DYNAMIC MAIL LABELLER

A project report submitted to

Rajiv Gandhi University of Knowledge Technologies

SRIKAKULAM

In partial fulfilment of the requirements for the
Award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

G SUNIL KUMAR	**S170904**
G N SAHITHI LAHARI	**S170034**
L DEEPIKA	**S170341**

3rd year B. Tech 2nd semester

Under the Esteemed Guidance of

Mr. Ch. Satish Kumar

Assistant Professor

Rajiv Gandhi University of Knowledge Technologies - Srikakulam

CERTIFICATE

This is to certify that the thesis work titled "DYNAMIC MAIL LABELLER" was successfully completed by **G SUNIL KUMAR (S170904), G NAGA SAHITHI LAHARI (S170034),** and **L DEEPIKA (S170341)**. In partial fulfilment of the requirements for the Mini Project in Computer Science and Engineering **of Rajiv Gandhi University of Knowledge Technologies** under my guidance and output of the work carried out is satisfactory.

Mr. Ch. Satish Kumar
Assistant Professor
Project Guide

Mr. K. Dileep Kumar
Assistant Professor
Project Coordinator

DECLARATION

I declare that this thesis work titled "Dynamic Mail Labeller" is carried out by me during the year 2021-22 in partial fulfilment of the requirements for the Mini Project **in Computer Science and Engineering.**

I further declare that this dissertation has not been submitted elsewhere for any Degree. The matter embodied in this dissertation report has not been submitted elsewhere for any other degree. Furthermore, the technical details furnished in various chapters of this thesis are purely relevant to the above project and there is no deviation from the theoretical point of view for design, development and implementation.

G SUNIL KUMAR **S170904**

G N SAHITHI LAHARI **S170034**

L DEEPIKA **S170341**

ACKNOWLEDGEMENT

I would like to articulate my profound gratitude and indebtedness to my project guide **Mr. Ch Satish Kumar**, **Assistant Professor** who has always been a constant motivation and guiding factor throughout the project time. It has been a great pleasure for me to get an opportunity to work under his guidance and complete the thesis work successfully.

I wish to extend my sincere thanks to Mrs. **S. Laxmi Sri**, **Head of the Computer Science and Engineering Department**, for her constant encouragement throughout the project.

I am also grateful to other members of the department without their support my work would not have been carried out so successfully.

I thank one and all who have rendered help to me directly or indirectly in the completion of my thesis work.

Project Associate

G SUNIL KUMAR **S170904**

G N SAHITHI LAHARI **S170034**

L DEEPIKA **S170341**

ABSTRACT

Dynamic Mail Labeller, a one of its kind chrome extensions which is exclusively developed for Gmail users. It dynamically assigns labels to each and every email without any human intervention. The extension follows various techniques to classify each and every mail into different categories. And a user can directly able to check the categorized mails and can redirect to the specific email. This makes the process of labelling and categorizing the mails easier and allows the user to easily identify and retrieve any peculiar mail.

Index Terms *– Mail Labelling, Mail Classification, Mail Filters, Chrome Plugins*

Table of Contents

Chapter-1

INTRODUCTION

1.1 Introduction

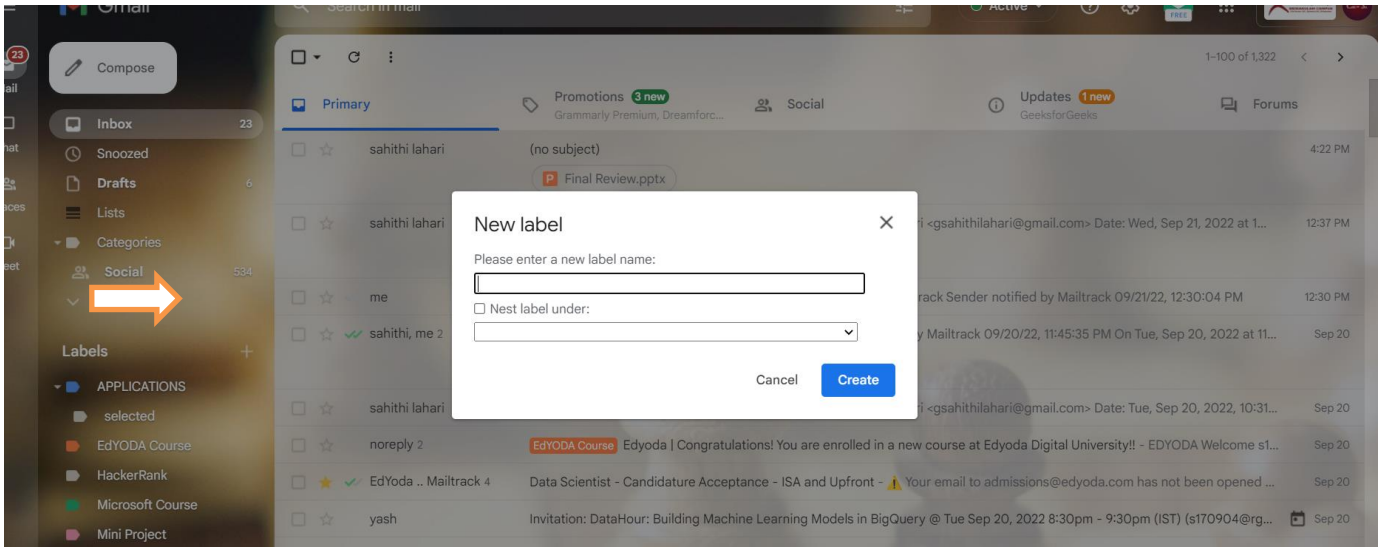
We encounter several system and application changes on a regular basis. The main reason for this is to implement new modifications that make work easier and less time and space consuming. On that topic, our project Dynamic Mail labeller is a method of labelling every mail for simpler management and identification. We are all aware that Gmail is the most often used programme for transferring communications, whether professional or personal. We receive a large number of emails in a single day. There is a potential that after a few days we may need a certain email and will need to look for it, which is a time-consuming operation.

Labelling is an existing function in Gmail that allows us to give labels to each email and add them to the appropriate category. And anytime we can classify emails automatically since we have filters that respect it. The issue here is that we must do this procedure manually for each and every message that has to be classified. As a result, we aim to tackle this by employing a mail labeller chrome plugin, which dynamically labels emails on the time they are received.

It produces filters and assigns labels dynamically. It produces filters depending on the email's TO | Sub | Content, Body, and other variables.

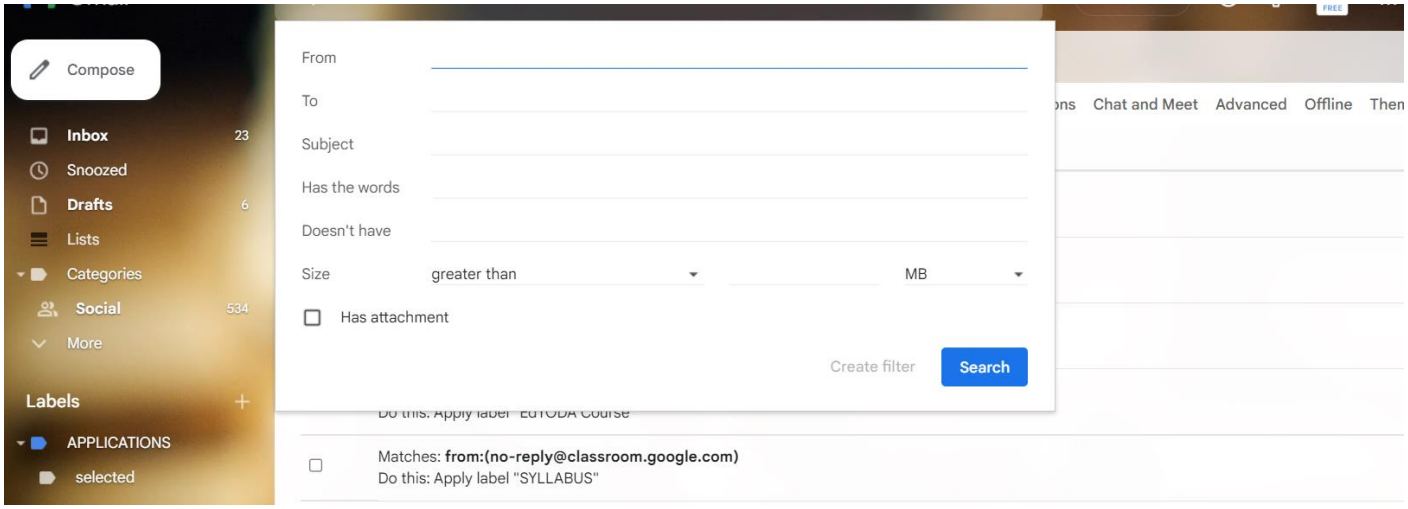
1.2 Statement of the problem

We all are aware that Gmail is the most widely used application for transferring messages, whether professional or personal. We receive a large number of emails every single day. There is a chance that after a few days we will need a specific email and will have to search for it, which is a time-consuming process.



Creation of Labels Manually

On that note, everyone uses labels to categorize their mails. The issue here is that we must perform this process manually for each and every mail that needs to be classified. Keeping the foregoing in mind, we wanted to create a chrome extension that automates labelling and has advanced ways to categorize them.



Creation of Filters

1.3 Objective

- A chrome extension that would be useful to all email users to categorize emails automatically.
- A person who is a chrome user will be able to add this to their chrome.
- A person can install this extension and run to label the emails for categorization.

1.4 Goals

- User Friendly
- Simple and Fast
- Low cost and effective
- Categorization
- Time Saving

1.5 Scope

- We are mainly focusing on every mail user to make their labels categorized.
- We are developing a chrome extension which will ease the user to customize their mails.

- It will be available in Google chrome extensions where the mail user needs to add that extension to their chrome.

1.6 Applications

What users can do with the chrome extension "Dynamic Mail Labeller?"

- **Mail labelling:**

  A person can make use of this chrome extension to dynamically label their emails and can also categorize them into various lists.

- **Mail filtering:**

  A person can also use this extension for adding filters, the extension will dynamically create filters and assign them to the emails without the need of human.

1.7 Limitations

- Only accessible by Google Chrome users.
- Only available to categorize the Gmail users.
- It will collect the user email addresses and password and should be enabled with IMAP.
- Only Categorizes the received mails.

Chapter-2

LITERATURE SURVEY

2.1 Collect Information

We have taken information from couple of Gmail users to know how they categorize/organize and label their emails to enable quick actions and we proposed this chrome extension by ourselves. To label the mails it uses the **Subject and From Address** of an e-mail, so this dynamic mail labeler will collect those From address and Subjects of every unread mail or newly received mail id's. This uses Python to read the emails and to collect the information. In Python it uses ***imap_tools*** library which is an elegant interface to communicate with an email provider using IMAP(which almost every email provider will have). First, to access the MailBox; for which it needs to get the IMAP server and login credentials (username and password). You should be able to find this in the email provider's help or settings (e.g. here's a guide for Gmail).

High level lib for work with email by IMAP:

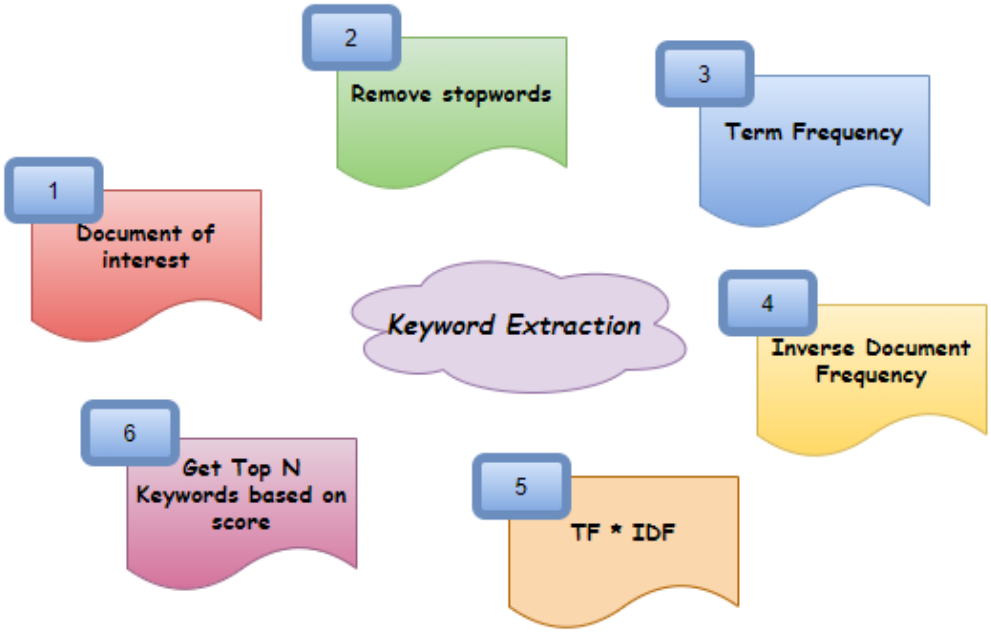
- Basic message operations: fetch, uids, numbers
- Parsed email message attributes
- Query builder for search criteria
- Actions with emails: copy, delete, flag, move, append
- Actions with folders: list, set, get, create, exists, rename, subscribe, delete, status
- IDLE commands: start, poll, stop, wait
- Exceptions on failed IMAP operations
- No external dependencies, tested

2.2 Study

NLP Concepts used:

Key Word/Phrase Extraction:

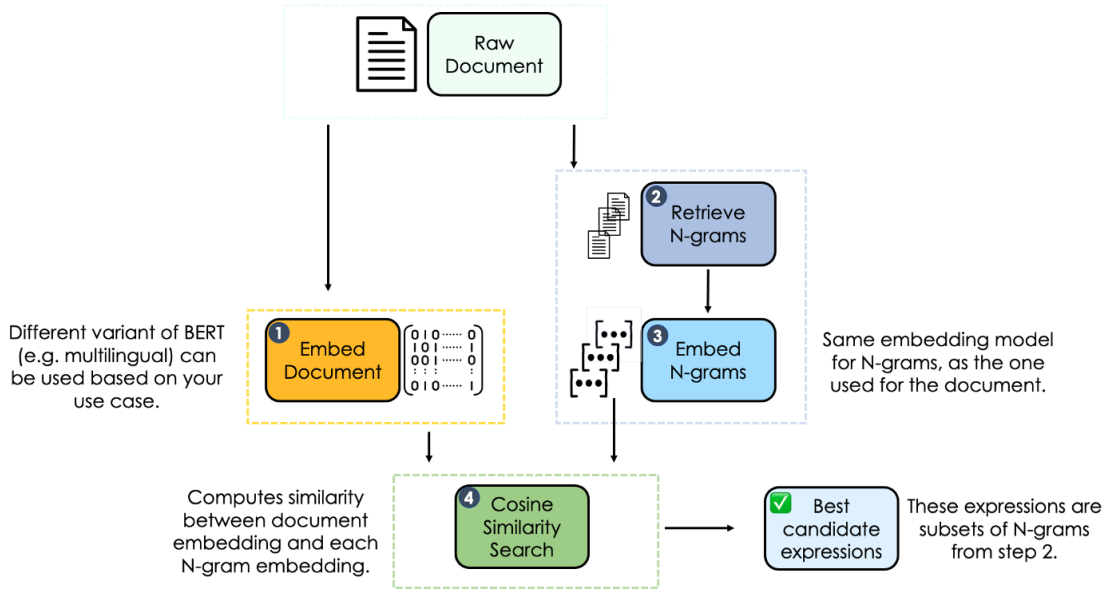
It is a text analysis technique. We can obtain important insights into the topic within a short span of time. It helps concise the text and obtain relevant keywords. It saves the time of going through the entire document. Example use-cases are finding topics of interest from a news article and identifying the problems based on customer reviews and so. One of the techniques used for Keyword Extraction is TF-IDF (Term Frequency – Inverse Document Frequency).



There are some pre-defined models or libraries in Python which will be able to extract the keywords from the given data.

KEYBERT MODEL:

Keyphrases provide a more accurate description than simple keywords and are therefore often the preferred choice. Thankfully, many open source solutions exist that allow us to automatically extract keyphrases from text. One of the recently very popular solutions is **KeyBERT**. It is an easy-to-use Python package for keyphrase extraction with BERT language models. Simply explained, KeyBERT works by first creating BERT embeddings of document texts. Afterwards, BERT keyphrase embeddings of word n-grams with predefined lengths are created. Finally, cosine similarities between document and keyphrase embeddings are calculated to extract the keyphrases that best describe the entire document.



KeyphraseVectorizers is a recently released package which can be used in addition to KeyBERT to extract enhanced keyphrases from text documents. This approach eliminates the need for user defined word n-gram ranges and extracts grammatically correct keyphrases. Furthermore, the approach can be applied to many different languages. Both open-source packages are easy to use and allow precise keyphrase extraction in just a few lines of code.

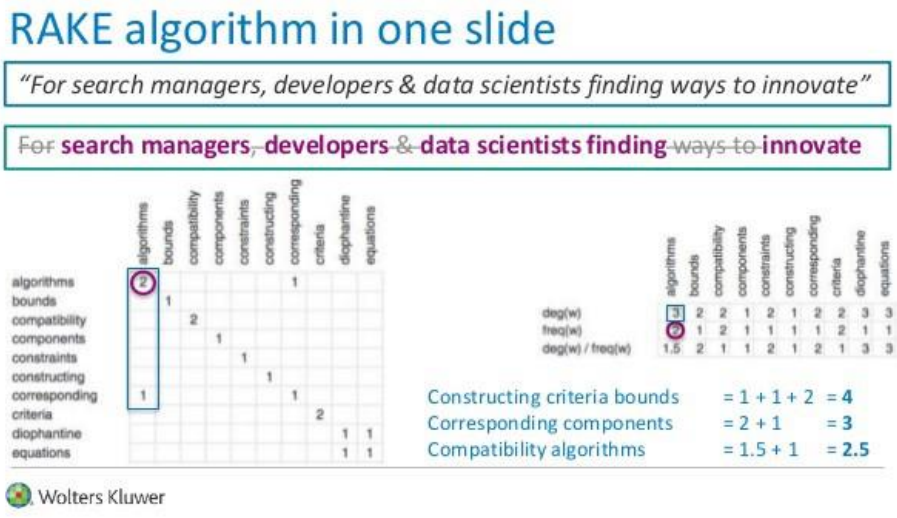
```
keywords = kw_model.extract_keywords(doc, highlight=True)
```

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a 'reasonable' way (see inductive bias).

MULTI – RAKE (RAKE – Rapid Automatic Keyword Extraction):

Rake also known as **Rapid Automatic Keyword Extraction** is a keyword extraction algorithm that is extremely efficient which operates on individual documents to enable an application to the **dynamic collection**, it can also be applied on the new domains very easily and also very effective in handling **multiple types of documents**, especially the type of text which follows **specific grammar conventions**.

Rake is based on the observations that keywords frequently contain multiple words with **standard punctuation** or **stop words** or we can say functioning words like **'and', 'of', 'the'**, etc with **minimum lexical meaning**. Stop words are typically dropped within all the **informational systems** and also not included in various text analyses as they are considered to be meaningless. Words that are considered to carry a meaning related to the text are described as the **content bearing** and are called as **content words.**



Features

- Automatic keyword extraction from text written in any language
- No need-to-know language of text beforehand
- No need to have list of stopwords
- 26 languages are currently available, for the rest - stopwords are generated from provided text
- Just configure rake, plug in text and get keywords (see implementation details)

Dynamic Mail Labeller key features:

- Mail Classification
- Mail Labeling

2.3 Benefits

- Can dynamically assign labels.
- Classifies every mail into different categories.
- Dynamically Categorization performed.

Summary

This system had built as a chrome extension with all the features to filter emails, classify and organize mails into different categorizes, and to assign a unique label to every category. For creation of the labels, it will generate labels based on the subject and From addresses of received mail and assigns labels according to it.

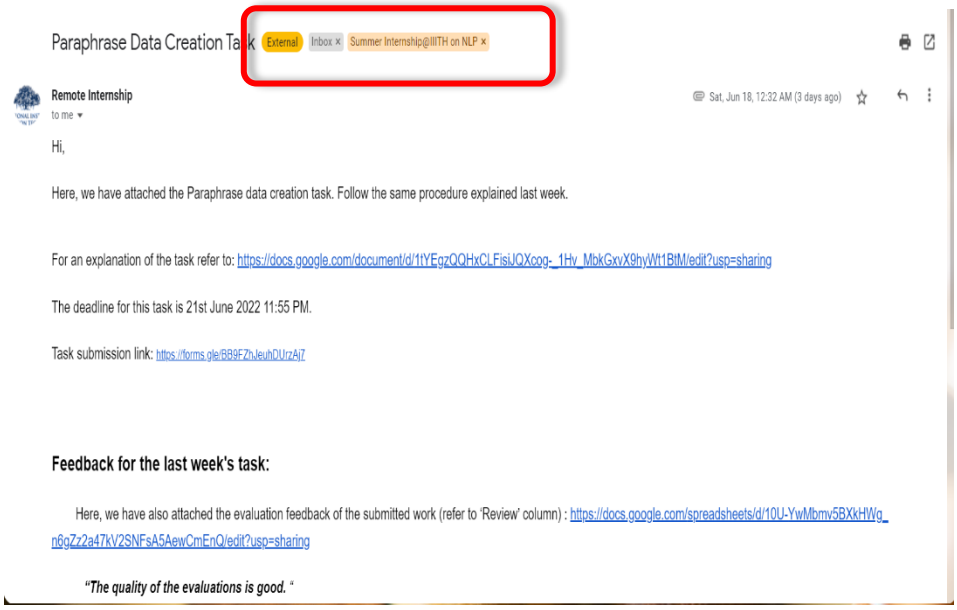
Chapter-3

ANALYSIS

3.1 Existing System

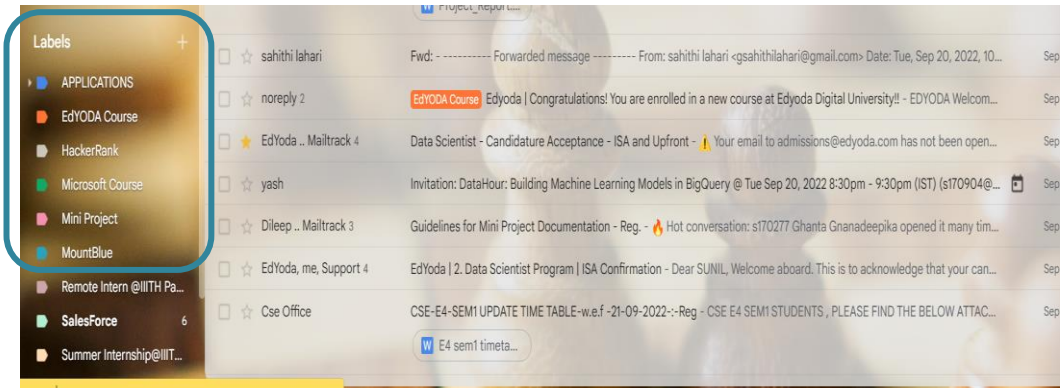
(GMAIL)

- In the existing system there is an option to label each and every mail. And after labeling we can create filters and categorize them. But in the existing system all the above process needs to be done manually.
- The following are some assigned labels, manually.



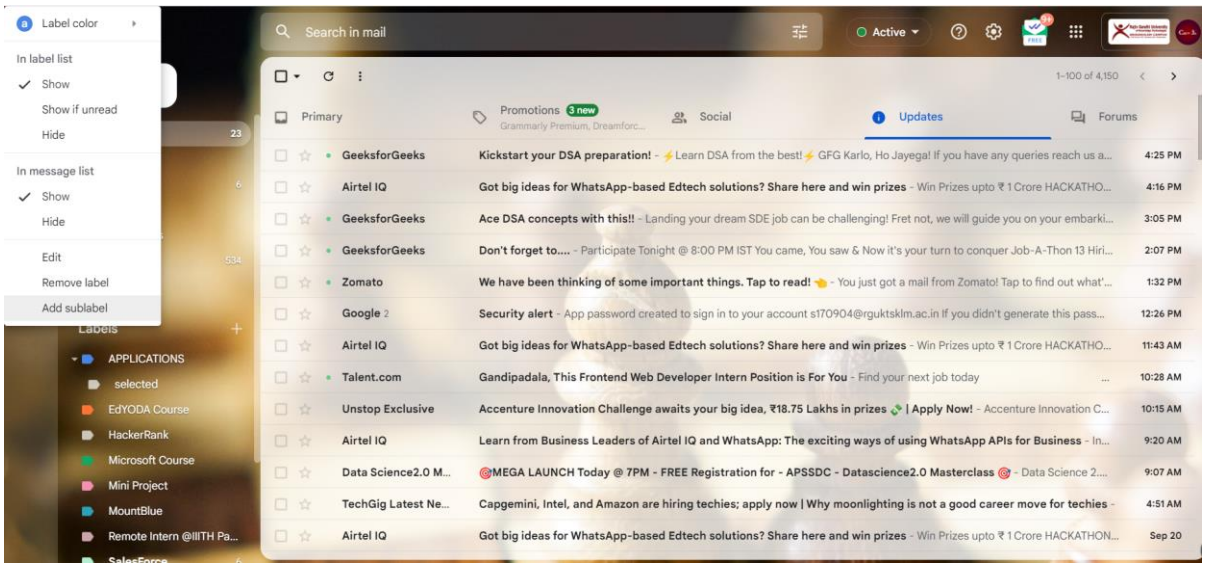
Manually assigned label

The mail existing feature:



Existing/ Manually created labels by user

We can create sublabels also manually



3.2 Disadvantages

- Time consuming
- Need human interaction every time
- Every time user needs to manually assign label and add filters for every mail.

3.3 Proposed System

- In the proposed system we can automate the process explained above and it does not require human contact always.
- The mails received will be automatically labeled through this extension.
- It dynamically assign labels to the e-mails.

3.4 Advantages

- Easy to use
- Time Saving
- No human interaction needed

3.5 System Requirements

Software Requirements:

- IMAP Tools (for EMAIL EXTRACTION)
- Python IDE (Jupyter Notebook).
- NLP (Natural Language Processing) with Python
    - Key Word Extraction using NLP
    - KEYBERT, RAKE models
- Updated Google Chrome
- JavaScript Enabled in Chrome
- Requirements needed for Extension

Chapter-4

SYSTEM DESIGN

4.1 Design of the System

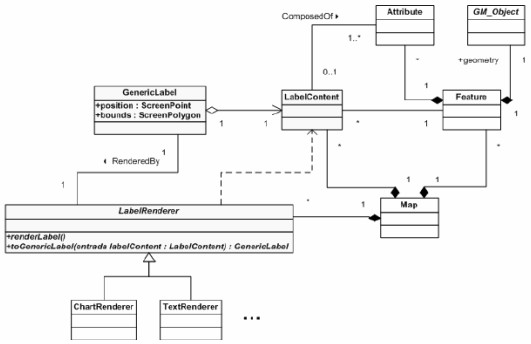
Unified Modelling Language (UML) was created in 1995 by using merging diagramming conventions used by three application development methodologies: OMT by James Rumbaugh, Objector y by Invar Jacobson and the Brooch procedure by using Grady Brooch. Previous to this time, these three amigos, together with a few dozen other practitioners had promoted competing methodologies for systematic program development, each and every with its possessed system of diagramming conventions. The methodologies adopted a sort of cookbook sort of pushing an application task via a succession of life cycle stages, culminating with delivered and documented software. One purpose of UML was once to slash the proliferation of diagramming techniques by way of standardizing on an original modelling language, as a result facilitating verbal exchange between builders. It performed that goal in 1997 when the (international) Object administration team (OMG) adopted it as a commonplace.

Some critics don't forget that UML is a bloated diagramming language written by means of a committee. That said, I do not forget it to be the nice manner to be had today for documenting object-oriented program progress. It has been and is fitting more and more utilized in industry and academia. Rational Rose is a pc Aided program Engineering (CASE) software developed by way of the rational organization underneath the course of Brooch, Jacobson and Rumbaugh to support application progress using UML. Rational Rose is always complex due to its mission of wholly supporting UML. Furthermore, Rational Rose has countless language extensions to Ada, C++, VB, Java, J2EE, and many others. Rational Rose supports ahead and reverse engineering to and from these langue ages.

However, Rational Rose does now not aid some usual design tactics as knowledge drift diagrams and CRC cards, due to the fact that these will not be a part of UML. Considering that Rational Rose has so many capabilities it is a daunting task to master it. Happily, loads can be executed making use of only a small subset of these capabilities. These notes are designed to introduce beginner builders into making productive use of the sort of subset.

4.1.1 Class Diagram

Class diagram in the Unified Modelling Language (UML), is a kind of static structure diagram hat describes the constitution of a process through showing the system's classes, their attributes, and the relationships between the classes. The motive of a class diagram is to depict the classes within a model. In object-oriented software, classes have attributes (member variables), operations (member capabilities) and relation.



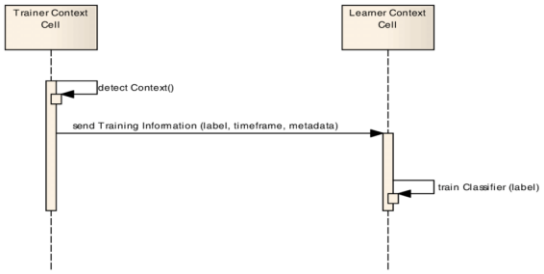
4.1.2 Use Case Diagram

It is a visually representation what happens when actor interacts with system. A use case diagram captures the functional aspects of a system.

The system is shown as a rectangle with name of the system inside ,the actor are shown as stick figures, the use case are shown as solid bordered ovals labelled with name of the use case and relationships are lines or arrows between actor and use cases. Symbols used in Use case are as follows-

4.1.3 Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) is one variety of interaction diagram that suggests how methods operate with one other and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are quite often referred to as event-hint diagrams, event situations, and timing diagrams. A sequence diagram suggests, as parallel vertical traces (lifelines), special systems or objects that are residing at the same time, and, as horizontal arrows, the messages exchanged between them, within the order the place they occur.

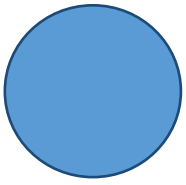


4.1.4 Data Flow Diagram

A data flow diagram or bubble chart (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

The primitive symbols used for constructing DFD's are:



A circle represents a process.

A rectangle represents external entity

A square defines a source or destination of the system data.

An arrow identifies dataflow.

Double line with one end closed indicates data store

Chapter-5

SYSTEM IMPLEMENTATION

5.1 Data Collection

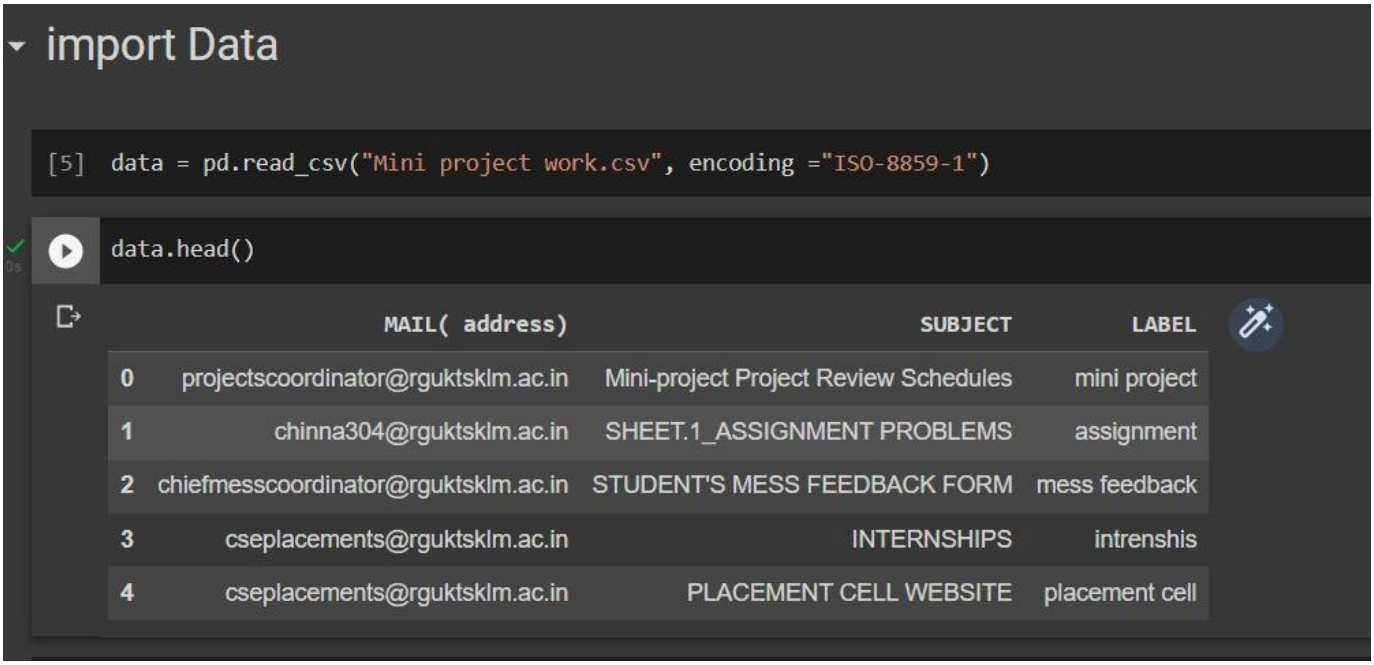
Reads the data from the mails which mails were unread till now which were received using the python Imap tools and saved them in **'Mini Project Work.csv'** and the CSV file contains mail From addresses and **Subject** of the received mail.

	A	B	C
1	mail-noreply@google.com	The best of Gmail, wherever you are	
2	mail-noreply@google.com	Tips for using your new inbox	
3	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
4	student@internshala.com	Verify Your Email	
5	noreply@updates.freelancer.com	The Pocket Guide To Building Your Brand	
6	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
7	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
8	noreply@notifications.freelancer.com	sunil kumar, these PHP, Website Design	
9	noreply@updates.freelancer.com	Get Exactly What You Need Done	
10	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
11	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
12	no-reply@accounts.google.com	Security alert	
13	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
14	noreply@updates.freelancer.com	5 Tips To Get The Most Out Of Your Content	
15	noreply@updates.freelancer.com	Tell us what you really think	
16	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
17	noreply@notifications.freelancer.com	sunil kumar, these PHP, HTML	
18	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
19	editor@internshala.com	Offer on Internshala Winter Trainings	
20	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
21	noreply@freelancer.com	Come back and start earning today!	
22	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
23	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
24	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
25	noreply@updates.freelancer.com	Corporate: Your Account That Means Business!	
26	noreply@notifications.freelancer.com	sunil kumar, these HTML, Website Design	
27	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
28	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	
29	noreply@notifications.freelancer.com	sunil kumar, these HTML, PHP	

mini_project_work

Ready Accessibility: Unavailable Count: 8057 100%

And the data imported from the csv file.



import Data

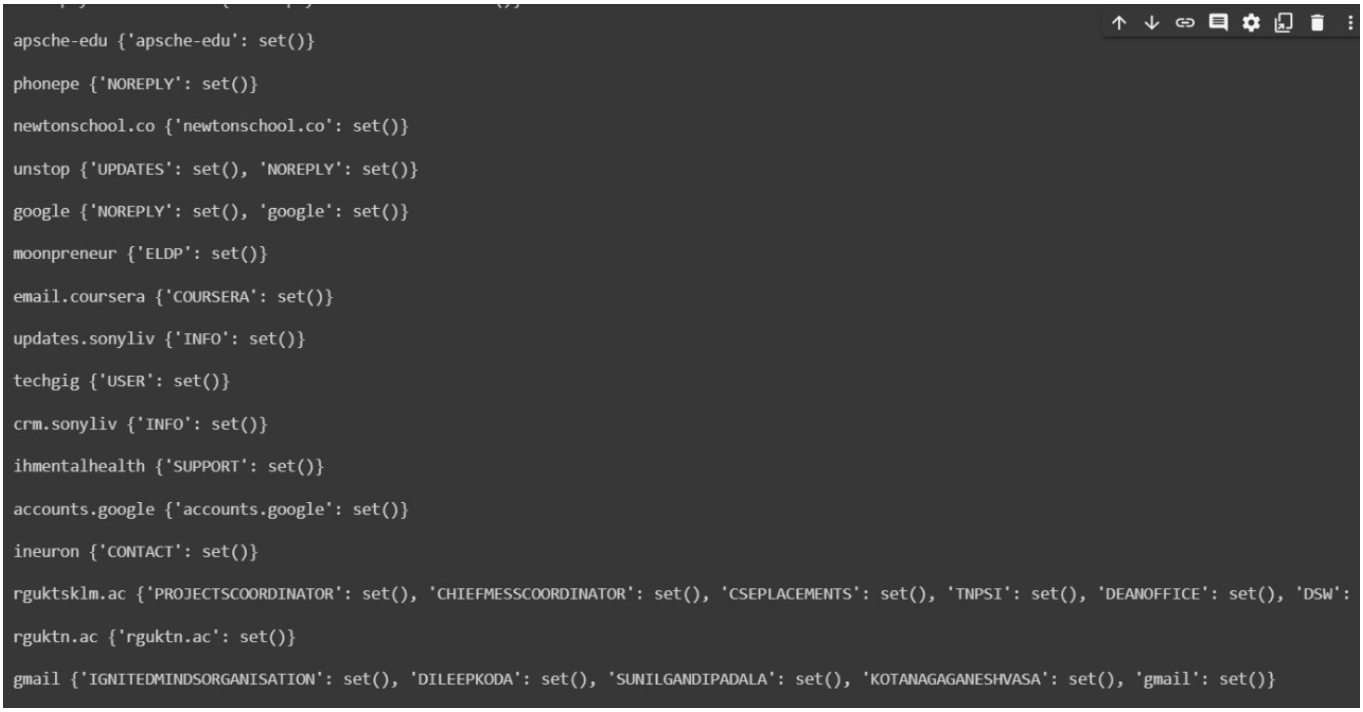
```
[5] data = pd.read_csv("Mini project work.csv", encoding ="ISO-8859-1")
```

```
data.head()
```

	MAIL(address)	SUBJECT	LABEL
0	projectscoordinator@rguktsklm.ac.in	Mini-project Project Review Schedules	mini project
1	chinna304@rguktsklm.ac.in	SHEET.1_ASSIGNMENT PROBLEMS	assignment
2	chiefmesscoordinator@rguktsklm.ac.in	STUDENT'S MESS FEEDBACK FORM	mess feedback
3	cseplacements@rguktsklm.ac.in	INTERNSHIPS	intrenshis
4	cseplacements@rguktsklm.ac.in	PLACEMENT CELL WEBSITE	placement cell

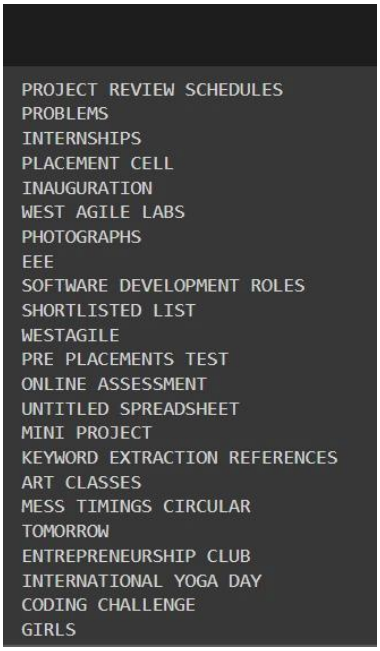
5.2 LABELS BASED ON TO ADDRESSES

These are the labels which were generated by the mail_addresses which we had imported from the collected data.



5.3 LABELS BASED ON SUBJECTS

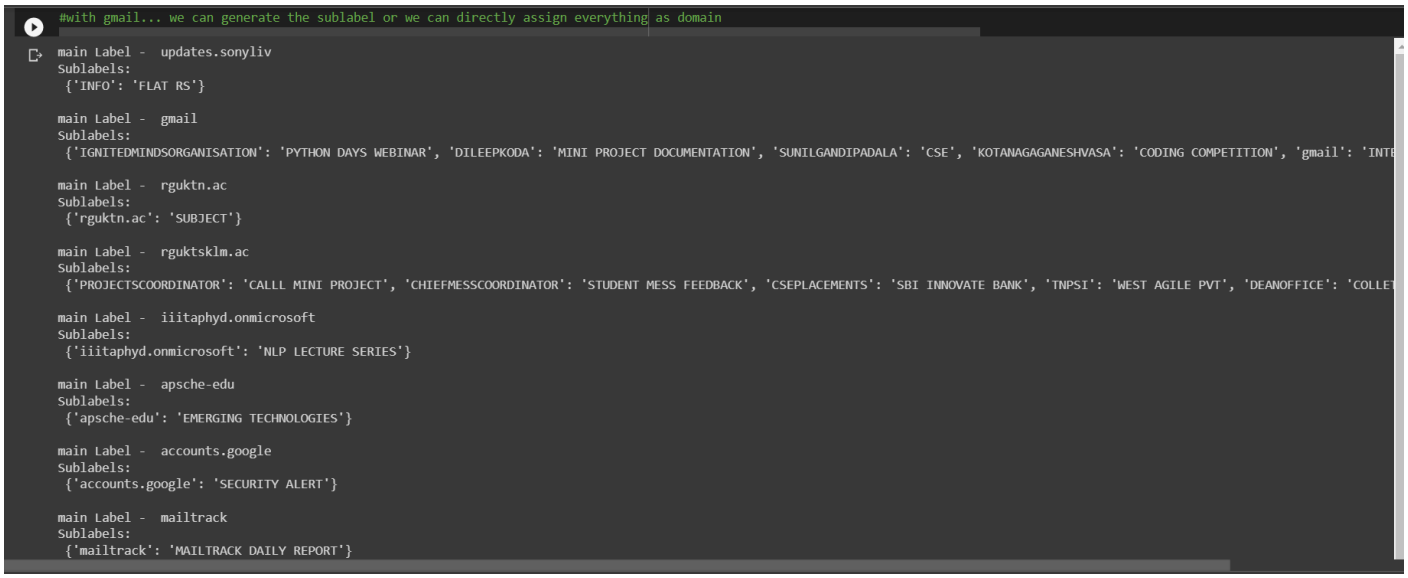
These are the labels which were generated based on the subjects of the mails which we were received.



5.4 Final Labels

These are the final labels for every mail,

Here the main label indicated **domain label** and the dictionary followed by mail id's related labels as the keys of each dictionary and the labels of subjects are the values of each dictionary those are a set.



```
#with gmail... we can generate the sublabel or we can directly assign everything as domain

main Label -  updates.sonyliv
Sublabels:
 {'INFO': 'FLAT RS'}

main Label -  gmail
Sublabels:
 {'IGNITEDMINDSORGANISATION': 'PYTHON DAYS WEBINAR', 'DILEEPKODA': 'MINI PROJECT DOCUMENTATION', 'SUNILGANDIPADALA': 'CSE', 'KOTANAGAGANESHVASA': 'CODING COMPETITION', 'gmail': 'INTE

main Label -  rguktn.ac
Sublabels:
 {'rguktn.ac': 'SUBJECT'}

main Label -  rguktsklm.ac
Sublabels:
 {'PROJECTSCOORDINATOR': 'CALLL MINI PROJECT', 'CHIEFMESSCOORDINATOR': 'STUDENT MESS FEEDBACK', 'CSEPLACEMENTS': 'SBI INNOVATE BANK', 'TNPSI': 'WEST AGILE PVT', 'DEANOFFICE': 'COLLE

main Label -  iiitaphyd.onmicrosoft
Sublabels:
 {'iiitaphyd.onmicrosoft': 'NLP LECTURE SERIES'}

main Label -  apsche-edu
Sublabels:
 {'apsche-edu': 'EMERGING TECHNOLOGIES'}

main Label -  accounts.google
Sublabels:
 {'accounts.google': 'SECURITY ALERT'}

main Label -  mailtrack
Sublabels:
 {'mailtrack': 'MAILTRACK DAILY REPORT'}
```

Final Labels

Chapter-6

SOURCE CODE

The system is implemented as 3 subparts

- Data Collection
- Model Implementation
- Chrome Extension

6.1 Data Collection

Here the emails which we received and unread will be collected and extracted the **subject and from address** and saved in a file called 'Mini project Work.csv'.

For this IMAP TOOLS (imap_tools library) used.

```
from imap_tools import MailBox, AND
import csv

# open the file in the write mode
f = open('mini_project_work.csv', 'a')

# Server is the address of the IMAP server
server = "imap.gmail.com" #there are other servers...
#yahooSmtpServer = "imap.mail.yahoo.com

user = 's170904@rguktsklm.ac.in'
password = 'jhscixihmivtuobv'
mb = MailBox(server).login(user, password)
# Fetch all unseen emails containing "xyz.com" in the from field
# Don't mark them as seen
# Set bulk=True to read them all into memory in one fetch
# (as opposed to in streaming which is slower but uses less memory)
messages = mb.fetch(criteria=AND(seen=False),
                    mark_seen=False,
                    bulk=True)
files = []
for msg in messages:
    # create the csv writer
    writer = csv.writer(f)
    mails = [msg.from_, msg.subject]
    print(mails)
    # write a row to the csv file
    writer.writerow(mails)

    # close the file
f.close()
```

6.2 Model Implementation

In this it is used to generate the dynamic mail labels from the mail address and subjects of the email.

Here the labels are generated by using Key Word/Phrase Extraction of NLP Technique by using already existed models. But the labels are generated by implementing different Models to get suitable label.

The models we used are:

- KeyBERT
- Multi_Rake
- PyTextRank

These all models were used when generating a label.

Code:

#Data Reading

```
import pandas as pd
data = pd.read_csv("Mini project work.csv", encoding ="ISO-8859-1")
mails = list(data["MAIL( address)"])
subjects=list(data["SUBJECT"])
```

#Data Pre-processing

```
import spacy
nlp = spacy.load("en_core_web_sm")
for i in range(len(subjects)):
  doc = nlp(subjects[i])
  newstring = [word.text for word in doc if word.text.isalpha()]
  newstring=" ".join(newstring)
  subjects[i]=newstring


mail =" ".join(mails)
#filtering mails
mail_ids = re.findall('([a-zA-Z0-9._-]+)@[a-zA-Z0-9._-]+\.[a-zA-Z0-9_-]+',mail)
```

#importing required libraries

```
import spacy,nltk
import re
import pytextrank
nlp = spacy.load("en_core_web_sm")
nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
#create a porterstemmer by using PorterStemmer class
ps= PorterStemmer()
```

```
from keybert import KeyBERT
# load a spaCy model, depending on language, scale, etc.
# add PyTextRank to the spaCy pipeline
nlp.add_pipe("textrank")
from multi_rake import Rake
rake = Rake()
kw_model = KeyBERT(model='all-mpnet-base-v2')
```

Generating Labels based on the **To address of the mail and subjects**

```
def subject_label(subject):
  doc = nlp(subject)
  # examine the top-ranked phrases in the document
  for phrase in doc._.phrases[:1]:
      keywords = rake.apply(phrase.text)
      keywords = [i[0].upper() for i in keywords]
      if(len(keywords)>0):
        return keywords[0]
      else:
        keywords = kw_model.extract_keywords(phrase.text,

                                        keyphrase_ngram_range=(1, 3),

                                        stop_words='english',

                                        highlight=False,

                                        top_n=10)

        keywords_list= list(dict(keywords).keys())
        return keywords_list[0].upper()

# #List out all the unique domains
import re
pattern = "@([a-zA-Z0-9._-]+)\.[a-zA-Z0-9_-]+"
mail_doamins = re.findall(pattern,mail)
labels = set(mail_doamins)
new_labels =[label.split() for label in labels]
mails_with_domains={}
for i in labels:
  mails_with_domains[i] = {mails[j][re.search('([a-zA-Z0-9._-]+)@[a-zA-Z0-9._-
]+\.[a-zA-Z0-9_-]+',mails[j]).start():re.search('@[a-zA-Z0-9._-]+\.[a-zA-Z0-9_-
]+',mails[j]).start()].upper():subject_label(subjects[j]) for j in range(len(mails)
) if i in mails[j]}

for domain in mails_with_domains.keys():
  dom=mails_with_domains[domain]
  keys = list(dom.keys())
  for id in keys:
    if id.isalpha():
```

```
      continue
    else:
      mails_with_domains[domain][domain]=mails_with_domains[domain][id]
      del mails_with_domains[domain][id]

# The Labels based on the mail domain..
  # the sublables based on the mail id's
      # The list represent the labels based on the subject
for key,value in mails_with_domains.items():
  print(key, value)
  print()
#with gmail... we can generate the sublabel or we can directly assign everything as
 domain
```

6.3 Chrome Extension

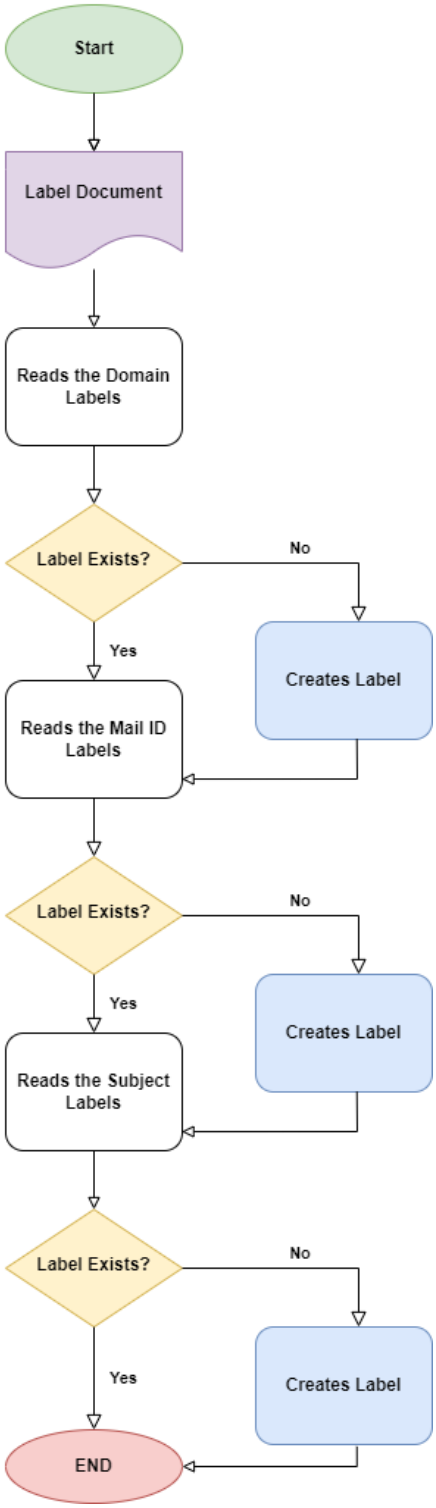
```
{
    "manifest_version":3,
    "name":"dYNAMIC mAIL lABELLER",
    "version": "1.0",
    "description": "mail labelling extension! ",
    "icon":{
        "128":"icon128.png",
        "48": "icon48.png",
        "16": "icon16.png"
    },
    "browser_action":{
        "default_icon": "icon16.png",
        "default_popup":"popup.html"
    },
    "homepage_url": "https://mail.google.com/mail/u/0/#inbox"
}
```

Basic Extension added without linking the code

This part isn't completed yet…

THE ALGORITHM WE DESIGNED FOR THIS CHROME EXTENSION TO WORK IS…

Extension Algorithm



Chapter-7

SYSTEM TESTING

7.1 INTRODUCTION

The cause of testing is to detect mistakes. Making an attempt out is the technique of looking for to realize each viable fault or weakness in a piece product. It presents a method to determine the performance of add-ons, sub-assemblies, assemblies and/or a completed product. It is the method of ex excising g program with the intent of constructing certain that the application procedure meets its necessities and client expectations and does no longer fail in an unacceptable process. There are rather plenty of forms of scan. Each experiment sort addresses a special trying out requirement.

7.2 TYPES OF TESTS

7.2.1 Unit Testing

Unit checking out involves the design of scan circumstances that validate that the internal application good judgment is functioning safely, and that program inputs produce legitimate outputs. All decision branches and interior code float must be validated. It's the checking out of character application items of the application. It is achieved after the completion of a person unit earlier than integration. It is a structural checking out, that relies on competencies of its construction and is invasive. Unit exams participate in common exams at component level and scan a distinct business approach, utility, and/or process configuration. Unit assessments are certain that every specified course of an industry method performs appropriately to the documented requisites and involves clearly outlined inputs and anticipated results.

7.2.2 Integration Testing

Integration Testing are designed to scan built-in program accessories to determine within the occasion that they evidently run as single software. Trying out is occasion driven and is more concerned with the fundamental final result of screens or fields. Integration assessments reveal that despite the fact that the accessories had been for my part pleasure, as proven through effectively unit checking out, the combo of accessories is correct and regular. Integration checking out is chiefly aimed at exposing the issues that come up from the performance of different components.

7.2.3 Functional Testing

Functional Testing checks provide systematic demonstrations that capabilities established are to be had as particular by means of the business and technical specifications, method documentation, and consumer manuals. Functional testing is working on below mentioned data:

Legitimate input: identified lessons of legitimate input ought to be accredited.

Invalid enter: recognized lessons of unacceptable effort must be rejected.

Capabilities: recognized features ought to be exercised.

Output: recognized courses of software outputs have got to be exercised.

Systems/Procedures: performance of the system here was invoked

Individual and team work of useful checks is fascinated by specifications, key capabilities, or special scan instances. Moreover, systematic insurance plan concerning establish business method flows; data fields, predefined processes, and successive strategies have to be regarded for trying out. Before useful trying out is whole, extra checks are recognized and the strong price of present checks be strong minded.

7.2.4 System Testing

Scheme difficult ensure so as to the whole included agenda process meets principles. It exams a pattern to make sure the outcome is identified and predictable. An illustration of procedure testing is the configuration-oriented approach integration scan. System testing is based on approach descriptions and flows, emphasizing pre-driven system links and integration aspects.

White Box Testing

White Box Testing is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.

Black Box Testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioural Testing.

7.3 LEVELS OF TESTING

7.3.1 Unit testing strategy

Unit checking out is most commonly performed as a part of a mixed code and unit experiment part of the software lifecycle, though it be not exceptional for coding and unit checking out to be performed as two targeted phases.

Test strategy and approach: Field testing out can be carried out manually and sensible assessments shall be written in element.

Test objectives:

- Each field must be work correctly.
- Each page must be activated through the specified link.
- Features to be tested Verify that the entries are of the correct format No duplicate entries should be allowed

7.3.2 Integration testing strategy

Software integration testing is the incremental integration checking out of two otherwise further included software gears on top of a solo stage to fabricate failure induced with the aid of interface defects. The project of the mixing scan is to check that components or program applications,

e.g., Components in a program approach or œ one step up œ software purposes at the company degree or interact without error.

Test Results:

All of the scan circumstances recounted above passed efficiently. No defects encountered.

7.3.3 Acceptance Testing Strategy

User Acceptance testing trying out is a crucial section of any mission and requires enormous participation by the tip user. It additionally ensures that the procedure meets the functional specifications.

Test Results:

The entire test cases recounted above passed effectually. No defects Encountered

Chapter-8

CONCLUSION

To sum up with, Dynamic Mail Labelling is an easy to install and very useful chrome extension that will be very beneficial to all the Gmail users. It enhances productivity and takes less time consuming for the tasks like mail filtering and labelling. It makes one's search for a specific mail easy and quick.

This extension generates labels to the subjects based on the principle of key phrase extraction technique of NLP. Which was implemented by KeyBERT model and RAKE model. The labels will be created as nested labels (Sub labels).

That is, the labels will be generated for three levels every time as domain label, which contains sublabel with the mail id(name of the mail id with out domain), and finally labels based on the subject label which will be a label nested to mail id label and which is nested nested to domain labels.

FUTURE ENHANCEMENT

With a goal to provide a seamless and quicker process for labelling emails, a convenient Chrome extension is developed using NLP, Web Scraping. To fulfil the above requirement, using Html, CSS, JS along with manifest JSON, a chrome extension was created. It is designed primarily to make the process of labelling a lot easier for all the Gmail users.

Based on the development it can further introduced into all kinds of browsers along with Google Chrome.

APPENDIX

References

https://deviloper.in/how-to-read-emails-using-python

https://pypi.org/project/imap-tools/

https://www.analyticsvidhya.com/blog/2022/03/keyword-extraction-methods-from-documents-in-nlp/

https://www.analyticsvidhya.com/blog/2022/01/four-of-the-easiest-and-most-effective-methods-of-keyword-extraction-from-a-single-text-using-python/

https://developer.chrome.com/docs/extensions/mv3/manifest/