Projekt śródsemestralny: parser CKY

termin oddawania: 13 maja

Zadanie polega na samodzielnej implementacji parsera CKY oraz przygotowaniu krótkiego (1–2 strony) raportu. Jako punkt wyjścia może posłużyć akceptor z fragmentu 4.4 rozdziału 8 *NLTK book* (`http://www.nltk.org/book/ch08.html`). O algorytmie CKY można też poczytać tutaj: `https://web.stanford.edu/~jurafsky/slp3/12.pdf`.

Proszę zaimplementować parser jako klasę dziedziczącą po `nltk.parse.api.ParserI` (`http://www.nltk.org/api/nltk.parse.html#nltk.parse.api.ParserI`), podobnie, jak parsery dostępne w NLTK. Poniższy fragment kodu przedstawia bardzo prostą klasę spełniającą ten wymóg:

```
from nltk import ParserI
from nltk import Tree

class DummyParser(ParserI):

    def __init__(self, grammar):

        # real parsers shouldn't ignore their grammars... :)
        pass

    def parse(self, sentence):

        # real parsers shouldn't ignore the passed sentence... :)
        just_a_dummy = Tree('NP', [
                            Tree('ADVP', [Tree('ADV', ['just'])]),
                            Tree('NP', [Tree('Det', ['a']), Tree('N', ['dummy'])])
                       ])
        dummy_tree = Tree('S', [Tree('NP', ['I']), Tree('VP', [Tree('V', ['am']), just_a_dummy])])
        return iter([dummy_tree])

parser = DummyParser(None)
for tree in parser.parse('time flies like an arrow'.split()):
    tree.draw()
```

Implementacja parsera powinna:

- uwzględniać niejednoznaczności składniowe i leksykalne,

- odtworzyć wszystkie zgodne z gramatyką drzewa składniowe dla danego zdania (jako obiekty klasy `nltk.Tree`).

Należy w tym celu odpowiednio wzbogacić strukturę danych przechowywaną w komórkach tablicy parsera. Uwaga: w ogólnym przypadku odtwarzanie wszystkich drzew może być bardziej kosztowne, niż samo budowanie tablicy – nie należy się tym przejmować.

Parser powinien operować wyłącznie na gramatykach w postaci normalnej Chomsky'ego. W wersji mimimalnej konstruktor parsera powinien sprawdzić, czy ma do czynienia z gramatyką w takiej postaci (warto zajrzeć do dokumentacji `nltk.CFG`) i rzucić wyjątkiem, jeśli nie. W wersji ambitniejszej parser może w razie potrzeby przekształcać gramatykę do CNF (i od tej pory „zapomnieć" pierwotną gramatykę).

Proszę również napisać gramatykę bezkontekstową w postaci normalnej Chomsky'ego (lub nie w CNF, jeśli zdecydowali się Państwo na parser, który sam przekształca gramatykę), która pozwoli na wieloznaczne sparsowanie:

- *I shot an elephant in my pajamas* (2 drzewa),
- *time flies like an arrow* (4 drzewa),

ale nie pozwoli sparsować:

- *I shot elephant in pajamas*,
- *I shot a flies*,
- *a an elephant flies in my pajamas*.

Proszę dodać w pliku z implementacją parsera krótki fragment kodu (demo), który zaprezentuje działanie parsera i gramatyki dla 5 podanych zdań.

Postać rozwiązania:

Proszę przesłać paczkę zawierającą:

- skrypt w Pythonie, zawierający Państwa implementację parsera oraz demo,
- gramatykę (w skrypcie lub w osobnym pliku),
- raport na 1–2 strony, (PDF złożony w LaTeX-u), opisujący rozwiązanie, w szczególności:
  - strukturę tablicy budowanej przez parser,
  - sposób odtworzenia drzew,
  - wszelkie elementy rozwiązania, które okazały się trudniejsze czy ciekawe dla Państwa,
  - wszelkie rozszerzenia (np. przekształcanie gramatyki), jeśli zechcą Państwo takowe zaimplementować.