Starting With Chatbots

There are three steps one should follow before building chatbots. We'll discuss each one of them briefly here.

1. Think about all the scenarios or tasks you want your chatbot to be able to do, and gather all related questions in different forms that can be asked to do those tasks. Every task that you want your chatbot to do will define an **intent**.
2. Each question that you list or intent can be represented in multiple ways. It depends on how the user expresses it.

   For example: Alexa, Switch off the light. Alexa, Would you please switch off the light? Can you please switch off the light? A user may use any of these sentences to instruct the bot to switch off the light. All of these have the same intent/task to switch off the light, but they are being asked in different **utterances/variances**.
3. Write all your logic to keep the user tied to the flow that you have chosen after you recognize the user's intent.

   For example, suppose you are building a bot to book a doctor's appointment. Then you ask your user to give a phone number, name, and specialist, and then you show the slots and then book it.

In this case you can expect the user to know these details and not try to accommodate all the things in the bot itself, like a specialist for an ear problem is called an ENT. However, doing this is not a big deal. So, again it comes back to deciding the scope of your bot, depending on the time and resource you have to build the application.

Decision Trees in Chatbots

If you know about decision trees, then that's very good because you will be needing that knowledge frequently when designing the flow of your chatbots. But if you don't know about the decision trees, then just Googling would help you learn this simple concept widely used in Computer Science.

Using Decision Trees in Chatbots

In the context of chatbots, a decision tree simply assists us in finding the exact answer to a user's question.

> *A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.*
>
> —Wikipedia

The most difficult part when building a chatbot is to keep track of if...else code blocks. The greater the number of decisions to make, the more frequently if...else comes up in the code. But at the same time these blocks are required to encode the complex conversational flows. If the problem is complex and requires a lot of if...else in real-life, then that will require code to adjust in the same way.

How Does a Decision Tree Help?

Decision trees are simple to write and understand, but they are a powerful representation of the solution made for the problem in question. They inherit a unique capability to help us understand a lot of things.

- Help in creating a full picture of the problem at hand. Looking at the decision tree, we can easily understand what's missing or what needs to be modified.
- Helps debug faster. Decision trees are like a short bible or, say, a visual representation of a software requirement specification document, which can be referred by developers, product managers, or leadership to explain the expected behavior or make any changes if needed.
- AI is still not at that stage that it can be trained with lots of data and perform with 100 percent accuracy. It still requires a lot of hand-holding by writing business logic and rules. Decision trees help wherever it becomes a little tough to ask a machine to learn and do it.

Let's take a simple example and try to understand how it helps in building chatbots. Look at the example diagram for a chatbot that starts with a question of whether the user is looking for a t-shirt or jeans, and based on the input the diagram flow goes further to give options related to the product by asking more questions. You don't need to create a full-fledged decision tree, but you should definitely have a flow of questions defined at every step before starting to build chatbots.

Suppose you were building a similar chatbot that helps people buy apparel online. The first thing you would do is to make a similar decision tree or a flowchart to help your chatbot ask appropriate questions at the right time. This is really needed to set the scope of each step and what needs to be done at that stage. You will need the state diagrams or a simple flowchart later when you actually code your first chatbot. Remember to not be too stringent while creating a diagram like Figure 1 ; keep it as simple as possible and then add the extended functionalities later. The benefit of such a process is the development time will be cut down, and later on the functionality will be loosely coupled and would start making sense as components. Like in the example, after creating the basic functionality, you can add color choices, price range, ratings, and discount options as well.

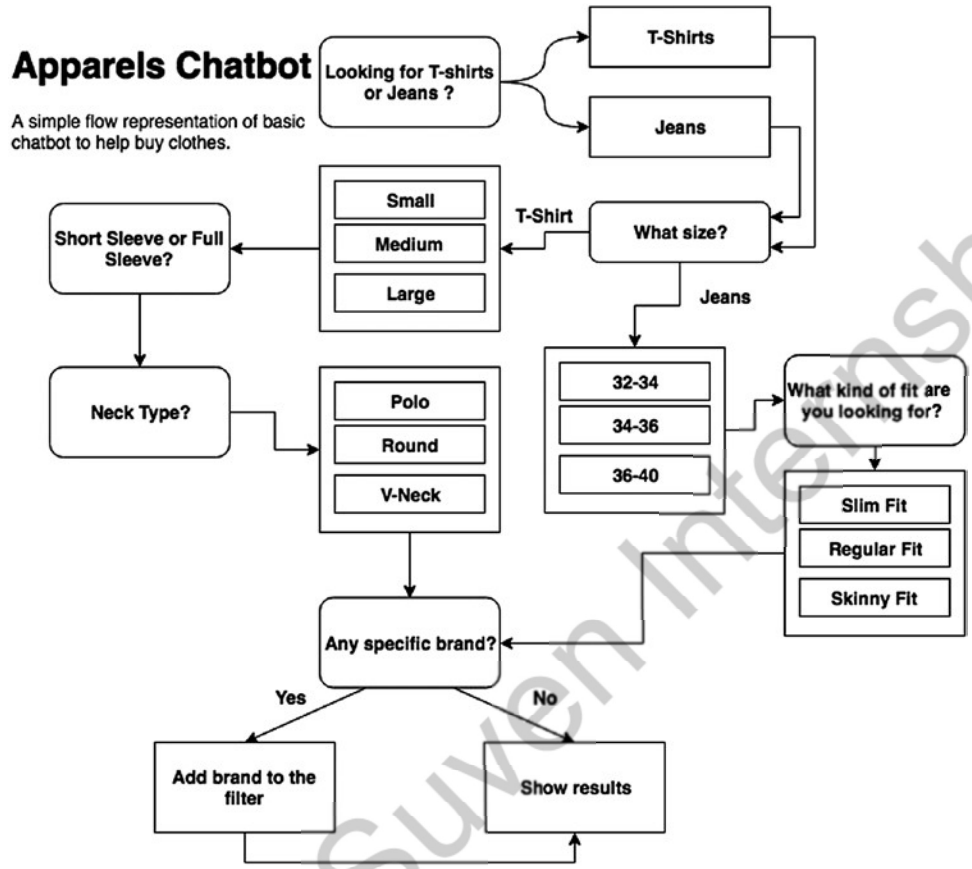

Figure 1. *A simple representation of an apparel chatbot for buying clothes online*

There are definitely more things you can add to the earlier use-case depending upon your requirements. But you have to make sure that you don't make it too complex for yourself as well as for the user.

A decision tree not only helps you to keep the user tied to the flow but also is a very effective way to identify the next intent that might be coming in the form of a question from the customer.

So, your bot will ask a series of questions following the decision tree that you have built. Each node narrows down on the customer's goal through chatbot intents.

Suppose you were creating a chatbot for a financial institution—say, a bank—that can do a money transfer based on your request after authentication. In this case, your bot may first want to verify the account details and ask the user to confirm the amount, and then the bot may ask to validate target account name, account number, account type, etc.

You cannot or would not want to invoke an OTP (one-time password) API unless you have validated if the user's account balance is more than the requested amount.

It happens with all of us and with customers as well. They get frustrated when their questions are not answered correctly. Using decision trees for your chatbot will definitely make the experience better for your users than it would be if your didn't use them.

Lots of times you will find issues solving some intents programmatically. So, the bottom line is, "*If you can't solve something programmatically then solve it by design.*"

Look at Figure 2 where the bot is trying to take a health quiz and wants to know if antibiotics can work for everything.

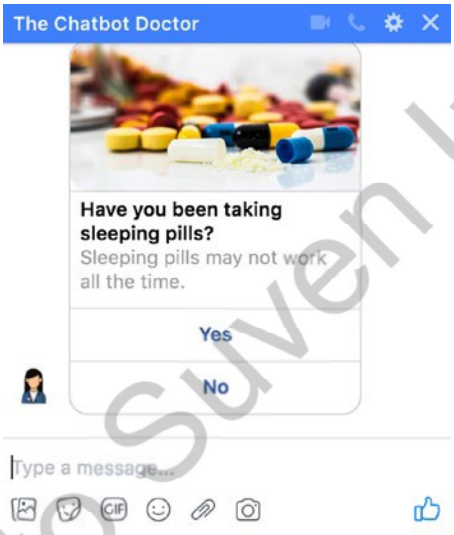

Figure 2. *Example of solving a use-case by design*

Since the answer is expected to be a Boolean (True/False), you give just two buttons for the user to click instead of letting them type and wait to fix their mistake.

This is solving by design rather than writing lots of code that will be handling unexpected user inputs. You will have so many scenarios while building the chatbots where by just giving buttons, you will be able to quickly know the intent of the user. It's important to understand such scenarios and provide buttons both for your own convenience as well as for users who don't need to type in obvious cases of optional answers.

The Best Chatbots/Bot Frameworks

- https://woebot.io/
  - Can track your mood
  - Helps you feel better
  - Gives you insights by seeing your mood pattern
  - Teaches you how to be positive and high-energy
- https://qnamaker.ai/
  - Build, train, and publish a simple question-and-answer bot based on FAQ, URLs, and structured documents in minutes.
  - Test and refine responses using a familiar chat interface.
- https://dialogflow.com/
  - Formerly known as api.ai and widely popular among chatbot enthusiasts.
  - Give users new ways to interact with your product by building engaging voice-and text-based conversational interfaces powered by AI.
  - Connect with users on the Google Assistant, Amazon Alexa, Facebook Messenger, and other popular platforms and devices.
  - Analyzes and understands the user's intent to help you respond in the most useful way.
- https://core.rasa.ai
  - A framework for building conversational software
  - You can implement the actions your bot can take in Python code.
  - Rather than a bunch of if…else statements, the logic of your bot is based on a probabilistic model trained on example conversations.

- https://wit.ai
  - Wit.ai makes it easy for developers to build applications and devices that you can talk or text to.
  - Acquired by Facebook within 21 months of its launch, wit.ai team contributes toward Facebook's own NLP engine inside Facebook.
  - You can use wit.ai for building chatbots, home automation, etc.
  - Wit.ai is similar to the way Dialogflow works but is not as feature-rich as Dialogflow. People initially used wit.ai, as it was free and Dialogflow was not, but later on Dialogflow became free as well.
- https://www.luis.ai/
  - A machine learning-based service to build natural language into apps, bots, and IoT devices.
  - Quickly create enterprise-ready, custom models that continuously improve.
- http://botkit.ai
  - Visual conversation builder
  - Built-in stats and metrics
  - Can be easily integrated with Facebook, Microsoft, IBM Watson, Slack, Telegram, etc.

Components of a Chatbot and Terminologies Used

Components of a chatbot system are very few. In this section we'll be briefly discussing the components of a chatbot that **you will come across in the internship project docs**.

Having a basic theoretical understanding of any system before diving deep is always helpful. You should have a fair idea after this section about technical terminologies used while building chatbots using Python. These terminologies will be used frequently in the internship project docs when we actually start building our chatbots.

Intent

When a user interacts with a chatbot, what is his intention to use the chatbot/what is he asking for?

For example, when a user says, "Book a movie ticket," to a chatbot, we as humans can understand that the user wants to book a movie ticket. This is intent for a bot. It could be named "*book_movie*" intent.

Another example could be when a user says, "I want to order food," or "Can you help me order food?" These could be named "*order_food*" intent. Likewise, you can define as many intents as you want.

Entities

Intents have metadata about the intent called "**Entities.**" In the example, "Book a movie ticket," booking a ticket could be an intent and the entity is "**movie,**" which could have been something else as well, like flight, concert, etc.

You can have general entities labeled for use throughout the intents. Entities could represent as a quantity, count, or volume. Intents can have multiple entities as well.

For example: Order me a shoe of size 8.
There could be two entities here:
Category: Shoe
Size: 8

Utterances

Utterances are nothing but different forms of the same question/intent your user may show.

- Remember we discussed the switching off the light intent? That was an example of how a user can use different utterances for the same intent.
- It is suggested to have an optimum 10 utterances per intent and a minimum of 5, but this is not restricted.

Training the Bot

Training essentially means to build a model that will learn from the existing set of defined intents/entities and utterances on how to categorize the new utterances and provide a confidence score along with it.

When we train the system using utterances, this is called **supervised learning**. We will soon be learning more about doing this practically.

Confidence Score

Every time you try to find what intent an utterance may belong to, your model will come up with a confidence score. This score tells you how confident your machine learning model is about recognizing the intent of the user.

That's all we wanted to cover in the Pre-read matter to **Building Chatbots with Python using NLP and ML Coding Internship.**

We learned about the different chatbot frameworks and also got to know about the terminology used for chatbots by example. We'll be using them in the **two internship project specs**. You should be at a stage now where you know what kind of chatbot you want to build and how it would behave when built.

Our **team@Suven** wishes all the best for this **2 weeks of learning and fun-filled Internship**.