INF344 2019–2020

Tableau de bord / Mes cours / INF344 2019–2020 / Représentation des connaissances, Extraction d'instances / Type Extraction / Description

Description Devoir rendu Edit Submission view

Type Extraction

Due date: Monday 1 June 2020, 23:59
Requested files: extractor.py, page.py, parser.py (Download)
Nombre maximal de fichiers: 20
Type of work: Individual work
Reduction by automatic evaluation: 10 **Free evaluations**: 4

Purpose

The goal of this lab is to extract the class to which an entity belongs from Wikipedia. For example, given the Wikipedia article about Leicester:

```
Leicester is a small city in England
```

the goal is to extract:

```
Leicester TAB city
```

Prerequisites

We provide

- a preprocessed version of the Simple Wikipedia, which looks like above.
- a template for your code (in the Moodle). It reads the Wikipedia file line by line and runs the function extractType() on each article. It prints a tuple, in which the components are separated by "\t" (tabulator).
- a gold standard sample for the task.

Task

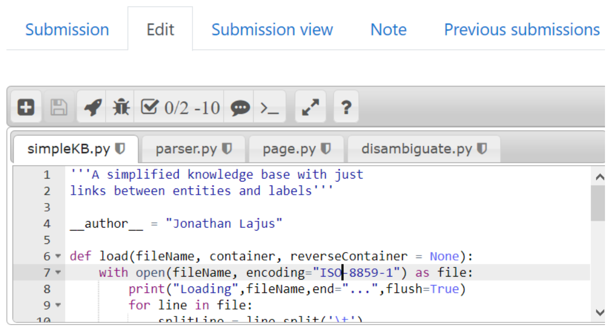
Complete the extractType(content) function so that it extracts the type of the article entity from the content. For example, for a content of "Leicester is a beautiful English city in the UK", it should return "city".

Exclude terms that are too abstract ("member of...", "way of..."), and try to extract exactly the noun(s). You can also skip articles (e.g. return None) if you are not sure or if the text does not contain any type.

The lab will be graded by a variant of the F1 score that gives higher weight to precision (with beta=0.5). You can approximate your precision by looking manually at the first 20 output triples or by comparing to the gold standard sample.

Code

This lab can be programmed, run and evaluated, and graded directly in the Moodle.



```
'''A simplified knowledge base with just
links between entities and labels'''

__author__ = "Jonathan Lajus"

def load(fileName, container, reverseContainer = None):
    with open(fileName, encoding="ISO-8859-1") as file:
        print("Loading",fileName,end="...",flush=True)
        for line in file:
```

In the “Edit” tab, you will find the file extractor.py you have to fill. You can directly modify the code in the Web interface. Click the disk symbol to save. Click the rocket to run it. Click the bug to debug it.

If you want to work offline: You can download the code file, modify it, and then upload it (in the tab “Submission”). The Web interface already knows Wikipedia, so you do not need to upload it.

You can (but don't have to) use NLTK. You can also use spacy with the following model:

```
import spacy
nlp = spacy.load("en_core_web_sm")
```

Please do not use other non-standard libraries.

Submitting the lab

When you are done, you can submit the lab (click the checkbox). This will directly compute an estimated grade on our gold standard. You can submit 2 times. If you submit more often, you lose points.

Plagiarism is sanctioned with a grade of 0/20 (plus potentially other sanctions).

NOTE: The submission server only works inside Télécom. Please visit https://www.telecom-paristech.fr/vivre-ecole/services-numeriques-dsi/connexion-depuis-lexterieur.html to know how to connect to the internal network from the outside.

Requested files

extractor.py

```python
'''Extracts type facts from a wikipedia file
usage: extractor.py wikipedia.txt output.txt

Every line of output.txt contains a fact of the form
    <title> TAB <type>
where <title> is the title of the Wikipedia page, and
<type> is a simple noun (excluding abstract types like
sort, kind, part, form, type, number, ...).

Note: the formatting of the output is already taken care of
by our template, you just have to complete the function
extractType below.

If you do not know the type of an entity, skip the article.
(Public skeleton code)'''

from parser import Parser
import sys
import re

if len(sys.argv) != 3:
    print(__doc__)
    sys.exit(-1)

def extractType(content):
    # Code goes here
    return None

with open(sys.argv[2], 'w', encoding="utf-8") as output:
    for page in Parser(sys.argv[1]):
        typ = extractType(page.content)
        if typ:
            output.write(page.title + "\t" + typ + "\n")

```

page.py

```python
import sys

class Page:
    def __init__(self, title, content):
        self.content = content
        self.title = title
        if sys.version_info[0] < 3:
            self.title = title.decode("utf-8")
            self.content = content.decode("utf-8")

    def __eq__(self, other):
        return isinstance(other, self.__class__) and self.title == other.title and self.content == other.content

    def __ne__(self, other):
        return not self.__eq__(other)

    def __hash__(self):
        return hash((self.title, self.content))

    def __str__(self):
        return 'Wikipedia page: "'+(self.title.encode("utf-8") if sys.version_info[0] < 3 else self.title)+'"'

    def __repr__(self):
        return self.__str__()

    def _to_tuple(self):
        return (self.title, self.content)

    # Only used for Disambiguation TP
    def label(self):
        return self.title[1:self.title.rindex("_")].replace("_", " ")

```

parser.py

```python
'''Parses a Wikipedia file, returns page objects'''
from page import Page
__author__ = "Jonathan Lajus"

class Parser:
    def __init__(self, wikipediaFile):
        self.file = wikipediaFile
    def __iter__(self):
        title, content = None,""
        with open(self.file, encoding='utf-8') as f:
            for line in f:
                line = line.strip()
                if not line and title is not None:
                    yield Page(title, content.rstrip())
                    title, content = None,""
                elif title is None:
                    title = line
                elif title is not None:
                    content += line + " "
```

VPL

◄ Mail Extraction

Aller à...

Résumé du cours (lexique) ►