Abot: An Open Source Tool to Create your own Digital Assistant!

Virtual assistants are huge pieces of complex software. Abot makes it easy and fun to create one, since it comes with everything you need to get started.

If you've always wished to create your own digital assistant that talks back to you and completes everyday tasks, it's easy with the Abot open source tool that's written in the Go programming language.

A digital assistant (also called a virtual assistant) is an application program that can understand human language and complete electronic tasks for the user. Such tasks include taking dictation, reading text or email messages aloud, looking up phone numbers, anticipating requests, placing calls and reminding the user about appointments.

Today's digital assistants like Apple's Siri, Google Now and Cortana, are programmed with artificial intelligence, voice recognition and machine learning technologies. As users interact with their digital assistants, the artificial intelligence programming uses sophisticated algorithms to learn from data inputs and becomes better at predicting the users' needs. Tomorrow's digital assistants will be built with more advanced cognitive computing technologies, enabling them to understand and carry out multi-step requests and perform more complex tasks.

You must be aware that digital assistants are huge complex pieces of software. Abot comes with pre-installed tools that manage and understand human language.

An introduction to the Go programming language

According to Wikipedia, Go is an open source programming language created at Google by Robert Griesemer, Rob Pike and Ken Thompson. Though the development of Go began in 2007, it was publicly released in November 2009.

Go is basically a general-purpose programming language with advanced features and a clean syntax (a C-like syntax with no semicolons). Because of its wide availability on a variety of platforms, its well-documented library, and its focus on good software engineering principles, Go is an ideal language to learn. It is unlike Java or Ruby. Its creators were not trying to make it a language theorist's dream. Go is not about type hierarchies. It is not about object-oriented programming or functional programming.

The process we use to write software with Go is straightforward:

- Gather requirements
- Find a solution
- Write source code to implement the solution
- Compile the source code into an executable
- Run and test the program to make sure it works

This process is iterative and the steps usually overlap.

Figure 1: Go's mascot: Gopher

What is Abot?

Abot is a digital assistant framework written in the Go programming language. Abot's framework is designed to help users in building their own assistant and customising it according to their needs—whether it's a computer that answers phones, intelligently routes calls and schedules your business travel, or just a better take on Siri that orders Ubers and does your laundry.

Abot exposes a simple HTTP API, so you can easily connect it to send, process and respond to texts, emails and more.

How the framework of Abot works

The brain of the Abot tool consists of three parts:

- An API (application program interface) that accepts natural language (human language) input.
- A state machine for tracking grammar and context across inputs to enable chaining of commands.
- A router to select the correct plugin that sends inputs based on the present commands and past context.

Abot combines these three parts and libraries to hasten the process of coding one's personal digital assistant.

Abot processes, routes and responds to every message that it receives. Actually deciding what to say is the role of plugins. Let's take a look at an example:

1. A user sends the following message via the console, SMS, email, etc: 'Show me Chinese restaurants nearby.'
2. Abot pre-processes the message, breaking it into key components:
   a. Commands: [Show]
   b. Objects: [me, Chinese restaurants nearby]
3. Abot routes this message to the respective plugin:
   a. Route: [find_chinese]
   b. Plugin: [restaurant]
4. The plugin then generates a response: 'Sure, how does Sun Shi sound? It's nearby.'
5. Abot sends the response to the user.

Abot sends the response through the same channel the user chose, so if you send Abot a text, it will respond in text. Thus, every message goes from *User --> Abot --> Plugin --> Abot --> User.*

How to build your own personal assistant

As mentioned earlier, Abot is a digital assistant framework that is based on the Go programming language. It requires some prerequisites like Go (version 1.6 or later) and PostgreSQL (version >= 9.5). Once you have ensured that you have these dependencies, we can move forward to running the Abot server on your terminal.

1. Fetch Abot using *go get:*

```
go get github.com/itsabot/abot
cd $GOPATH/src/github.com/itsabot/abot
go get github.com/robfig/glock
```

2. Set the Glock binary path:

```
export PATH=$PATH:/home/ubuntu/workspace/bin/
```

3. Run the setup script, by passing it in your Postgres credentials/host, if needed:

```
cd $GOPATH/src/github.com/itsabot/abot
cmd/setup.sh [username[:password]@host[:port]]
```

If you don't pass anything to the script, Postgres will, by default, set the host = 127.0.0.1, port = 5432 and username = postgres.

4. Once the script is completed, run the following command:

```
$ abot server
```

And then visit Abot at *localhost:4200*.

Once you have everything installed and running, you can start communicating with Abot. To do so locally, use the command *$ abot* at the console. But before that, you'll need to sign up. To do so, go to *http://localhost:4200* and click on 'Sign up' in the header. Fill in the appropriate information, and remember the phone number you use (which should be in the format of +918889992221 — no spaces, and with a leading +91).

Once you've created an account, type the following code:

```
$ abot console +918889992221
> Hi
Hi there. :)
```

Replace the number with the one you used to sign up. Abot won't do much at this point; we will need to add plugins to add more functionalities.

Installing plugins

When Abot boots, it starts every plugin listed in your *plugins.json* file.

Open up *plugins.json* in your text editor of choice, and add to the *Dependencies*, so it looks like what's shown below:

```
{
    "Dependencies": {
        "github.com/itsabot/plugin_onboard": "*",
        "github.com/itsabot/plugin_weather": "*"
    }
}
```

Then, in a terminal, run the following:

```
$ abot plugin install

Fetching 2 plugins...
Fetching dependencies...
Installing plugins...
Success!

$ abot server
```

And from another terminal, run the following:

```
$ abot c +918889992221

> What's the weather like in Mumbai?
It's 74 and sunny in Mumbai.
```

Creating your first Abot plugin

Abot provides all the tools you need to quickly and easily write these plugins through its shared library. To do so, you should know the Go programming language.

Let's take a look at a 'Hello World' plugin, which will introduce you to the plugin API. Let's download the plugin by adding it to our *plugins.json* file and installing it as follows:

```
$ tail $GOPATH/src/github.com/itsabot/abot/plugins.json
{
    ...
    "Dependencies": {
        "github.com/itsabot/plugin_onboard": "*",
        "github.com/itsabot/plugin_hello": "*"
    }
}

$ abot plugin install
Fetching 2 plugins...
Installing plugins...
Success!
```

Now, let's take a look at the contents of the plugin. Pay close attention to the inline comments, which explain each piece of the plugin API as it is introduced.

```
// Package hello responds to "Say something" with "Hello
World".

package hello

import (
    "log"

    "github.com/itsabot/abot/shared/datatypes"
    "github.com/itsabot/abot/shared/nlp"
    "github.com/itsabot/abot/shared/plugin"
)

var p *dt.Plugin
var sm *dt.StateMachine

func init() {
    // When Abot receives a message, it'll route the message
to the correct
    // package. Doing that requires a trigger, which tells
Abot to send the
    // response to this package when Commands include "say"
and Objects
    // include "something", "hello", etc. Case should always
be lowercase,
    // and the words will be stemmed automatically, so
there's no need to
    // include variations like "cat" and "cats".
    trigger := &nlp.StructuredInput{
        Commands: []string{"say"},
        Objects:  []string{"something", "hello", "hi"},
    }

    // Tell Abot how we'll respond to first messages and
follow up messages
    // from a user in a conversation.
    fns := &dt.PluginFns{Run: Run, FollowUp: FollowUp}

    // Create the package, setting it up to communicate with
Abot through
    // the functions we specified.
    var err error
    p, err = plugin.New("github.com/itsabot/plugin_hello",
trigger, fns)
    if err != nil {
        log.Fatalln("building", err)
    }

    // Abot includes a state machine designed to have
conversations. This
```

```
    // is the simplest possible example, but we'll cover more
advanced
    // cases with branching conversations, conditional next
states, memory,
    // jumps and more in other guides.
    //
    // For more information on state machines in general, see:
    // https://en.wikipedia.org/wiki/Finite-state_machine
    sm = dt.NewStateMachine(p)
    sm.SetStates(
        []dt.State{
            {
                OnEntry: func(in *dt.Msg) string {
                    return "Hello world!"
                },
                OnInput: func(in *dt.Msg) {
                },
                Complete: func(in *dt.Msg) (bool, string) {
                    return true, ""
                },
            },
        },
    )
}

// Run is called when the user first enters the package in a
conversation. Here
// we simply reset the state of the state machine and pass it
on to FollowUp().
func Run(in *dt.Msg) (string, error) {
    sm.Reset(in)
    return FollowUp(in, resp)
}

// FollowUp is called as the user continues to message the
package after the
// user's first message. Here we tell the state machine to
move to the next
// available state and return the response from that move--
since there's only
// one state, it will always be "Hello world!" from the
OnEntry function.
func FollowUp(in *dt.Msg) (string, error) {
    return sm.Next(in), nil
}
```

You can find and edit this file at *$GOPATH/src/github.com/itsabot/plugin_hello*. You should see Abot boot with a line or two mentioning our new plugin, 'Hello World!'. Open another terminal while the Abot server is still running, and type (using the same phone number you used to sign up at *http://localhost:4200*):

```
$ abot console +918889992221
> Say hi
Hello World!
```

Abot just routed your message to the plugin based on the trigger defined in our *abot_hello.go*. The state machine told it to respond with 'Hello World!' when it entered its first state, and since there are no other states, this state is replayed every time a new message matching our trigger is sent to Abot.

Deploying Abot

To make Abot available to the outside world, we'll have to deploy it to a server. We will be using Heroku for now. Go and Abot do make it easy to work on any Web server, though. Open a terminal and run the following code:

```
$ echo 'web: abot server' > Procfile
$ heroku create --buildpack https://github.com/itsabot/
heroku-buildpack-go
$ heroku addons:create heroku-postgresql:hobby-dev --version
9.5
$ cat db/migrations/up/*.sql | heroku pg:psql
$ cat data/cities.csv | heroku pg:psql -c "COPY cities(name,
countrycode) FROM stdin DELIMITER ',' CSV;"
$ heroku config:add ABOT_ENV=production \
ABOT_URL="https://YOURURL" \
ABOT_SECRET=$(< /dev/urandom tr -dc _A-Z-a-z-0-9 | head
-c${1:-64};echo;)
$ git push heroku master
$ heroku open
```

Be sure to replace YOURURL above with a user-presentable URL. If everything booted correctly, you'll see the 'Congratulations, you're running Abot!' screen. If not, you can track down any issues with *heroku logs --tail*.

Once you see the 'Congratulations' screen, try communicating with your Abot. Sign up for an account using the Web interface, then use the following commands:

```
$ abot console https://yourapp.herokuapp.com +yourphone
> Hi
Hello there!
```

Replace *yourapp* and *yourphone* with the appropriate values, and you should see Abot respond to your message!

Abot is a new framework, and more versions will be released soon. One can also contribute to the project. END

References

[1] *https://github.com/itsabot/abot*
[2] *https://en.wikipedia.org/wiki/Go_(programming_language)*
[3] *https://github.com/itsabot/abot/wiki/Roadmap*

By: Srijan Agarwal

The author is an open source fan and a Web developer, by default. You can contact him at *srijanagarwal.cse@gmail.com*.