Google Calendar Analytics

Andrea Chamorro
anch9699@colorado.edu
106628307
CSCI 4502

Xinyu Jiang
xiji6874@colorado.edu
109036441
CSCI 4502

Qiuyang Wang
qiwa8995@colorado.edu
108741913
CSCI 4502

Vladimir Zhdanov
vlzh1513@colorado.edu
104573924
CSCI 5502

ABSTRACT

The problem which this project sets to address is unconscientious time consumption. People spend their time sub-optimally throughout their day, and this can be improved by providing insights into a user's daily life. With this project, we analyze Google Calendar data in attempt to categorize, analyze, and visualize events. Prior work on this topic has been minimal, with very few tools having accomplished analyzing a person's calendar data.

We took the approach of tagging the calendar events into several distinct categories: Academic, Entertainment, Health, Personal, Miscellaneous, and Travel. We performed K-fold validation on these tags, found frequent patterns in the events, and performed statistical extraction on the data. From this, we are able to generate textual summarizations specifying a user's most interacted with people, places, airlines, and attended classes. We also create graphs and heat maps for our data at varying granularities, each of which exposed insights that could be used under different contexts the user might need. Finally, we discuss immediate scenarios in which the data visualizations created might be useful, and propose future steps that might improve the quality of our item categorization algorithms in the future.

Author Keywords

Calendar, K-Fold, Pattern Recognition, Statistical Extraction, Tagging, Temporal, Visualization

INTRODUCTION

The world we live in today revolves around productivity. Many of the events we go to, the meetings the we attend, or the choices we make, have some sort of structure to them. As such, many people tend to employ the use of a calendar to keep track of their events.

With this project, we intend to attempt to mine multiple users' Google Calendar data to gain useful analytics and insights for a user about their time usage from their calendar data, and display it visually.

Motivation

Calendars play an important role in our lives. A calendar can help a user create a routine in their day, and can help one stay on top of all of their daily activities. It also allows a user to prioritize certain events over others, stay on task, and even remember to take breaks. In short, a calendar is a powerful and useful tool.

Recently, the music company *Spotify* released their annual Year-In-Review feature (2019 Wrapped), which commemorated all of a user's interactions with Spotify and its content. This included a lot of statistical information on a user's top music, time spent listening, and favorite genres. These types of generalizations are very insightful because they provide a myriad of information about habits which users may not even know about or realize. As a result, they are able to reflect on their choices, and are able to get to know themselves better.

As part of this project, we hope to accomplish something similar, but with a user's Google Calendar data. We hope to provide analytics and information based on the collected calendar data so that a user will be able to better analyze their calendar usage and habits. This sort of analysis is valuable, as it can help gain some key insights into the individual's time consumption, to then allow them to effectively reflect and adjust their time allocations in the future and gain greater control over their time.

Problem

The problem which we are trying to address is the idea of unconscientious time consumption. The idea being that one spends time sub-optimally throughout the day, and these sub-optimal time allocations are likely to be linked to a certain pattern, category, or activity.

We plan to categorize events together which are not perfectly similar and label them under a category. Under these categories, we will extract patterns from the category features. By extracting subset rules that exist and displaying correlations visually, it will be easier for a user to gauge their habits and make informed changes to their time consumption to better meet their needs.

Also, we plan to provide statistical data on a user's events in the form of temporal and categorical charts and visualizations. These will allow a user to see a better breakdown of their calendar usage, both by category and over time, further allowing them to make informed decisions about their life.

Prior Work

Overall, there has not been much prior work in attempting to analyze a user's calendar data. There seem to be only a

few tools which currently exist which do this, and they are heavily limited to the domain of helping employees within a company schedule their time around each other, or the like. Few are intended specifically for personal insight on one's unproductive time. In addition, these tools rarely attempt to group a user's activities into categories or cluster which might be useful to the user for summarization purposes, or provide statistics specific to a user's interests.

Challenges

There exist many challenges that stem from working with a Google Calendar dataset. While this dataset is filled with much useful information, the SUMMARY and NOTES sections pose a large natural language processing challenge. This is especially true since we are working with multiple users' datasets, and it is quite likely that their language style varies dramatically.

Specifically, while NLP tasks in the past often have much more data to work with to disambiguate sentences [3], we have fewer word descriptions to work with than the most similar styles of language processing. Luckily, these words will mostly have a certain domain they exist in: Action, Person, or Location, and so it may be possible to preset our parts of speech analysis to revolve around these.

Also, in working with multi-dimensional data such as this, a challenge lays in choice exclusion or inclusion of some fields over others in our analysis. So, even just focusing on the textual descriptions and times of events will be deeply informative.

Another challenge, specifically with this dataset, is that not all fields are required for each calendar event. Apart from the summary, start time, and end time, all the other fields are optional, and are thus only filled out for a smaller subset of the entire dataset. However, because of the nature of Google Calendar, we can rely on temporal data and a header description to be present in almost every case.

Possible Contributions

By analyzing a person's calendar, we can have a fuller understanding about our daily life. This answers questions such as where do we spend our time the most and what our daily routines are. We provide specific analytics which will be useful to a large majority of people, such as calendar use over time and frequent reoccurring groups of events. By aggregating events under specific tags, we are able to gain a better understanding of time consumption by category in one's life.

Organization

This report is organized into several main sections: methodology, evaluation and discussion. In our methodology, we describe the datasets, steps taken during preprocessing, design, implementation, and data analysis. The implementation section discusses the implementation of naive tagging, predictive tagging, k-fold validation, frequent pattern detection, statistical extraction, and textual evaluation. In our evaluation, we describe metrics and key results. And, in our discussion, we describe lessons learned and limitations of our current results.

RELATED WORK

Calendar Analytics Tools

There exist several online tools which measure analytics of a user's calendar data, the most pertinent of which are *Google Calendar Hours* and *Calendar Analytics*.

The first tools, *Google Calendar Hours*, is solely used to show a user the total allocated time of all the events which they have in their calendar, as can be seen in Figure 1. Although this is a useful statistic, we plan to provide a more in depth breakdown of a user's calendar usage as part of this project.

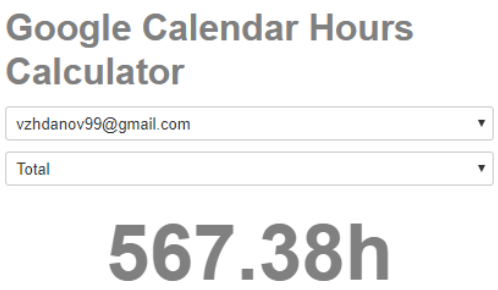

Figure 1. Example Usage of *Google Calendar Hours* Tool

The second tool, *Calendar Analytics*, focuses on breaking down where and with whom a user's time is spent, providing insights on when to most effectively schedule work-related meetings [2]. While this can be a useful tool in certain contexts, it is intended to be used in a work environment. Thus, this tool is not as useful for a large subset of users, especially those which are still in school, or do not regularly schedule and attend meetings.

We differ in our approach because we want to find out *what kind* of activities a user is partaking in, even if these activities are not labeled quite yet. We also hope to provide more useful analytics on activities broken down by category, along with a better representation of a user's calendar use over time.

Disambiguation

Short text disambiguation is prominent in search engines and in virtual assistants, trained specifically to recognize inputs in the format "Schedule meeting at 6pm with Joe" similar to those we are working with. These tasks fall in the domain of Natural Language Understanding, and established ways to solve these kinds of problems include using part-of-speech (POS) tagging. Stanford research actually provides a great POS API which we could make use of.

Feature Extraction

Feature extraction has been used to extract data from stock markets by setting two indicators to extract characteristic factors. Data is split into two groups: multi-day and intra-day, and analyzed separately.

There are also grouped time intervals (months and weeks) which are used to evaluate which group has the best performance of extracting features. With this, an experimenter uses technical indicators to analyze the data in both

different-feature-combination perspective and interval perspective. Then, they use the indicators to cluster the data into different groups [6]. For our own project, we can employ a similar method to extract certain features from our datasets.

Supervised Learning

Supervised learning has been used for classifying data in many applications. It has been used to label users' activities based on the locations where the applications were used. There were two main approaches for classifying data: visit approach and places approach. Also, there are 5 main methods to evaluate the data for both approaches: Naive Bayes, Decision Tree, Bagged Tree, Neutral Network and K-Nearest Neighbours. For classifying the data stored in phones, instead of approaching data chronically, it is categorized according locations where the events took place. [4]. While this paper pointed us to a number of supervised learning techniques we could apply to solve our problem, we will likely go with the Neural Network technique, which has vast support in tools like Tensorflow, along with a wide range of libraries which we can take advantage of. However, if we have time, it might be helpful to compare our outcomes with solutions implemented using other methods of supervised learning.

Unsupervised Learning

Unsupervised learning has been used for grouping data, for example using Hidden Markov Models and Markov Chain. People use Markov chains to predict the probability that an event could happen. A Markov Chain model primarily contains of a set of transition matrices. Each matrix contains the probability that an event will happen given that a different event has occurred [5].

While unsupervised learning has been attempted for these types of classification problems, specifically for those managing highly predictable structures as we are, the recommended approach has been using supervised learning techniques [7].

METHODOLOGY

Problem Formulation

The foundation of our identified problem comes from personal reflection on the part of our team members. We all know how it is to have an increasingly busy schedule, and how nice it is to be be conscious and aware of one's own schedule to not allow ourselves to become overbooked, double-schedule appointments and classes, or over promise that one will be available when we cannot be. When looking forward for big events, like a very-needed vacation, it is typically nice to know around what dates of the year one is very busy and will be needing that break the most.

When planning to reorganize one's daily routines, it is nice to have concrete data specifying how we spend our time, and then be able to track how our time has changed after an important lifestyle choice was made, to see our progress. Lastly, as the end of the year comes around, it is nice to reflect on what the last year has brought, who has been there the most to make it a great one, and what places and locations made the top of our list for that year. Given all of this, we saw an immediate opportunity to attain all of this information, exploiting a key data source that is used extensively by some of our group members and friends. We decided to make abstract and decentralized data concrete by performing statistical analysis of the trends, durations, and entities that make up our Google Calendar data.

Datasets

The data acquired for this project consists of several Google calendar datasets from group members and acquaintances:

- **Set 1** (from group member Andrea) provides data over 3 years of extensive usage.
- **Set 2** contains 4 years of moderate to extensive calendar usage.
- **Set 3** has data on approximately two years of usage grouped across multiple calendars (work, school, and personal calendars).

Since this is private user calendar data, there is no publicly accessible source link for this data which we have gathered.

Preprocessing

By default, the Google Calendar dataset contains the following fields:

Column	Description
SUMMARY	Calendar Event Name
DSTART	Start Date (mm/dd/yyyy hh:mm AM/PM)
DEND	End Date (mm/dd/yyyy hh:mm AM/PM)
DUE	Due Date (Optional)
NOTES	Event Notes and Recurrence Rules
ATTENDEE	Names of Event Attendees (Optional)
LOCATION	Location of Event (Optional)
PRIORITY	Event Priority (Optional)
URL	URL for Event (Optional)
CALENDAR	Calendar Specifier (if user has multiple)
UID	Calendar Component Unique Identifier

Before these datasets could be properly analyzed, it was necessary to clean and categorize the data. To clean the data, we did the following:

- Drop any rows which had NaN or missing values for the SUMMARY, DSTART, and DEND columns, since these values would be necessary for the majority of our analysis.
- Consolidate any Event Notes under the NOTES category into SUMMARY category to make keyword tagging more effective.
- Rename all the columns to be more descriptive and easier to access.

For recurring events, some extra processing was required as well. Once the Event Notes contained in the NOTES category was consolidated, all that was left in the NOTES category were recurrence rules (RRULEs) for recurring events. Since each recurring event was in the calendar data once, it was necessary to parse these RRULE strings to create the necessary duplicate events on correct intervals.

For example, one such string could be:

`FREQ=WEEKLY;UNTIL=20170118T065959Z;BYDAY=WE`

With this string, the original event would be duplicated for Wednesday every week beginning from the DTSTART value of the event until 01/18/2017. This process was repeated for every RRULE in each dataset.

In addition to this parsing, some additional fields were added to each row to be easily able to categorize and sort the data by these values:

- Activity Duration - Calculated as the difference between DTSTART and DTEND
- Time of Day - Morning (before 12:00pm), Afternoon (before 6:00pm), and Evening
- Day of the Week - An integer value between 0 and 6, where 0 is Monday and 6 is Sunday
- Holiday - Whether the event occurs during a school holiday (Winter, Spring, Summer, and Fall breaks) or not

Design

The design and implementation of all of the following algorithms was done in Python and Jupyter Notebooks. The most notable libraries used were `numpy`, `pandas`, `matplotlib`, `pyfpgrowth`, `sklearn`, and `nltk`.

Implementation

Our implementation was split into several subtasks.

Additional Information Calculation

As part of our data cleaning, it was necessary to create additional columns in our dataset to store time information such as weekday and time of day.

To determine the duration of an event, it was necessary to extract the start and end time for each event in the dataset, and convert them to `datetime` objects. Once this is done, these values are simply subtracted to determine the duration. For the majority of the data, this method works, but there are some cases where the start and end time are confusing. If, for example, an event is ongoing for an entire week, this algorithm may determine that that the duration is 7 days, which may be an overall outlier in the data. Once the duration is calculated, it is converted to an hour datatype for analysis. Since our time are already in the `datetime` format, this value is simply divided by 3600 to acquire the number of hours the activities lasted.

To determine the part of day, we compare the start time of an activity to cutoffs which we set as determining Morning, Afternoon, and Evening. Then, each activity is tagged with the corresponding period.

To determine whether an event takes place during a holiday, we needed to find specific dates for such events. We define four main holiday times: Spring break, Summer break, Thanksgiving Break, and Winter Break. These are calculated based off of the current school year calendar for CU Boulder, which gives us a rough approximation of when these holidays should be. Then, the start date is compared with these holiday date ranges to determine whether the event took place during a holiday. Although this is not an entirely robust function, since holiday times shift slightly from year to year, it gives us a baseline to determine whether an event is during a holiday or not.

To determine what day of the week an event occurs, we used Python's built in weekday function (`datetime.weekday()`), and labelled each day with the resulting value.

Finally, with all of these resulting columns, a new dataframe was generated which contains all of the new information along with any pertinent columns from the original data.

Naive Tagging

One method which was used to categorize our datasets was naive tagging. We tried three different algorithms to implement naive tagging.

Initially, we simply used Python's string searching functionality (`str.contains()`) to see if keywords were contained in each event string. If this was the case, corresponding activities were tagged with specific keywords. However, this method resulted in an excessive number of unique tags which messed with our analysis. For example, two tags which may have been created were "HW" and "Homework", which are both relating to the same overall category, which this algorithm would not have been able to detect. So, we implemented the following algorithm to synonymize these tags into broader categories.

The synonimization algorithm uses the `checktag` function update tag information. Initially, all events are tagged as "Miscellaneous". But, the `checktag` function has synonymous lists for each type of tag. For example, the tag "Travel" may be synonymous with words which are related to this subject, such as "Flight", "Ride" or "Airport". As long as we find these keywords in our data, then we are able to accurately label each activity with its corresponding correct tag. Such an aggregation method allows us to have only 6 total tags, which compared with the previous method, is much more accurate and simplified.

The last algorithm we implemented for naive tagging included searching and finding Regex strings in the data, which typically always tend to match to a certain type of activity. This was successfully able to tag data into the categories "Academic" and "Travel" with a high degree of accuracy. This method required having deep knowledge of the dataset structure, but given that this is generalizable or standardized information, they could be relied on to extend successfully to other datasets and still expect high levels of accuracy. While this method did not rely on finding keywords in the text, but rather, on dataset structure, it would not successfully find all the items that needed to be tagged under these categories, and was better used as a complimentary technique to our previous implementations.

Predictive Tagging

Another tagging option which was explored was that of predictive tagging. Although the naive tagging method worked quite reliably on the datasets we had, it would not work on every dataset without modification, since every calendar user

has a slightly different style of language. As such, the idea of predictive tagging could be a viable alternative.

Initially, this algorithm builds a word matrix. It iterates through every event in the dataset, and counts the number of occurrences of each word longer than 3 characters. Then, the algorithm filters out any of the top 100 English words, since most of these are just conjunctions such as "and", "or" and "but", and do not provide any useful information to the user.

Then, using this word matrix, the algorithm takes a greedy approach to determine each tag. It finds the word with the greatest number of occurrences for each event, and assigns this as the tag for that event. This is an effective approach, as it easily clusters events together, and will work for any dataset without alterations.

K-Fold Validation

To extract more information from the Google Calendar data, we added another column named importance level. This column uses activity's tag as a basis for the actual importance level.

- Relation:
  - {"Health", "Academic", "Travel"}⊆{"Important"}
  - {"Personal","Entertainment","Miscellaneous"}⊆{"Not Important"}

For example, we have six primary tags for all our events: ""Health", "Academic", "Travel", "Personal","Entertainment", and "Miscellaneous". If one event is tagged as "Health", the importance level would be marked as "important", but if the activity is tagged as ""Miscellaneous", the importance level would be marked as "not important".

One machine learning method that was used to predict the importance level is K-Fold Validation using the Python `sklearn` library. Since the `sklearn` API only works for float data types, we must convert the labels for "Hours", "Part of day", and "Holiday" to float numbers using the following conversions:

- {"Duration"}⊆{"Hours"} gets converted from 0 days 01 : 30 : 00 to 1.5 hours.
- {"Morning, Afternoon, Evening"}⊆{"Part of Day"} gets converted to $\{1,2,3\}$.
- {"Yes", "No"}⊆{"Holiday"} gets converted to $\{1,2\}$.

Once these features are converted, by folding the data with 10 splits, we use these features as training sets, and the actual importance level as the testing set. By comparing the predicted value with the actual value, we get the Area Under the Curve (AUROC) value of each fold as the prediction accuracy. Then, by calculating the average and standard deviation of the AUROC of each fold, we can determine the global accuracy.

To predict the accuracy of the importance level prediction, we use seven different classifiers: "Logistic Regression", "Naive Bayes", "C-Support Vector", "Decision Tree", "Random Forest", "Multi-layer Perceptron", and "Orthogonal Matching Pursuit". Based on the implementation of the code, "Decision Tree" and "Random Forest" yield the highest accuracy value and trade off on the highest standard deviation value. This can be seen in Figures 2 and 3.

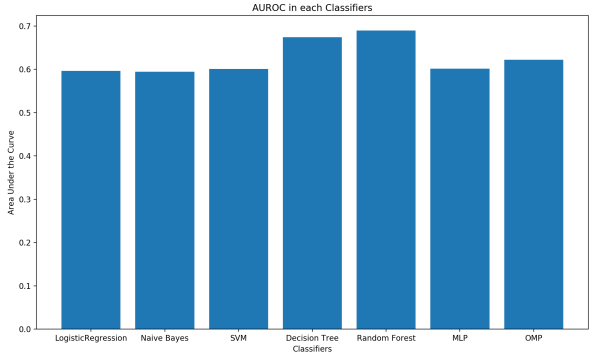

Figure 2. AUROC Average Value for each Classification Method

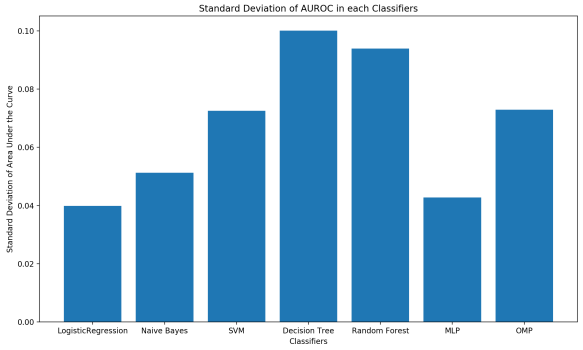

Figure 3. AUROC Standard Deviation for each Classification Method

Frequent Patterns

The FP-growth algorithm was utilized to find frequent patterns in the data with a granularity of weekday. There already exists a Python library `pyfpgrowth` which is able to apply the algorithm to numbered data. So, some data processing was required to apply this algorithm on our datasets.

For each unique event name in the dataset, a corresponding mapping to an index was created. Then, based on the day of the week specified to the algorithm, an array of event indices based on day was created. With this data structure, we were able to apply the `find_frequent_patterns` algorithm, and then revert the mapping to get a list of frequent events for each weekday.

Statistical Extraction

We performed some preliminary Clustering mechanism which used regex queries and our knowledge of how Google Calendar and people typically use Google Calendar data, to classify our activities into some of the following categories. For example, Google Calendar formats flight events in a format similar to

the following, which could be regex-searched and upon finding this type of formatting, we could successfully say we'd found an activity of type "Travel":

> Flight to Philadelphia (AA 1781),2019-09-06 17:20:00+00:00,2019-09-06 21:04:00+00:00,"American Airlines flight 1781 Denver DEN 11:20am (local time) - Philadelphia PHL 4:53pm (local time)

With this kind of formatting, it was simple to extract additional information such as flight airline taken, and keep a running list of Places visited by the user over time, which we did.

Another type of formatting that proved to be effective in classifying events of type "Academic" is looking for a tagging mechanism which is naturally used by students to delineate some classes using their class code. For example, if one encountered the formatting of type ABCD1234 such as:

> CSCI4502 - HW2,2019-09-18 15:30:00+00:00,2019-09-18 16:30:00+00:00,,

or similar, one could assume that a class was referenced and tag this event as a class, if no other method of classifying this event was available. This was effective in helping to classify some values in more than one dataset.

EVALUATION

Key Results

Textual Summarization

Using the above Regex methods, we collected information on the following categories in the dataset:

- People
- Places
- Airlines
- Classes

With this data, we were able to create textual summaries like we set out to do, much like those provided by Spotify with their 2019 Year-In-Review feature.

```
This is data for user: Andrea, file: datasets/and_dataset.ics.orage
Your total flight time ever was: 159 hours, 13.0 minutes
You were on 46 flights according to your data
Your top three most used airlines were:
1. Frontier
2. Frontier
3. Spirit
Your top three people were:
1. Chairman
2. Gustavo
3. Gustavo
Your top three places were:
1. Denver
2. Denver
3. Denver
Your top most attended classes were:
1. ECEN4593
2. CSCI4502
3. ECEN4593
```

Figure 4. Data Summary (Dataset 1)

```
This is data for user: Anna, file: datasets/anan_dataset.ics.orage
Your total flight time ever was: 14 hours, 0.0 minutes
You were on 7 flights according to your data
Your top three most used airlines were:
1. Delta
2. None
3. None
Your top three people were:
1. Aaron
2. Jenny
3. David
Your top three places were:
1. Din
2. Salt Lake City
3. None
Your top most attended classes were:
1. EXEN 3250
2. ECEN 3300
3. APPM 2350
```

Figure 5. Data Summary (Dataset 2)

Data Analysis

We generated series of histograms and bar charts based on the features we implemented. The tag frequencies histograms which tell us which tag appears the most in each data set.

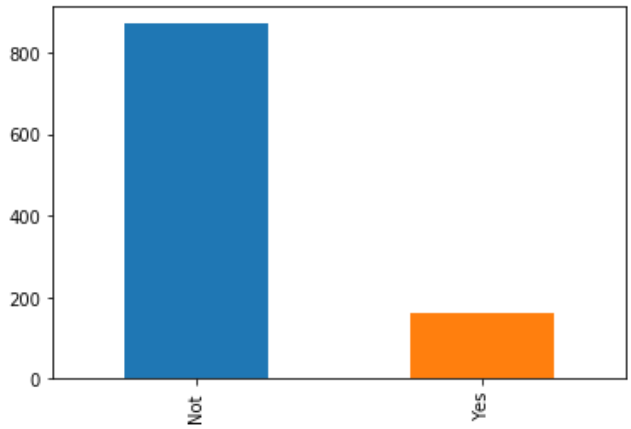

Figure 6. Holiday Productivity Metrics (Dataset 2)

As shown in Figure 6, the holiday frequencies histograms show how many events took place when the user was on holiday, and how many of them took place when the user was not in holiday. One could easily use this to compare their productivity over different periods of the year and shown at a more specific granularity, could show a user how much time more he or she can expect to take to complete an assignment, say over Thanksgiving break, as opposed to during the school week, given their pasts history. A part-of-day histogram, as seen in Figure 7 below, can be equally insightful in helping a user plan what time of day he or she is the most productive in, comparatively, and when he or she should plan to get their most important work done.

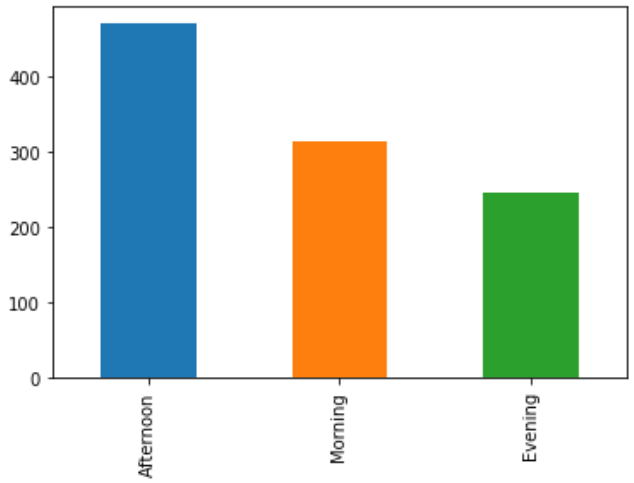

Figure 7. Time of Day Productivity Metrics (Dataset 2)

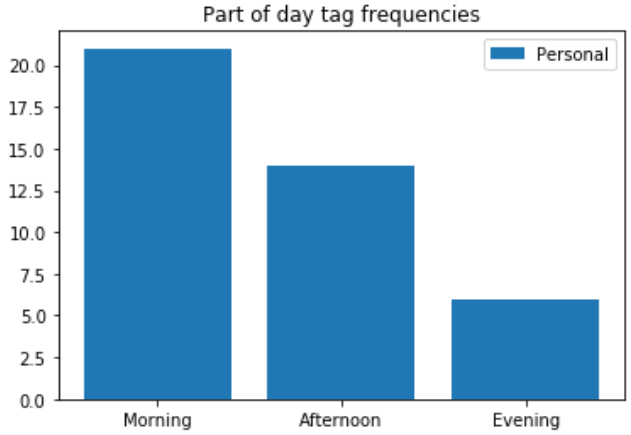

Figure 9. Personal Activity via Time-of-Day (Dataset 1)

Next, we generated tag bar charts for each tag in our data set, such as the "Academic" tag, as seen in Figure 8 below. We performed this for each user, and for each Categorical classification we grouped activities into. Bar charts like the one below associate how many activities were typically performed on each day of the week (labeled 0-6), where 0 is Monday, and 6 is Sunday.

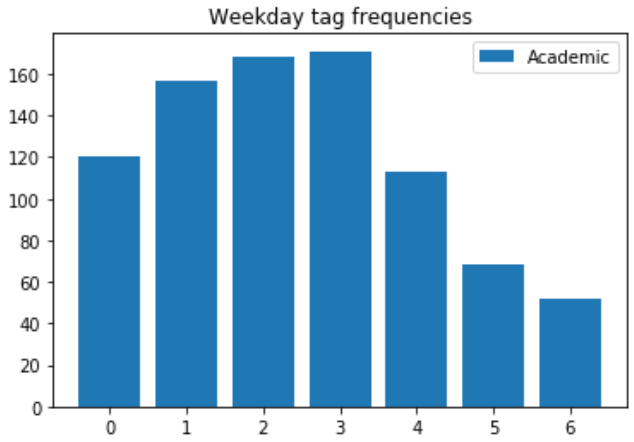

Figure 8. Weekday Academic Activity (Dataset 3)

Therefore, the user can observe how productive they typically are on a given day of the week for academic activities. It seems this user starts the week off strong, peaks at about the middle of the week on Wednesday and Thursday, and then finishes the week off at a more relaxed pace.

We generated bar graphs like those shown in Figure 9 above, which gives us insight into when a user is most likely to spend personal time. Future visualizations we would perform might specify the time frame of this data to be more insightful to the user.

One of the other utilities of tag frequency bar charts like those shown above is that they can visualize how efficient of our naive tagging methods are working. When we used synonymous word tagging, "Miscellaneous" was the most frequent tag, and this was evident in our graphs. For one data set, the frequency of "Miscellaneous" could take up to 70-80 percent of the tags that were being generated. We decided to change the algorithm for naive tagging, and upon doing so observed the percentage of "Miscellaneous" decreased and the percentage of other tags like "Academic" increase by a lot. This observation proves that our change of the naive tagging algorithm is now more effective, and that the efficiency of naive tagging was improved.

Event Heatmap

As part of our visualizations, we also generated a heatmap which showed the number of events a user had per day over the course of a year. This heatmap could be specified to show information for a specific year, or historically for all data a user had available. Figure 10 shows such a heatmap for dataset 1, both historically and for the year 2019. As we can see, there are specific dates which have significantly more events than others, and these dates appear to line up with the beginning of each academic semester.

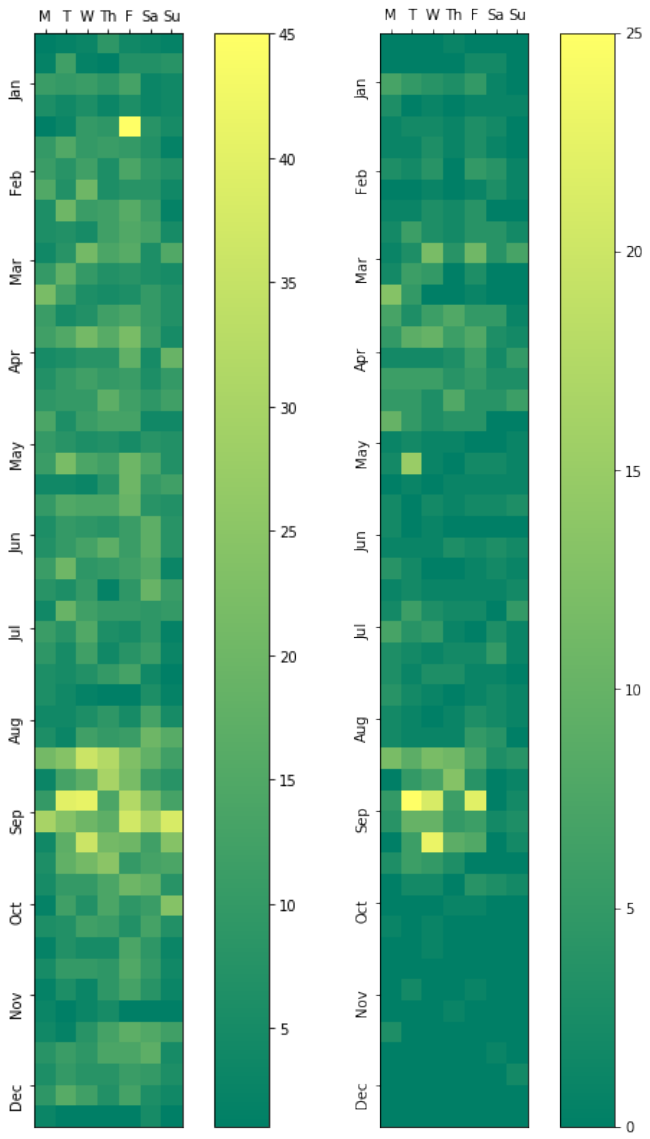

Figure 10. Historical (Left) and 2019 (Right) Heatmap for Dataset 1

Milestones

We were able to achieve the following of the milestones we set out at the beginning of the semester:

- **Milestone 1:** Acquire multiple sets of Google Calendar data, and clean the data into a uniform .csv format.
- **Milestone 2:** Add fields for "naive tags" based on exact similarities in the data. For example, grouping "CSCI 4502-Presentation" and "MIDTERM-CS4502-Presentation" as `#CSCI4502`. Also, detect simple patterns and correlations between events which tend to occur at the same time.
- **Milestone 3:** Add fields for "smart tags" based on inexact similarities. For example, grouping "CSCI4502-Presentation" and "ECEN3953-Phase3" as `#homework`.
- **Milestone 4:** Using these smart tags and category classifications, search for frequent patterns in each dataset, and analyze to see if there are certain reoccurring events. Using these patterns, determine if it is feasible to naively predict future events as a way to understand user daily patterns.
- **Final Milestone:** Create a visual display of data, including nice charts and graphics identifying the most pertinent findings and discoveries.

The milestones we were not able to achieve this time, though we attempted, were:

- **Milestone 5:** Train our neural net using set aside training data.
- **Milestone 6:** Test our neural net classification using the set aside training data ,using the fields we extracted as additional features, and extracting information classifying the data into categories. We can then provide statistical summaries of these categories including count, distribution, spatial-temporal, changes over time, and influence across the data points.

The reason we were not able to implement these milestones with the data we had available was because implementations that rely on perceptrons to classify rely on having arrays of large amounts of distinct features for each of the items in our dataset. The perceptron then weights each of the features for a given item more heavily or less heavily based on what its training data has shown it to, and repeats until the data converges to an answer. In this case, we were not able to allocate enough time to identifying distinct features on which to train the data on. We explored creating features based on parts of speech: for example, extracting verbs from an event name or notes, to make its own distinct feature, and running sentiment analysis APIs on our data to create new features. In the future, we would research how to create distinct feature arrays from our data to continue with this implementation.

DISCUSSION

Lessons Learned

Creating algorithms, and empirically testing their effectiveness with a given type of data is a skill we developed through this project. We also developed our ability to think about large quantities of data and how they might be best visualized to provide insight to a user.

We encountered challenges with our more intensive proposed implementations, including Neural Nets and implementing them using Natural Language Processing (although, notably, we found that some of the Natural Language Processing APIs we used actually implemented Neural Nets to train themselves to classify and tag words with their correct part of speech). However, when learning to prioritize our time on what mattered most to gain a lot of insight fast, we found it was typically okay to opt for simpler algorithms, which still resulted in delightful and useful results. We learned that usually, just a little bit more computation than a human is willing to make will still make for interesting data to present to a user in summary format, which libraries like the ones we utilized (like `numpy`, `pandas`, `scipy`, and `matplotlib`) do well.

Limitations

Naive Tagging Limitations

One problem that is difficult to handle is dirty data sets. For instance, each user has their own distinctive writing style, meaning they abbreviate words, use different words to describe same activity, write words which other people can't understand (such as using terms to represent things other than what they truly mean). These situations are difficult to handle using the current algorithm, since we have to come up with many specific keywords to search instead of using universal keywords. When we were creating keywords, we had to deeply examine each data set, trying to search for special keywords which are unique to each data set. For improvement, we can first search for the most frequent words which appear in data sets, relate those words to the 6 tags we have, then run naive tagging on the data sets, which is much more efficient than manually adding keywords to synonymous lists.

Although this improvement will result in us not having to manually input specific keywords, we still will need to manually distribute them into corresponding tags, which is not efficient enough when we are handling large data sets. Therefore, one of the limitations of our algorithm is an incapacity of handling large data sets.

Besides the inability of processing large data files, the current tagging algorithm doesn't consider users' writing styles and habits. These are unique for each user, so the events they write down could be totally different. For instance, one user might write down some words which we have never seen before, but that event is related to entertainment (such as a movie). We will never know this event is relevant to this specific movie, but the algorithm will recognize this event as "Miscellaneous". Therefore the algorithm will misclassify events when users write down weird words, which is another limitation.

Also, we did not handle typos and misspelled word errors as well as we could have. In the future, we might run a dictionary error-correcting API on our data during the preproccessing phase to address this problem, or allow words with a certain degree of similarity to equate to each other. In the data sets we processed, there were some incomplete values. For missing value data (like missing date/event), we dropped them. However, we must be careful with this technique, as dropping certain events could result in us unintentionally losing valuable and key data.

Lastly, we only considered a niche type of dataset, as we only had available to us datasets from students. It might be interesting to see how this would look for people with different lives and evaluate whether our classification algorithms still uphold in these scenarios.

K-Fold Validation Limitations

Because of the data samples that we have, it was difficult to convert more features to floating point numbers so that our data would efficiently work with the `sklearn` library. Therefore, with a limited number of features, the prediction value was not as accurate as we initially expected.

Another limitation is that the relation of the features chosen did not necessarily have a strong relation with the activity tags. For our current K-Fold validation, it uses "Hours", "Part of Day" and "Holiday" as features to predict the importance level (importance level is based on activities tag). For example, it is not always the case that activities with a long duration or occurring on a weekday will always be marked as an important activity.

POSSIBLE FUTURE WORK AND APPLICATIONS

There are many possibilities for future work related to mining Google Calendar data, along with applications of our current work.

An extension of frequent pattern detection is the generation of association rules related to these frequent patterns. With association rules, we can set a support and confidence threshold on seeing how often one event happening implies another will happen. This will allow us to be able to predict a user's future habits.

Possible Future Visualizations

This project can be extended by making the variables of the data more easily modified by users. We would plan to create a website displaying all the charts, graphs, and textual summaries we produced during this project, and allow users to upload their Google calendars to get their respective summaries. If temporal granularity was easily modifiable, this would make our analytics tool much more robust.For example, a user would be able to select from a drop-down menu between: Past Year, Past 3 years, Historical Data, and get the corresponding data shown to them. A user might also be able list exact dates for which they wanted their data displayed. The idea of visualizing the data this way is to allow the user to quickly consume the statistical summary we are trying to portray in a way that is engaging and informative.

Perceptrons and Neural Nets

If we are able to acquire more user Google Calendar datasets, it would be interesting to attempt to automate tagging with a supervised learning method, such as a neural network. With many datasets, it would be easier to train a neural network over a myriad of cases and language styles, and would result in a robust solution to logically tagging new events.

We would propose training a muti-class, multi-layer perceptron implementation, called a neural net, to classify our data points into our stated categories. Perceptrons, given labeled data containing sets of features, are trained to weight each feature, and use the labeled weights to calculate whether a new given input is one of several categories. The interesting thing about using neural nets like this one is that we can extract features previously unknown to us from the data, for example, it might use come up with a temporal-linguistic combination feature that we would not have thought was a good predictor to label our dataset. The reason for this is that they use features extracted through many layers (some of which are not the original features) to come up with an ultimate decision [1].

Lastly, given more time we would have focused more on temporal-related data and analytics. There seem to be many analytics related to our data which are yet unexplored by our current analysis. As mentioned above, it may be convenient to

integrate all of our tagging, analytics, and information together into one coherent display.

CONCLUSION

Google Calendar data, and other data surrounding the way people interact, spend their time, and learn new things, remains an area of continual exploration and interest for data mining. Gaining insights on a person's time utilization habits has deep impact not just on an individual, but on society as a whole. These insights help us in becoming more productive, conscientious, or intelligent individuals, whichever it is that we value the most. The key to enabling this type of discovery is to recognize the tools that we use in everyday life that are a mass storage of our personal data, and take initiative to use our own data to our advantage when we have the proper tools to analyze it. With the mass quantity of datasets available, open-sourced or otherwise like the Google Calendar ical data, there is really no question that, without the proper mining techniques, should go unanswered.

REFERENCES

[1] 2012. University of California Berkeley Intro to AI Course. (2012). http://ai.berkeley.edu/

[2] 2019. Calendar Analytics. (Oct 2019). https://www.calendar.com/analytics/

[3] Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 1375–1384. http://dl.acm.org/citation.cfm?id=2002472.2002642

[4] A. Rivero-Rodriguez, H. Leppäkoski, and R. Piché. 2014. Semantic labeling of places based on phone usage features using supervised learning. In *2014 Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS)*. 97–102. DOI: http://dx.doi.org/10.1109/UPINLBS.2014.7033715

[5] Devin Soni. 2019. Introduction to Markov Chains. (Jul 2019). https://towardsdatascience.com/introduction-to-markov-chains-50da3645a50d

[6] H. Tung, C. Cheng, Y. Chen, Y. Chen, S. Huang, and A. Chen. 2016. Binary Classification and Data Analysis for Modeling Calendar Anomalies in Financial Markets. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. 116–121. DOI: http://dx.doi.org/10.1109/CCBD.2016.032

[7] Andreas Vlachos. 2011. Evaluating Unsupervised Learning for Natural Language Processing Tasks. In *Proceedings of the First Workshop on Unsupervised Learning in NLP (EMNLP '11)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 35–42. http://dl.acm.org.colorado.idm.oclc.org/citation.cfm?id=2140458.2140463

APPENDIX

Honor Code Pledge

On my honor, as a University of Colorado Boulder student, I have neither given nor received unauthorized assistance.

Group Member Contributions

- **Andrea**
  - Project Formulation
  - Statistical Extraction
  - Textual Evaluation
- **Qiuyang**
  - Additional Information Calculation
  - Naive Tagging
  - Tag Data Visualization
- **Vladimir**
  - Recurring Event Processing
  - Predictive Tagging
  - Event Category Counting
  - Calendar Event Heatmap
  - Frequent Pattern Analysis
- **Xinyu**
  - Feature Extraction
  - K-Fold Validation
  - Classifier Analysis