Homework 10: Web Crawling. POS Tagging. Relation Extraction.

Benjamin Roth, Marina Sedinkina
Symbolische Programmiersprache

Due: Thursday January 24, 2019, 16:00

In this exercise you will:

- process text from the given URL
- find homographs within this text
- extract relations between phrases and entities with spaCy

Exercise 1: Text Processing and Homographs [6 points]

This homework will be graded using unit tests by running: `python3 -m unittest -v hw10_crawling/test_analysis.py`

Implement following methods that can process the text from the given URL:

- `get_text(html)` - creates the list of clean paragraphs (no HTML markup) from the given html string (use `BeautifulSoup`) and returns paragraphs as a string. `Hint`: join the list of paragraphs by newline.
- `get_headline(html)` - returns the headline from the given html string.
- `get_normalized_tokens(text)` - should tokenize the text with NLTK and return the list of lower case tokens without stopwords (use NLTK to remove stopwords).

Use NLTK to find all homographs within the text. We use the following definition: Distinct words that have the same written form are called homographs. In other words, homographs are words with the same spelling and different POS.

- Implement a function `get_pos_dict(tokens)` that stores mapping between words and their possible POS tags. `Hint:` use `defaultdict`, a subclass of the built-in `dict` class. Setting `defaultdict` to `set` makes the `defaultdict` useful for building a dictionary of sets:

```
>>> s = [('red', 1), ('blue', 2), ('red', 3), ('blue', 4),
        ('red', 1), ('blue', 4)]
>>> d = defaultdict(set)
>>> for k, v in s:
...     d[k].add(v)
...
>>> d.items()
[('blue', set([2, 4])), ('red', set([1, 3]))]
```

- Implement a function `filter_dict_homographs(word_dict_h)` that deletes an entry from the dictionary, if this entry is not a homograph.
- Implement a function `find_homographs(tokens)` that returns a dictionary which holds homographs. Use already implemented methods.

Exercise 2: Relation Extraction with spaCy [10 points]

- First, install spaCy (see `https://spacy.io/usage/` for details):
  `pip install -U spacy`
- After installation you need to download a language model for English:
  `python -m spacy download en`

This homework will be graded using unit tests by running: `python3 -m unittest -v hw10_crawling/test_spaCy.py`

In this exercise, you will write a program to extract relations between noun phrases and MONEY entities with spaCy. Download the file `hydrogenics_report.txt` into the `data/` folder of your project. Take a look at the file `hw10_crawling/relation_extractor.py`. Implement the methods to make it work:

- `__init__(self,path,nlp)` - reads the text as a string and tokenizes it by sentences (use NLTK to tokenize the text by sentences). `self.sentences` is the list of sentences from the text of the given path [1 point]
- `entities_and_nounChunks(self,doc)` - extracts all named entities and noun phrases and saves them into one list (use spaCy to extract entities and noun chunks). [1 point]
- `update_tokenizer(self,spans)` - make from entities and noun chunks a single token (use spaCy). [1 point]
- `extract_money_relations(self,doc)` - extract the noun phrases and MONEY items they refer to:
  - iterate over tokens in the given document (entities and noun phrases are a single token now)
  - check if entity type is MONEY

  - check if MONEY is attribute (`attr`) (e.g. in the sentence "Net income was $9.4 million", the MONEY item "$9.4 million" is attribute, the head of the MONEY item "$9.4 million" is the word "was")
  - find a noun phrase for this `attr` (should be of type `nsubj` (nominal subject)) (e.g. in the sentence "Net income was $9.4 million", the noun phrase for the MONEY item "$9.4 million" is "Net income" which is the nominal subject and the left element of the MONEY item's head "was")
  - save this relation (tuple `(noun phrase, MONEY item)` e.g. `(''Net income'', ''$9.4 million'')`) in a list
  - check if MONEY is a preposition object (`pobj`) and its head is a prepositional modifier (`prep`)) (e.g. in the phrase "the prior year of $2.7 million", the MONEY item "$2.7 million" is a preposition object, the head of the MONEY item "$2.7 million" is the prepositional modifier "of")
  - find a noun phrase for this `pobj` (which is the head of the prepositional modifier (`prep`)) (e.g. in the phrase "the prior year of $2.7 million", the noun phrase for the MONEY item "$2.7 million" is "the prior year" which is the head of the prepositional modifier "of")
  - also save this relation in a list [4 points]

- `extract_relations(self)` - extract relations from the text (use implemented methods)
  - iterate over sentences (`self.sentences`)
  - convert each sentence in a spaCy object
  - extract entities and noun chunks
  - update tokenizer by converting entities and noun chunks into one token
  - extract money relations in each sentence
  - save the list with relations in a list [4 points]