International Journal of Computer Applications (0975 - 8887)
Volume 176 - No.41, July 2020

Event Extraction from Emails

Anupama M. Nair
Dept. of Computer Science and Engineering
Sree Chitra Thirunal College of Engineering
Trivandrum, India

Anusha Aji Justus
Dept. of Computer Science and Engineering
Sree Chitra Thirunal College of Engineering
Trivandrum, India

Arjun Ramesh
Dept. of Computer Science and Engineering
Sree Chitra Thirunal College of Engineering
Trivandrum, India

Binu Rajan M. R.
Assistant Professor
Dept. of Computer Science and Engineering
Sree Chitra Thirunal College of Engineering
Trivandrum, India

ABSTRACT

This is an era where people are too busy to check out their inbox. They swipe off notifications, not knowing they might miss something important. These could be anything like, personal messages asking whether you are free to meet up or reminders to events that you need to attend or some emails regarding interesting events that are happening around you. The need of the hour is an efficient way for keeping track of the important events from the vast number of incoming messages. This paper proposes a solution based on event extraction from emails. The scope of the project is limited to Gmail. The primary component is an android application that identifies event containing emails, extracts important details and provides an automated reminder system for the same depending on user needs. Various machine learning techniques along with natural language processing are used for the fulfillment for this project.

General Terms

Machine Learning, Natural Language Processing

Keywords

Email Classification, Event Extraction, Machine Learning, Named Entity Recognition, Natural Language Processing

1. INTRODUCTION

In this digitizing world, all are receiving numerous emails. It is a much easier and faster way to communicate than the traditional mail. Nowadays, an average person receives about 84 notifications in their phone per day. Due to busy schedules most people just swipe through them and miss important notifications. People receive a lot of text messages or emails about several events that they are interested in. They usually keep track of these events by adding them to their calender or some reminder application or any means that would notify them about the event. They try not miss out those events by any means possible but the issue in all these systems is that all these are to be done manually. In this hectic world, reading the mail manually and searching for event and then adding data to a calendar is a very time-consuming and chaotic job. Emails can be personal messages, community events or official notifications. The main domain of concern is the set of emails that arrive with an event in it. It includes emails having a date, time and venue or either of these present which indicates an event.

In this scenario where a single person receives an unmanageable number of emails and other messages, the dependence on reminder applications elevates. The core focus is on emails. This project aims at automating the process of extraction of event and adding it on to the calendar. Using this system, the manual efforts of viewing a mail, understanding the event information, and manually adding to calendar will be eliminated.

The core objectives of the proposed system include:

(1) Retrieve event information from emails in a real-time environment.
(2) Avoid wastage of time by manually opening the mail.
(3) Eliminate any possibility of human negligence.
(4) Enable customization capabilities for the information retrieved.
(5) Developing a user-friendly application.

The proposed application can be used for fetching events from any users Gmail inbox. If tweaked accordingly, it can be used for getting event details from social messaging applications as well.

2. MOTIVATION

Many of us get a large volume of email hitting the inbox. We receive emails on certain mailing lists about events that may interest us such as recruiting events, talks, and other social events and it would be convenient if the event could be added with one click instead of opening the calendar and entering the title, date, time and venue manually. Indeed, Gmail already has such a feature where it suggests a one-click add to a users calendar in certain cases, but it is observed that in many cases Gmail does not provide

such a suggestion even when the email contains event information and the user would like to add it to his/her calendar quickly.

3. RELATED WORKS

The proposed system has been implemented after extensively studying various related works in email classification and event extraction.

3.1 Email Classification

Rita McCue in [1] used SVMLib to do spam email classification. This application has four available kernels: linear, polynomial, radial basis function, and sigmoid. Support Vector Machines require parameter tuning in order to improve the accuracy of their test results. She used 5-fold cross validation for each of the four kernels. A logarithmic grid search method was performed to find the best choices for C and gamma, those that had the best average cross-validation accuracy are the ones chosen for use on the test data. The RBF kernel maps samples into the higher dimensional space in a non-linear fashion, so unlike the linear kernel, it can handle cases where the class labels and attributes are related in a non-linear manner. They obtained an accuracy of above 90% using SVM.

Shashank Senapaty in [2] tried the Naive Bayes Multinomial Event model using a specialized tokenization of the input. These include special tokens like MONTH, TIME, DATE to replace months, times and dates respectively in addition to basic tokens like HTTPADDR, EMAILADDR, and NUMBER to replace URLs, email addresses, and numbers. While this algorithm has certain shortcomings for the task at hand, such as it doesnt capture the sequence structure of the text, it still serves well as a baseline since it is an algorithm known to perform well for a text classification task.

Julie A. Black and Nisheeth Ranjan in [3], have identified three types of emails as: *Official meeting* emails are all messages that contain event information clearly delimited from all other text in the email. *Personal meeting emails* are meeting proposals in which the specifics of the meeting are presented within the body of the email itself. *Other emails* are all incoming messages that are non-event related. The raw email data is exported to a single text file where personal, official, and other emails are demarcated by XML tags. The email file is then processed by Perl and categorized into three using a similarity measure based on TF-IDF (Term Frequency-Inverse Document Frequency) and numerous domain specific heuristics with hand-tuned weights.

Ola Amayri and Nizar Bouguila in [4] have researched in developing suitable filters to separate spam emails. They used Support Vector Machines through text classification approaches. Instead of traditional kernels, various string kernels methods were advocated to implement a solution for the existing spam filtering problem. An online active framework system is proposed to cope with real time situations.

3.2 Event Extraction

Aneesh G Nath, Krishnanth V, Kevin Biju Mathew, Pranav T S and Sarath Gopi in [5] aim at automating the process of extraction of event and adding it on to the calendar. Aswar Shreyas, Gaikwad Priyanka, Merlyn Pearl and Shinde Swapnal in [6] do the same with emails. Both use NLP to classify the event details in the arriving email into subject, date, time and place. The extraction is mainly grammar based. An extension was developed in order to implement this by integrating python code. Grammar helps to construct a tree which will help in event detail extraction. The different stages of the project were tokenization, POS tagging, parser, event extraction, validation, mapping to calendar. The final results were not so accurate. In some cases it showed error while identifying time, date or location. Moreover, the title of the event could not be identified correctly in most cases.

Julie A. Black and Nisheeth Ranjan in [3] proposed an architecture for automated event extraction from incoming email messages. A successful system will classify event containing messages, do information extraction, and present this information to the user for confirmation. ANNIE (a Nearly New Information Extraction System) was used as a named entity recognizer after the preprocessing step. ANNIE helps to provide tags of people, locations, dates, parts of speech, and sentence boundaries. These tags appear in the output body of the raw email as XML tags wrapping the recognized text. ANNIE provides a tokenizer, a gazetteer, a sentence splitter, a part of speech tagger, a semantic tagger, named entity transducer, an orthographic coreferencer, and a pronominal coreferencer. In order to extract information from the categorized email messages, the RAPIER algorithm that employs a bottom up learning algorithm to learn patterns was used. It uses pairs of sample documents and filled templates to induce pattern match rules that directly extract fillers for the slots in the template. It trains on a training set where each training example is a collection of three files: the original email, the original email after passing it through a sentence splitter and part of speech tagger, a filled event template containing the date, time, location, and title of the event contained in the email. The focus was on extracting only the date, location and title of the event using a few key features. They werent able to achieve the performance necessary for deploying the information extraction system as a plugin to an email client.

Eleni Partalidou and others in [7] built a part of speech tagger that can detect the morphology of the tokens. For named entity recognition using spaCy, a model that extends the standard ENAMEX type was built. SpaCy uses sophisticated neural network based models for the implementation of Natural Language Processing components. In spaCys approach text is inserted in the model in the form of unique numerical values for every input that can represent a token of a corpus or a class of the NLP task (part of speech tag, named entity class). At the embedding stage, features such as the prefix, the suffix, the shape and the lowercase form of a word are used for the extraction of hashed values that reflect word similarities. At this stage a vocabulary with hashed values and their vectors exist in the model. For the exploitation of adjacent vectors in the state of encoding, values pass through the Convolutional Neural Network (CNN) and get merged with their context. The result of the encoding process is a matrix of vectors that represents information. Before the prediction of an ID, the matrix has to be passed through the Attention Layer of the CNN, using a query vector to summarize the input. At prediction, a Softmax function is used for the prediction of a super tag with part of speech and morphology information. Similarly, for named entities, the available class is predicted. After the training process of the model, the CNN is able to be used for NLP tasks.

Shashank Senapaty in [2] used a Maximum Entropy Markov Model (MEMM), which combines a Maximum Entropy classifier with a Viterbi decoder for event extraction. The MEMM classifier

assigns a label among TITLE, VENUE, DATE, START TIME, END TIME and OTHER to each token corresponding to the attributes of interest. For each token in the email, features are computed based on the token itself, the neighboring tokens, and the label of the previous token. The conditional probability of a class for a particular token is modeled as a linear combination of the features combined by a softmax function. Two MEMM classifiers are used, one for the subject and one for the mail body. For extracting the value of an attribute, all candidates labeled with that label are considered and one of the values is chosen based on the probabilities associated with each of them. If a candidate for an attribute is available from the subject, it is preferred over a candidate from the body. The recall for the venue was a bit low because in many cases there is not enough contextual information surrounding the venue. This problem can be fixed by increasing the training size to a more representative set or by building a corpus of locations.

Ashraf Q. Mahlawi and Sreela Sasi in [8] have discussed the process of extraction of structured data from emails. Their research is done mainly in three phases: data cleaning, data extraction and data consolidation. To improve the quality of data, first, email data is cleaned to avoid data in incorrect format. Data extraction phase comprises of data mining and NLP techniques to extract specific parts of information from the emails. Finally, data consolidation is done to combine all the various extracted data to get the structured data present in the respective email. Named Entity Recognition for event extraction has been also mentioned by Michal Laclavik, Stefan Dlugolinsky, Martin Seleng, Marcel Kvassay, Emil Gatial, Zoltan Balogh and Ladislav Hluchy in [9].

4. DATASET

4.1 Existing Dataset

Emails are quite private, so email datasets were least available. One of the few available ones was the Enron Dataset. This dataset was collected and prepared by the CALO Project. It contains mails from about 150 users, mainly from the senior management of Enron. The corpus contains a total of about 0.5 million messages. This is the most complete email corpus available. The corpus is one of the few publicly available mass collections of real emails easily available for study, because such collections are typically bound by numerous privacy and legal restrictions which render them prohibitively difficult to access.

The existing dataset was insufficient to give a higher performance owing to the following reasons:

(1) The Enron Dataset was huge, but the number of emails that matched the concerned scenario was very few.

(2) Event email classification requires emails containing events in order to set a clear demarcation from the other non- event mails.

(3) The dataset contained mainly informal conversation mails stating unclear events only.

(4) Using such a dataset would cause overfitting and misclassification.

(5) Enron dataset is useful for spam mail classification, sentiment analysis etc. and not for Event mail classification.

Thus, a new dataset was made, which was more problem specific.

4.2 Our Dataset

To train the Event Mail Classifier, a new dataset of 470 emails was prepared. It was a balanced dataset with 235 Event mails and 235 Non-Event mails. Emails were collected from Gmail users from various professions, different age groups and thus maximum variance in emails was ensured. Since this dataset was made in a problem specific manner, the accuracy and performance was increased to a good level. Emails containing most date formats, time formats and indirect events were included in dataset preparation. Then, the Event Extraction NLP model was also trained using a related Custom Tagged Entity dataset, having nearly 12000 entities. The mails are tokenized into words using the regex tokenizer. The required entities like Time, Date and Venue were tagged.

4.3 Data Preprocessing

For proper training of the classifier, the emails in dataset were preprocessed for:

(1) Removing HTML tags using BeautifulSoup4
(2) Removing space issues
(3) Removing quoted text using regular expressions
(4) Removing encoding issues that prevailed in Subject part of mails.

5. METHODOLOGY

The proposed system has the following phases: (1) Event mail classifier (2) Event extraction module (3) Android application (4) Google Calendar connectivity. The block diagram in Fig. 1

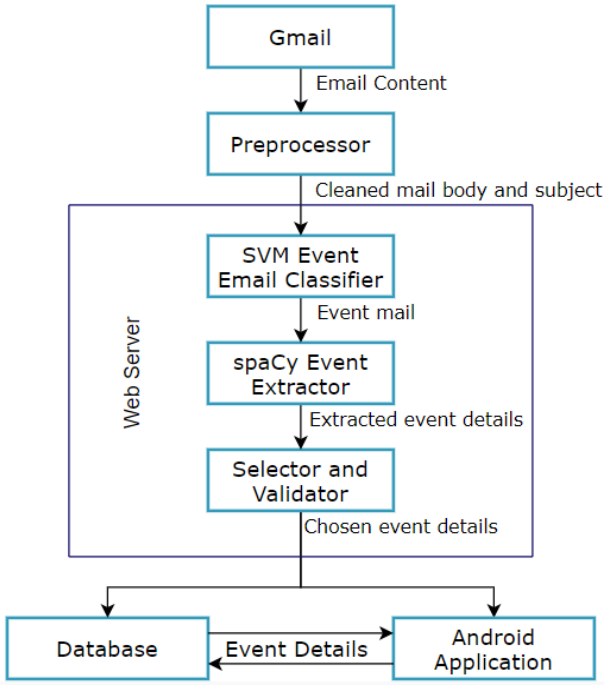

Fig. 1. System Architecture

explains the system architecture of the proposed system. In Fig. 1, the major modules can be seen. This includes the Data

Preprocessor, Email Classifier based on SVM, a spaCy Extractor and finally a Selector integrated with a Validator. Initially, the raw Gmail content will pass through a Data Preprocessor where the HTML tag removal, quoted text removal etc. will take place. The cleaned mail body will be concatenated with the subject, and will be transferred to the Support Vector Machine based Event Mail Classifier. If it is classified as an Event Email, it is transferred to the spaCy Extractor Module that will identify the event related information such as date, time, venue and link (if any). These details will be sent to a Validator to validate the date and time. In this module, the right details will be chosen from the list of entities identified by spaCy model. These processes take place in a web server and then the event details are stored in the database. An Android Application will provide the event details as well as reminders, using the information from the database.

5.1 Machine Learning Framework for Classification

Although event details could be directly extracted from an email body, the machine learning based classifier reduces the overload of searching for date, time etc. in mails that do not contain an event. Among the several machine learning techniques to design a binary classifier, the one using Support Vector Machine was found to be very simple and efficient. This technique is inspired from the detailing of SVM in [1], where different methods were used to perform spam email classification and SVM was proved to be much better than others. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes. In case of linearly inseparable data points, SVM uses a kernel trick to transform a low-dimensional input space and transforms it into a higher dimensional space.

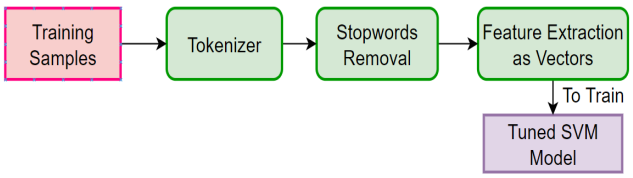

Fig. 2. Training SVM Classifier

Fig. 2 illustrates the classifier training. The SVM classifier is designed using a train-test split of 90:10. The train data is tokenized into individual words and stopwords are removed from the list of tokens to make the dataset more simple and specific. The words form the vocabulary. All unique tokens (numbers, words, special symbols, etc.) in the entire training corpus are identified and each one is treated as a single feature. A feature selection is applied to choose the most important words and reduce dimensionality. Each mail body is then represented by a vector that contains a normalized weighting for every word based on its importance. An instance of CountVectorizer class is created and the fit_transform function is called, that learns the vocabulary dictionary and return term-document matrix. Then the hyper parameter tuning is done using Grid Search Cross Validation. The RBF kernel function is used to transform the given dataset input data into the required form. The RBF kernel on two samples x and x_i, represented as feature vectors in some input space, is defined as:

$$K(x, x_i) = exp(-gamma * sum((x - x_i^2)) \quad (1)$$

where, gamma defines how far the influence of a single training example reaches, with high values meaning close and low values meaning far. C or the regularization parameter is used to maintain regularization. It is the penalty parameter, which represents misclassification or error term that tells the SVM optimization how much error is bearable. This way the trade-off between decision boundary and misclassification term can be controlled. Thus, the model is tuned with the chosen hyperparameters and then trained with the document matrix. This model is used to perform the Event Email classification.

5.2 Natural Language Processing framework for Event Extraction

An NER (Named Entity Recognition) system is used for event details extraction, based on the idea advocated in [7], with lot of customization and improvising. NER is a standard NLP problem which involves detection of named entities from a chunk of text, and classifying them into predefined set of categories. SpaCy is an open-source library that provides an efficient system for NER in python, which can assign labels to groups of tokens which are contiguous. It provides a default model which can recognize a wide range of named or numerical entities, which include person, organization, language, event etc. Apart from these default entities, spaCy allows adding arbitrary classes to the NER model, by training the model to update it with newer trained examples. SpaCy first tokenizes the text into words or word embedding. Words are then split into features and then aggregated to a representative number. This number is then fed to a fully connected neural structure, which makes a classification based on the weight assigned to each feature within the text.

The dataset required for training consists of nearly 220 mails. The sentences were tokenized into words using the regex tokenizer which avoided the problems of mistokenizing while using the default NLTK tokenizer. Regular expressions were formed for various formats of dates, time etc. and python nltk.tokenize.regexp module was used. Each token was POS tagged using NLTK POS tagger. Since this is a supervised learning problem, the training data should be annotated manually. The dataset consists of the following tags- DATE, TIME, VENUE and LINK. The dataset follows a BIO type tagging. The BIO format (beginning, inside, outside) is a common tagging format in named-entity recognition.

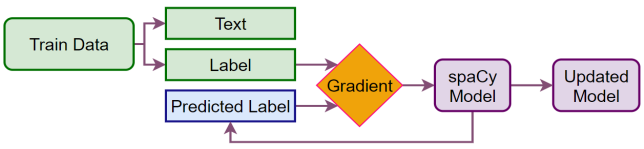

Fig. 3. Training spaCy model

The .csv file containing entities with POS tags and custom tags, was converted to a .tsv file then converted to a .json file. The .json was converted into a format needed by spaCy. SpaCy NER already supports the entity types like- PERSON, NORP, FAC, ORG, etc. The aim is to further train this model to incorporate new custom entities present in the dataset. SpaCy training can be seen in Fig. 3.The existing en (English) model was initially loaded and all other pipeline components were disabled during training using nlp.disable_pipes. This way, only the entity recognizer gets

trained. New entity labels are added to the entity recognizer using the add_label method. At each word, it makes a prediction. It then consults the annotations, to see whether it was right. If it was wrong, it adjusts its weights so that the correct action will score higher next time. Gradient is used to calculate weight change to improve the predictions after each iteration. On training spaCy model with several numbers of iterations, the 275 iteration model gave the best results. This spaCy model was used to perform event detail extraction.

5.3 Selector and Validator Module

All the details extracted by spaCy module are not necessary to depict the event. The extracted data is validated and required ones are selected using Selector and Validator module. It is simply a Python code that works with the output of the extraction module. Validation is done using Regular Expressions for Date, Time and Link. Initially, the best date is selected. Then the time, venue etc. are chosen based on that date's position. The position of each entity in the tagged entity list is used throughout for finding proximity. All the date entities will be converted into YYYY-MM-DD format and the least among them will be found. Past dates are eliminated by comparing with today's date. The position of the least date will be kept for reference. If there are multiple occurrences of the least date, all those positions will be preserved as a list. For time selection, all valid times will be converted into hh:mm hrs format. To start with, find the first time to the right of chosen date. If found, it will be selected as the time, else find the least of all times to the left of the chosen date. For selecting venue, find nearest B-VENUE and I-VENUE tags on either sides of the chosen date, since venue can be a multi-entity value. If closest one is a B-VENUE on right, append that tag's value with the values in I-VENUE tags coming continuously right after that, to get the venue. If closest one is an I-VENUE on left, search for a B-VENUE to its left through a sequence of I-VENUEs. If found, append together all values from tag B-VENUE to the I-VENUE which was initially found to get the venue, else discard the I-VENUE since its invalid. If closest one is an I-VENUE on right, discard it. If closest one is a B-VENUE on left, its value is the venue. If some invalid venue pops up in any case, check for closest B-VENUE to either side of chosen date and proceed as before. Online sessions, webinars etc. are identified and such mails are not searched for venue. Instead, the corresponding event like Webinar, Online Session etc. will be set as venue. Links are also chosen based on least distance from chosen date. If there are multiple occurrences of least date, time will be found considering each occurrence and then the date-time pair with minimum separation is selected. The same is done in case of venue. This way, date (DD-MM-YYYY), time (hh:mm hrs), venue and link are extracted by this module.

6. IMPLEMENTATION

6.1 Technology Stack

The technology stack for the proposed system is illustrated in Fig. 4. The machine learning model i.e. classifier is implemented using Scikit_learn. It is a free software machine learning library for Python. Pandas as well as NLTK are used for the classification phase. The NLP model for Event Extraction has been implemented using spaCy, an open-source software library for advanced natural language processing written in Cython and Python. Regex and dateutil libraries have been used for validation and other purposes. The Android application has been developed

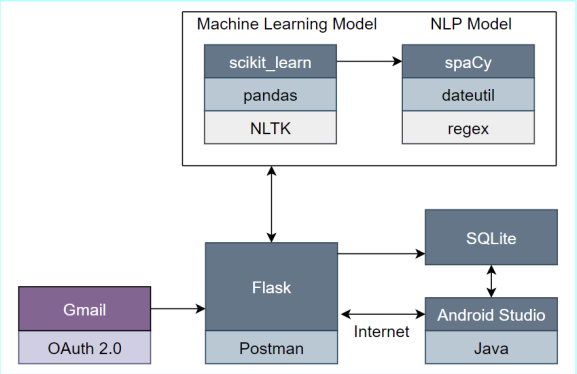

Fig. 4. Technology Stack

using Android Studio that uses Java. This provides the user interface as well. SQLite is used as the Database Management System. It is a relational database management system contained in C library. This is used to create a Client side database. Web server is created using Flask, a micro web framework written in Python. Further, this server is tested using Postman. The emails are accessed via a Gmail API which is authorized using OAuth 2.0.

6.2 User-Interface

The user-interface of android application is created using Android Studio. Upon logging in using Gmail credentials, user can view the list of subject of event mails in the $Events$ tab. On clicking each subject, the user gets the details of that particular event such as date (DD-MM-YYYY), time (hh:mm hrs), venue and link, where the user gets options to open Gmail, star the event, add a reminder for the event to the Google Calendar or delete the event. The $Starred$ tab will display the starred events. The $Week$ tab will give the list of upcoming events during the ongoing week. Using the profile icon, the user can view account details and can logout.

6.3 Database Implementation

A client-side database is implemented using SQLite used to avoid overheads. The json response from the web server will be parsed to obtain the event details. These details will be stored in the database. When the application is installed in a device for the first time, a database will be created in that device by the name $Userbase$. When a user logs in for the first time, a table will be created for him, with columns such as msg_id, subject, date, time, venue, link, calendar and starred. msg_id stores message ID which will be unique for each email and thus it will be taken as the primary key. Starred is to identify starred events and Calendar is to identify events that have been added to the Google Calendar. Being a client-side database, it has some drawbacks like reduced portability of information. To cope up with this, when a user logs in a new device for the first time, his table will be created in $Userbase$ since there is no old table corresponding to him and the event extractor model will run on the latest 50 mails and corresponding events will be stored in the new table in database.

6.4 Gmail Connectivity

Gmail connectivity is realized using Gmail API. When the user logs into the application using Google Signin, he/she is prompted

for allowing access to sensitive data like Gmail mailbox and Google Calendar. During this, a server authorization code is requested, which is later exchanged for an access token. The value of the access token is stored in the Authorization table in the SQLite database, which in turn is used to build the Gmail API service for accessing the mails. Various client libraries are used to obtain a list of message resources containing message IDs and some metadata. These message IDs are used to get the actual body of the mails.

6.5 Google Calendar Mapping

Since authorization is common to both Gmail and Google Calendar, access token already stored in the authorization table is once again used to build the Google Calendar in this case. The event is to be added to the Google Calendar of the current user. The event date converted to YYYY-MM-DD format is given as the start and end date. The subject is set as the title of the event. The time, venue and link are also mapped to the Calendar as event description. All the mapping is done using various client libraries provided by Google. Upon clicking *Add reminder* button, all the event details mentioned above are automatically added to the primary calendar of the user in his/her Google Calendar.

7. RESULTS

7.1 Model Performance

The performance of the model is calculated by monitoring the individual modules by using a test set of new emails that are from various people and related to different events. The emails are given to the model in the exact way in which they are rendered in the final model.

1) *SVM Classifier Performance*: The SVM Classifier model exhibited the highest performance when the RBF Kernel was used with hyperparameters C as 1000 and gamma as 0.001. The comparative performances on using other pairs of C and gamma value are depicted in the graph in Fig. 5.

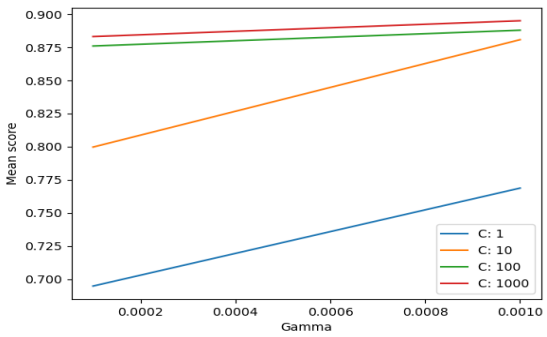

Fig. 5. Best Accuracy with C=1000 and gamma=0.001 (89.5%)

A training accuracy of 89.5% was obtained. On testing with 150 new emails, 131 were correctly classified and 19 were misclassified. Thus, an accuracy of 87.33% was gained. Out of these 19, 7 are non-event mails misclassified as event mails. This error is rectified in the event extraction module as they are discarded since no event is extracted from them. This gives an effective accuracy of 92% for the classifier model. The Confusion Matrix and Classification Report of testing can be seen in Fig. 6 and Fig. 7 respectively.

```
CONFUSION MATRIX

[[40  7]
 [12 91]]
```

Fig. 6. Confusion Matrix - Testing

```
          TESTING - CLASSIFICATION REPORT

              precision    recall  f1-score   support

           0       0.77      0.85      0.81        47
           1       0.93      0.88      0.91       103

    accuracy                           0.87       150
   macro avg       0.85      0.87      0.86       150
weighted avg       0.88      0.87      0.87       150

Test Accuracy:  0.8733333333333333
```

Fig. 7. Classification Report - Testing

2) *Event Extraction Model Performance*: The overall accuracy of a spaCy model could not be calculated using any methods. For 275 iteration model, the loss was initially 6276.3559 which gradually reduced to 4776.3059 at the end of 275 iterations. Increase in number of iterations cause overfitting reduction leads to the omission of important tagged entities. The model was tested using 100 mails and model performance was analyzed by calculating the accuracy taking into consideration various cases. Out of the total 100 mails, 83 mails gave all the extraction details accurately and 17 mails had at least one mistake in extraction. Thus the spaCy model has an accuracy of 83% in this regard. Out of these 17 mails, date and time were correct and venue was wrong for 12 mails. So the effective accuracy is to 95% ignoring venue.

3) *Integrated Model Performance*: It was done after merging classifier and extraction modules. Integrated model was tested using a dataset of 150 mails containing 103 event mails and 47 non-event mails. Out of this, 121 emails gave accurate results, yielding an accuracy of 81.33% for the whole system. On ignoring venue, the performance has been increased to an accuracy of 88.67%. Moreover, 5 out of 7 misclassified non-event mails have been discarded in extraction module successfully. The reduction in effective number of misclassifications was successfully achieved by the extraction module.

7.2 Testing and Validation

As stated in the previous section, each component of the system has been tested separately. Integration testing was also performed after the final system was built. The project was successfully validated and verified to ensure that all the requirements as per the problem statement, have been met. Several test cases used to test the working of the system are outlined in this section from Table 1 to Table 4.

Table 1.

Test Case	Event Mail with all Details
Preconditions	The application is linked to user's Gmail Account
Test Data	Dear sir, a meeting of Research Council is scheduled at 3 PM to 3.20PM on 29th August, 2020 in the Conference hall. All members are requested to attend.. With regards, Dr. Chithraprasad, Dean Academic
Expected Result	DATE : 29-08-2020 TIME : 15:00 hrs VENUE : CONFERENCE HALL Event details displayed in Android App with an option to set reminder.
Actual Result	["B-TIME 3 PM","B-TIME 3.20PM","B-DATE 29th August, 2020","B-VENUE Conference", "I-VENUE hall"] DATE : 29-08-2020 TIME : 15:00 hrs VENUE : CONFERENCE HALL
Postconditions	Event Details display in Android Application
Status	Pass

Table 2.

Test Case	Event Mail with fewer details
Preconditions	The application is linked to user's Gmail Account
Test Data	Subject: Forest Department Notification for Forest Guard posts Webveus KFD Jobs Recruitment of Forest Guard Posts Forest Department Jobs Recruitment Notification 2020 Forest Department (KFD) inviting applications for the positions of Forest Guard. Last Date for Submission is : May 15th, 2020. APPLY NOW webveus All Rights Reserved. To stop receiving these emails please click here to unsubscribe.
Expected Result	DATE : 15-05-2020 TIME : Not Specified VENUE : Not Specified Event details displayed in Android App with an option to set reminder.
Actual Result	["B-DATE May 15th, 2020"] DATE : 15-05-2020 TIME : Not Specified VENUE : Not Specified
Postconditions	Event Details display in Android Application
Status	Pass

Table 3.

Test Case	Non-Event Mail
Preconditions	The application is linked to user's Gmail Account
Test Data	Unlock your creativity and win exciting prizes Participate now for a #BreakFromBoredom Email not displaying correctly? View in Browser [image: HDFC Bank NetBanking — Your home branch way from home.] [image: HDFC Bank Net Banking — Your home branch away from home.] Participate Now Warm regards, HDFC Bank Terms & conditions apply — Unsubscribe Based on Retail Loan book size (excluding mortgages). Source: Annual Reports FY 18-19 and No.1 on market capitalisation based on BSE data as on 31st Dec, 2019.
Expected Result	Email will be classified as Non-Event mail and it will be discarded.
Actual Result	Email Classifier classified the mail as Non-Event and discarded
Postconditions	No action taken.
Status	Pass

Table 4.

Test Case	Non-Event Mail
Preconditions	The application is linked to user's Gmail Account
Test Data	Scheme to implement the PMGKY package for credit of Provident Fund Contributions for three months by Govt. of India Dear employer, The Govt. of India on 26.03.2020 announced Rs.1.70 Lakh Crore relief package under PMGKY for the poor to help them fight the battle against Corona Virus Pandemic. The Central Govt. proposes to pay 24 percent of the monthly wages into EPF accounts for next three months of Wage earners below Rupees fifteen thousand per month, who are employed in establishments having up to one hundred employees, with 90% or more of such employees earning monthly wages less than Rs.15000/-. Regards, Regional PF Commissioner, Thiruvananthapuram
Expected Result	Email will be classified as Non-Event mail and it will be discarded.
Actual Result	["B-DATE 26.03.2020","B-VENUE Central", "B-DATE 90%"] Email Classifier classified the mail as Event mail but it was rectified in Extraction phase as no proper date was traced and discarded.
Postconditions	No action taken.
Status	Pass

8. FUTURE SCOPE

This project is developed using the newly made custom datasets, but such an approach is bound to have some flaws in the long run. The two core models, SVM classifier and spaCy extractor can be improved by using better datasets. The proposed application is supposed to notify users in real time when he/she receives an event mail but on a practical level this is quite difficult to achieve. The current system finds out event mails from Gmail on each time the

app opens. This is enough in user's perspective, but it is not a real-time process when Gmail is concerned. One of the methods of achieving real-time processing is by using webhooks. However, the way the backend server is setup, such an approach is out of the question. Thus, the server setup can be improved in future. Various features like setting custom reminders to events, editing Google Calendar from within the app, etc. This project is based specifically on Gmail and can be extended to other platforms like Yahoo, Outlook, WhatsApp, Facebook Messenger, etc.

9. CONCLUSION

Daily life is getting more hectic and the new goal is to alleviate some of it. On average, a person receives about 120 mails per day according to studies. Hidden among this vast plethora of mails, there are many important ones. These may range from invitation to some event to some important official appointments. Going through all the 120 mails is a huge task. This project makes this easier by filtering out mails that contain any kind of event and extracting the important details using NLP techniques. A user friendly android application with features like adding events to favorites, viewing the mail in Gmail app and most importantly, a system was implemented that can map the events automatically to Google Calendar on a single click to get notified each time a new event is received has also been developed successfully. Data privacy is also respected, since a client sided approach is used for maintaining the database of the project. The source code for the proposed system has been made available in GitHub under open source license [10]. A project video [11] was also launched on YouTube, which summarizes the purpose and working of the system, along with the UI of the EventEx App. We expect this project to be helpful to all those people that have a heavy mailbox at the end of the day.

10. REFERENCES

[1] Rita McCue. "A Comparison of the Accuracy of Support Vector Machine and Naive Bayes Algorithms In Spam Classification," University of California at Santa Cruz, 2009

[2] Shashank Senapaty. "Detection and Extraction of Events from Emails," Department of Computer Science, Stanford University, Stanford CA, December 12, 2008.

[3] Julie A. Black and Nisheeth Ranjan. "Automated Event Extraction from Email," Stanford University, 2004.

[4] Ola Amayri, Nizar Bouguila. "A Study of Spam Filtering Using Support Vector Machines," Artif. Intell. Rev.. 34. 73-108. 10.1007/s10462-010-9166-x, 2010.

[5] Aneesh G. Nath, Krishnanth V, Kevin Biju Mathew, Pranav T S, Sarath Gopi. "NLP based Event Extraction from Text Messages," International Conference on Future Technology in Engineering, 2016.

[6] Aswar Shreyas, Gaikwad Priyanka, Merlyn Pearl, Shinde Swapnal. "Event information extraction from email and updating event in calendar," Vol-4 Issue-3 2018, IJARIIE-ISSN(O)-2395-4396.

[7] Eleni Partalidou, Eleftherios Spyromitros-Xioufis, Stavros Doropoulos, Stavros Vologiannidis, Konstantinos I. Diamantaras. "Design and implementation of an open source Greek POS Tagger and Entity Recognizer using spaCy," IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2019

[8] Ashraf Q. Mahlawi, Sreela Sasi. "Structured Data Extraction from Emails," International Conference on Networks Advances in Computational Technologies (NetACT), vol. 37, pp. 323-328, 2017.

[9] Michal Laclavk, Stefan Dlugolinsky, Martin Seleng, Marcel Kvassay, Emil Gatial, Zolt an Balogh, Ladislav Hluchy. "Email analysis and information extraction for enterprise benefit," Computing and Informatics, Vol. 30, 2011

[10] https://github.com/arjrrk/EventEX-Final-year-project

[11] https://youtu.be/WmAPrTs9cnk