Pavel Surmenok Follow
Engineering Manager at JustAnswer. Machine learning engineering and chatbots. All opinions are my own.
Sep 12, 2016 · 8 min read

Chatbot Architecture

Chatbots are on the rise. Startups are building chatbots, platforms, APIs, tools, analytics. Microsoft, Google, Facebook introduce tools and frameworks, and build smart assistants on top of these frameworks. Multiple blogs, magazines, podcasts report on news in this industry, and chatbot developers gather on meetups and conferences.

I have been working on chatbot software for a while, and I have been looking on what is going on in the industry. See my previous posts:

- Natural Language Pipeline for Chatbots
- Intelligence Platform Stack
- Character-level Convolutional Networks for Text Classification

In this article, I will dive into architecture of chatbots.

What kinds of bots are there?

There are two major types of chatbots: chatbots for entertainment and chatbots for business.

Engineers have been developing chatbots for entertainment for decades since famous chatbot-psychotherapist ELIZA was introduced in 1966. Creators of these chatbots usually try to make a bot which can look like a human, pass the Turing test. Perhaps all of the bots which participate in Loebner's prize and similar competitions are in this group. Microsoft's bots Xiaoice and Tay have similar behavior. The most recent example is "Spock" bot in Skype: "Chat with Spock, second in command of the USS Enterprise, to learn the ways of Vulcan logic!"

Chatbot responses to user messages should be smart enough for user to continue the conversation. The chatbot doesn't need to understand what user is saying and doesn't have to remember all the details of the dialogue.

One way to assess an entertainment bot is to compare the bot with a human (Turing test). Other, quantitative, metrics are an average length of conversation between the bot and end users or average time spent by a user per week. If conversations are short then the bot is not entertaining enough.

Chatbots for business are often transactional, and they have a specific purpose. Conversation is typically focused on user's needs. Travel chatbot is providing an information about flights, hotels, and tours and helps to find the best package according to user's criteria. Google Assistant readily provides information requested by the user. Uber bot takes a ride request.

Conversations are typically short, less than 15 minutes. Each conversation has a goal, and quality of the bot can be assessed by how many users get to the goal. Has the user found information she was looking for? Has the user successfully booked a flight and a hotel? Has the user bought products which help to solve the problem at hand? Usually, these metrics are easy to track.

Perhaps some bots don't fit into this classification, but it should be good enough to work for the majority of bots which are live now.

Another useful classification is based on a type of conversation: one-to-one or one-to-many, if the chatbot is added into a group chat. Dynamics of conversation, use cases, complexity of chatbot software is very different for these cases.

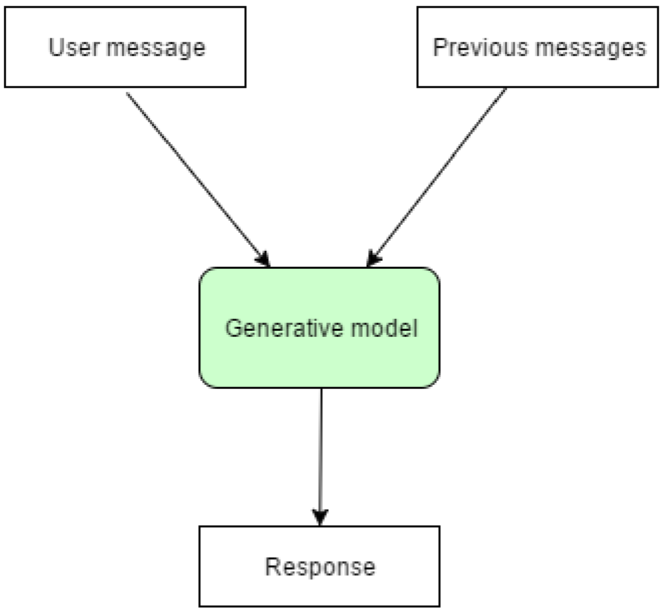
Models

As Denny Britz wrote in "Deep learning for chatbots", the chatbot can either generate responses from scratch based on machine learning models or use some heuristic to select a response from a library of predefined responses.

Generative models are harder to build and train. Typically it requires millions of examples to train a deep learning model to get decent quality of conversation, and still you can't be totally sure what responses the model will generate.

Generative models

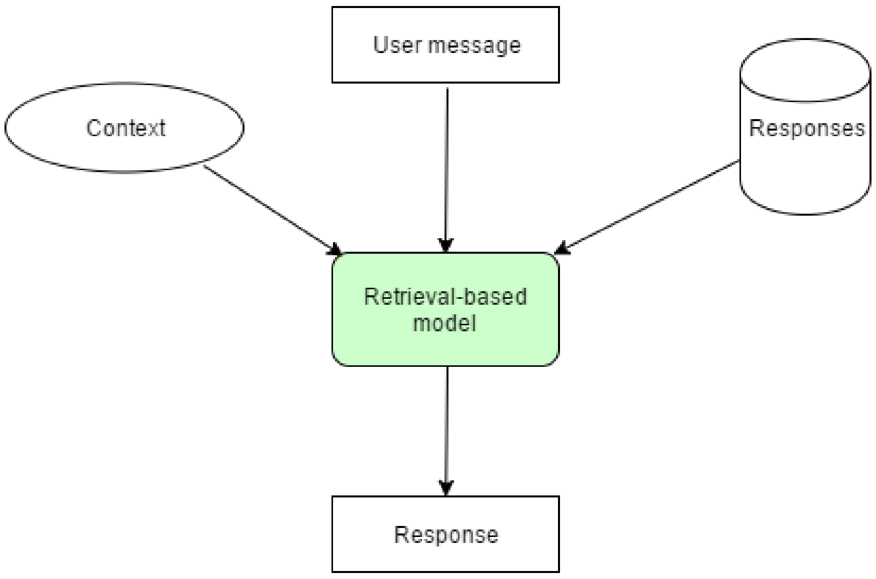
Generative models are the future of chatbots, they make bots smarter. This approach is not widely used by chatbot developers, it is mostly in the labs now.



Retrieval-based models

Retrieval-based models are much easier to build. They also provide more predictable results. You probably won't get 100% accuracy of responses, but at least you know all possible responses and can make sure that there are no inappropriate or grammatically incorrect responses.

Retrieval-based models are more practical at the moment, many algorithms and APIs are readily available for developers.



The chatbot uses the message and context of conversation for selecting the best response from a predefined list of bot messages. The context can include current position in the dialog tree, all previous messages in the conversation, previously saved variables (e.g. username).

If the bot doesn't use context then it is stateless. It will only respond to the latest user message, disregarding all the history of theconversation.

Pattern-based heuristics

Heuristics for selecting a response can be engineered in many different ways, from if-else conditional logic to machine learning classifiers. The simplest technology is using a set of rules with patterns as conditions for the rules. This type of models is very popular for entertainment bots. AIML is a widely used language for writing patterns and response templates. Bot developers write code in AIML language, code can include multiple units like this:

> ***<category>***
> ***<pattern>****WHAT IS YOUR NAME****</pattern>***
> ***<template>****My name is Michael N.S Evanious.****</template>***
> ***</category>***

When the chatbot receives a message, it goes through all the patterns until finds a pattern which matches user message. If the match is found, the chatbot uses the corresponding template to generate a response.

ChatScript is a modern implementation of this idea. It is an open source chatbot engine which allows defining a chatbot in a rule-based language. Each rule contains a pattern and an output:

> *s: (because [someday "one day"]) That won't be soon.*

ChatScript engine has a powerful natural language processing pipeline and a rich pattern language. Using ChatScript you can do much more than with AIML. It will parse user message, tag parts of speech, find synonyms and concepts, and find which rule matches the input. In addition to NLP abilities, ChatScript will keep track of dialog, so that you can design long scripts which cover different topics. It won't do anything fancy, though. It won't run machine learning algorithms and won't access external knowledge bases or 3rd party APIs unless you do all the necessary programming.

Machine learning for intent classification

The inherent problem of pattern-based heuristics is that patterns should be programmed manually, and it is not an easy task, especially if the chatbot has to correctly distinguish hundreds of intents. Imagine that you are building a customer service bot and the bot should respond to a refund request. Users can express it in hundreds of different ways: "I want a refund", "Refund my money", "I need my money back". At the same time, the bot should respond differently if the same words are used in another context: "Can I request a refund if I don't like the service?", "What is your refund policy?". Humans are not good at writing patterns and rules for natural language understanding, computers are much better at this task.

Machine learning lets us train an intent classification algorithm. You just need a training set of a few hundred or thousands of examples, and it will pick up patterns in the data.

Such algorithms can be built using any popular machine learning library like scikit-learn. Another option is to use one of cloud API: wit.ai, api.ai, Microsoft LUIS. Wit.ai was probably the first machine learning API for chatbots, it was bought by Facebook this year, and it is free.

Response generation

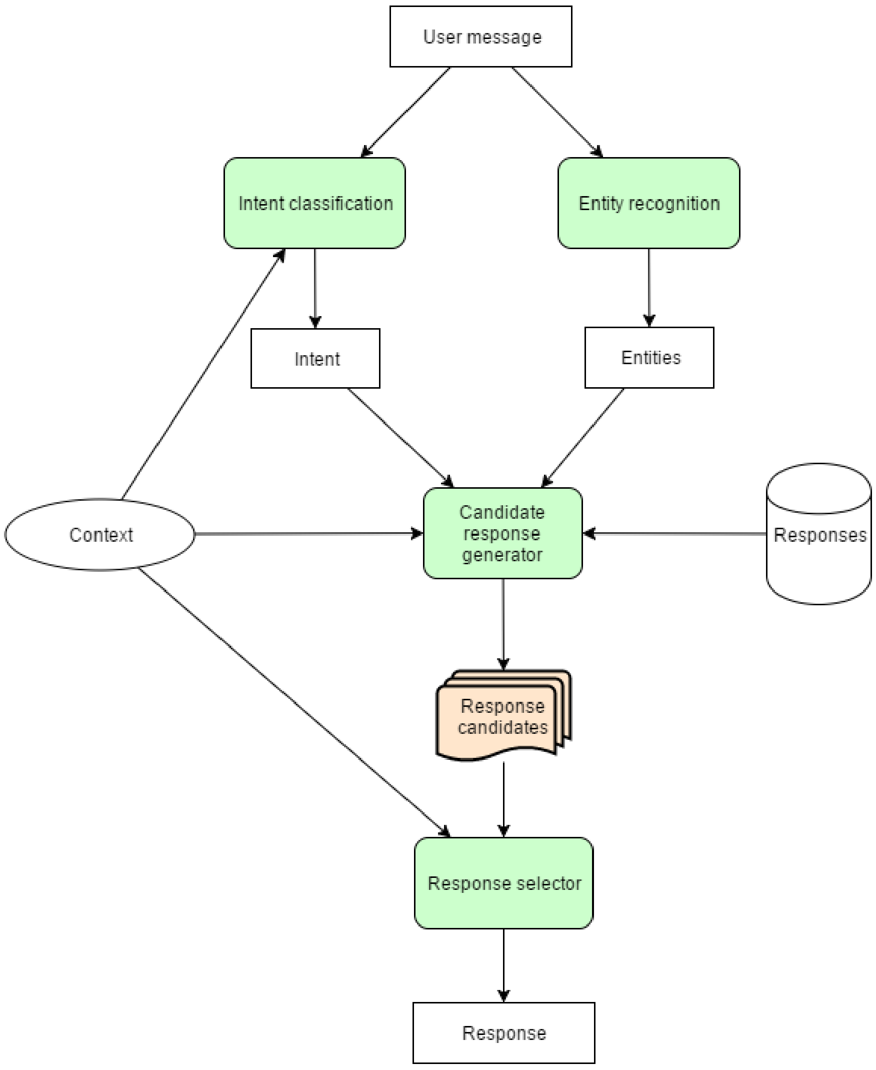
Patterns or machine learning classification algorithms help to understand what user message means. When the chatbot gets the intent of the message, it shall generate a response. How can the bot do it? The simplest way is just to respond with a static response, one for each intent. Or, perhaps, get a template based on intent and put in some variables. It is what ChatScript based bots and most of other contemporary bots are doing.

How can bots do better? There is no single answer. Response generation mechanism must depend on the task at hand. A medical chatbot will probably use a statistical model of symptoms and conditions to decide which questions to ask to clarify a diagnosis. A question-answering bot will dig into a knowledge graph, generate potential answers and then use other algorithms to score these answers, see how IBM Watson is doing it. A weather bot will just access an API to get a weather forecast for a given location.

Architecture with response selection

The chatbot can express the same message using different words. A weather bot can say "It's rainy", or "Probability of rain is 80%" or "Please carry an umbrella today". Which one will work the best for the user? Different users prefer different styles of response. The bot can analyze previous chats and associated metrics (length of the conversation, probability of sale, rating of customer satisfaction, etc.) to tailor responses for the user.

The chatbot can have separate response generation and response selection modules, as shown in the diagram below.



Message processing begins from understanding what the user is talking about. Intent classification module identifies the intent of user message. Typically it is selection of one out of a number of predefined intents, though more sophisticated bots can identify multiple intents from one message. Intent classification can use context information, such as intents of previous messages, user profile, and preferences. Entity recognition module extracts structured bits of information from the message. The weather bot can extract location and date.

The candidate response generator is doing all the domain-specific calculations to process the user request. It can use different algorithms, call a few external APIs, or even ask a human to help with response generation. The result of these calculations is a list of response candidates. All these responses should be correct according to domain-specific logic, it can't be just tons of random responses. The response generator must use the context of the conversation as well as intent and entities extracted from the last user message, otherwise, it can't support multi-message conversations.

The response selector just scores all the response candidate and selects a response which should work better for the user.

How are you designing software for your bots, and what algorithms, libraries, and APIs are you using?

. . .

See also:

Natural Language Pipeline for Chatbots

Character-level Convolutional Networks for Text Classification

Best Sources of Deep Learning News

. . .

The article was originally published on http://pavel.surmenok.com/2016/09/11/chatbot-architecture/