AN INTELLIGENT MOBILE APPLICATION TO AUTOMATE THE CONVERSATION OF EMAILS TO TASK MANAGEMENT USING AI AND MACHINE LEARNING

Yi Li[1] and Yu Sun[2]

[1]Seattle Academy, 1201 E Union St, Seattle, WA 98122
[2]California State Polytechnic University,
Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

It is no secret that a large portion of our population struggle with task management [1]. According to a 2021 research, twenty five percent of people do not employ a task management system and simply work on "whatever seems the most important at the moment". Among the population that do use some sort of task management, the most popular form of managing personal tasks is through to do lists (33%) followed by using their email inbox (24%). Thus, I thought to combine these two most common methods by creating a to do list automatically generated from the email. We used the open sourced software natural language processing (NLP) to pick out the important sentences in the text and convert them into tasks for the to-do list. We used keywords such as "tomorrow", "next", "month", etc combined with the date the email was sent to determine the "due date" of the to do list. Because the to-do list extracts information directly from the inbox without any participation from a human, unlike many other apps, this could be effectively used by those that do not check their email.

KEYWORDS

Mobile platform, Machine Learning, NLP.

1. INTRODUCTION

According to surveys, self reported reading either their personal or work email regularly. This makes preexisting lists and calendars all but ineffective as they require a human to add task to the task management system. A second problem that is also quite prevalent is that there is the chance that even if you have read your email, you neglect to add it to your calendar. As someone that has dealt with these issues personally, I wanted to find a way to reduce the likelihood that such things happen because the cost of missing opportunities is too high. In order to solve this problem, we designed and created an App that will automatically generate a to do list from your inbox [2].

This topic is very important because it will help people make the most of their opportunities -- by letting them know that they have the opportunity.

Currently there is no shortage of apps that try to address this problem. The most popular Include google calendar, smart sheets, notion, and various digital to do lists [3]. These apps are all very well developed and used with many advanced features. However, these platforms share a weakness---they cannot automatically extract tasks directly from an email and add it to the to do

David C. Wyld et al. (Eds): FCST, CMIT, SE, SIPM, SAIM, SNLP - 2022
pp. 13-21, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.120802

lists unless the senders of the email send invites [4]. The issue is that it is completely impractical to expect such consideration from everyone that emails you, and this consideration would be impossible if the email is a school announcement, or any sort of advertisement meant for many people as they do not know you are coming. Another issue, as mentioned before, that is more commonplace than people would admit is that many emails are left unread. 55% of all email users admit that they don't open either business or personal emails regularly. This means that 55% of the population does not even get the chance to use these other calendars or to-do lists effectively as they all require human involvement. The benefit of this app is that it boils down emails to the tasks you need to complete so you can quickly skim over all of them and be reminded of the things you need to do or you can do. It is important to note that this app is not strictly a to do list in the traditional sense, it is more like a reminder list of the things that you can do. Overall, there are some differences in the intent of this app compared with preexisting ones. Instead of focusing on creating an impeccably tailored list of the things you will do, we chose to focus on making sure that you are remembering that certain events exist so you don't accidentally miss an event because you did not see it when perusing your email or failing to do so.

Using NLTK, we understand every word within the sentences in the email [5]. There are two rough parts to generating a task. Understanding the task and extracting the date and time. For the actual task content, our algorithm uses NLTK to parse the sentences and tag each word with the category it falls under. I.e. nouns, verbs, numbers etc. Using this information, our program especially targets action verbs to understand what the email is asking the user to do. On the other hand, in order to obtain the date, the program uses a variety of approaches. NLTK parses the sentences and gives us a "tag" of what type of word it is (or if it is a number). This narrows down the text we need to search for keywords. For example, we only need to search for proper nouns and numbers when trying to find the date. The algorithm checks for key words such as "January" and variations such as "Jan", "jan" for every month, day of the week. In addition to this, it also checks for relative time such as "tomorrow", "next month" "next week" and etc to determine the date if it is not explicitly listed. It also detects whether a few strings of numbers are in fact critical dates or times. Once the task and the date is determined, we export our results to our front end. The back and front ends are connected through the web framework flask. Our front end is built with dirt. We chose this as it allowed us to create beautiful interfaces relatively easily.

The road map for the rest of the paper is as follows: section two will describe the road bumps and challenges we ran into during the project. Section three details how we resolved the problems outlined in section two. In Section four we will explain our testing process and the subsequent results about the app's performance. Section five presents related works done by others to make a side by side comparison of our app. Lastly, section six includes concluding remarks as well as areas needing further research.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. How to determine the due date

A challenge we encountered was determining when an action is "due" when no specific date and time were given in the email. For example, the program needs to understand what day "tomorrow" is in the sentence "please arrive by tomorrow." This is more complex than one might initially think. As humans use various forms of speech to describe the same thing, the computer needs to be able to understand all forms of communication [6][7]. It needs to understand the words such as "minutes" and "hours", all of the days of the week, all the months of the day when spelled out

such as “January” “as well as that “1/19” is also in January. Beyond these basics, it also needs to understand things like “next”, “tomorrow”, and “yesterday”. Furthermore, the program needs to be able to make smart assumptions about what the “due date” is when it does not have definitive information. For example, it needs to be understood that unless stated otherwise, the phrase “see you on Tuesday” usually means the upcoming Tuesday.

2.2. How to make sure the computer is intelligent enough

Another major issue we faced when writing the algorithm is making sure that the computer is intelligent enough to be useful. Which in this case means the ability to discern what information is relevant and ought to be extracted and what is irrelevant noise. This is no easy feat for a computer. Although we have access to the nltk library, it can only go so far as to help us analyze the sentence structure-- which, although is necessary, it is far from being sufficient. The computer needs to know either what you need to do, what event is happening, what registration is closing, as well as what day that is happening, at what time, and at what place. Simply an analysis of what is a date, or what is time is not going to cut it. Furthermore, as this is the core of the app, it is incredibly important that it functions without a hitch.

2.3. How to publish APP

Another road bump we faced occurred during app publication [8]. Things did go smoothly with the app login and required multiple adjustments in Firebase for it to function properly [9]. Furthermore, during our testing period (after we fixed the previous issue), for some inexplicable reason, our app stopped loading the login that it was capable of just a few days before. We eventually were able to find a potential fix for the issue, but the app will need to be re-approved before use ---which stalled the testing process even more.

3. SOLUTION

TODO is a smart automatic to do list generator that uses your email inbox to generate a list of things happening each day that might be easily forgotten otherwise. It is capable of finding and extracting concrete things to do from an email. Or in other words, it is not only a to-do list, but it can also function as an email summarizer even though that is not its intended purpose. The process of its operation are as follows. First, whenever you receive an email, TODO will be able to access it and analyze the sentences using an open sourced library called nltk (natural language toolkit). Next the program goes through each sentence to see if a “due date” is stated within the sentence. If it is, this sentence is deemed the “action” of the “task”. We chose to use date as an indication of an “action” because in virtually every task, there will be a time when you need to be finished. Furthermore, this will likely be reflected in practically all sentences with actions. For example: “Let’s talk on 9/30/21”. First of all, it would only make sense if there is a date there, second of all, it would not be a pressing matter to do if no time was specified. Although this is an assumption, we believe it is reasonable [10]. After we have obtained the due date and the action, we will stitch them together as one task and send them to the app to be displayed.

The first step is to access the email of the user. This is quite a simple task; we simply have to set up the correct email verification. After we ’ ve done this, access to all of the text within the g mail inbox is at our program’ s disposal.

Our second step is to iterate through every sentence of each new email tokenize (split text into chunks for the ease of analysis) and analyze them. For the analysis, We used labels provided by the nltk open sourced library. Examples of the labels we employed are “DATE” , “TIME” ,

"PLACE". The nltk library is able to automatically determine if a text is a date, time, place and etcetera and subsequently label it through their trained algorithm.

```
def parse_todo_list(sentences):
    todo = []

    sentence = remove(sentences)
    # print(sentence)
    tag = tokenize(sentence)
    for a in tag:
        tree = entities(a[0])
        print(a[1])
        try:
            parsed_date_time = str(parse(a[1], fuzzy=True).strftime('%m-%d-%Y
%H:%M'))
            parsed_actions = parse_action(tree)
            todo.append((parsed_date_time, parsed_actions))
        except:
            print('exception')
            pass
    return todo
```

Figure 1. Screenshot of to do list code

Our next step is to obtain the "due date" corresponding with that email [11]. This is harder than it seems. Humans tend to write dates in very nonuniform ways. Some write it as "dd/mm/yyyy", some simply write "tomorrow", some say "next Tuesday", and much more. We had originally written code to be able to recognize this, but later we found a library that handled this much more efficiently than ours so we adopted their system. We set whatever date and time we find to Parsed_date_time. The code for this is the first line in the "try" block in the figure above.

After this, we called the parse_action function and passed in the tree (sentences labeled with grammar, and other labels such as date) to find the task we were to do. As can be seen in the code below, we iterated through every node in the tree and all appended sentences with dates to "node". Following this, we checked if the nodes with dates also have a specified time. All nodes that do will be stored in node1 and the previous node returned to an empty state. We then checked to see if a location was specified. After this, we try to see if the node also has a specified place. The last if statement checks that if the node is not empty (has date), the node (sentence) will be the action of this email's task. It is important that all actions within the to-do list have a due date for the sake of prioritization and organization.

After completing this, we appended the parsed_date_time (from the previous image) with the action we just obtained and appended them together. Which becomes a "task" ready to be uploaded to the to do list.

```
def parse_action(tree):
    node = []
    node1 = []
    node3 = []

    for i in tree:
        node.extend(extract_nodes(i, 'DATE'))
    for i in node:
        node1.extend(extract_nodes(i, 'TIME'))
    node = []
    for i in node1:
        node.extend(extract_nodes(i, 'PLACE'))
    for i in node:
        if isinstance(i, tuple):
            node3.append(i[0])
        else:
            node3.append(i[0][0])
    return ' '.join(node3)
```

Figure 2. Screenshot of parse section code

Our final step is to upload the "task" onto the app. We did this through a very simple function as can be seen in the code below. We called the functions mentioned before and sent it to the app via json where it was displayed.

```
@app.route("/", methods= ['POST'])
def getTodoList():
    sentences = request.data.decode('utf-8')
    print('sentence:', sentences)
    result = functions.parse_todo_list(sentences)
    result = json.dumps(result)
    print(result)
    return result

app.run(host = '0.0.0.0')
```

Figure 3. Screenshot of app route code

4. EXPERIMENT

Experiment 1

In order to verify that our solution can effectively solve problems at different levels and have good user feedback, we decided to select multiple experimental groups and comparison groups for several experiments.

For the first experiment, we want to prove that our solution works stable and continuously, so we choose a group size of 40 different Tasks in 4 different types. The 5 different types of tasks are Course Schedule, Volunteer Activity, Driving Activity, and Sports Activity. The goal of the first experiment is to verify if the Task Manage System works well for different types of tasksThrough sampling 4 groups of tasks of different types and asking the same person to finish all these tasks with the schedule of our app. Results are collected by statistics if the app helps the user save time by scheduling all the tasks automatically than not using the app. Experiments have shown that all tasks in different types have a high rate of saving time. Class Schedule has the most high saving rates, which means our AI works much better in class scheduling field. This experiment could explain that the task types do have an obvious impact on the arrange results. The average saving time (in minutes) of 4 different types of the task shows below:

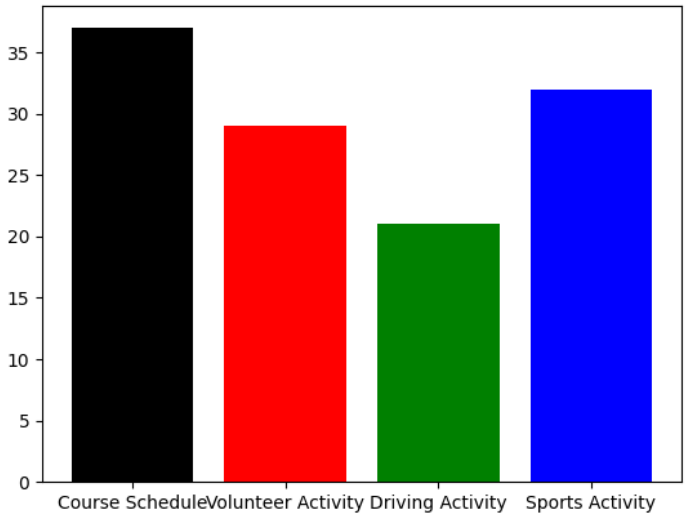

Figure 4. Result of Experiment 1

Experiment 2

A good user experience is as important as a good product. So a perfect solution should have excellent user experience feedback. In order to prove that our solution has the best user feedback, we specially designed a user experience questionnaire. We statistics the feedback result from 100 users, we divide those users into Five different groups. The first group of users ages from 10 - 20, the second group of users ages from 20 - 30, the third group of users ages from 30 - 40, the fourth group of users ages from 40 - 50, the fifth group of users ages from 50 - 60. The goal of the first experiment is to verify high feedback scores shows high performance We collected the feedback scores form these 5 different groups of users and analyze it. Experiments have shown that users who ages from 10 - 20 give the highest result feedback to our app. Which may because of paywall link appears more in the game searching The experiment graph shows below:

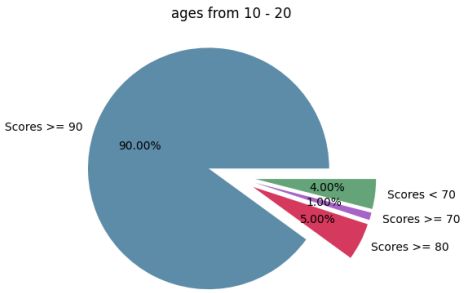

Figure 7. Result of age 10-20

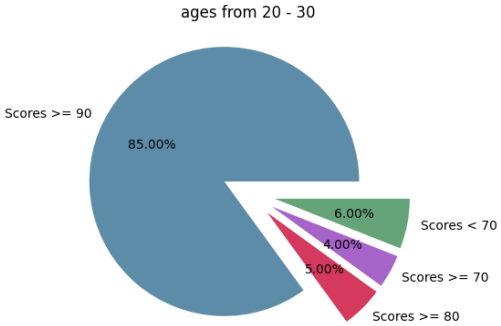

Figure 5. Result of age 20-30

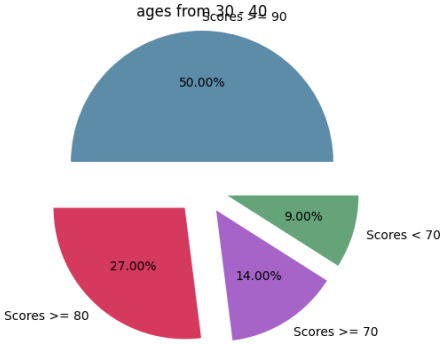

Figure 6. Result of age 30-40

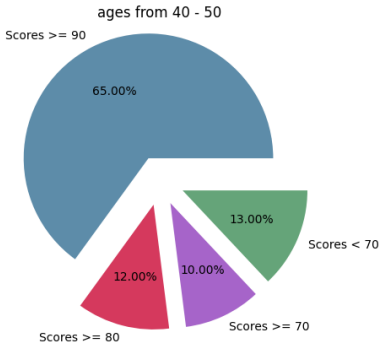

Figure 9. Result of age 40-50

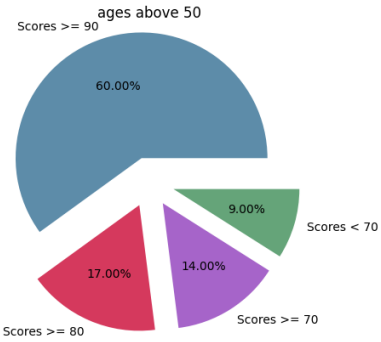

Figure 8. Result of age above 50

5. RELATED WORK

Talking diary is an app for android that takes audio notes to process and sort into tasks [12]. It has three main components: Classifier, Scheduler, and a working hour calculator. Note: in this case, the classifier simply determines the type of tasks (ie. work, home, school, etc). When comparing this app with TODO, there are many differences. For one, although both use NLP, the point of TODO is to detach all human involvement in the generation of the to -do list while Talking Diary wants people to leave verbal notes. As for the organization and scheduling portion of the app, it is quite clear that Talking Diary has the advantage. It is able to know how much time you have, how much work you have to do, etc.

Pyhop1 is a hierarchical task network planner written in Python sharing similarities with SHOP [13]. It uses no loop detection to accomplish its tasks. With in this platform, there are three different categories of items one can enter: tasks, goals, and action. The main difference between SPyhop and TODO is that Pyhop focuses on task management while TODO's key point is task acquisition. Their algorithm in task organization is without a doubt, far superior and more complex than that of TODO's but these two platforms are meant to accomplish different things. One is meant for prioritizing and organization while the second is to simply remind you of the miscellaneous things you should do based on your inbox.

Schedule Me has four main components: Data engineering, Intelligent task breakdown and Scheduling using constraint programming, Reinforcement Learning for Personalized Task Scheduling, and User Centered Interaction Design. In data engineering, Schedule Me is able to scrape information from all relevant websites (such as school websites) and store them into a database [14]. It is also able to automatically break down tasks for you using constraint programming. Furthermore, it will organize your tasks to optimise information retention. Lastly, it makes it easy to use. In comparison with TODO, this is a very advanced scheduling system that both organizes tasks as well as acquire tasks. The difference between the two apps is that TODO is more targeted towards the email while Schedule ME is all encompassing. Furthermore, TODO does not schedule your time, it simply reminds you of the things happening today.

6. CONCLUSIONS

Task management is an important and complex issue whose consequences are significant. A missed email could be a missed opportunity. By using NLP to automatically compile a to-do list, it makes it more difficult to forget or miss something. That it can extract tasks completely

autonomously means that It does not require people to already have good habits to use the app effectively. Experiments show that the app is most capable of recognizing and extracting tasks related to school work. User feedback indicates general positive reactions to the app, especially among the 10-20 age group.

There are still great limitations on the rate at which the app is identifying and saving the task from the user email–especially tasks that are non-coursework related. This impacts the reliability and dependability of the app. In addition, there is room for improvement in the user experience — especially for populations older than 30.

Further research will involve using sample emails with a wider array of topics during App development which should increase its adaptability and dependability [15]. In addition, we need to collect more user feedback, especially written user feedback so that not only do we know which age group is enjoying the app the most, we can understand why that is. This information would be important for future improvements.

REFERENCES

[1] Bellotti, Victoria, et al. "What a to-do: studies of task management towards the design of a personal task list manager." Proceedings of the SIGCHI conference on Human factors in computing systems. 2004.
[2] Joorabchi, Mona Erfani, Ali Mesbah, and Philippe Kruchten. "Real challenges in mobile app development." 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. IEEE, 2013.
[3] Feddern-Bekcan, Tanya. "Google calendar." Journal of the Medical Library Association: JMLA 96.4 (2008): 394.
[4] Gillespie, Tarleton. "The politics of 'platforms'." New media & society 12.3 (2010): 347-364.
[5] Loper, Edward, and Steven Bird. "Nltk: The natural language toolkit." arXiv preprint cs/0205028 (2002).
[6] Lüders, Marika. "Converging forms of communication?." Ambivalence towards Convergence: Digitalization and Media Change, Gothenburg: Nordicom (2007): 179-98.
[7] Janoschka, Anja. Web advertising: new forms of communication on the Internet. Vol. 131. John Benjamins Publishing, 2004.
[8] Ithnin, Muslimah, et al. "Mobile app design, development, and publication for adverse drug reaction assessments of causality, severity, and preventability." JMIR mHealth and uHealth 5.5 (2017): e6261.
[9] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.
[10] Lawrence, W. Gordon, Alaistair Bain, and Laurence Gould. "The fifth basic assumption." FREE ASSOCIATIONS-LONDON- (1996): 28-56.
[11] Keskinocak, Pinar, and Sridhar Tayur. "Due date management policies." Handbook of quantitative supply chain analysis. Springer, Boston, MA, 2004. 485-554.
[12] Munir, A. Hani, Abubakar Manzoor, and Utba Aziz. "Talking Diary: A Novel Approach for Automatic Audio Note Categorization and Event Scheduling for Android Application." (2020).
[13] Stanton, Neville A. "Hierarchical task analysis: Developments, applications, and extensions." Applied ergonomics 37.1 (2006): 55-79.
[14] Liyanage, A. N., et al. "ScheduleME-Smart Digital Personal Assistant for Automatic Priority Based Task Scheduling and Time Management." 2021 2nd Global Conference for Advancement in Technology (GCAT). IEEE, 2021.
[15] Mota, José Miguel, et al. "Augmented reality mobile app development for all." Computers & Electrical Engineering 65 (2018): 250-260.