|  |
| --- |
| Introduction to RASA Chatbot Framework |
|  |
| April 2020  Authored by: Siddhartha Ranjan Bhattacharya |



Contents

[Setting the Context 3](#_Toc37786026)

[What is a Chatbot? 4](#_Toc37786027)

[Intro to RASA 5](#_Toc37786028)

[Components of RASA 5](#_Toc37786029)

[Installing RASA 5](#_Toc37786030)

[Building the BOT 9](#_Toc37786031)

[Problem Statement 9](#_Toc37786032)

[Prepare Training Data 9](#_Toc37786033)

[Prepare the Response 11](#_Toc37786034)

[Quick Recap 12](#_Toc37786035)

[Training the BOT 12](#_Toc37786036)

[Test the BOT 13](#_Toc37786037)

[Integration with 3rd party channels 14](#_Toc37786038)

[Wrapping it up 16](#_Toc37786039)

[References 17](#_Toc37786040)

# Setting the Context

*This document provides an introduction of the RASA NLU chatbot framework. The intent is to cover the basics so that anyone can start building his/her own chatbot within a day. The framework is quite huge as a result not all the aspects have been covered. The key components have been touched upon and it is expected that as someone starts doing practical work, further exploration can be done by the individual. Also, the language chosen is business friendly and not technical oriented owing to the very nature of my background 😊.*

# What is a Chatbot?

When I searched the internet, figured out almost thousand of definitions. So, thought of simplifying the same in my write-up. As the name suggests, there are 2 parts.

* Chat – which means capable of doing a conversation
* Bot – which means it is non-human being (often times a software application)

To put in context, a chatbot is something which will understand human language and reply back accordingly and carry on with the conversation till you are fulfilled. With the rapid advancement of AI & Machine Learning, conversational AI has improved a lot. Chatbot is nothing but conversational AI in action. While traditional chatbots used to be rule based, which is very cumbersome to be maintained, majority of the current chatbot frameworks are AI based.

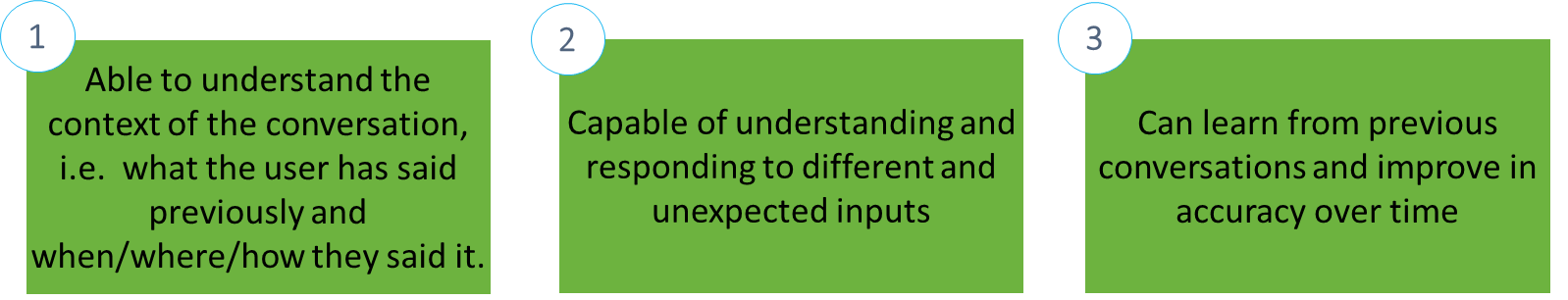
*Caution- Please don’t expect that chatbots are emotional beings. These are applications which are trained to do some pre-defined conversation. So, it will not be able to fulfil all types of conversation as a normal human being can.*

Some of the key benefits of chatbots are –

* Customer Service – Almost the entire service sector (banks, insurance, college etc) have deployed chatbots in their commnuication channels as a 1st touchpoint to their customers.
* Cost Effective – Once build & deployed, running cost is almost neglible.
* Automation – zero human touch
* Time to launch – time to build & deploy is in days. Also these applications can be easily integrated with whole host of channels. (company website, facebook messenger, twitter etc)

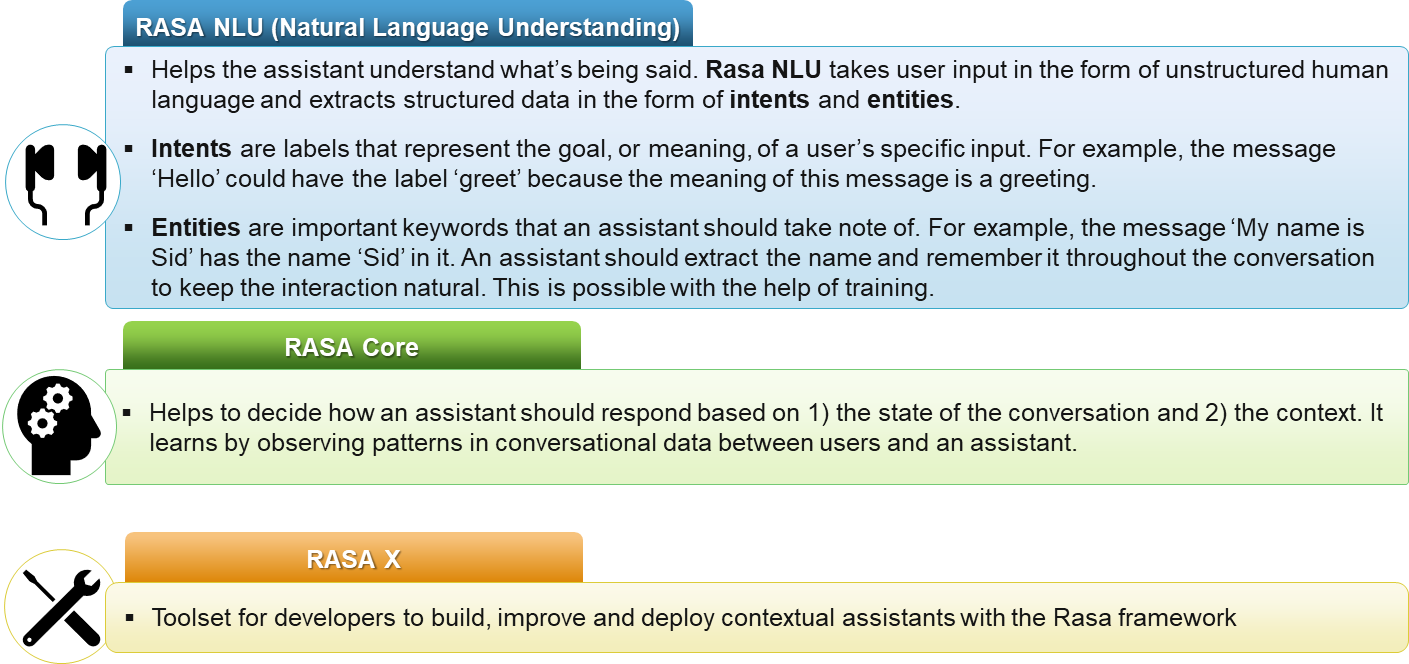
# Intro to RASA

RASA is an open source platform capable of building contextual assistants or AI based chatbots. Key features of such a chatbot are –



## Components of RASA

The following depiction provides a view of the RASA components.



## Installing RASA

Let us now focus on the installation of RASA framework. Note that I have provided steps for a Windows machine. Individual can refer to the following link for all the details. [Official RASA installation guide](https://rasa.com/docs/rasa/user-guide/installation/#installation-guide)

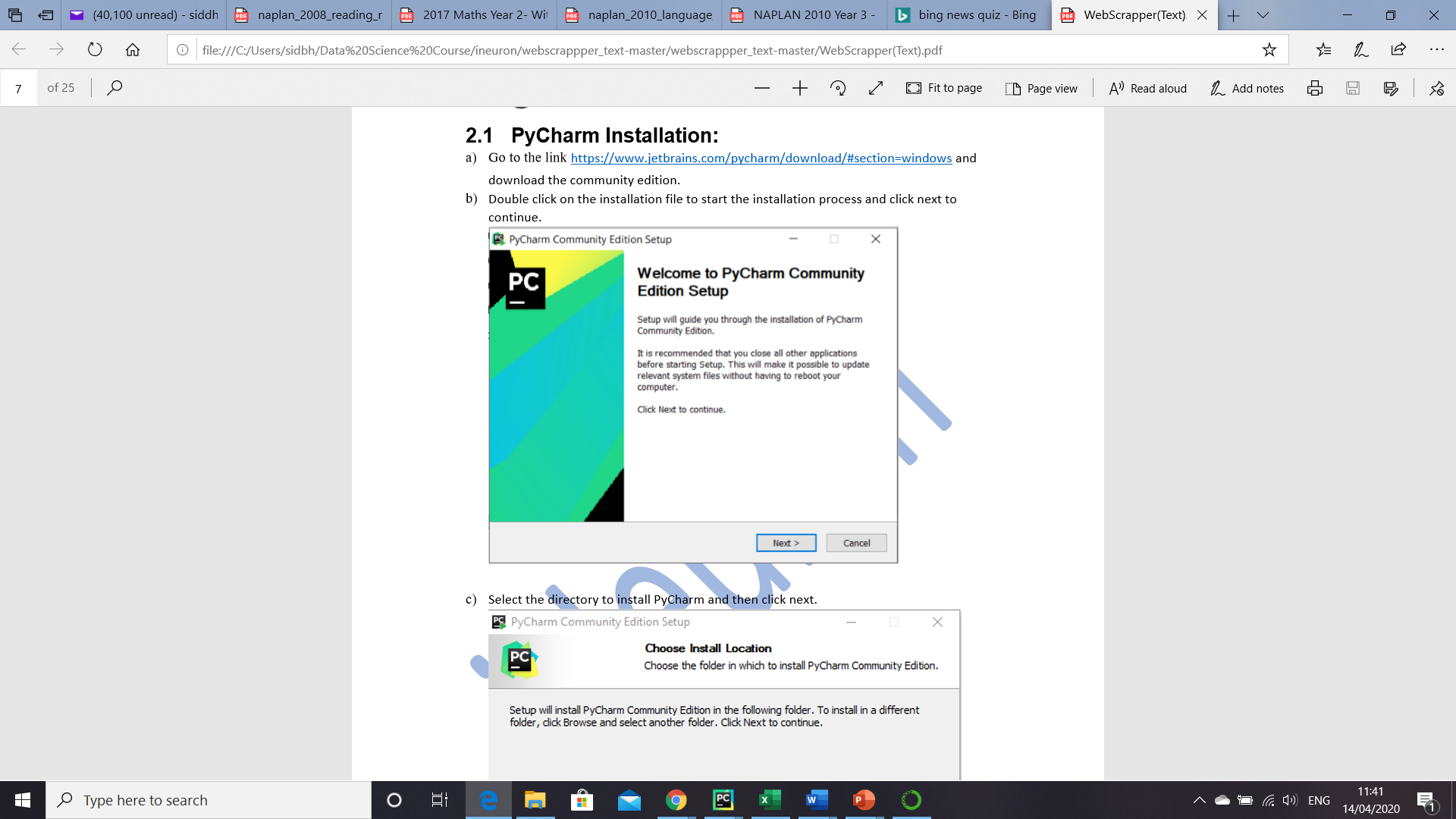
Prerequisite - Make sure the Microsoft VC++ Compiler is installed, so python can compile any dependencies. You can get the compiler from [Visual Studio](https://visualstudio.microsoft.com/visual-cpp-build-tools/). Download the installer and select VC++ Build tools in the list.

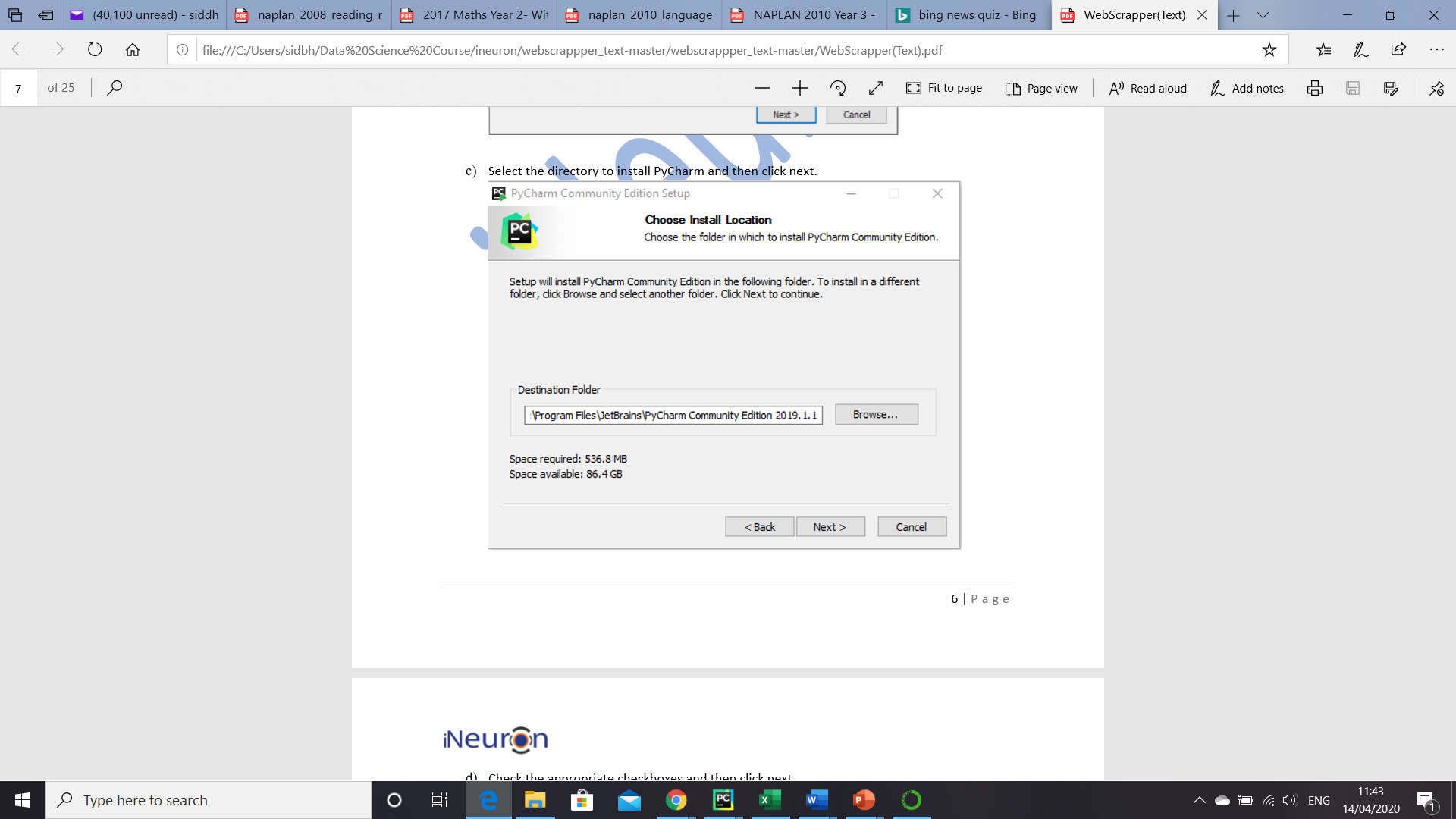
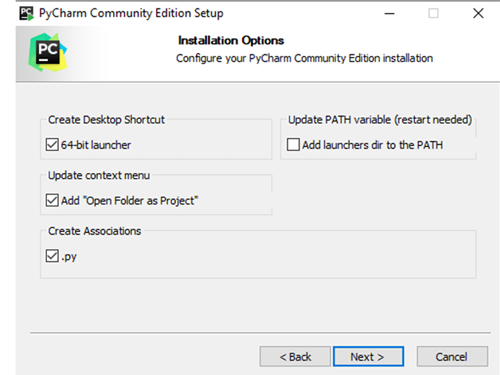
RASA requires python 3.6/3.7. It is strongly recommended to create a new virtual env for RASA using conda. Commands to use using anaconda prompt –

conda create -n rasa python=3.6

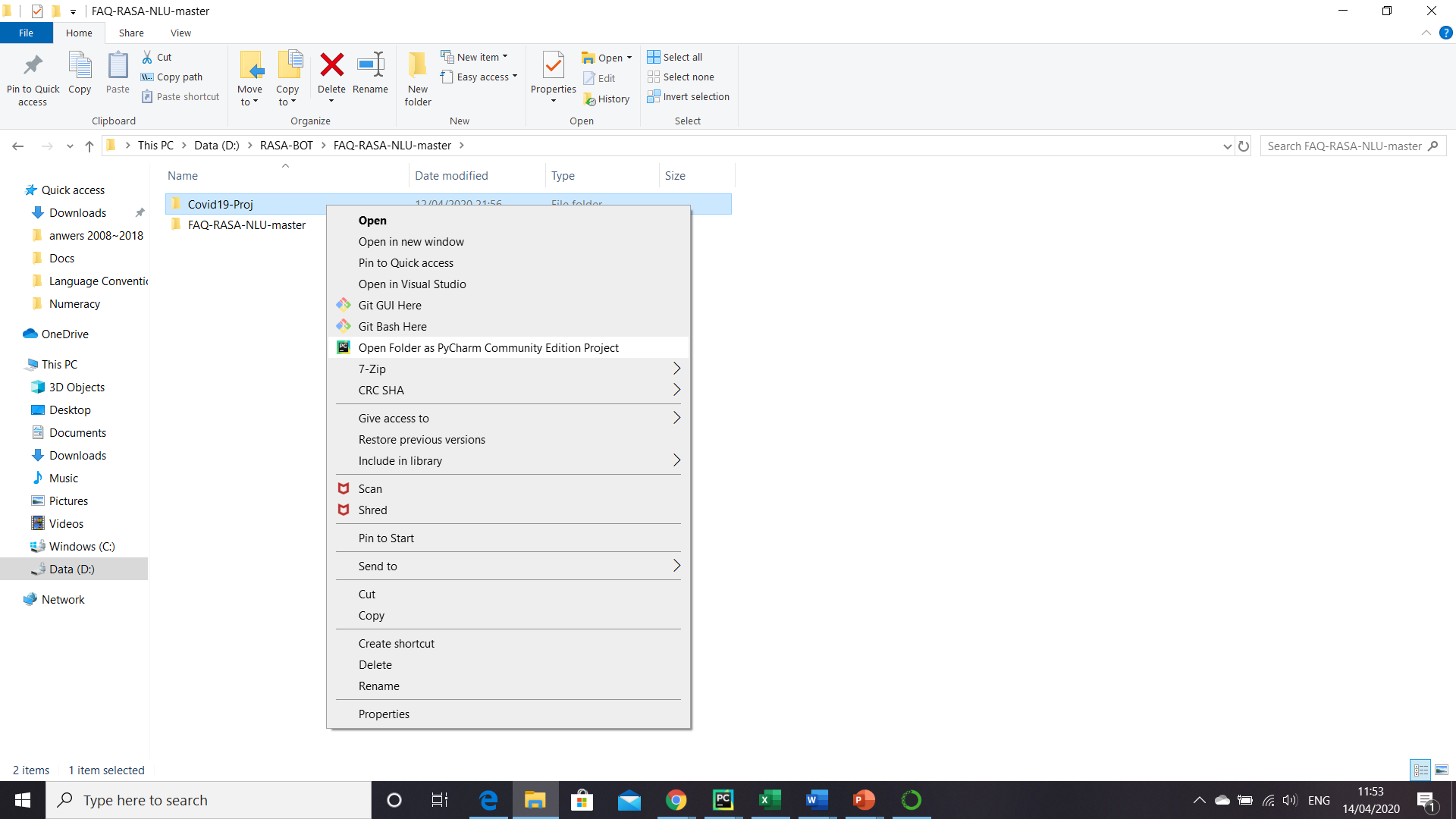
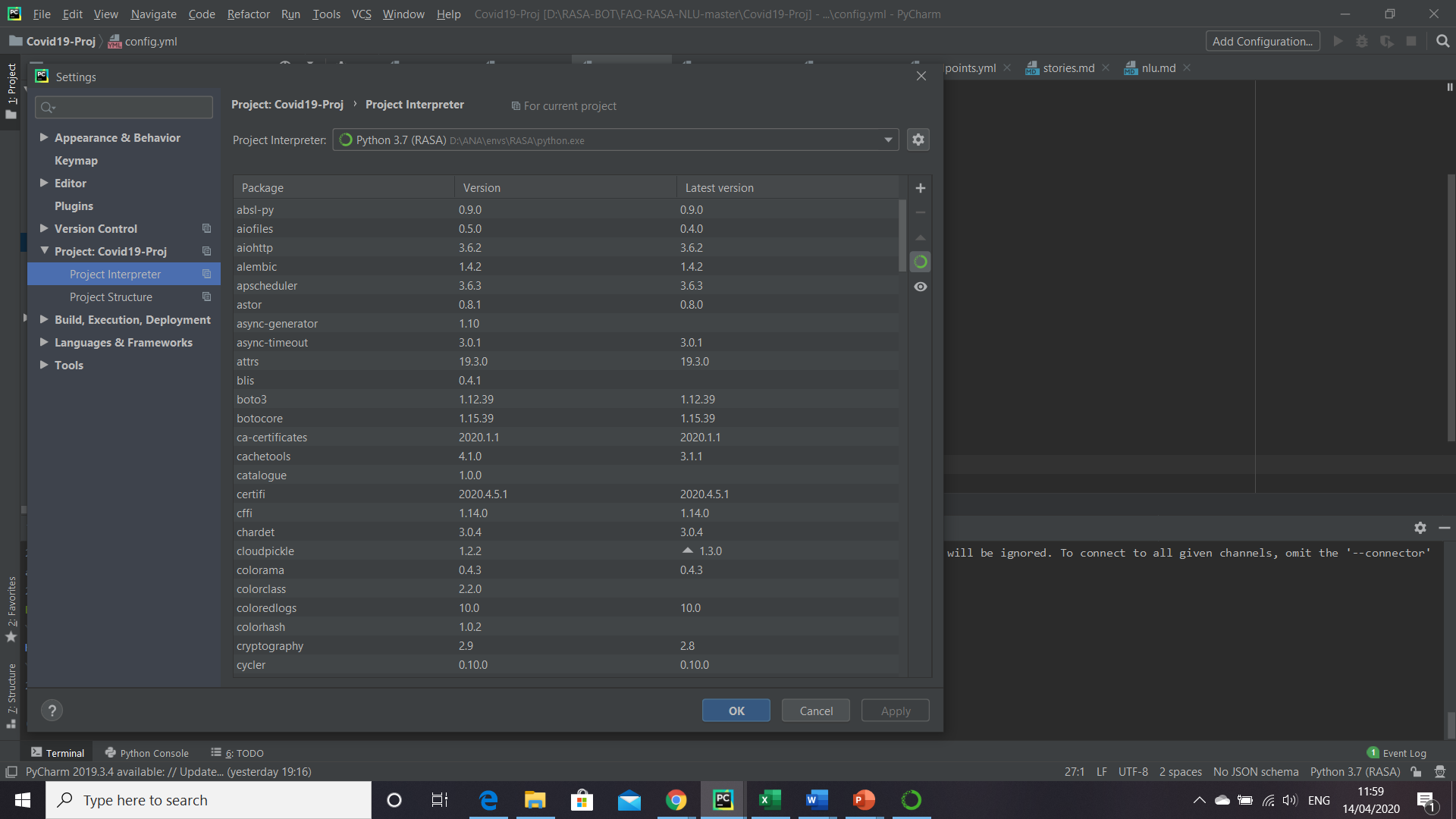
conda activate rasa

Please note – to setup conda refer to the following [anaconda site](https://www.anaconda.com/) which has all the details.

Next step is to install a python IDE where we will carry on with the next steps of RASA setup. My personal favourite is PyCharm. Quick steps to install PyCharm.

1. Go to the [link](https://www.jetbrains.com/pycharm/download/#section=windows and download the community edition.)
2. Double click on the installation file to start the installation process and click next to continue.
3. Select the directory to install PyCharm and then click next
4. Select the appropriate checkboxes as shown below. Don’t miss to check the option “Open Folder as Project”
5. Choose the name of the start menu folder and then click on install to finish the installation.

The next set of steps are installing the RASA framework.

1. Create a new folder – Name it according to the project you are delving with. My project name is “Covid19-Proj”
2. Right Click the folder and open PyCharm editor as shown in the diagram.
3. Once you are in the PyCharm editor follow the path File -> Settings. The settings window will show the name of the folder in the Project. Select the Project Interpreter and select the environment that we have created earlier. A snapshot of the same is provided below. Now your python environment is all set for the RASA framework.
4. Open the terminal below in the PyCharm editor. Fire the following commands to install RASA.

pip install rasa-x (pls note sometimes this might give error. But it will provide the actual RASA link to use in the pip command in the error message. This will install all RASA components and dependencies)

pip install spacy (this will install the spacy library)

Then fire the following commands in this sequence-

python -m spacy download en

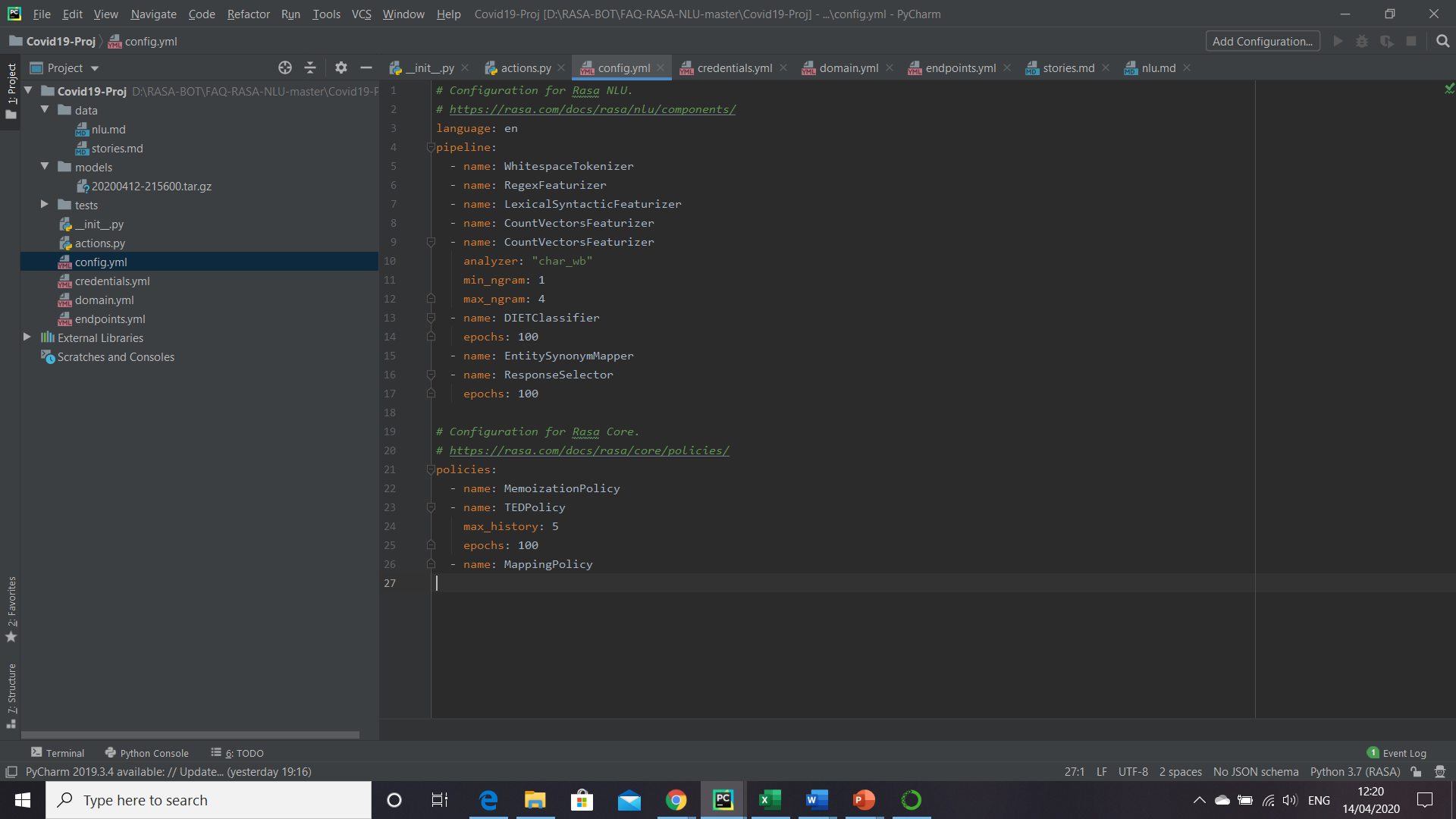
python -m spacy download en\_core\_web\_md

python -m spacy link en\_core\_web\_md en (to fire this run as administrator. It might give error otherwise in windows)

1. Finally execute the command –

rasa init (for all the subsequent actions choose Y)

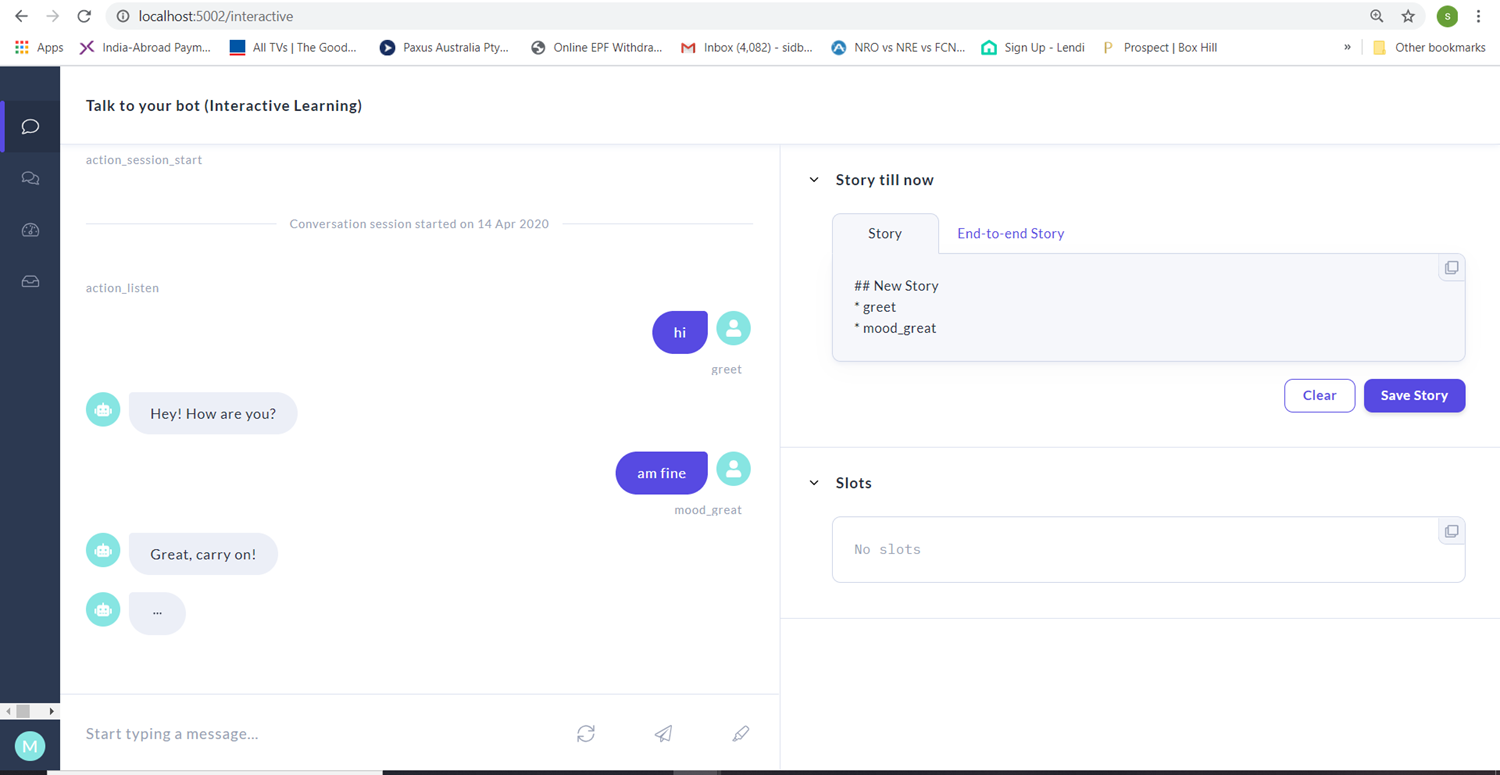


Once the above command is executed successfully, we will end up with the following folder structure inside the project as shown. This also ends up with the installation process. In the next section we will see the functionality of each of the files and how we can configure the same based on our business problem.

RASA comes up with some pre-loaded data, so that the BOT is up & running from your installation. Fire the command –

rasa x (this will open up a webUI of the default BOT)

The below is the 1st view of the BOT. Isn’t it so quick 😊



# Building the BOT

## Problem Statement

Before we start building the BOT it is critical that we define the problem statement. For the current project the aim is to build a chatbot which can answer queries related to the COVID-19 disease. In order to build it is key to come up with the conversational design. The following are the key factors for this design –

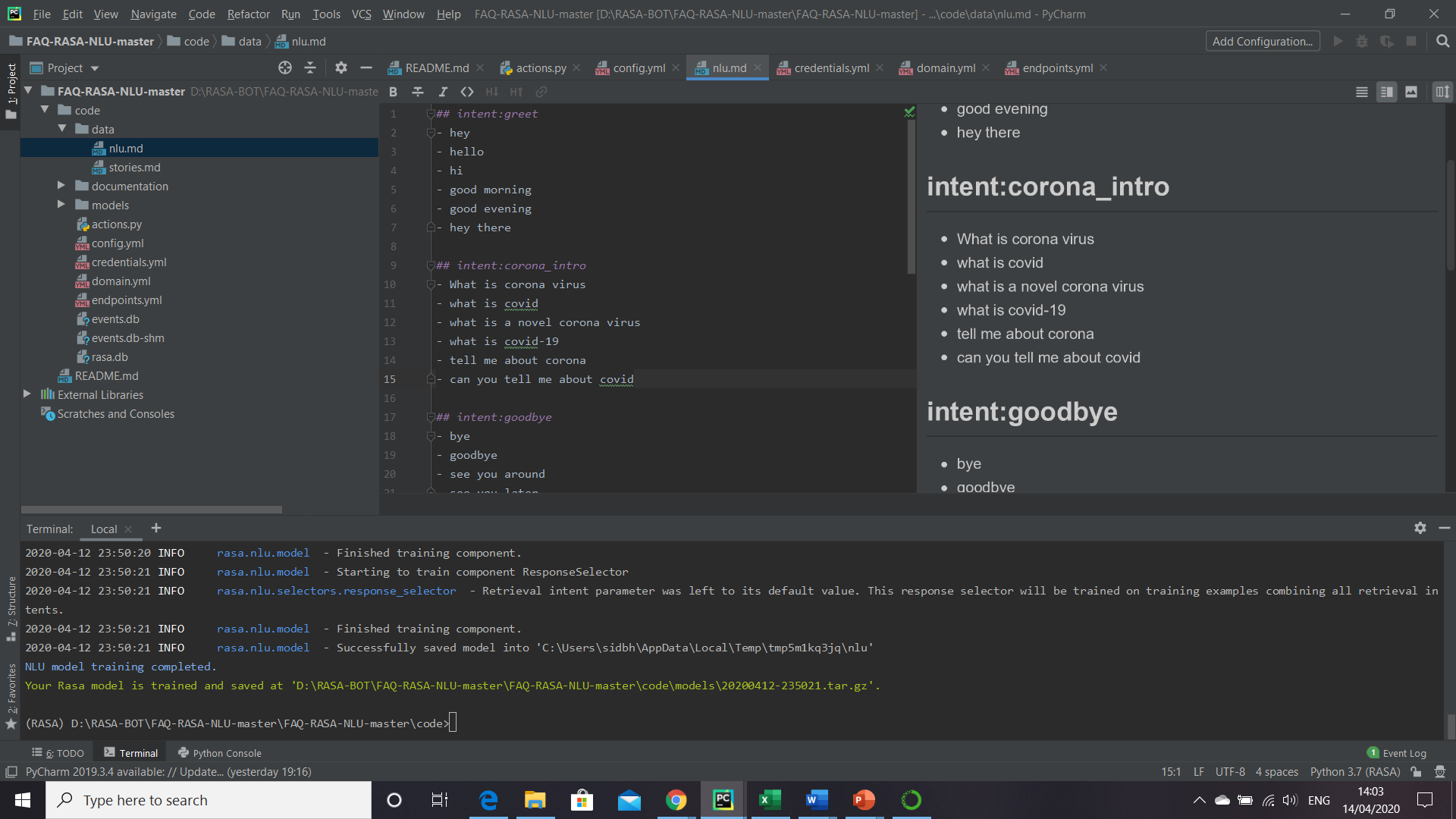
* Who are the users?
* Understand the purpose of the BOT
* Finally coming up the conversational paths that a user will have with the BOT

To come up with the most common set of questions the following sources can be helpful –

* Consulting with a domain expert (in this case a doctor)
* Common queries already being raised in other platforms. A good source will be already any customer service agents managing such requests.

## Prepare Training Data

The step of the conversation will be initiated by the user. The BOT should be in a position to understand the text and response back. Under the data directory there is a file called nlu.md. This file acts as the training data for the BOT. The file includes intents, which are user goals, and example utterances that represent those intents. A snapshot of the nlu.md from our Covid BOT.

Intents are defined using a double hashtag. Each intent is followed by multiple examples of how a user might express that intent. As depicted, we have an intent called ‘greet’. So, whenever user says ‘hey’, BOT will able to co-relate it with intent ‘greet’. The same applies for ‘corona\_intro’.

Quick Tips

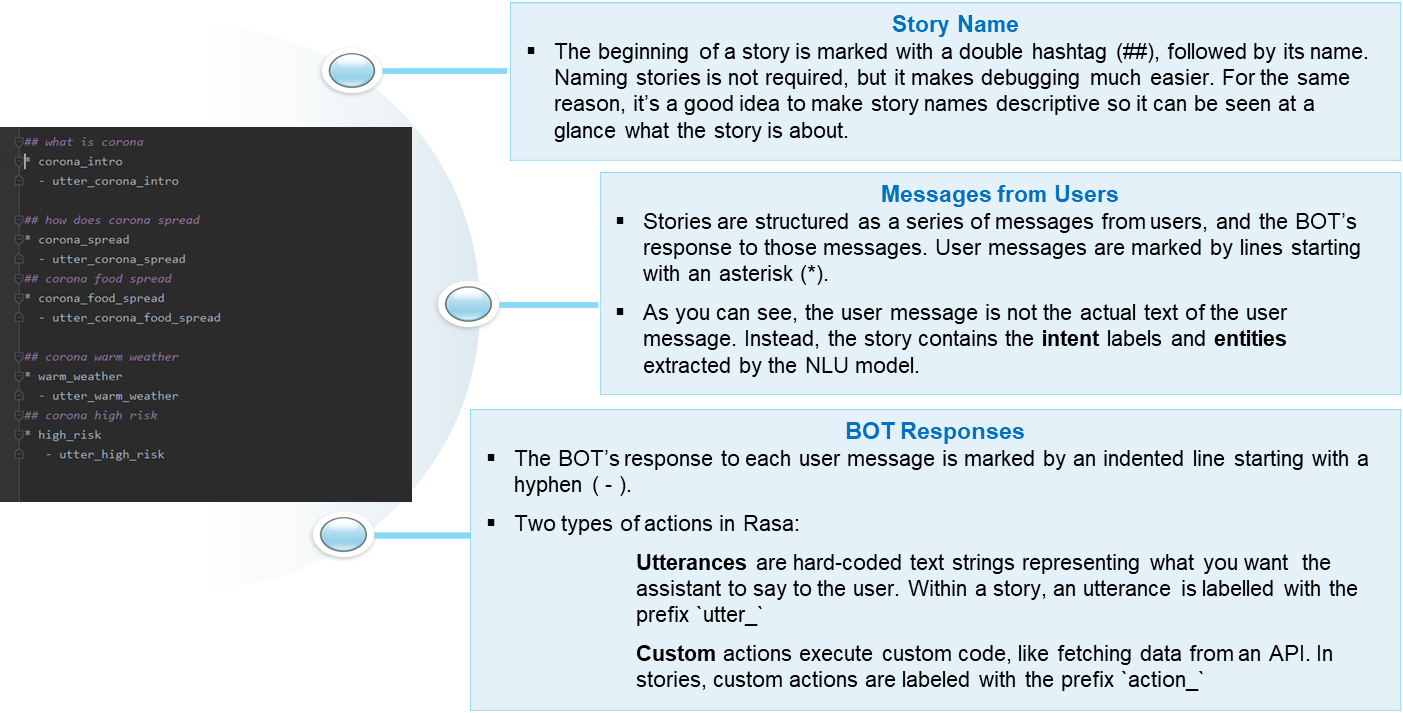
Will be good if we provide 10-15 examples for each intent. No need to provide all possible utterances

Utterances should be relevant to the intent. Ensure diversity in the vocabulary to make a robust BOT

Attaching the full list of intents created for this project.



The next logical step is to identify the stories. Stories are basically short scripts that details the conversation path between user and BOT. This is another entity that will be fed for training purpose. The file to edit is stories.md under data directory. A snapshot is provided for the Covid project. The following are the key features of defining a story.



Attaching the full list of stories created for this project.



Point to be noted - At the end of the story, we need to create a new line and begin the next story with a double hashtag.

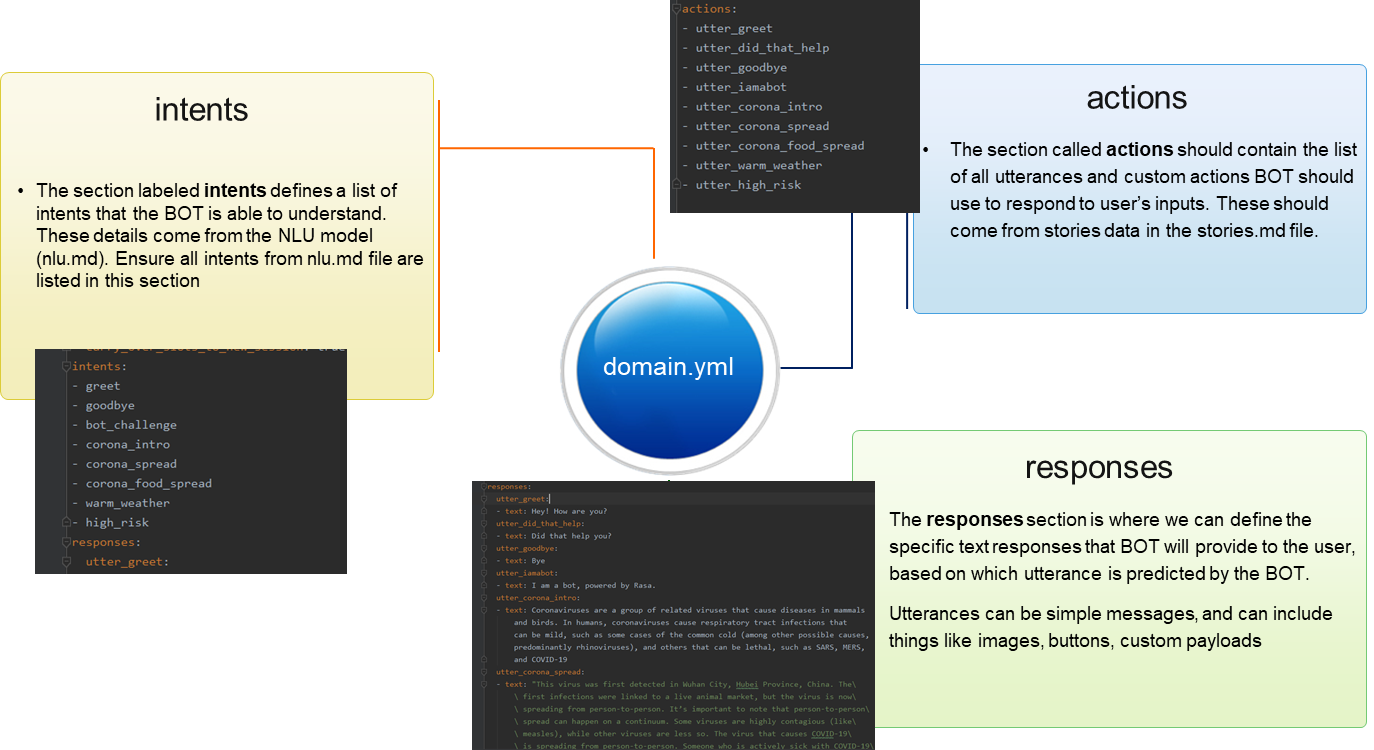
Quick Tips

The more the no of stories the better. We need to provide diverse examples so that BOT can learn and generalize and potentially manage unseen inputs

It is always preferred to get inputs from real users rather cooking up paths by oneself. So once the base model is ready, it is always recommended to test it by actual users so that we can figure out the stories that are not managed by the BOT

## Prepare the Response

Once we have prepared the training data it is time to build the response. For this step we need to edit the ‘domain.yml’ file. This file will have the actual responses that BOT will provide to the user. The following depicts the mandatory sections of the file.



Attaching the domain.yml created for this project.

Apart from the above components RASA allows the following components as well –

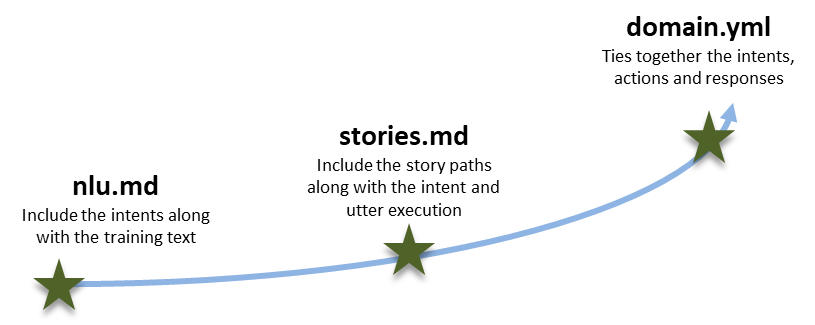
* Entities
* Custom actions
* Slots

These are not used in the Covid project & hence not elaborated. Refer to RASA docs on usage.

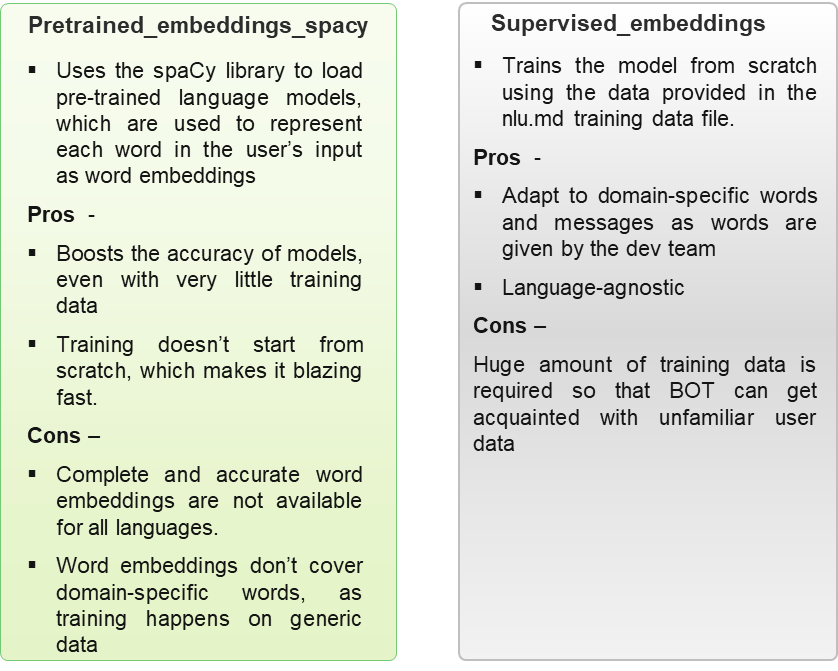


## Quick Recap

Thought of doing a quick recap of what all steps we have performed till now. Highlighting the salient steps only –



## Training the BOT

It’s time to train the BOT as data preparation is done. RASA comes with two pre-configured training pipelines. It is flexible to add any custom training pipeline as well. Let’s understand the pre-configured ones at a high level. Am not delving in the details of the algorithm as that is beyond the scope of this document. The file config.yml provides a view of the machine learning pipeline and policies that will be used to train the BOT with the data. The below provides some of the characteristics of the pre-defined pipelines.

For our project we have used

Supervised\_embeddings

Command to fire –

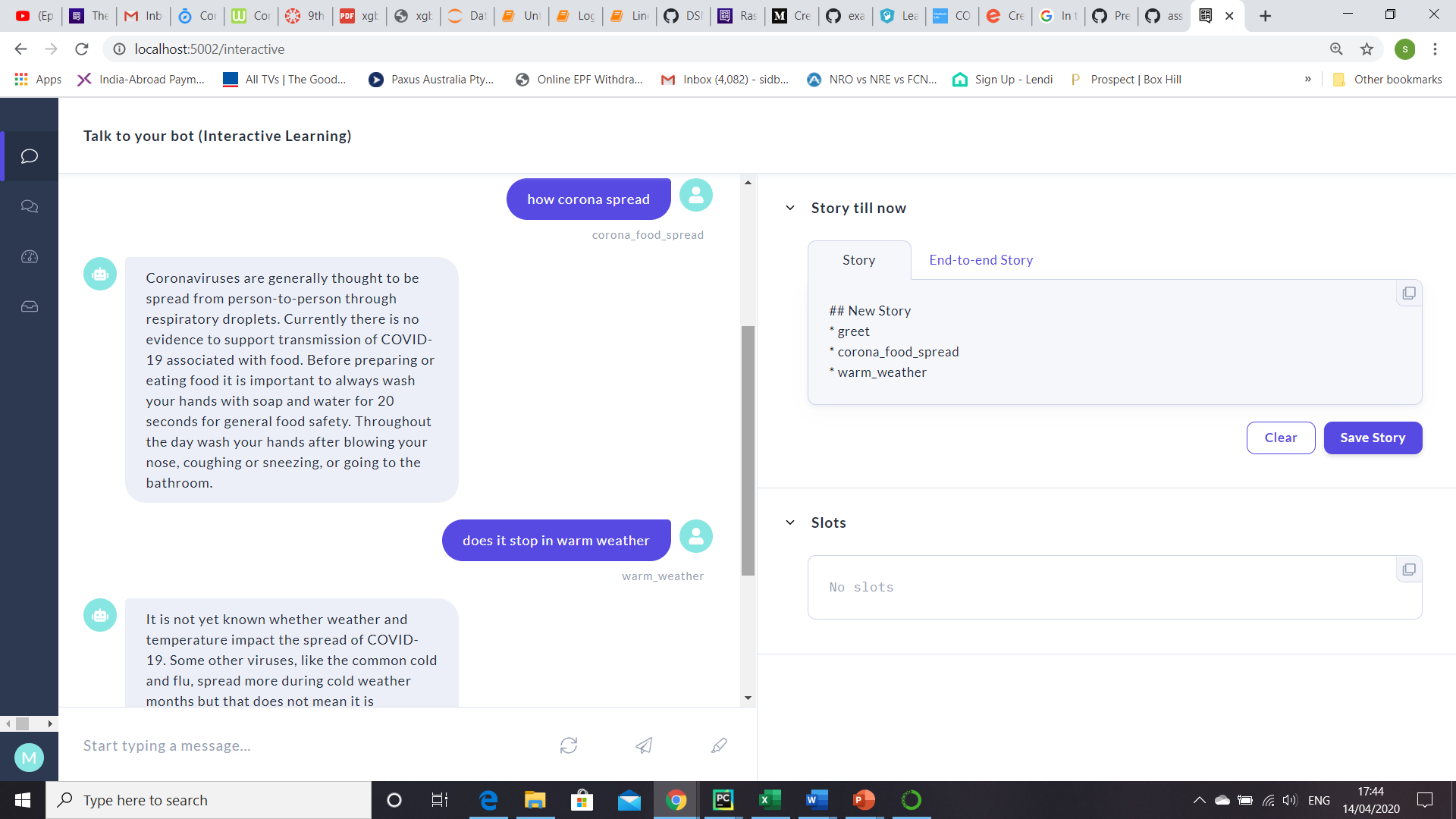
rasa train

Attaching the config.yml created for this project



## Test the BOT

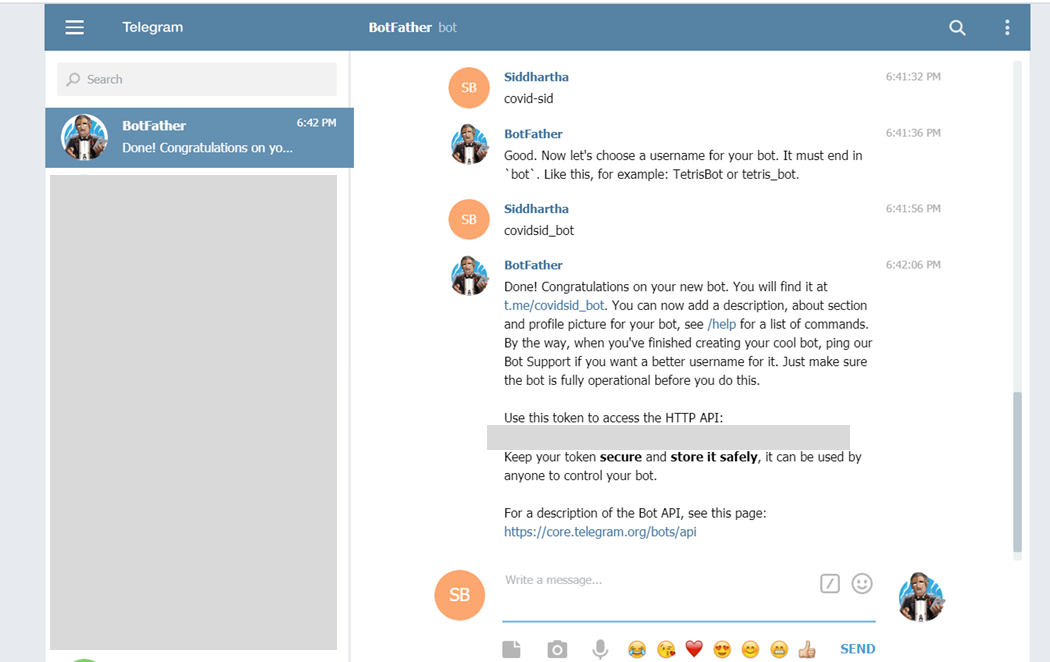
Now it is time to check how the trained BOT is responding. To open the BOT execute rasa x. It will provide the localhost URL. PFB a screenshot of the BOT on some user questions. Seems it is working. If you notice on the right-hand side it is showing which intent BOT is able to recognize. Accordingly, BOT is responding based on domain.yml configuration. You can experiment further on variety of questions and check the areas where BOT is failing to respond properly or out of context.



# Integration with 3rd party channels

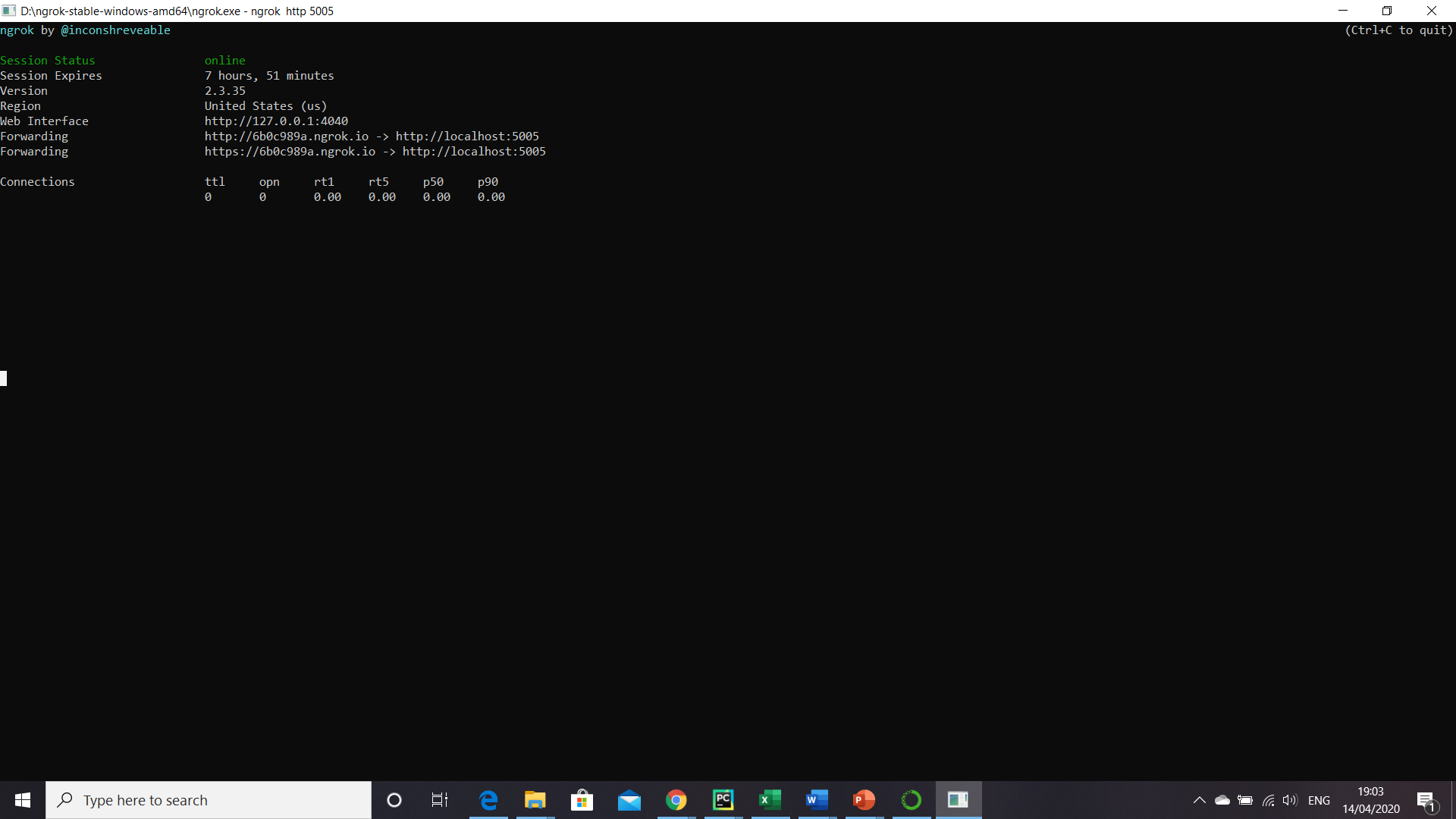
One of the key features of any chatbot application is how easy to integrate the chatbot application with other messaging platforms. Typically, in production, a chatbot will be not a standalone application but will be plugged along with another channel like company website etc. Rasa comes pre-configured to connect with popular messaging channels like Slack, Facebook Messenger, Telegram, and many more. For this project we have integrated the BOT with Telegram. The following are the steps –

1. Login to Telegram web. Search for BotFather. This helps developers to register a new bot in Telegram application. Provide the command /newbot. Post that it will ask name & username of the BOT. Username must end with bot. When it is done it will provide a token. This will be utilized in step 3. A snapshot of Telegram is provided below.



1. Download ngrok from <https://ngrok.com/download>. After extracting the zip file, open the ngrok file and run it. In ngrok, enter the command ‘ngrok http 5005 ’:

The following is a screenshot of the ngrok command.



Forwarding URL will be used as ngrokurl in the step 3.

1. Finally edit the file ‘credentials.yml’.

telegram:

access\_token: "obtained from telegram"

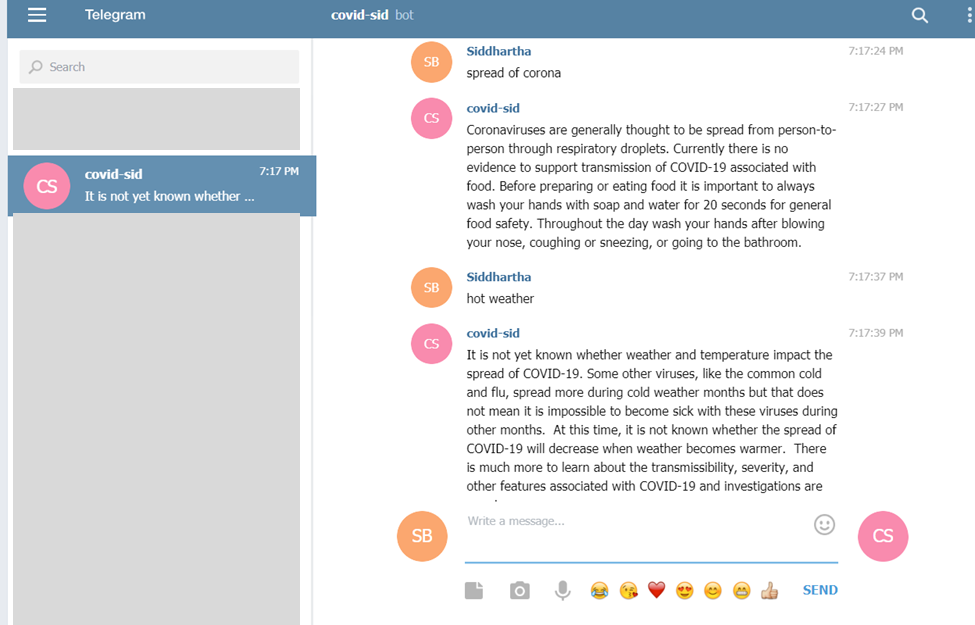
verify: "your bot username"

webhook\_url: https://<ngrokurl>/webhooks/telegram/webhook

1. Go to terminal and enter the command ‘rasa run’
2. Open one more terminal and run the command ‘rasa run actions’

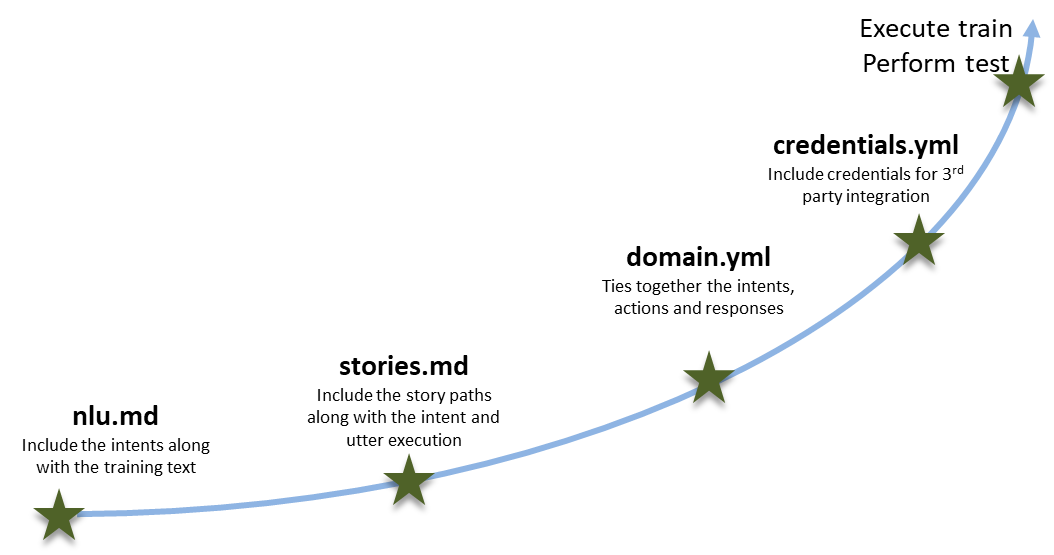
This concludes the integration with Telegram.

PFB the screenshot of the BOT from my Telegram. Hurray finally able to check my BOT.



# Wrapping it up

Finally, it is time to wrap up the document. We have covered a long stretch starting from describing a bot, the key benefits of a BOT and then deep diving in the RASA framework. Providing a view of the key steps of setting up the BOT again for quick recap.



Steps followed for Covid Project

Hope you have enjoyed reading about chatbot & RASA. Enjoy building bots. Signing off.

# References

<https://rasa.com/>

[https://blog.rasa.com/](https://blog.rasa.com/the-rasa-masterclass-handbook-episode-8/)

<https://www.cdc.gov/>