

Text Mining Lab

Training Rasa-Chatbots with Text

Project Report

David Fuhry
Leonard Haas
Lukas Gehrke
Lucas Schons
Jonas Wolff

Winter Term 2018/2019

Contents

1	Project Description	2
1.1	Conversational AI and Training	2
1.2	Rasa Framework	2
1.3	Research Question	2
1.4	Project Goals	2
2	Data Processing	3
2.1	R Package 'wikiproc'	3
2.2	Data Acquisition	3
2.3	Fact Extraction	3
3	Chatbot Architecture	4
4	Results	5

1 Project Description

1.1 Conversational AI and Training

Conversational AI describes computer systems that users can interact with by having a conversation. One important goal is to make the conversation seem as natural as possible. Ideally, an interaction with the bot should be indistinguishable from one with a human. This can make communication with a computer become very pleasant and easy for humans as they are simply using their natural language.

Conversational AI can be used in Voice Assistants that communicate through spoken words or through chatbots that imitate a human by sending text messages.

1.2 Rasa Framework

Rasa is a collection of tools for conversational AI software. The *Rasa Stack* consists of two open source libraries called *Rasa NLU* and *Rasa Core* that can be used to create contextual chatbots.

A Rasa Bot needs training data to work properly.

1.3 Research Question

The objective of this project is to find out, whether chatbots can be trained with natural language texts *automatically*. There are two initial research questions:

- Can these facts be extracted from natural language text?
- Can this be done automatically?

1.4 Project Goals

In regard to the given research questions, this project aims at implementing procedures to extract information from natural language text and make that information accessible to a chatbot.

1. Define possible intents fitting the given domain.
2. Configure a chatbot that recognizes these intents and linked entities.
3. Acquisition of data and implementation of processing that extracts required information.
4. The chatbot is given access to the extracted data to create answers to given entities and intents.

Development of the bot is focused on proof of concept instead of production ready conversation flow. Therefore the natural conversation abilities of the bot will be limited.

2 Data Processing

2.1 R Package 'wikiproc'

All functionality to extract facts, download data from wikipedia as well as some utility functions is encapsulated inside the *wikiproc* R package. This allows for a better management of dependencies as well as inclusion of unit tests for fact extraction methods.

Function	Category
<code>clean_html</code>	Utility
<code>create_annotations</code>	Utility
<code>init_nlp</code>	Utility
<code>get_data</code>	Data scraping
<code>get_awards</code>	Fact extraction
<code>get_birthdate</code>	Fact extraction
<code>get_birthplace</code>	Fact extraction
<code>get_spouse</code>	Fact extraction
<code>get_university</code>	Fact extraction

Table 1: Exported functions of the *wikiproc* package

2.2 Data Acquisition

Wikipedia was chosen as resource as it provides texts of relatively long length in a somewhat uniform manner. While Wikipedia does have a *Physicists* category¹, it is fragmented into somewhat arbitrary subcategories and thus not optimal to use as a collection. However Wikipedia also has a *List of physicists*² which contains 981 physicists and was used to build the collection used.

Data scraping was done using the R Package *WikipediR*, a wrapper around the Wikipedia API. Articles were downloaded as HTML³ and afterwards stripped of all HTML tags and quotation marks.

2.3 Fact Extraction

Fact extraction approaches greatly vary depending on the nature of the fact to extract. As all approaches leverage on some form of NER or POS tagging, annotations were created for all texts. This was done using the R Package *cleanNLP* with a spaCy backend to create NER and POS tags, as well as lemmatization. Fact extraction for physicists spouses was done using pre-defined patterns on word lemmata.⁴ A pattern consists of word lemmata to be matched (including wildcards) as well as defined places to look for the name of the physicist and his/her spouse. When a matching phrase is found the results are verified by checking that the correct physicist is mentioned as well as the potential spouse being detected as a person by the NER tagger. A different approach is used for

¹<https://en.wikipedia.org/wiki/Category:Physicists>

²https://en.wikipedia.org/wiki/List_of_physicists

³HTML was chosen over wikitext to ease text cleaning

⁴Functionality to use patterns on POS tags is also available but did not yield a better outcome.

the `get_awards()` function. The approach is based on the assumption that the NER tagger will tag the awards as some kind of entity. A set of keywords is the used to extract all entities of interest, the awards.

3 Chatbot Architecture

The chatbot built for this project uses both Rasa Stack components - *Rasa Core* and *Rasa NLU*. The *Rasa NLU* component takes care of getting user input and matching it with the respective intents. The *Rasa-Core* component executes all actions associated with the determined intent. Configuration has been organized in reference to examples from the Rasa github repository⁵.

Rasa NLU has been trained with example questions in markdown format that contain highlighted entities. This ensures that the bot is able to understand intents and extract the entities inside the sentences. One example can be seen in listing ??.

```
9      ## intent:nationality
10     - what nation is [Albert Einstein](physicist) from
11     - what nationality does [Albert Einstein](physicist) have
12     - where is [Galileo Galilei](physicist) from
```

Rasa Core has been configured with *stories* that contain example conversation flows as training data (listing ??) and the *domain* of the bot. The domain contains all actions, entities, slots, intents, and templates the bot deals with. *Templates* are pattern strings for bot utterances. *Slots* are variables that can hold different values. The bot proposed in this project uses a slot to store the name of a recognized physicist entity. According to the Rasa website⁶, the domain is *the universe the bot is living in*.

```
13     ## nationality
14     * nationality{"physicist": "albert einstein"}
15     - action_search_nationality
16     - action_utter_nationality
```

The bot recognizes the intents shown in table 2. It can be started by issuing *MAKE*-commands. For further details, refer to the README⁷.

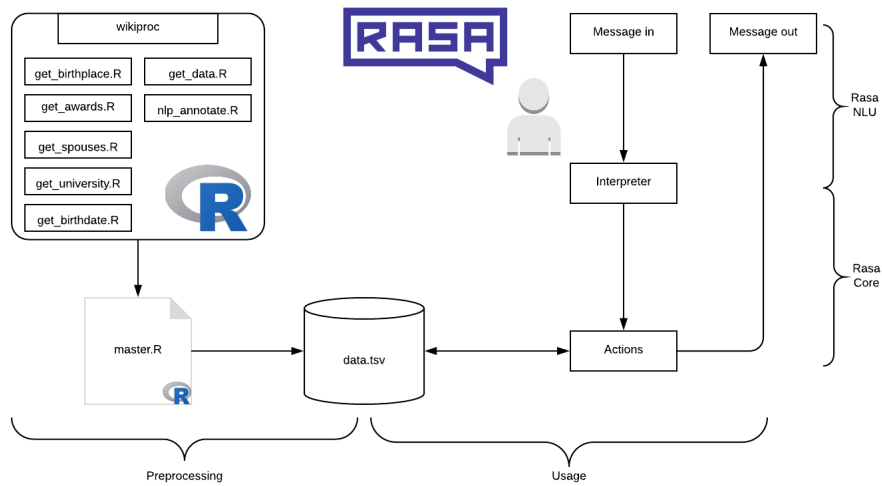
⁵https://github.com/RasaHQ/rasa_core/tree/master/examples

⁶https://rasa.com/docs/get_started_step2/

⁷<https://git.informatik.uni-leipzig.de/text-mining-chatbot/wiki-rasa/blob/master/README.md>

No	Intent	Example
1	birthdate	When was Albert Einstein born
2	nationality	Where was Albert Einstein born
3	day of death	When did Albert Einstein die
4	place of death	Where did Albert Einstein die
5	is alive	Is Albert Einstein still alive
6	spouse	Who was Albert Einstein married to
7	primary education	Where did Albert Einstein go to school
8	university	Which university did Albert Einstein attend
9	area of research	What was Albert Einstein area of research
10	workplace	Where did Albert Einstein work
11	awards	What awards did Albert Einstein win

Table 2: Intents that are recognized by the bot



4 Results

Evaluating the Rasa framework we find an ambivalent result. On the one hand, in the beginning of the project the setup and configuration of the bot led to considerable problems because of outdated documentation. Therefore a lot of time had to be spent on trial-and-error procedures to understand the functionality of the framework. On the other hand, the NLU functionality of the Rasa stack has a high precision in recognizing intents expressed in the input, even far beyond the provided training examples. It was possible to configure the bot to meet our needs without any restrictions.

Wikipedia articles are particularly well suited for the process of information extraction, because they generally are composed consistently. The different levels of detail and therefore information were an issue when dealing in using these articles.

Concluding the textmining part of our project we can assess that the func-

tions using mainly NER tags (`get_awards.R` and `get_university.R`) have high recall and relatively low precision. The function `get_spouses.R`, which is working with pattern matching, has low recall and high precision. It needs to be emphasized that the quality the results is strongly dependent by the provided data.

In regards to answering our research question: We were able to extract facts about pre-determined intents from text and make them available to the bot. The next logical step, which we were unable to address in the context of this project, would be generating intents automatically from text.