Final Project Spec

Jake Seaton - jseaton@college.harvard.edu, jakeseaton13@gmail.com
Harnek Gulati - harnekgulati@college.harvard.edu gulatiharnek@gmail.com

Updated Overview

We want to design an algorithm that will parse user input, as part of a program that translates it into commands that are executable by the computer. We want to make it simpler to access documents and programs that you use frequently, without having to navigate directories or go through multiple hoops for relatively simple actions. The idea springs from potential improvements over existing programs such as spotlight and alfred that make the system more user friendly. The computationally significant portion of the project will be the natural language processing algorithm, the significance of which we will attempt to translate into increased user friendliness.

Our main focus is to implement various machine learning algorithms. The parser will take input at the command line and utilize machine learning to sort it to be carried out by one of the various modules we will be writing. This algorithm will first search for key words that identify commands as falling into the categories we define. Next, it will compare it against previous commands. Each module will also have machine learning components that assume knowledge of previous actions by the user.

Feature List

Core Features

- Natural Language Processor
- Simple Commands (Shut Down, Restart, etc.)
- Complex Commands (Opening/Closing applications etc.)
- Events (Calendar additions/subtractions)
- People (sending messages, emails, opening social media, etc.)

Cool Extensions

- Convert file types
- Netflix Watch Prediction
- Study Time (Blocks cites)
- Facebook commands
- Weather
- Stock Lookup
- Math

· UI
· Searching the Web

Technical Specification

Hierarchy Of Functions

- Parser
    - Takes user input at the command line
    - Checks for a finite list of simple commands
        -
    - Checks for a complex command by searching for key words
        - Open, Close, Print, etc.
        - Determines the command and the file/directory that the command concerns
    - Checks for a food query
    - Checks for a person
        - Checks the input against contacts
        - Determines the input to the people module (person, action)
    - Checks for an event
        - looks for key words at, before, after, pm, am, morning, afternoon, months, days, numbers
        - Determines the inputs to the event module. (Time, event, date)
    - If none of these conditions are met, the command will be considered “incomplete” and we will check it against the starts of previous commands, and execute the most recent
    - Failing this, an error will be returned
    - stores it in a data structure with previous commands

Simple Commands Module

- **Parameters: Command**
- **Start/Restart/Sleep**
- **Shutdown: Powers off the computer**
- **Restart:Restarts the computer**
- **Lock: Locks the computer**

Complex Commands Module

- **Parameters: Command and the file/application**
    - **Open:** Opens the file/application
    - **Delete:** Deletes the file/application
    - **Print:** Prints the read file.
        - **Print _ copies of:** Prints x amount of copies of a file to default printer.
    - **Search:** uses the in-computer functionality to search through it all
    - **Errors**
    - **Return Done when done**

- **Move directories**
- **Move: Takes a document or directory to be moved and a new location, and moves it. Move _ to _**
- Machine Learning Aspect: Most recently used directories, documents, and applications first.
    - Data Structure: Huffman Coding

Calendar Additions

- **Parameters:** Date, Time, Activity
    - **Add:** Add calendar to the event.
    - **Change:** Be able to recognize ("change tomorrow's dinner to 5 o'clock") and react accordingly.
    - **Done:** Outputs with confirmation with right information
    - **Warning:** Prompts for more information.
    - Machine Learning Aspect: Will determine allocation of time periods, i.e. does meal at 7 mean breakfast at 7 or dinner at 7, based on previous work.
        - Data Structure: Hash table with changing values: keys of times of day and all events with that time

People

- **Parameter:** Person
    - **Hierarchy of interactions:** Based on how you have been interacting recently with that person, predicts whether you want their contaict info, their factbook page, to send them a message, face time, etc.
    - **Ideas:**
        - **How is "Friend":** Will return the earliest status message of said friend.
        - **"Stalking"** Returns pictures of that friend from facebook.
        - **#tbt:** Returns facebook photos/status messages of friend starting from latest facebook date.
    - Machine Learning Aspect:The Hierarchy of interactions described above will determine the action, ie open email, facebook. etc.
    - Data Structure: Dictionary with list of how you interact with the person.

Food

- **Ideas**
    - **I'm Hungry:** Returns a table of nearby food restaurants of food that you have not tried recently. Ideas:
        - Reverses the unique list of food related calendar events, and returns what you got least recently.
        - assumes a data structure containing nearby restaurants and filters out recently eaten food.
    - **Date Night:** Determines the best place for a date night depending on two lists of data.
    - Machine Learning Aspect: Will determine recommendations based on constantly changing list of nearby foods.

- Data Structure: Array-list dictionary with tags.

Useful Documentation:
https://developers.google.com/google-apps/calendar/v2/developers_guide_python
https://pythonhosted.org/plex/
http://stackoverflow.com/questions/7821661/how-to-code-autocompletion-in-python
http://www.learnpython.org/
http://isites.harvard.edu/icb/icb.do?keyword=k100309 CS181!