Natural Language Interface to DataTable

Previous post
Next post

Tags: AI, Chatbot, Natural Language Processing, NLP

You have to write SQL queries to query data from a relational database. Sometimes, you even have to write complex queries to do that. Won't it be amazing if you could use a chatbot to retrieve data from a database using simple English? That's what this tutorial is all about.

---

comments

By Yogesh Kulkarni, Yati.io

A good amount of structured data is stored in the form of relational databases as tables. Considering just one table as a simplified case, even to access the values in various cells, one needs to either program the logic or open it in a spreadsheet software and find them manually. Won't it be better if one can just query the fields using a natural language, like English.

For example, if you have table like in **Figure 1**, and it would be nicer to just send a query "Which City held Olympics in 2008?" in a chatbot and you would get the answer as, "Beijing". This is far more user-friendly than writing a program or manually searching a huge table.

Year	City	Country	Nations
1896	Athens	Greece	14
1900	Paris	France	24
1904	St. Louis	USA	12
...	...	...	...
2004	Athens	Greece	201
2008	Beijing	China	204
2012	London	UK	204

x = Greece held its last Summer Olympics in which year?

y = 2004

Figure 1: Wiki Table Questions (Source: Stanford NLP Group Research Blog)

The example given above is surely a very simplistic version of the enormous possibilities one can imagine for providing natural language interface to the datasets or databases. Natural Language Understanding (NLU) is the key technology here, which would parse the query in the natural language like English, 'understand' it, and then fire data-format-specific query to fetch the desired answer. This article demonstrates how this can be achieved using Rasa NLU framework.

Rasa is an open source chatbot framework. It has two parts, 'NLU' for understanding intents and entities from the user-text whereas 'Core' is for dialog management. One of the salient points of Rasa framework is that its functionalities are machine/deep learning based and are trained locally. There is no need to send your data (ie user text) over the net to some hosted service for NLU or dialog management.

Example Chatbot

Say, you wish to build a General Knowledge chatbot for querying information of various countries in the world. Queries like “What’s the population of China?”, “What is the area of India?”, etc.

The World FactBook is a wonderful source of such data about the countries. Information is arranged in the form of country-wise-webpages, having sections like “Geography”, “Economy”, etc. To find the population of China, one needs to first scroll to China in the drop-down list, find “People and Society” section and then you get to see that the population of China, which is “1,384,688,986 (July 2018 est.)”. A chatbot would be far more effective in such cases.

Data Collection

First important step is to get the data, which is spread across various webpages, into a single table. This KDNuggets Article explains the process of web scraping, nicely. The output is a table with rows as countries and various fields as columns.

Now we need to build a natural language interface to fields or cells of this table. For query like “What’s the population of China?”, we need to fetch value in the cell which is at intersection of a row for “Country==China” and a column named “Population”. To put simply, from the natural language query, we need to extract two entities, “Population as Column” and “China as Row” (in the column “Country”). Rasa NLU is being used for the entity extraction here.

Entity Extraction

Entity extraction is a process of finding words/tokens in the sentence mapping to pre-defined entity types. Named Entity Recognition (NER) is the supervised method to extract the entities, using algorithms such as Conditional Random Fields (CRF), etc.

To train RASA NLU’s NER, we need to provide training data which has sample sentences with entities annotated. Sample training file is shown in **Figure 2**.

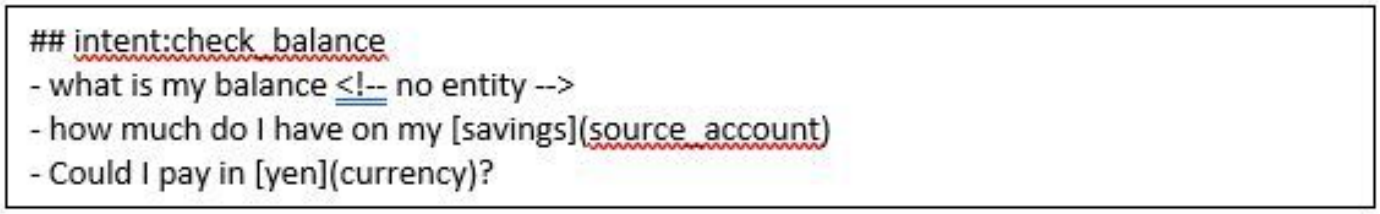


Figure 2: RASA NLU training file in Markdown format (Source: Rasa NLU docs)

Here “savings” and “yen” are the actual values of the entity types “source account” and “currency” respectively.

Building such training data for the given table, would again be a manual and thus a tedious process. The annotation process can be automated by iterating over rows and columns of the table . There is no need to provide examples of all the columns or rows, as the abstract pattern is learnt and not the hardcoded actual values in the cells.

Code sample for generation of training file can be seen below:

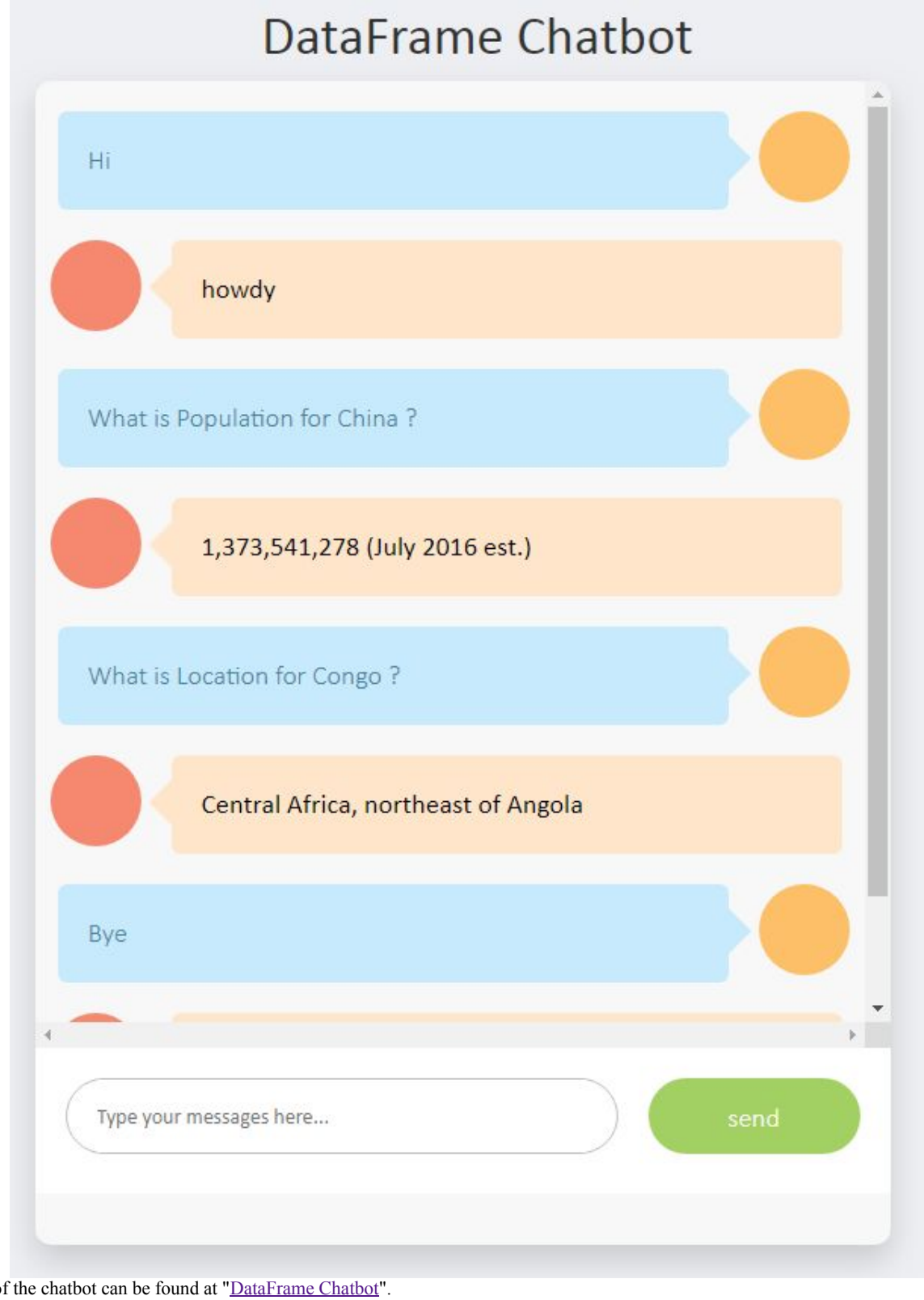
```
        n_sample_countries = 5
        n_sample_columns = 5
        self.all_columns = [col for col in list(self.df.columns) if col != self.primarycolumnname]
        sample_sentences = []
        for c in self.primary_key_values_list[:n_sample_countries]:
            for col in self.all_columns[:n_sample_columns]:
                sentenceformat1 = "What is [" + col + "](column) for [" + c + "](row) ?"
                sentenceformat2 = "For  [" + c + "](row), what is the [" + col + "](column) ?"
                sample_sentences.append(sentenceformat1)
                sample_sentences.append(sentenceformat2)
```

Although only two sentence formats are shown above, one can add more variations. The automatically generated NLU training file looks like:

```
## intent:query
- What is [Area](column) for [Afghanistan](row) ?
- For  [Afghanistan](row), what is the [Area](column) ?
- What is [Background](column) for [Akrotiri](row) ?
- For  [Akrotiri](row), what is the [Background](column) ?
- What is [Location](column) for [Akrotiri](row) ?
- For  [Akrotiri](row), what is the [Location](column) ?
- What is [Map references](column) for [Akrotiri](row) ?
- For  [Akrotiri](row), what is the [Map references](column) ?
:
:
```

Using the training data, NER model is trained. When a user query comes, the model is then able to extract “row” and “column” names. With both, one can locate the value in the given table. The value is returned to the user.

Sample conversation workflow is shown below:



Source code of the chatbot can be found at "DataFrame Chatbot".

Limitations and Future Scope

The chatbot example shown is very simplistic. It can be enhanced further with following capabilities:

- Queries across multiple tables.
- Queries with aggregations and relations, like “Which countries have more than 100 Million population and GDP per capita less than $1000?”.
- Partial/Full SQL support ie to convert natural language query into an equivalent SQL query.

Conclusion

This article demonstrates building of a chatbot to interface with a datatable in a user-friendly manner. It has varied applications for the domains where such tabular data is available.

Bio: Yogesh Kulkarni is a Data Science Instructor-Researcher at Yati.io.

Related:

- Build Your First Chatbot Using Python & NLTK
- TNLP – Building a Question Answering Model
- Using Deep Learning To Extract Knowledge From Job Descriptions