A chatbot assistant

implemented using NLP and ML by

Băilă Romina & Cocoş Tudor

Version	Details	Date
1.0	First report	27.03.2019
1.1	Second report + basic chatbot	09.04.2019

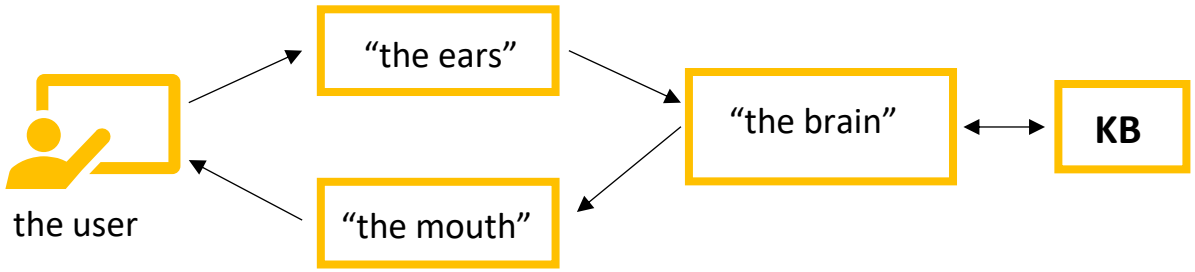
Proposed idea

We would like to develop a chatbot that would act like a personal assistant while it resides in a private chat room with yourself in an application like Slack or MS Teams.

The full functionality will be described by two main features: detecting the intent of the user by analysing a text input in chat and the second one, learning more and more about us by storing relevant parts of our discussions in a knowledge base. By doing so, it will need to ask further less questions when asked to do something, because it will extract that information from the context that it has built.

Analysis & design

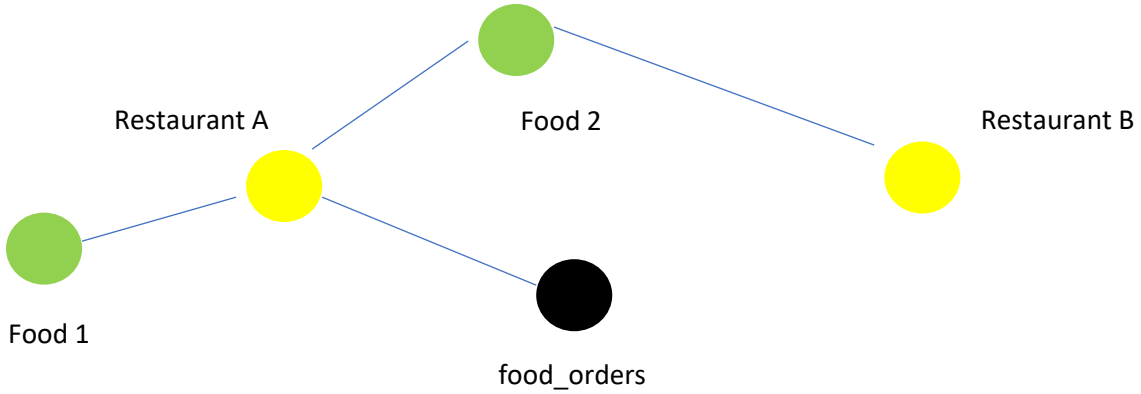
As a general concept, seen throughout the articles that we have read, the main components of such a system would be:



Once the user starts communication with the bot, it will read the text input using "the ears" and turn it into relevant input for the Natural Language Processor (NLP). Having parsed the text, the tokens contained in the output of this component will be passed to the brain where the intent is being detected. To this, context information will be added if it exists in the knowledge base (KB). Also, if the intent is valid and the new information that was inputted is relevant and not present in the KB, "the brain" will also store this new content for later use and an improved context. Having found an intent or not, or needing additional information, will be communicated back to the user via "the mouth" which will transform information and answers/requests into natural language for the user.

Development

To start, we will try to implement a basic functionality for the chatbot: ordering food for the lunch break. It will take as input a text that contains a name of a restaurant or fast-food joint, what to order and in what quantity and maybe even some details about the order. The delivery details should be already in the KB as it should know at the start who we are, where do we work and what is our email address or phone number. Also, we would like to see that once we ordered from a restaurant, for example, next time we order something and we don't specify the place, but the food is similar, it should know what restaurant we might refer to from its KB, thus asking and needing less and less information from the user.



Here is an example of how the knowledge base could look like.

If we order food of type “Food 1”, no need to worry, it is only associated with Restaurant A. However, there could be a conflict if we want to order “Food 2” and here the chatbot would still have to ask us from where should it order the food, but at least it will offer some options that it already knows of and not put us in a situation to go look ourselves to make the choice.

Once we manage to achieve a working prototype of this functionality, we can start to diversify what the chatbot can do, especially on the business side of things (reports, appointments, to-do agenda, etc).

Technologies used

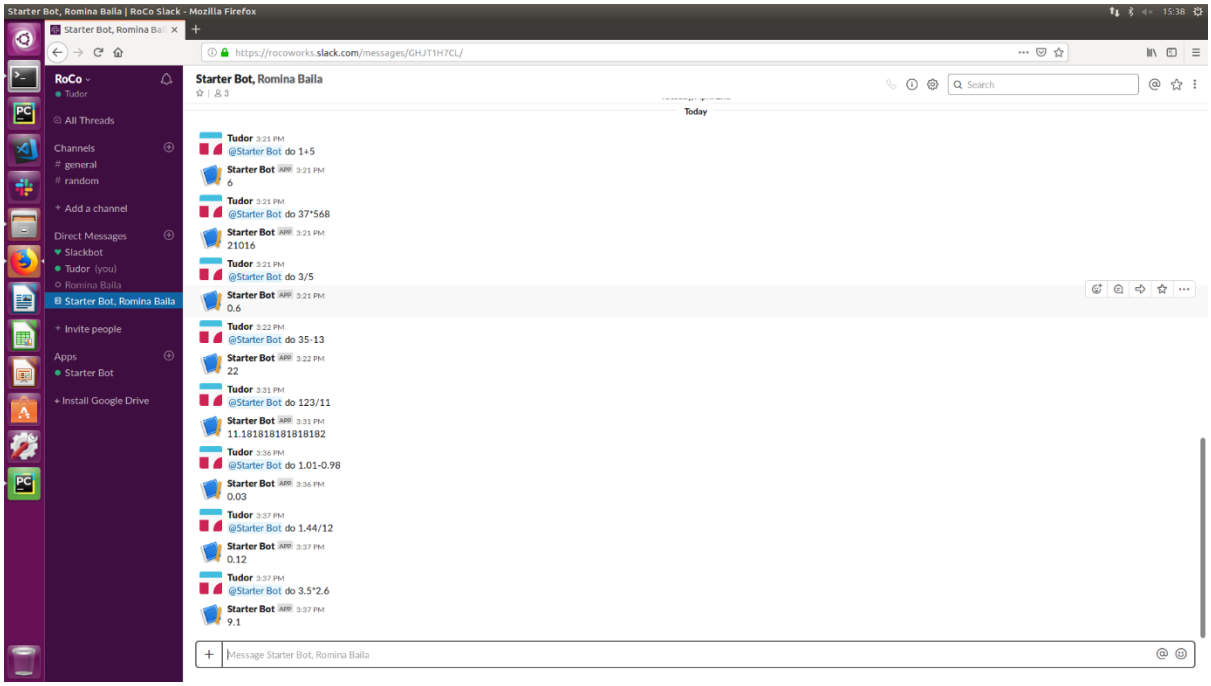
We would like to develop the chatbot in Python and as far as language processing goes, we will make use of NLTK or AllenNLP (we still have to try and test to see which one works better with the chatbot).

Implementation details

Using the Slack API, we created a simple chatbot that can analyse the messages in a conversation on Slack. The bot, called Starterbot, reads all the messages and reacts only when it’s summoned. Then, it takes the command and figures out what the user wants it to do by searching for keywords.

So far, Starterbot is only able to perform a simple computation (ex. 1+5 or 3.7/1.2) as we focused more on integrating it with Slack. To do this we set up a Slack application on our Workspace with our bot and then we added it to a conversation with the two of us. Next, we took the access token for the API and saved it in our virtual environment on the local machine. As we run the bot, using python, it will use the access token to connect to our workspace and be able to “see” the conversations. To be able to tell if it is summoned using “@Starter Bot”, a regex is used, and if it is the case, the command is processed in the “handle_command” method.

The next target will be to make it so that the intent will be automatically detected, and also try to experiment with Dialogflow (https://dialogflow.com/).



Bibliography

1. https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS17/paper/download/15426/14918
2. https://medium.com/@phanimarupaka/nlp-design-for-chatbots-4954c2527d88
3. https://medium.com/@BhashkarKunal/conversational-ai-chatbot-using-rasa-nlu-rasa-core-how-dialogue-handling-with-rasa-core-can-use-331e7024f733?fbclid=IwAR1TJY_iwvqBoLx-2Ske9V85zUjD1sZ3Dh45-bwrORBMSePUXCPX_Gg_-TA
4. https://www.fullstackpython.com/blog/build-first-slack-bot-python.html
5. https://medium.com/swlh/deep-learning-for-text-made-easy-with-allennlp-62bc79d41f31