Homework 6:
POS Tagging

Robert Litschko*
Symbolische Programmiersprache

Due: Thursday December 1, 2022, 12:00 (noon)

In this exercise you will:

- match sequences of POS-tags in sentences

Exercise 1: POS-Tags [10 Points]

In this exercise, you will write a program to match sequences of POS tags in sentences. In this exercise we will use `hydrogenics_report.txt`, please make sure it is in your `hw06_pos` folder. Take a look at the file `hw06_pos/pos_match.py`. Implement the remaining unimplemented methods to make it work:

This part of the homework will be graded using unit tests by running:

```
python3 -m unittest -v hw06_pos/test_pos_match.py
```

1. `Sentences.from_file(cls, path)` – reads the file at the given path, splits the sentences, tokenizes each sentence (use NLTK), pos-tags each sentence and returns a new instance of the `Sentences` class. `Hint:` The constructor of `Sentences` expects a list of tagged sentences (each sentence being a list of pos-tagged words). [2 points]

2. `PosExpr.from_string(cls, expr)` – creates an instance of PosExpr from a string expression. `Hint:` The constructor of `PosExpr` expect a list of strings; take a look at the tests to see how the function is used. [1 points]

3. `PosExpr.match_expr(expr, pos)` – returns True if `expr` matches `pos`. An expression *XX* matches the pos-tag *XX*, the expression *** matches any pos-tag and an expression *XX** matches the pos-tags *XX*, *XXY*, ... For example *NN** should return True for the tags *NN*, *NNP* and *NNPS*. [2 points]

*Credit: Exercises are based on previous iterations from Katerina Kalouli.

4. `PosExpr.matches(sequence)` – returns a list of matches in the given sequence (a sequence here is a list of (word, pos)-pairs – see following example). A single match is a list of (word, pos)-pairs, where the tags in the sequence match all expressions provided by PosExpr for all possible positions. [3 points]

   Example:

   ```
   >>> p = PosPattern.from_string("X Y")
   >>> seq = [('a','X'), ('b', 'Y'), ('c', 'Z'),('d', 'X'), ('e', 'Y'))]
   >>> matches = p.match_seq(seq)
   >>> matches
   [[('a', 'X'), ('b', 'Y')],[('d', 'X'), ('e', 'Y')]]
   ```

5. `find_str(sentences, expr)` – returns a list of strings (not (word, pos)-pairs) that match the given expression in all sentences. For example `find_string(sentences, "JJ NN")` should return the flat list `[...,"prior year",...]`. [2 points]