DSCI 558: Building Knowledge Graphs

Homework 2: Information Extraction

Released: Sep 4th, 2020

Due: Sep 13th, 2020 @ 23:59

Ground Rules

This homework must be done individually. You can ask others for help with the tools, however, the submitted homework has to be your own work.

Summary

In this homework, you will extract data from unstructured text. You will be using the data you extracted in the previous homework (from IMDb) and Spacy (https://spacy.io/), an open-source software library for advanced natural language processing (NLP).

Task 1: Preprocessing (1 point)

Store the biographies of the top 1000 entities you scraped in the previous homework (Homework 01, Task 2) in a tab-separated values (`tsv`) file (implement your own python script to achieve that). Each row should represent a single entity, and should contain two values: URL and biography. See the attached file `entities_bio_sample.tsv` to understand the format.

Task 2: SpaCy (9 points)

In this task you will extract structured data, for each entity, from the unstructured biography text. You will perform **Rule-Based Extraction**. We are interested in the following attributes:

`parent`	Name(s) of the entity's parents
`education`	Educational institution(s) attended by the entity
`debuted_in`	Name(s) of the performance in which the entity debuted
`starred_in`	Name(s) of the performance(s) in which the entity had some role
`spouse`	Name(s) of the husband(s) or wife(s) of the entity

Figure 1 shows these attributes over an example biography text.

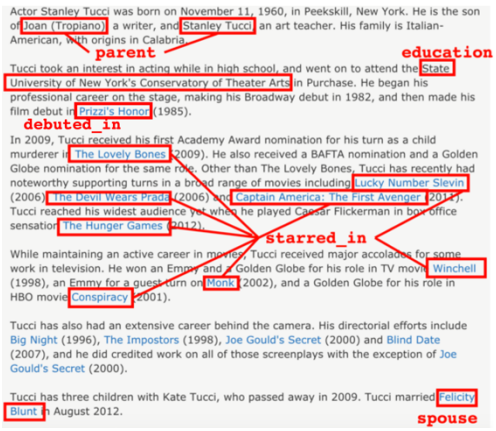

Figure 1: An example biography text with the required attributes marked

Before moving to this section's subtasks, familiarize yourself with SpaCy. We provide a python notebook (`Task2.ipynb`), which contains instructions, code and descriptions on how to perform information extraction using SpaCy and how to implement Rule-Based Matching patterns and functionalities.

Task 2.1 (1 point)

Pick one webpage of a single entity from what you have scraped (out of the 1000). Screenshot the page and highlight the required attributes over the screenshot (similar to what is shown in Figure 1).

Task 2.2 (1 point)

Pick one sentence from the bio which includes a reference to a spouse (such as the sentence shown in Figure 2 that is marked with a `starred_in` relation). Use the provided notebook (or use the code provided in there) to visualize its dependency parse-tree (using displaCy). An example of such tree is shown in Figure 2.

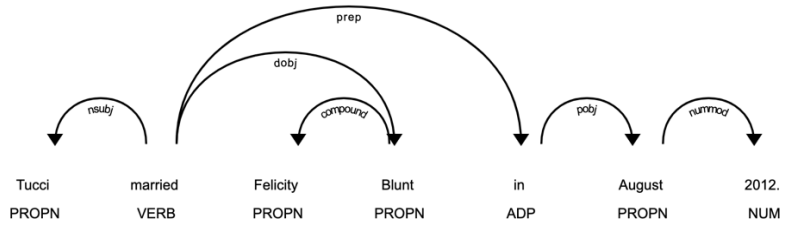

Figure 2: A dependency parse-tree of a sentence

Task 2.3 (5 points)

Implement two extractors for each attribute (`parent`, `education`, `debuted_in`, `starred_in`, `spouse`):

- Lexical: One that uses only lexical relation phrases (textual).
- Syntactic: One that uses the POS tags and/or dependency parse-tree and lexical phrases.

The code should be implemented in python, the script should accept a single argument (your `tsv` file from previous task) and produce a single `jl` file which includes all the extracted values of each attribute and the original `url` (for each entry). This file will be submitted as well. Each value should be a list (except for the `url` and `debuted_in`). An example file is provided (`sample__task_2_3.jl`).

Notes:

- Your script file (`.py`) should include 10 extractors (5 attributes x 2 types of extractors)
- Your output file (`.jl`) should include 1000 entries.
- In the provided notebook, we present a sample code that can be used as a starting point for extracting the `spouse` attribute.

Task 2.4 (2 points)

In this task you will validate your extractors. Start by building a ground-truth by picking 20 entries out of the 1000 entries (any that you want), then label them manually. Store your labeled data in 5 excel files (one for each attribute): indicate `0` if your extractor failed to acquire the value you labeled or `1` if it succeeded. Finally, report the **recall** of each one of your extractors on the labeled data (**success** divided by **total**) in the report file.

For example, for the `starred_in` attribute the file would look as shown in Figure 3. As seen in the figure, the first entity has two values (ground truth, which we manually inserted/labeled). The first extractor failed to extract the first movie but succeeded in the second one. The second extractor succeeded in both. So the recall of the lexical extractor of this attribute would be 50% and the recall for the syntactic one would be 100%.

A	B	C	D
url	**ground_truth__starred_in**	**extracted by lexical?**	**extracted by syntactic?**
https://www.imdb.com/name/nm0001804	Lucky Number Slevin	0	1
https://www.imdb.com/name/nm0001804	The Devil Wears Prada	1	1
...	...	...	...

Figure 3: An example of an evaluation file

Submission Instructions

You must submit (via Blackboard) the following files/folders in a single `.zip` archive named `Firstname_Lastname_hw02.zip`:

- `Firstname_Lastname_hw02_report.pdf`: pdf file with your answers to Tasks 2.1, 2.2, 2.4
- `Firstname_Lastname_hw02_bios.tsv`: as described in Task 1
- `Firstname_Lastname_hw02_cast.jl`: as described in Task 2.3
- Excel files containing the evaluation results as described in Task 2.4:
  - `Firstname_Lastname_hw02_task_2_4__parent.xlsx`
  - `Firstname_Lastname_hw02_task_2_4__education.xlsx`
  - `Firstname_Lastname_hw02_task_2_4__debuted_in.xlsx`
  - `Firstname_Lastname_hw02_task_2_4__starred_in.xlsx`
  - `Firstname_Lastname_hw02_task_2_4__spouse.xlsx`
- `source`: This folder includes all the code you wrote to accomplish Tasks 1 and 2 (i.e. your crawler/parser, your extractors code, etc…)