 'XYZ worst service', 'worst service provider', 'worst customer service', 'service worst'] | 'worst service' | 7 |
| Customer Agony | ['unnecessary trouble', 'never able', 'misplaced transit', 'unable trace', 'humble request', 'no action', 'call center number', 'mental agony', 'mental torture', 'hard earned money', 'hard copy', 'stands misplaced', 'blame game', 'much disappointed', 'false promises', 'careless handling', 'going worse day', 'blame game continued', 'unable track'] | 'blame game' | 7 |
| Charged Extra Amount | ['said sorry', 'sorry say', 'sorry inform till', 'charge sorry inform', 'sorry inform', 'speed charge sorry', 'charge sorry'] | 'sorry inform' | 5 |
| Complained About Company | ['response complained', 'even complained', 'complained company', 'also complained'] | 'response complained' | 4 |
| Immediate Action | ['immediately destroy copies', 'immediately destroy', 'sender immediately destroy'] | 'immediately destroy' | 2 |

If distributed to 6 classes:-

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Keyphrases** | **Centroid** | **Count** |
| Pathetic Service | ['horrible experience', 'service bad', 'pathetic service', 'bad experience', 'disappointed service', 'bad experiences', 'bad service', 'pathetic service XYZ', 'bad customer', 'bad customer service', 'really bad', 'bad experience XYZ'] | 'pathetic service' | 305 |
| Not Willing To Cooperate | ['not good', 'not sure', 'not available', 'not aware', 'not able', 'not ready', 'not satisfied', 'not willing', 'not interested'] | 'not willing' | 236 |
| Customer Agony / Wrong Address / Tracking lost | ['said sorry', 'sorry say', 'pathetic customer', 'XYZ failed', 'unnecessary trouble', 'never able', 'response complained', 'XYZ pathetic', 'sorry inform till', 'charge sorry inform', 'wrong address', 'giving wrong', 'misplaced transit', 'unable trace', 'humble request', 'no action', 'call center number', 'immediately destroy copies', 'mental agony', 'mental torture', 'even complained', 'hard earned money', 'complained company', 'hard copy', 'stands misplaced', 'wrong hands', 'blame game', 'immediately destroy', 'much disappointed', 'wrong information', 'false promises', 'careless handling', 'going worse day', 'sorry inform', 'sender immediately destroy', 'kind pathetic', 'speed charge sorry', 'courier pathetic', 'showing wrong', 'also complained', 'charge sorry', 'blame game continued', 'unable track'] | 'sorry inform' | 133 |
| Worst Courier Service | ['really worst', 'courier service worst', 'worst part', 'worst service', 'courier worst', 'XYZ worst service', 'worst service provider', 'worst courier service', 'worst customer service', 'service worst', 'XYZ courier worst', 'worst customer', 'worst website', 'XYZ worst', 'worst thing', 'XYZ worst courier', 'worst experience', 'worst courier'] | 'XYZ worst' | 62 |
| Lodging Complaint | ['file complaint', 'like lodge complaint', 'consider complaint', 'customer complaint', 'consumer complaint', 'company complaint', 'registered complaint', 'complaint XYZ courier', 'complaint consumer', 'investigate complaint', 'lodged complaint', 'complaint redressal mechanism', 'complaint told', 'raise complaint', 'subject complaint', 'register complaint', 'written complaint', 'complaint courier', 'lodge complaint', 'raised complaint', 'request investigate complaint', 'complaint request', 'investigate complaint request', 'complaint XYZ']  4 ['not good', 'not sure', 'not available', 'not aware', 'not able', 'not ready', 'not satisfied', 'not willing', 'not interested'] | 'investigate complaint' | 42 |
| Missing Packet | ['missing consignment', 'missing packet', 'missing items', 'consignment missing', 'courier missing', 'missing item', 'items missing', 'box missing'] | 'missing packet' | 22 |

A comparison of the highest and lowest distributions i.e. of 15 and 6 clusters is shown. However, the experiment was performed with four clusters of sizes 6, 9, 12, and 15 clusters to find out which cluster produced the optimum labels.

**Cluster Analysis**

Supervised algorithms have lots of metrics to check their goodness of fit like accuracy, r-square value, sensitivity, specificity etc. However, measuring the goodness of unsupervised algorithms in the form of Clustering is a complex task. To aid this process, various Cluster analysis indexes come into play.

Terms to be noted:

* Intercluster distance is the distance between two objects belonging to two different clusters.
* Intracluster distance is the distance between two objects belonging to same cluster.

**Silhouette Index**

When dealing with higher dimensions, the silhouette score is quite useful to validate the working of clustering algorithm as we can’t use any type of visualization to validate clustering when dimensions are greater than 3.

* Silhouette Coefficient or Silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.
* 1: Means clusters are well apart from each other and clearly distinguished.
* 0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.
* -1: Means clusters are assigned in the wrong way.

Silhouette Score = (b-a)/max(a,b)

where, a= average intra-cluster distance i.e. the average distance between each point within a cluster.

b= average inter-cluster distance i.e. the average distance between all clusters.

The Silhouette validation technique calculates the silhouette index for each sample, average silhouette index for each cluster and overall average silhouette index for a dataset. Using the approach each cluster could be represented by Silhouette index, which is based on the comparison of its tightness and separation*.*

**Calinski-Harabasz Score**

The CH Index (also known as **Variance ratio criterion**) is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

Here cohesion (intra-cluster distance) is estimated based on the distances from the data points in a cluster to its cluster centroid and separation (inter-cluster distance) is based on the distance of the cluster centroids from the global centroid.

CH index has a form of (a . Separation) / (b . Cohesion), where a and b are weights. The score is unbounded, so higher the score the better.

The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

**5-Fold Cross Validation**

A five fold cross validation was performed on the entire dataset of 1000 emails to identify the optimum set of test emails, of size 0.2 i.e , 200 out of 1000 emails. The fold with the best cluster distribution was looked into.

**Analysis of this cluster for test data :-**

For choosing the right cluster, Calinski Harabasz Score, Silhouette Score, and Scatter plot are kept as the measures, which are performed experimentally like this.

print("Calinski Harabasz score: ",metrics.calinski\_harabasz\_score(B,labels\_ward)

print("Silhouette Score: ",metrics.silhouette\_score(B, labels\_ward))

val = labels\_ward

X = np.arange(len(val)) + 1

plt.scatter(val , X)

plt.xlabel("Clusters")

plt.ylabel("Extracted Features")

plt.show()

**Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated**

**Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated**

**Choosing the right Cluster:**

**1.Scatter plot** - A plot with the Extracted Features on the y-axis and Classes on the x-axis was formed for each Cluster. The clusters were checked for even distribution of features.

*Cluster 6 was discarded as it was showing the most skewed scattering.*

*So,* ***Clusters 9, 12 and 15*** *were chosen over Cluster 6.*

**2.Calinski Harabasz Score** – The higher the CH Score better is the cluster. *Here, the cluster with the least score was discarded. So,* ***Cluster 12 and 15*** *were chosen over cluster 9.*

**3.Silhouette Score** - The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean inter-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters.

*Here, the cluster with a score closer to 1 was retained. So,* ***Cluster 12*** *was chosen over cluster 9.*

The detailed representation of this distribution over 12 clusters for the set of chosen 200 emails is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Keyphrases** | **Representative** | **Count** |
| Not Good Response | ['not sure', 'not able', 'not available', 'not ready', 'not aware', 'not good', 'not sure whether', 'not interested', 'not satisfied', 'not willing'] | 'not good' | 34 |
| Missing Consignment | ['missing items', 'parcel missing', 'courier missing', 'missing consignment', 'missing item', 'items missing', 'missing packet', 'consignment missing'] | 'missing consignment' | 31 |
| Bad Courier Service | ['bad service', 'horrible service', 'service bad', 'disappointed service', 'bad customer service'] | 'bad service' | 25 |
| Bad Customer Experience | ['horrible experience', 'bad experience', 'bad experience XYZ', 'really bad', 'bad customer', 'bad experiences'] | 'bad experiences' | 24 |
| Disappointed at Wrong/ Late Delivery | ['mental harassment', 'stands misplaced', 'much disappointed', 'response complained', 'lame excuses', 'also complained', 'mental agony', 'mental torture', 'unable trace', 'failed deliver', 'false promises', 'humble request', 'unable track', 'call center number', 'immediately destroy copies', 'hard earned money', 'going worse day', 'rude way', 'complained company', 'hard copy', 'careless handling', 'sender immediately destroy', 'immediately destroy', 'no action'] | 'response complained' | 23 |
| XYZ Worst Service Provider | ['XYZ worst service', 'worst service provider', 'worst service', 'service worst service', 'worst customer service', 'service worst'] | 'XYZ worst service' | 23 |
| Complaint Against Company | ['investigate complaint', 'complaint told', 'lodge complaint', 'company complaint', 'complaint XYZ', 'written complaint', 'consumer complaint', 'reading complaint', 'request investigate complaint', 'complaint courier', 'file complaint', 'complaint number', 'registered complaint', 'like lodge complaint', 'raised complaint', 'customer complaint', 'subject complaint', 'register complaint', 'investigate complaint request', 'complaint XYZ courier', 'complaint request', 'lodged complaint', 'consider complaint', 'file police complaint', 'complaint consumer'] | 'lodged complaint' | 10 |
| XYZ Pathetic | ['XYZ pathetic', 'XYZ failed', 'misplaced XYZ'] | 'XYZ pathetic' | 10 |
| Worst Experience/Service | ['XYZ courier worst', 'worst part', 'courier worst', 'worst experience', 'worst customer care', 'XYZ worst', 'worst customer', 'worst thing', 'worst courier', 'really worst', 'worst courier service', 'XYZ worst courier'] | 'really worst' | 10 |
| Pathetic Service | ['courier pathetic', 'really pathetic', 'pathetic service XYZ', 'pathetic customer', 'kind pathetic', 'pathetic service'] | 'kind pathetic' | 6 |
| Charged Extra Amount | ['sorry inconvenience', 'charge sorry inform', 'sorry inform till', 'speed charge sorry', 'sorry inform', 'said sorry', 'charge sorry'] | 'charge sorry inform' | 3 |
| Wrong Hands | ['delivered wrong address', 'wrong address', 'delivered wrong', 'thing wrong', 'wrong place', 'wrong hands', 'giving wrong', 'wrong information', 'wrong attitude'] | 'wrong hands' | 1 |

**Fitting the emails into classes:**

The entire model developed was tested on the test set of 200 emails to see how the emails were categorized into classes. Each email was assigned a particular complaint category a seen above.

**Accuracy:**

The final accuracy was calculated by the number of emails correctly labelled over total test size. The machine given labels were compared against manual labels given in two runs and the number of emails correctly labelled were checked.

In our case, 168 emails out of 200 were correctly labelled which resulted in test accuracy of 84%.

**Multi-Label Clustering:**

Modern Classification problems are so complex that they often require prediction of multiple labels simultaneously associated with a single instance. This is called *Multi-Label Classification*. It originated from the investigation of text categorization problems, wherein one instance might belong to more than one classes, at the same time.

A similar sounding but completely different concept to this is *Multi-Class Classification*. The difference between Multi-Class and Multi-Label Classification is that in multi-class problems the classes are usually mutually exclusive, whereas in multi-label problems, each label represents a different class of actions or outputs though the tasks are somewhat related in a broader sense. For example, multi-class classification makes the assumption that each sample is assigned to one and only one label: a car can be of either Maruti company or under the Hyundai brand but not both at the same time. Whereas, an instance of multi-label classification can be that a text might be about any of religion, politics, finance or education at the same time or in fact none of these.

The dataset of our problem is a highly imbalanced text dataset, with raw complaints from users. So, the task we have performed till now is a form of Multi-Class Classification of the emails into one class each. However, in reality each complaint might fall under multiple complaint categories. A customer might have a problem with “bad customer service” as well as their “courier not yet delivered” at the same time. The training mechanism to predict such outputs accurately by a machine becomes cumbersome.

Thus, a lot of Multi-Label Classification techniques are available which can be implemented tactically to handle such problems: -

* **OneVsRest:** The most intuitive approach to solve a multilabel problem is by decomposing the task at hand into individual single-label problems. In the one-vs-rest strategy, the aim is to build multiple independent classifiers, and choose the class with maximum confidence for the unseen instance. Here, labels are assumed to be mutually exclusive, with the underlying correlation between them being ignored.
* **Binary Relevance:** This is the easiest to implement Multilabel Classifier, wherein an ensemble of single-label classifiers is trained to give binary output, that is whether it becomes to a particular class or not and then the union of all the classes that were predicted is taken as the multi-label output. This only works well when there is not much dependencies underlying between the labels.
* **Classifier Chains:** Both the above two methods are simple to implement, however they do not take into account the correlation between labels. This method does. A chain of binary classifiers, {} is constructed, where a classifier uses predictions from all the classifiers constructed, , j < i. The total number of classifiers required here equals the total number of classes. Thus, this method takes into account the underlying correlation between labels. However, it is a bit complex to train this model.
* **Label Powerset:** This method is different from the above method in that it considers each member of the power set of labels in the training set as a single label. However, this method has high computational complexity and when the number of classes increases the size of the power sets grow exponentially and might cause computational havoc in some cases.
* **Adapted Algorithm:** The skmultilearn.adapt module of sklearn library provides this type of multilabel approaches. Here, single-label classification algorithms are adapted to multilabel problems by simple changes in the decision function. The most common way is using the ML - KNN approach, adapted from the KNN, K-Nearest Neighbour algorithm of single label classification.

The same problem can be solved using LSTMs in deep learning. For more speed we could use decision trees and for a reasonable trade-off between speed and accuracy we could also opt for ensemble models. Other frameworks such as MEKA can be used to deal with multi-label classification problems.

**Chapter 6**

**Conclusion**

The aim of the present work is to design a classification system for classifying complaints about a service provider, courier service in this case when the complaints are typically written in simple text but often having ungrammatical English and spelling mistakes. We propose an unsupervised Machine Learning model to deal with the problem, and streamline our approach to be following Data Pre-processing, Feature Extraction and Modification and Sentiment Analysis for building the model.

We have familiarized ourselves with Natural Language Processing tools and techniques, and carried out some initial experiments towards pre-processing and keyword extraction using sci-kit based NLP software. We have worked on n-grams, TF-IDF and RAKE Models to extract keyphrases. These keyphrases were checked for negative sentiment and those qualifying were extracted to serve as candidate keyphrases. Then the target was to identify labels out of these keyphrases for classifying the emails. Since this is a form of unsupervised learning, the aim was to cluster them into several classes which might serve as labels. Clustering algorithms like K - Means Clustering and Agglomerative Clustering were used further to group the extracted candidate keyphrases into meaningful labels.

The design of the complete software achieves desired results with around 84% accuracy. However, it is not completely flawless. Moreover, the entire algorithm can be modified to improve its efficiency and performance. The software we made served till achieving Multi-Class classification. The next improvement can be to venture into Multi-Label classification. One email can be classified under one or more complaint sections. So, this would provide a better understanding of the results. The future work includes delving into different Multi-Label Techniques Classification to tackle this problem.

**References**

[1] Chatterjee, Niladri & Mohan, Shiwali. (2008). Discovering Word Senses from Text Using Random Indexing. 299-310. 10.1007/978-3-540-78135-6\_25.

[2] N. Bindal and N. Chatterjee, “A Two-Step Method for Sentiment Analysis of Tweets,” 2016 International Conference on Information Technology (ICIT), 2016, pp. 218-224, doi: 10.1109/ICIT.2016.052.

[3] shalini, A., & Jain, U. (2017). Text Classification Using Centroid Technique. International Journal of Engineering and Computer Science, 6

[4] Shafeeq, Ahamed. (2012). Dynamic Clustering of Data with Modified K-Means Algorithm. 10.13140/2.1.4972.3840.

[5] https://www.analyticsvidhya.com/blog/2020/12/understanding-text-classification-in-nlp-with-movie-review-example-example/

[6] https://www.omnisci.com/technical-glossary/feature-selection