

Semantic Web Project

Extract jena rules from dbpedia to predict numerical properties

Sergio Enrico Coppolecchia | 384188 | 2017/2018

# The problem

Often, data extracted from Wikipedia are incomplete. Some property can be wrong, because sometimes automatic methods that extract automatically data may not work very well or in some case data are not available. So, on DBpedia there are many missing data.

## the solution

Use machine learning on present features to predict some missing property. In this case study I have used regression trees trained on numerical data extracted from dbpedia to predict numerical value. Then, I have converted the regression tree rules in Jena rules to reason about triples that haven’t the target property.

# ARCHITECTURE OVERVIEW

SPARQLWrapper

DBpedia

Data without target

NEW KNOWLEDGE

APACHE JENA

JENA Rules

CONVERTER

SCIKIT-LEARN

The system is composed by 2 principal software:

1. The first software in Python used to create the regression tree and then to convert the regression tree into Jena rules
2. The second software written in Java used to reason on data without the target property.

## Architecture components

### SPARQLWrapper

I used SPARQLWrapper library to query DBpedia and extract both data to the learning and reasoning. SPARQLWrapper is an endpoint interface for Python. This is a wrapper around SPARQL service. It helps in creating the query URI and convert the result into a more manageable format.

In order to let scikit-learn processable data, triples are transformed into a matrix where the column are the features(properties) and the rows corresponds to a sample (Subjects), so the values of the cells are the Objects of the triples.

While the target values are represented in a vector.

While the Subjects that haven’t the target value, are stored in a file in order to being used during the Jena reasoning.

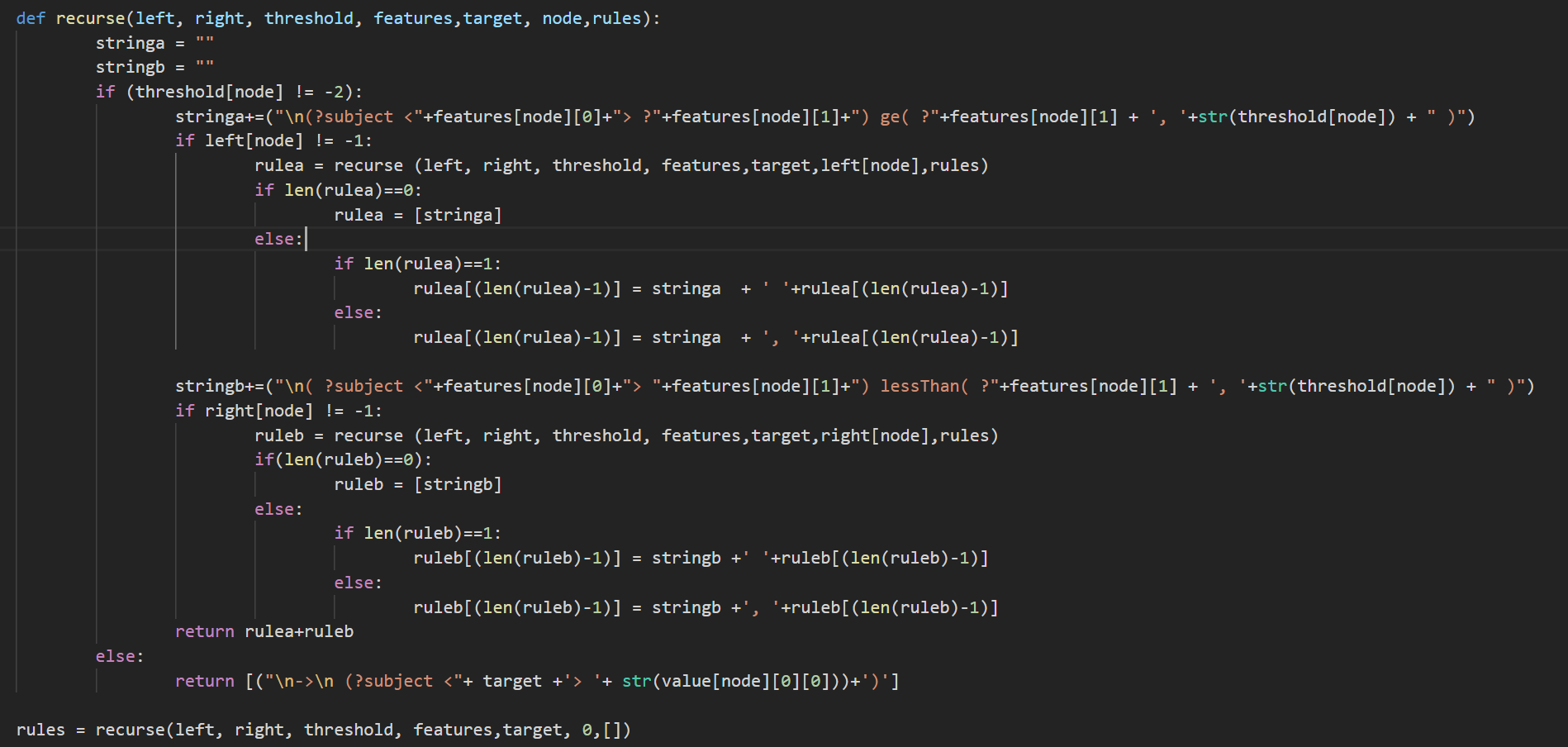
### SCIKIT-LEARN

Simple and efficient tools for data mining and data analysis. I used this library to train a regression tree and to evaluate it. A regression tree is a tree where every node is a test on a feature of the dataset and every edge is a possible value or interval of that feature. The leaf nodes are the target value predicted.

To evaluate the model I have split the dataset into training (85% of data) and test set(15% of data).

### CONVERTER

I have implemented a function that converts the regression tree learned into Jena rules.



The algorithm is recursive and at each call, it checks if its is a leaf node. If this is true returns the conclusion of the rule with the triple constructed with the predicted value, else the recursive function is called both the right and the left edge, adding the two values of the test like clause of the rule to the result of the recursive call.

Then the result rules are written into a file.

### APACHE JENA

Apache Jena is a free and open source Java framework for building Semantic Web and Linked Data applications. I used its reasoning component to add new knowledge to a knowledge graph. By the rules generated by Python module, the target values are inferenced.

So, the component simply reads data with the missing target value stored previously in the first SPARQL query and the rules learned and reasoning on this data. Then all the new data are stored in a file.