Introduction to Chatbot with Rasa
Yogesh Kulkarni

Introduction to Chatbots

Introduction

Calling the Call Center

- Calling to an IVR (Integrated Voice Response)
- A pre-recorded menu selection.
- "Please press 1 for Account Details, Please press 2 for . . . "
- Till it comes to your option.
- Else, you are given access to a person to talk to.

Boring? Annoying? But still heavily used . . . , Why?

(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

Instead, how about typing/saying your query directly and getting the answer right away?

Solution

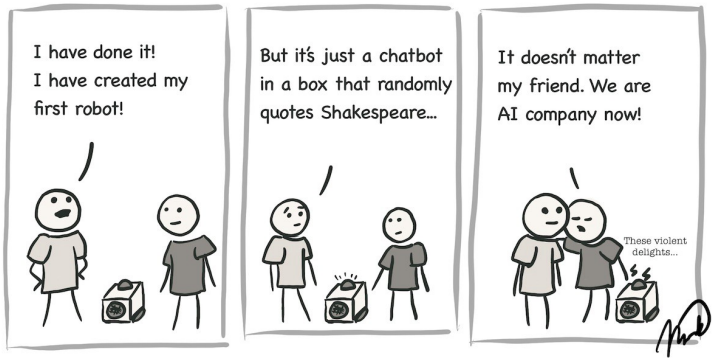
Chatbots!!!

- Which problem of IVR it is solving?
- Advantages?
- Disadvantages?
- Gaining popularity . . .
- Many platforms
- Any local chatbot companies/platforms?

Crystal Ball

"The global chatbot market is expected to reach $1.23 billion by 2025" - Business Insider

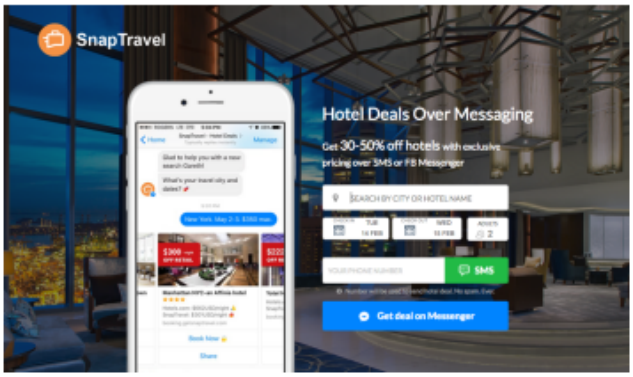
Chatbot == AI



(Ref: How to build awesome Rasa chatbot for a web - Martin Novak)

Sample Application

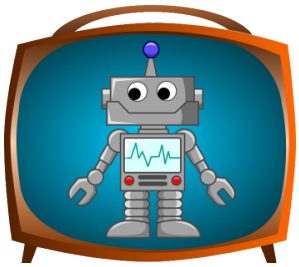
SnapTravel has processed $1 million in hotel bookings inside Messenger.



(Ref: Innovation in Health - Ritesh Ptael, et al)

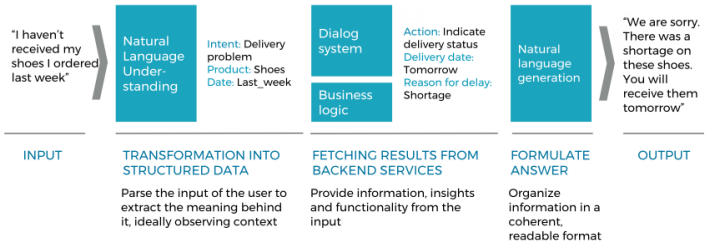
So, What is a Chatbot?

"Chatbots are a form of human-computer dialog system which operates through natural language via text or speech"- Deryugina, 2010; Sansonnet et al., 2006.



(Ref: Rasa - mdd01 course on github)

Anatomy of a Chatbot



(Ref: Chatbots and AI - botfuel)

Why so many chatbot startups?

- VCs appear excited with this new tool, more services, more opportunities, new battlegrounds for the big players (likely leading to acquisitions).
- So even without real technological breakthroughs, there is at least some money to be made investing in bot startups.
- But the real issue is : Truly 'conversational' software is a difficult problem to solve.

(Ref: We don't know how to build conversational software yet (Alan Nichol Apr 2016))

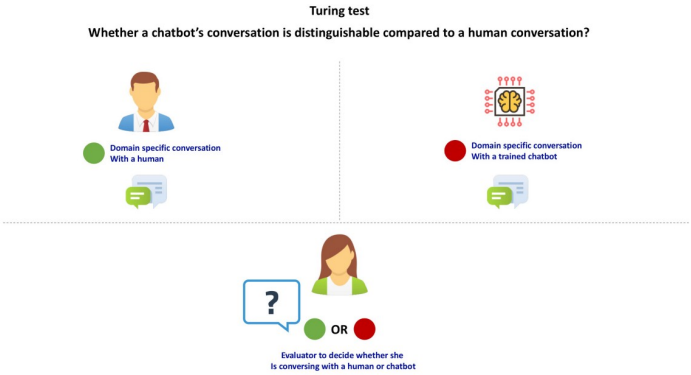
How difficult?

So what tools do developers need to do better than this?

(Ref: A New Approach to Conversational Software - Alan Nichol)

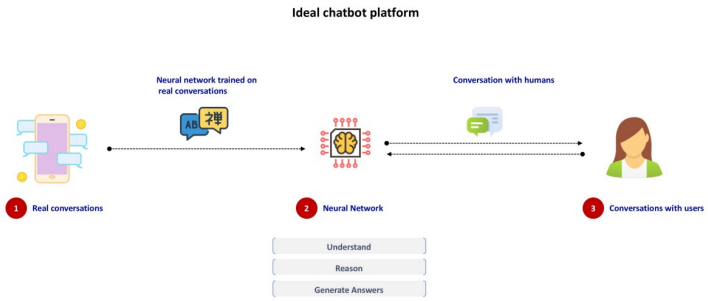
NLU is AI

Understanding Natural Language is Hallmark of Artificial Intelligence!!



(Ref: Conversational AI: Understanding the Basics and Building a Chatbot in Rasa module - Manikandan Jeeva)

NLU is AI



But the current bots are not this generic.

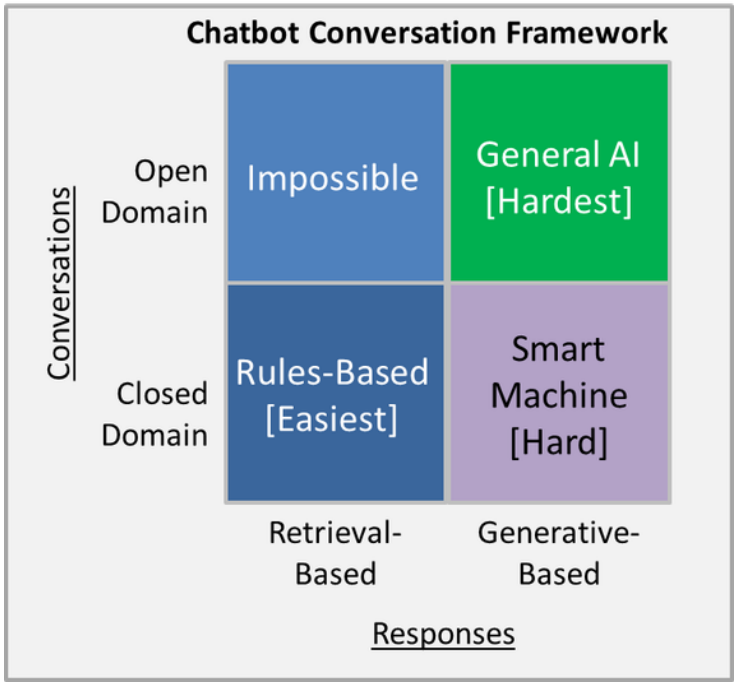
(Ref: Conversational AI: Understanding the Basics and Building a Chatbot in Rasa module - Manikandan Jeeva)

Types of Chatbots

- Command & response: Stateless bots are essentially a command line app over HTTP
- Hard-coded conversation flows: navigate a flow chart defined. http://superscriptjs.com/ allows that. Evi is an intelligent bot built with "knowledge base" technology.
- Fuzzy/continuous/fluid state: that's the goal. Human conversations don't follow a template

(Ref: We don't know how to build conversational software yet (Alan Nichol Apr 2016))

Classification of Chatbots

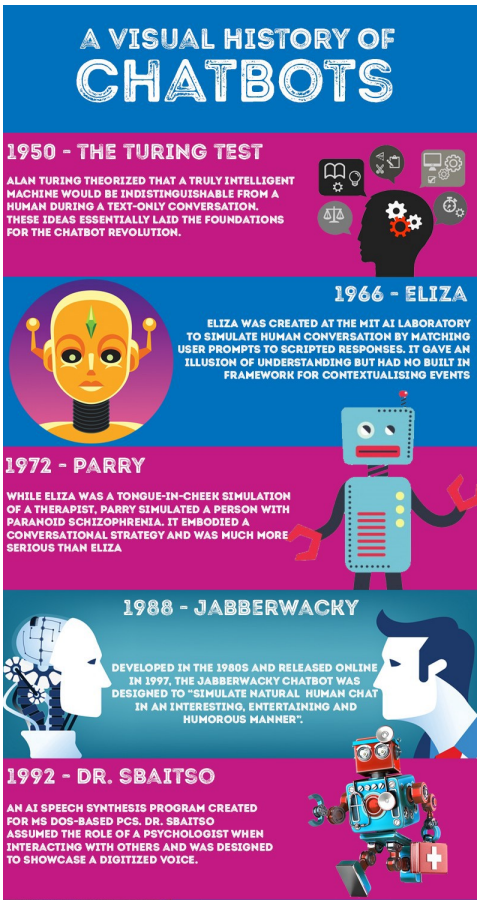


- Retrieval-based models (easier) use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context.
- Generative models (harder) are based on Machine Translation techniques, but instead of translating from one language to another, we "translate" from an input to an output (response).

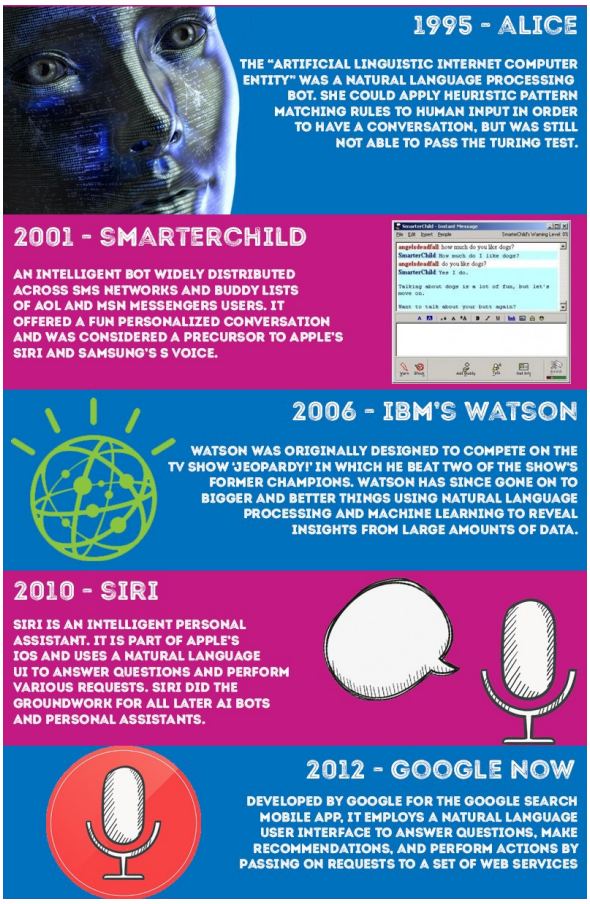
History

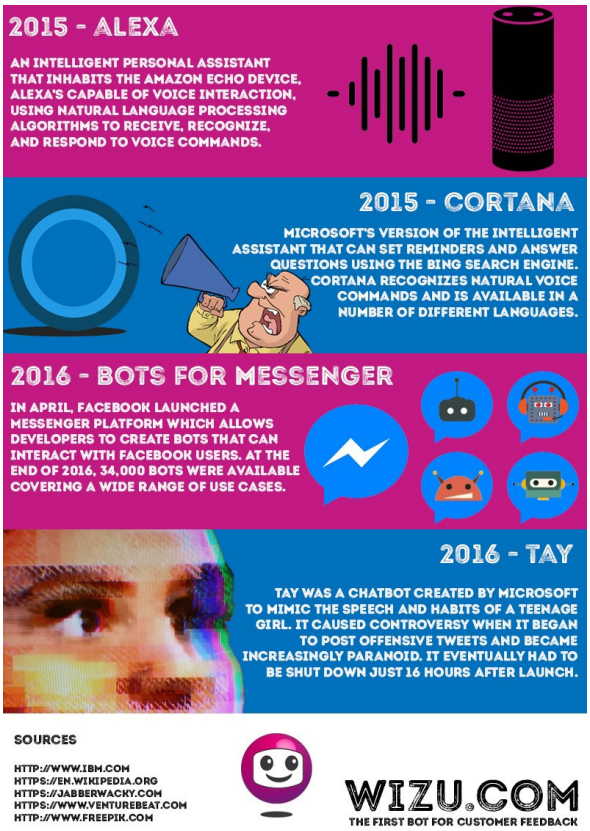
- Chatbot history: starts in 1960s.
- Eliza by MIT professor Joseph Weizenbaum: a psychotherapist, Pattern based.
- ALICE ("Artificial Linguistic Internet Computer Entity"), 1995, Richard Wallace using AIML (artificial intelligence markup language)
- Then of course, most tech giants

(Ref: Understanding AI Chatbots, Challenges, Opportunities & Beyond - Pramod Chandrayan)

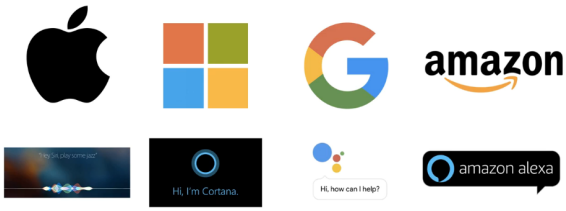


History





The Giants are at it ...



(Ref: Deep Learning and NLP A-Z - Kirill Eremenko)

If you want to develop one

- Chatbots or QA systems, predominantly voice based,
- Underlying processing is primarily Natural Language Processing (NLP).
- You can have your own chatbot, specific to you!!
- NLP is the core skill needed.

Steps for building Chatbot

- Decide domain (better if smaller)
- Design conversations (list all possible questions, answers)
- List intents (verbs), entities (nouns), actions (call-backs), response (query results)
- Train AI/ML engine
- Write backend Db code
- Create and update knowledge-base (offline, with new info)
- Test scenarios and improve

Comparison

- Rule Based (AIML)
  - Decision tree, with small samples ok
  - Pre-defined responses, so predictable
- ML Based (Rasa)
  - Large Samples a must
  - More natural responses, but initially unpredictable

Start with AIML, once data is received go with ML based

Why so much popularity?

Chatbots are:

- Autonomous and Always Available
- Drive Conversation
- Able to handle millions of requests, scalable.

But to have a good Chatbot, at core, we would need expertize in NLP!!

Challenges for Chatbot

- Security: should ensure that only relevant data is being asked and captured as an input and also is being securely transmitted over the Internet.
- Making Chatbot stick, like-able and functioning
- Language Modeling: meaning based vectorization, even for vernacular.
- etc ...

Pros

- Anytime, day or night
- Can handle repetitive, boring tasks
- Scalable
- Consistent
- Can gather data

Cons

- NLU is hard, AI is not GENERAL yet
- Cant design for ANY interaction
- Can be Risky
- Cant trust for sensitive data

Platforms

Chatbot Platforms

Bot Building Platforms

- Set of tools and architecture
- To help you design unique conversation scenarios, define corresponding actions and analyze interactions.
- Understand Natural language (NLU—Natural language understanding),
- Process the conversation text and extracts information (NLP—Natural Language Processing) and
- Respond to the user preserving the context of the conversation (NLG—Natural Language Generation).

(Ref: Chatbots 101 - Architecture & Terminologies - Bhavani Ravi)

Conversation Platforms

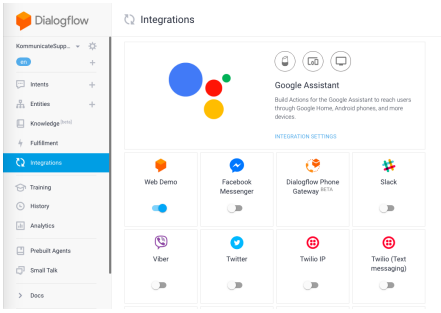
Established players

- Google DialogFlow : https://dialogflow.com
- Facebook Wit.ai : https://www.wit.ai
- IBM Watson Assistant : https://www.ibm.com/cloud/watson-assistant/
- Microsoft LUIS : https://www.luis.ai/
- Amazon Lex : https://aws.amazon.com/lex
- RASA : https://www.rasa.com/

(Ref : Dialogflow vs Lex vs Watson vs Wit vs Azure Bot — Which Chatbot Service Platform To Use?)

Google Dialogflow

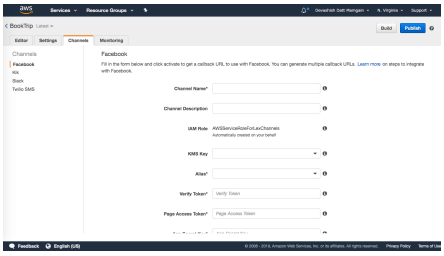
- Previous known as API.ai
- Completely closed-source product with APIs and web interface.
- Voice and text-based conversational interface
- Easy to even non-techies to create basic bots.



(Ref : Dialogflow vs Lex vs Watson vs Wit vs Azure Bot — Which Chatbot Service Platform To Use?)

Amazon Lex

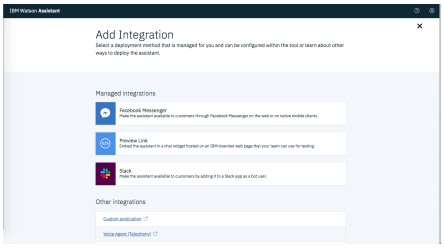
- Same deep learning technologies as Alexa
- Voice and text-based conversational interface
- Provides a web interface to create and launch bots.



(Ref : Dialogflow vs Lex vs Watson vs Wit vs Azure Bot — Which Chatbot Service Platform To Use?)

IBM Watson Assistant

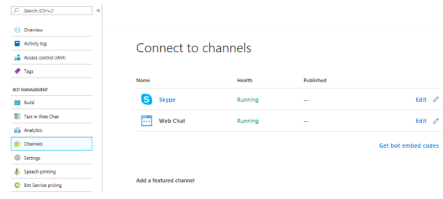
- Has support for searching for an answer from the knowledge base
- First, you need to create a Skill and then go to Assistant to integrate it with other channels.



(Ref : Dialogflow vs Lex vs Watson vs Wit vs Azure Bot — Which Chatbot Service Platform To Use?)

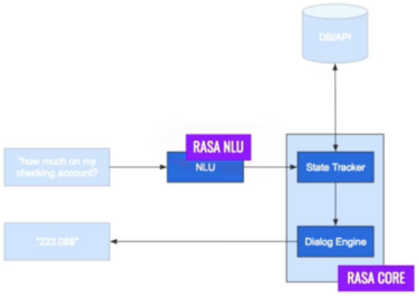
Microsoft LUIS, Azure Bot Service

- Web interface is available to create and publish bots which is fairly easy to understand.



(Ref : Dialogflow vs Lex vs Watson vs Wit vs Azure Bot — Which Chatbot Service Platform To Use?)

Rasa Chatbot Architecture



Machine Learning based and in Python.

(Ref: The talk would be about Rasa, an open-source chatbots platform - Nathan Zylbersztejn)

Introduction to Rasa

Theory Behind Rasa Platform

(Ref: Conversational AI:Building clever chatbots - Tom Bocklisch and Deprecating the state machine: building conversational AI with the Rasa stack - Justina Petraitytė)

NLU

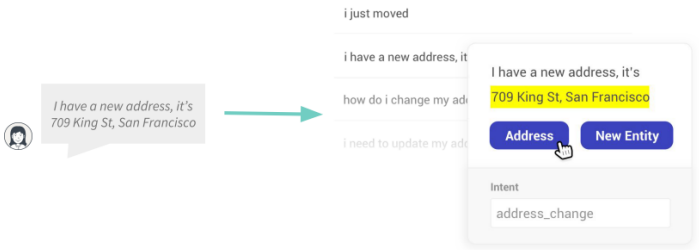
NLU Training

"train" function iterates through the pipeline and performs the NLP tasks

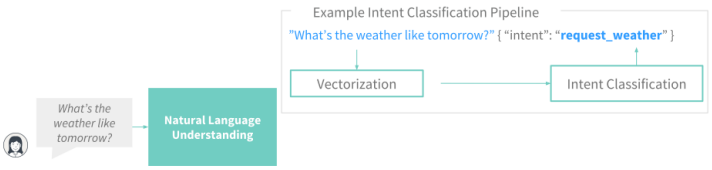
- The preprocessing step: Where the data is transformed to extract the required information. Eg. SpacyTokenizer, SpacyFeaturizer
- Entity Extractor & Intent Classifier: The preprocessed data is used to create the ML models that perform intent classification and entity extraction. NER_CRF EntityExactor, SklearnIntentClassifier
- Persistence : Storing the result

Natural Language Understanding (NLU)

Goal: create structured data



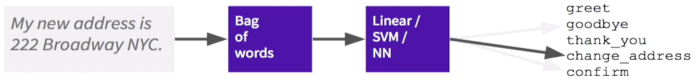
Intent Classification



Intent Classification

Intent Classification with GloVe

Bag of words sentence representation: $\{v_1, \ldots, v_s\} \rightarrow \frac{1}{s}\sum_i v_i$

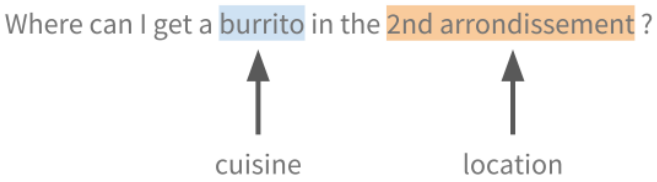


This works *embarrassingly* well, but has limitations:

- Out of vocab words: *"zanzusatzversicherung"*
- Domain-specific meaning: "balance" vs "cash"
- Single intent per message: *"yes please! Oh and can you .."*

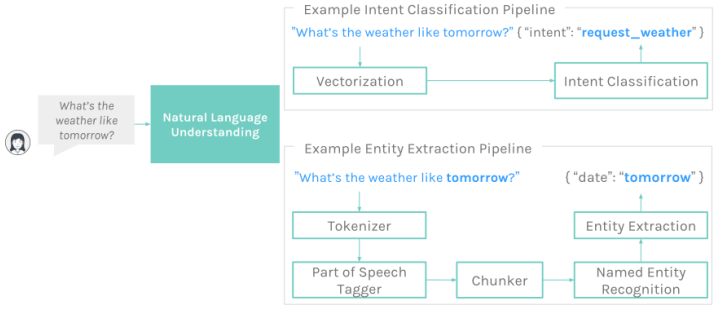
- Get word embeddings of each word, form sentence embeddings by averaging, run a classifier to find max probability intent.
- Classifier does not matter much but the quality of embedding does.
- Need domain specific vocab!!

Entity Extraction



- Can be done in phases.
- First, have a Binary classifier just to detect if the sentence has entities or not.
- Next, it would be multi class classifier to find WHICH entity is there?
- NER with Conditional Random Fields

NLU Full Workflow



NLU is Hard

Especially negations:

"No I don't want sushi"

"I'd go hungry before eating sushi"

- Both the sentences have same meaning, but how to detect!!
- First one is easy, find the negative "not" (LSTM can look for these associations), but how about the second one.

NLU is Impossible (*mushkil hi nahi, na mumkin hai*)
Some hopes: better (sentence level) embeddings.

Core

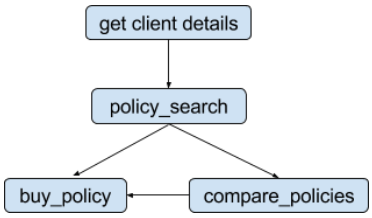
Rasa Core: Getting Rid of State Machines

The main idea behind Rasa Core:

- Thinking of conversations as a flowchart is WRONG
- Implementing conversations as state machine is WRONG
- You would need to come up with ALL possible conversations upfront and explicitly. Not possible!!

State Machines

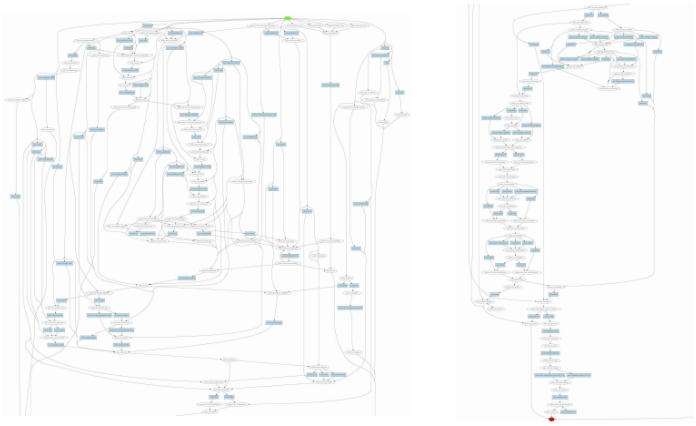
Say, for Insurance Purchase bot, the state machine could be:



It can take any branch!! What's more likely at a particular state, can be learnt by past data only, ie Machine Learning.

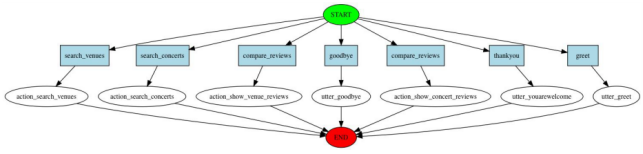
State Machines

State Machines are infeasible

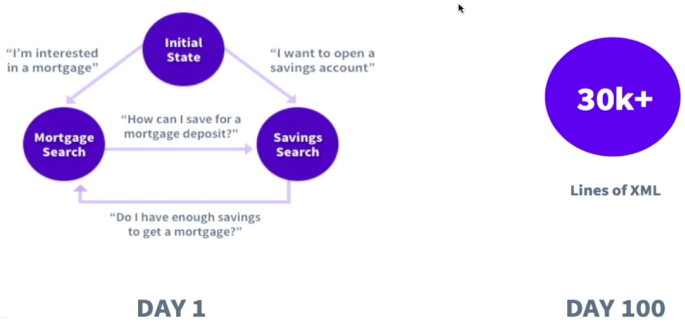


State Machines

State Machines don't scale



Why Machine Learning?



Example above is showing real numbers!!

(Ref: Building Conversational AI w Rasa Stack - Alan Nichol at PyBay2018)

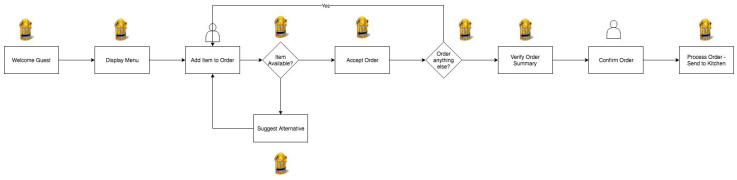
Rasa Core Approach

- Machine learning will be predicting the next action.
- But new approach cannot change radically an existing process: If you want a freaking pizza, you MUST tell a chatbot what type of base, what toppings you want and where you want it delivered.
- But let's be clever about it, there are always exceptions like in the case of pizza: ALLERGIES !! something unexpected.
- Sure you can define a logic around but how many such logic are you going to code each day.
- Keep in mind, Rasa core is not changing your process neither the machine is generating responses, it just allows you to handle exceptions better. Your rules still rule

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

Example: Ordering Food

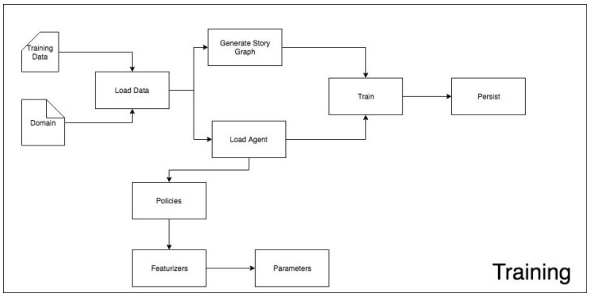
Let's build the different states



- As you can already see, a simple ordering conversation is quite complicated already,
- Covered one of the exceptions, where a given item is not available , the bot can suggest some alternatives.
- There could many such alternatives and how do we deal with item

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

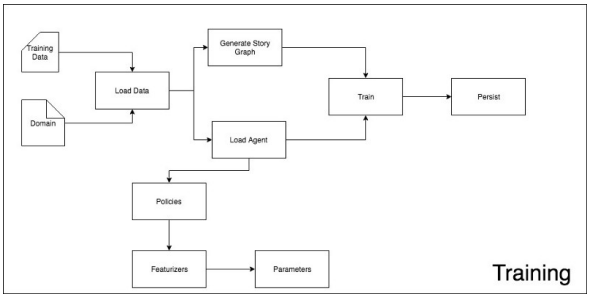
Training Steps



- Training Data: This is essentially all the stories where you typically define what is a normal conversation for your process.
- Domain basically determines what your chatbot should understand, what the chatbot can do and what kind of information is necessary for your chatbot's context so it understands the user better.

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

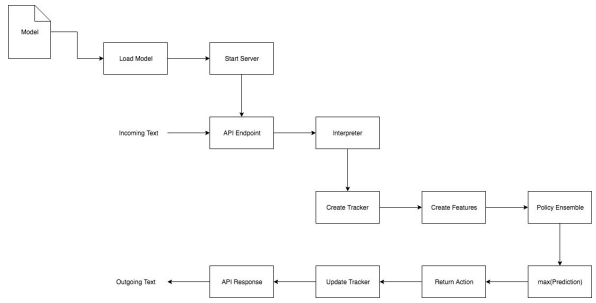
Training Steps



- Load Agent: Agent(or the bot) is first loaded with some parameters that determines how the training data will be converted into features for training the agent. Here, really important parameter is "Policy"
- A policy is what will define what is going to be the next action. As Rasa core is open-source, you can indeed create your own policy but let's get the basics right and see what are the already available policies that are used by default.

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

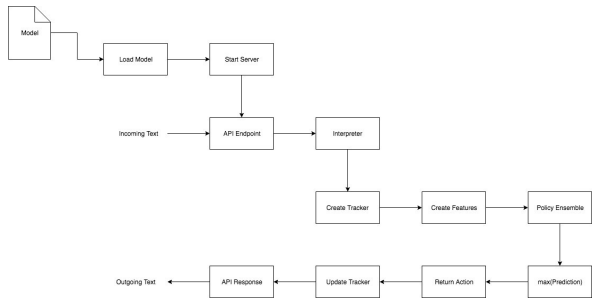
Prediction Process



- Load Model in memory before serving using a Server(Flask) and exposing an endpoint related to a particular channel, in our case we will deal with a REST API.
- Interpreter is able to read the raw text coming in from user and throw out the intention of the user along with some meaning entities, these entities which we are saving as slots that will drive the conversation.

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

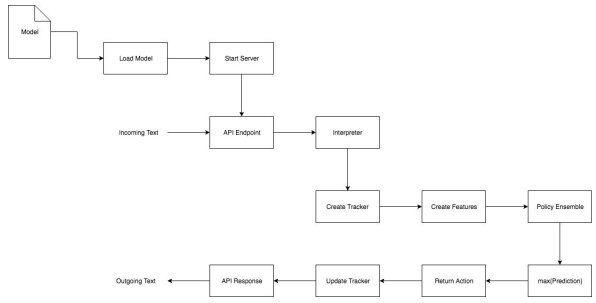
Prediction Process



- CreateOrUpdate Tracker: If this is the first message of the conversation, rasa core will create a tracker object with the key "sender_id" which is the incoming identifier of the user. Tracker object is usually stored in a tracker_store which is by default is in InMemory.
- Features are generated from contents of the tracker based on the policy
- Features will given to the policy ensemble which will determine the final outcome.

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

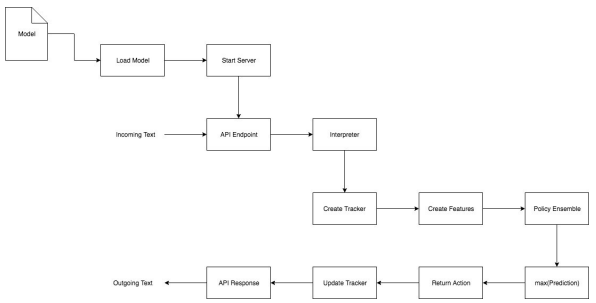
Prediction Process



- PolicyEnsemble: Since, we have trained different policies, when it comes to predicting the next action, each of these policies will provide a score for the particular action.
- Then there is a max taken from all scores given by every policy and whichever wins, will be the next action

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

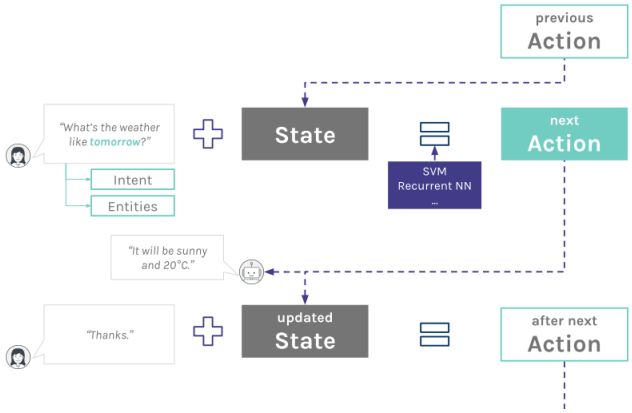
Prediction Process



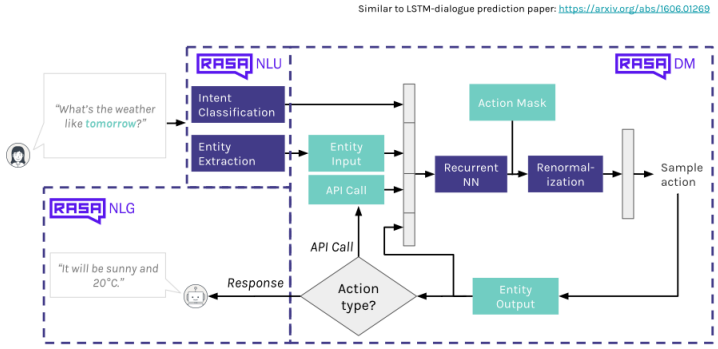
- Update Tracker: Once you have the action predicted, you will need to update the tracker for the next turn
- ExecuteAction: Now you will be finally executing your action, be it an API call or a message sent back to the user

(Ref: Contextual Conversational Engine— The Rasa Core Approach: Part 1 - Souvik Ghosh)

Summary: Overall Workflow



Summary: Overall Workflow



Getting Started

Getting Started

Installation

Python

- Install Anaconda for Python 3.7
- Else from Python.org and then additionally all requisite libraries
- Ubuntu :

```
sudo apt-get install build-essential python-dev git
```

- Windows Build tools: Make sure the Microsoft $VC++$ Compiler Visual Studio 2015 is installed, so python can compile any dependencies or https://visualstudio.microsoft.com/visual-cpp-build-tools/ Download the installer and select VC++ Build tools in the list.

Conda Installation

- Install the Conda(miniconda) from https://docs.conda.io/en/latest/miniconda.html as per the OS
- Check the Conda version `conda --version` conda 4.7.10
- In case need to upgrade, run below command `conda update conda`

Setup in Virtual Environment

- By installing conda, you get base or the root environment, which is the default.
- Practical tip: DO NOT install any packages in the root. ALWAYS create and env and install inside the new env.
- Env is needed especially for fragile packages like Python (its treated as a package) and rasa.
- So, `conda create -n rasa_env python=3.7`
- Python 3.6 as different asyncio format, so better to do it in 3.7
- Activate the new environment to use it

```
LINUX, macOS: conda activate rasa
WINDOWS: activate rasa
```

(Note: Env management is again a sour point. It creates complete copy (deeeep) of python 3.7 and all other packages inside "envs" folder. Goes to 1.5 GB!! Can someone optimize it?)

Rasa Installation

- Install latest Rasa stack.
- Rasa NLU + Core is now in the single package. Do not install them separately like in past. Your mileage may vary.
- `pip install rasa`

If you get Microsoft Build tool error:

- Go to https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2017
- Build tools for Visual Studio 19 are fine too. Select only $C++$ Build tools. Install.
- Re-run rasa install command

Spacy Installation

Additionally, spaCy:

```
pip install rasa[spacy]
python -m spacy download en
python -m spacy download en_core_web_md
python -m spacy link en_core_web_md en
```

If you get linking permission error:

- Run cmd as administrator,
- Activate rasa_env
- Do all the above spacy commands.

Other Installations

To show conda envs in notebook `conda install nb_conda_kernels`
Apart from this, may need to

- `pip install nest_asyncio`
- Write following code to test, in ipynb put this in the first cell

```
import nest_asyncio

nest_asyncio.apply()
print("Event loop ready.")
```

Some more (optional):

- Download and install ngrok from https://ngrok.com/download
- Need to have API keys for Slack, CRICINFO, ZOMATO, etc

Demo

Rasa Init

```
mkdir firstbot
cd firstbot
activate rasa_env
rasa init  --no-prompt
```

- Creates project structure and dummy files.
- Good to run and execute
- You can run as is or add your data to nlu.md, stories.md and domain.yml; retrain and run.
- If you get any warnings replace "rasa" with "python -W ignore -m rasa "

(Ref: How to build awesome Rasa chatbot for a web - Martin Novak)

Rasa Shell

To run the bot:

```
python -W ignore -m rasa shell --quiet --cors * -m models


# Sample Chat
Bot loaded. Type a message and press enter (use '/stop' to
    exit):
Your input ->  hi
Hey! How are you?
Your input ->  perfect
Great, carry on!
Your input ->  are you a bot?
I am a bot, powered by Rasa.
Your input ->  /stop
```

Now, sky is the limit . . .

The End

Conclusions

Take aways

3 take home thoughts:

- Conversational AI is a big part of the future
- ML techniques help advance state-of-the-art NLU and conversational AI
- Open source is strategically important for enterprises implementing AI

Conclusions

- RASA Is an Open Sourced Python implementation for NLP Engine / Intent Extraction / Dialogue → in which all of the above run on your machine / On premise → NO CLOUD!
- RASA can be integrated with different front ends like Slack, Facebook Messenger, or your own web app.

What Next?

- Try it out, tutorials . . .
- Subscribe to Rasa Newsletter (Rasa X has arrived!! Rasa NLU and Core got merged into Rasa 1.0 . . .)
- Build your own chatbot . . .

References

Many publicly available resources have been refereed for making this presentation. Some of the notable ones are:

- RASA-NLU setup, installation, https://github.com/RASAHQ/rasa_nlu
- Chatbots 101 - Architecture & Terminologies - Bhavani Ravi and the event-bot code
- Building chatbots using Python/Django - Youtube video.
- GST FAQ http://www.cbec.gov.in/resources//htdocs-cbec/deptt_offcr/faq-on-gst.pdf
- "Building a Conversational Chatbot for Slack using Rasa and Python" - Parul Pandey
- "The next generation of AI assistants in enterprise" - Alan Nichol
- Top 8 Healthcare Predictions for 2019 - FROST & SULLIVAN/ Reenita Das
- How Artificial Intelligence is Changing the Healthcare Industry - Sumi menon
- Can Healthcare Chatbots Improve the Patient Experience? - Intakeq
- Pydata Berlin talk by Tom Bocklisch
- Conversational AI: Design & Build a Contextual AI Assistant - Mady Mantha

---

Copyleft ⓒ Send suggestions to yogeshkulkarni@yahoo.com