Stephanie

Virtual Assistant At Your Service

Little About Me

I am Ujjwal Gupta.

Created Stephanie and Sounder along with few other open source libraries.

Works as a software engineer.

Website: slapbot.github.io
Github: github.com/slapbot/stephanie-va

Introduction

Why Stephanie?

- Completely Open Source.
- Minimalistic hardware requirements.
- 100% Customizable.
- Developer Friendly.
- Framework rather than application.

Outlines of this talk

1. Installation
2. Configuration
3. Usage
4. System Architecture
5. Sounder (Intent Recognition Algorithm)
6. Developer API
7. Pre-installed Modules

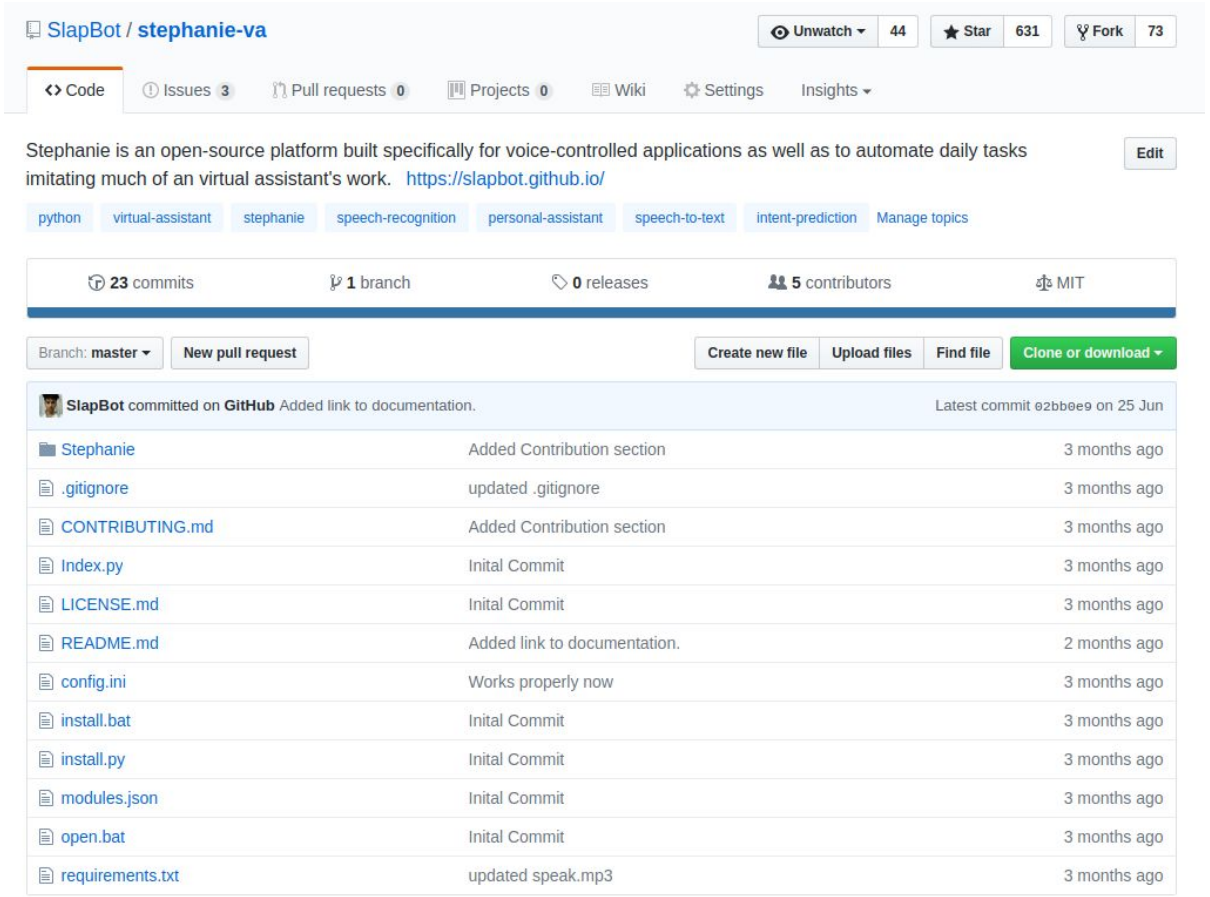
slapbot.github.io/documentation/installation

Installation

Requires no more than few clicks.

Installation

1. Go to the Github page.
2. Clone the repo.
3. Run install.py
4. ?
5. Profit.



slapbot.github.io/documentation/configuration

Configuration

Completely customizable to your needs.

Configuration

- Application
  - Version, update_check
- User
  - Name, age, gender, city, etc.
- System
  - Assistant_name, welcome_message, wake_up_engine, always_on_engine, etc.
- Modules
  - Facebook, twitter, evernote, gmail, calendar, weather, etc.
- Etc.

Usage

Controlled by your voice.

Usage

Run index.py to boot up the application.

Different ways to give commands

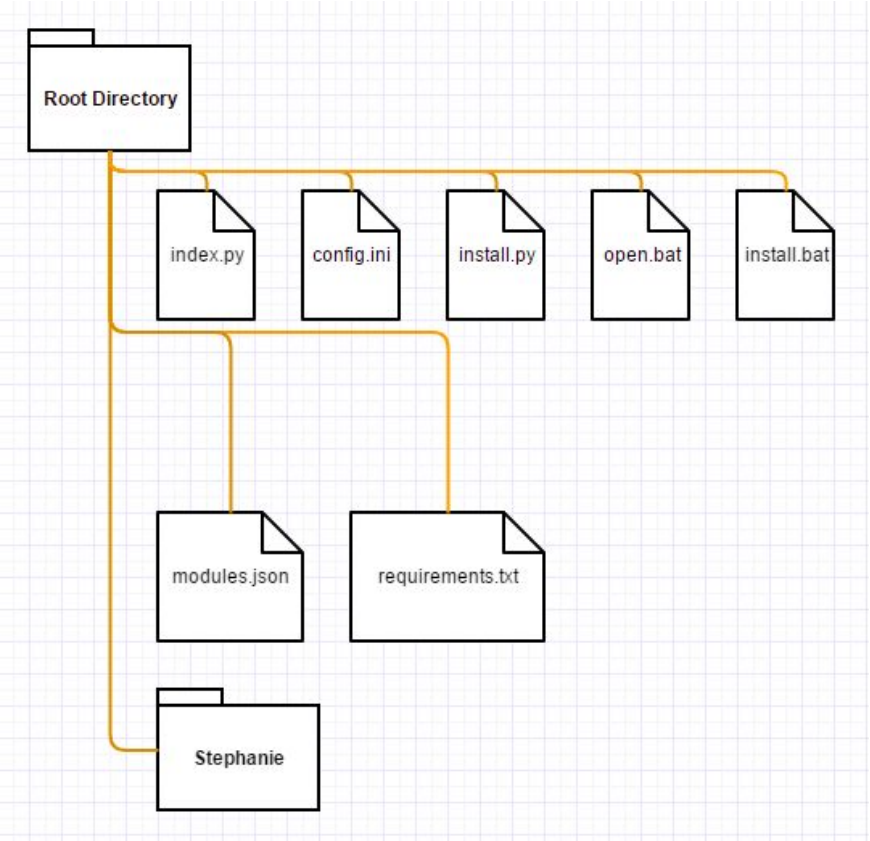
- Hey Stephanie, Do a status update on facebook.
- Stephanie, What’s trending on twitter?
- Can you note something for me?
- Yo Stephanie, do I have any calendar events pending for today?

System Architecture

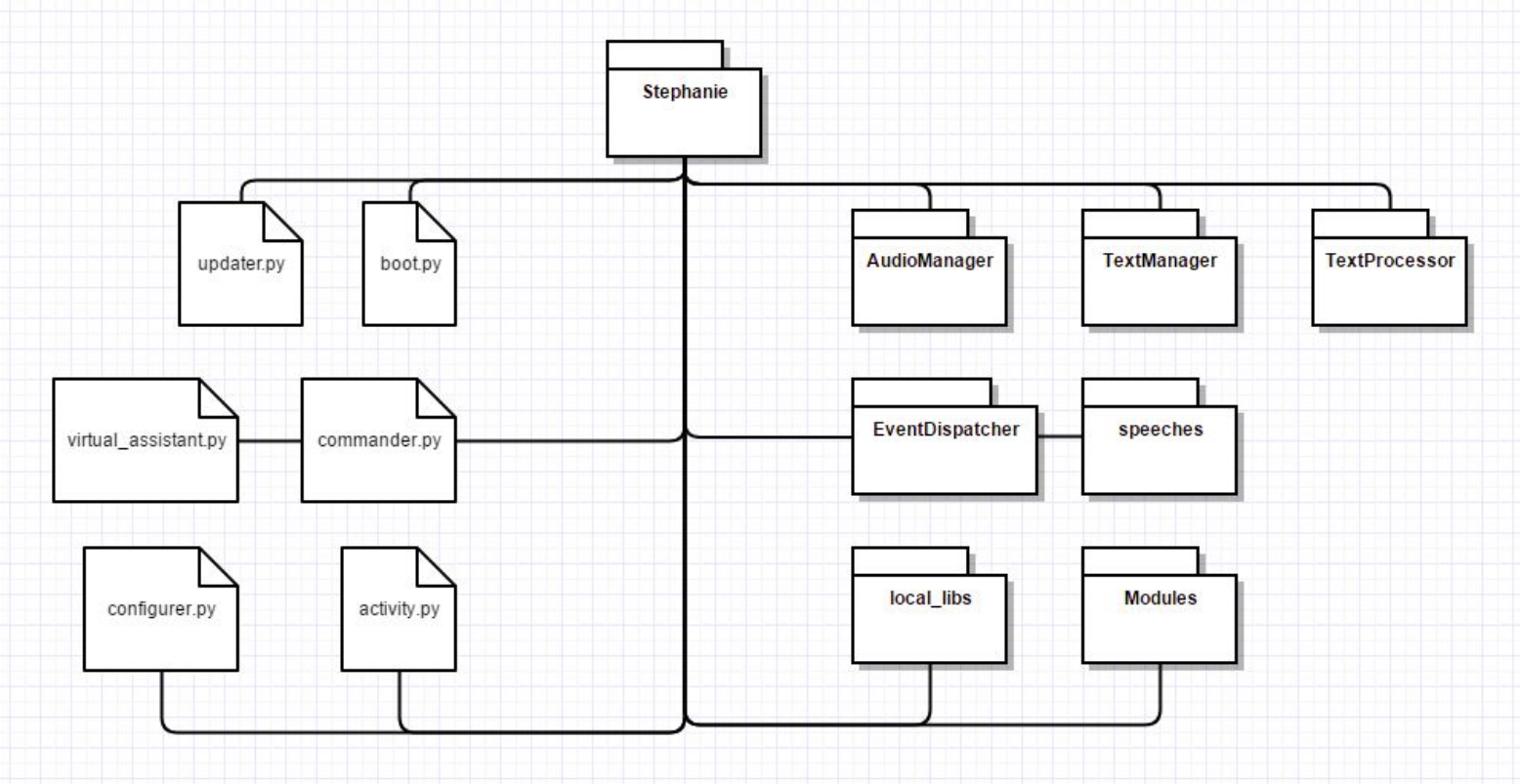
General overview of the internal API

Root Directory Structure

- **index.py** - Entry point to boot up the application.
- **config.ini** - Manages configuration settings
- **install.py** - Installer to install dependencies
- **open.bat** - wrapper to call index.py
- **install.bat** - wrapper to call install.py
- **requirements.txt** - Required dependencies
- **modules.json** - Modules Information
- **Stephanie** - Actual package which controls everything.



Stephanie Directory Structure



slapbot.github.io/documentation/resources/algorithm/sounder.pdf

Sounder

Language parsing algorithm to predict the intent

Sounder

Sounder is an intent-recognizing algorithm which uses five main principles:

1. Stopwords filtering
2. Phonetic Algorithm (Metaphone)
3. String metric (Levenstein Edit Distance)
4. Maximum weight matching (Munkres Algorithm)
5. Average likelihood (Mean + High Sum Value)

Stopwords filtering

- Stop Words are words which do not contain important significance.
- They return vast amount of unnecessary information.
- This technique is used quite often in NLP algorithms to filter out meaningful keywords from a given sentence
- For example: a, an, the, has, have, had, etc.

Phonetic Algorithm (Metaphone)

- Metaphone is used to return an approximate phonetic value.
- It is used to break a complex word into a simple one.
- It simplifies a word to a level of fuzziness to allow smart comparisons.
- it creates the same key for similar sounding words

For an example: “Stephanie” and “Stephany” both becomes “STFN”, whereas “Teffany” becomes “TFN”.

String Metric (Levenstein Edit Distance)

- Used for measuring the difference between two sequences (often strings)
- calculates the least number of edit operations required to change one string to the other
- Edit operation can be any of insertion, deletion and substitution.

"=" Match
"o" Substitution
"+" Insertion
"-" Deletion

It basically returns a value between 0.0 to 1.0 where 1.0 means exact match.

Maximum Weight Matching (Munkres Algorithm)

- Also known as Hungarian Algorithm solves the assignment problem
- Combinatorial optimization algorithm to compute maximum output

Name	Maths	Physics	Chemistry
Karan	75	55	89
Aarush	69	42	80
Renu	80	73	85

Allotting subjects such that overall marks are the highest can be computed using it: karan => 89, Aarush => 69, Renu => 73. Total => 231

Average Likelihood (Mean + High Sum Value)

- Computing the mean of each of the record present in dataset
- Choosing the maximum mean value
- In case of equal averages, choose the one with higher sum value

Putting It All Together

- Sentence is first broken down into keywords by filtering out subwords
- It is then converted into metaphone to ensure robustness (if needed)
- Each word is searched against dataset to compute edit distance
- Above matrix is then used to calculate the maximum cost using munkres algorithm
- Each cost is then averaged across the dataset.
- Maximum averaged value is chosen as the final result.

Rough Example

>> “What is the date today?”

>> [‘date’, ‘today’]

>> compared against dataset of [[‘date’, ‘today’], [‘twitter’, ‘notifications’], ...]

>> received maximum cost of [[100.0, 100.0], [36.0, 22.0], ...]

>> compute the averages [100.0, 29.0, ...]

>> Pick the highest value.

slapbot.github.io/documentation/developer

Developer API

Framework rather than an application

Developer API

- Built with a clear thought of providing a framework than being an application.
- Offers a rich well documented API to extend the functionality
- Easy to understand and get started with
- Can handle complex business logic
- Scales nicely to one’s need

Demo (Test Module)

```
from Stephanie.Modules.base_module import BaseModule


class TestModule(BaseModule):
    def __init__(self, *args):
        super(TestModule, self).__init__(*args)
```

```
def ask_favorite_food(self):
    self.assistant.say(“What is your favorite food?”)
    user_command = self.assistant.listen().decipher()
    response = “Oh really? My favorite food is ” + user_command + “ too!”
    self.assistant.say(response)
```

Now simply add the instructions at modules.json file:

```
[
            .
            .
            [..],
            [..],
            ["FacebookModule@GetNotifications",["facebook", "notifications"]],
            ["ZomatoModule@handle",["feeling", "hungry"]],
            ["TestModule@AskFavoriteFood",["ask", "favorite", "food"]]
]

["ClassName@FunctionName", ["keywords", "trigger", "module"]]
```

And that’s it

You successfully created your first dummy module.

Simply restart the Stephanie, and use any of the following commands like:

- Hey Stephanie, ask my favorite food.
- Do you wanna know my favorite food?
- Ask which kind of food is my favorite.

API reference

- Properties
  - raw_text, sub_words, key_words, key_words_assigned, etc.
- Assistant Object
  - say(text), listen(), decipher(), set_modules(sub_dataset), learn(raw_text), understand(), etc.
- Events Object
  - add(event_name), trigger(event_name), etc.
- Config Object
  - get_configuration(key, section=”modules”)
- etc.

Advanced Usage

```
from Stephanie.Modules.base_module import BaseModule


        class ChatModule(BaseModule):
                modules = [
                            "ChatModule@BadDay", ["bad", "day"],
                            "ChatModule@GoodDay", ["good", "day"],
                            "ChatModule@FineDay", ["fine", "day"]
                          ]

                def __init__(self, *args):
                        super().__init__(*args)
                        name = self.get_configuration('name', section='USER')

                def handle(self):
                    self.assistant.say("Hey %s, How was your day?" % (self.name))
                    user_command = self.assistant.listen().decipher()
                    module_function_that_is_guess = self.assistant.understand(self.modules, text)
                    func = gettattr(self, module_function_that_is_guess)
                    func()

                def good_day(self):
                    self.assistant.say("Oh wow that was awesome!" + self.name)

                def bad_day(self):
                    self.assistant.say("Don't worry ," + self.name + " tomorrow will be better!").

                def fine_day(self):
                    self.assistant.say("Oh okay, let's make it better.")
```

Stephanie is a framework designed to create your own virtual assistants

Modules

Builtin modules shipped out of the box

Modules

- Questions (Wolfarama Alpha)
- News (Newsapi.org)
- Weather (Open Weather)
- Notes (Evernote)
- Emails (Gmail)
- Twitter (twitter.com)
- Facebook (facebook.com)
- Calendar (Google Calendar)

Few more of them...

- Football (football-data.org)
- Restaurants (Zomato)
- Wikipedia (Wikipedia Foundation)
- Movies (Omdb)
- System (OS related)

600+

Stephanie has received an immense support in the social media such as github, hackernews, reddit, etc.

Thanks!

Contact me:

Ujjwal Gupta

Website: slapbot.github.io
Github: github.com/slapbot
Email: ugupta41@gmail.com

Questions?