Pavel Dvořák, 172770

Implementace CKY algoritmu pomocí knihovny NLTK

Cíl projektu

Napsat syntaktický analyzátor češtiny využívající algoritmus CKY (*Cocke–Kasami–Younger*) a pythonovou knihovnu NLTK (*Natural Language Toolkit*).

Předpoklady pro spuštění programu

Python, knihovna NLTK, morfologický analyzátor Ajka, označkovaný korpus.

Ovládání programu

Program se sestává z jediného souboru, `cky.py`, který lze ovládat pouze pomocí příkazové řádky. Po napsání příkazu `./cky.py -h` se vypíše nápověda v angličtině:

```
Usage: cky.py [OPTIONS]
The Cocke-Kasami-Younger algorithm performs the syntactic analysis
of a sentence using the grammar in a Chomsky normal form (CNF) and
outputs a parsing table.
The analysed sentence is expected to be entered by the standard input
(e.g. echo "Model hradu v použitelném stavu." | cky.py).

OPTIONS:
        -h, --help              display this help and exit
        -g, --generate=CORPUS   generate a grammar file of the CORPUS
        -v, --verbose           verbose output

The CORPUS file (e.g. /nlp/corpora/vertical/desam/source) should be
saved in the latin2 encoding.
```

Uživatel má na výběr mezi dvěma funkcemi: generováním gramatiky z korpusu a syntaktickou analýzou české věty. Parametr `-v` umožňuje vypsat podrobnější výstup.

Generování gramatiky

Protože pro knihovnu NLTK neexistuje žádná volně dostupná definice české gramatiky, program se snaží odvodit gramatická pravidla z označkovaného korpusu. Vstupem je soubor s korpusem (určuje se pomocí přepínače `-g`, tedy například napsáním příkazu `./cky -g /nlp/corpora/vertical/desam/source`) a výstupem je gramatika v CNF (*Chomského normální formě*). Protože k následné analýze nepotřebujeme žádné meta informace z korpusu, program pro jeho procházení nevyužívá žádnou XML knihovnu, ale prochází ho lineárně a pomocí sady regulárních výrazů z něj vybírá jednotlivé věty.

Sledy morfologických značek z vět jsou ukládány do stromové struktury obsahující kořenový symbol S, která je následně převedena do CNF a poté na gramatiku (pravidla typu A → BC). Aby byl převod na gramatiku jednoznačný, neterminály jsou odlišeny od terminálů připojením prefixu sestávajícího se z písmene X. Protože má tento převod určité nároky na tvar neterminálů, některé speciální znaky jsou nahrazeny jejich textovou reprezentací.

Gramatika v CNF je velmi obsáhlá a její generování proto trvá dlouhou dobu. Z toho důvodu se ve výchozím nastavení nezpracovává celý korpus, ale pouze jeho část – konkrétně prvních sto tisíc řádků. Tato část korpusu zabírá okolo dvou megabajtů místa na disku, přičemž vygenerovaná gramatika „nabobtná" až na devadesát megabajtů a její generování trvá na serveru Aurora téměř půl minuty. Výchozí soubor, do kterého se gramatika ukládá, je `cky.data`.

CKY algoritmus

Když máme přichystanou gramatiku, můžeme se pustit do analýzy. Stačí programu poslat přes standardní vstup libovolnou českou větu v kódování UTF-8. Program tuto větu rozdělí na jednotlivá slova a na speciální symboly (interpunkce apod.). Tato slova přepošle externímu morfologickému analyzátoru Ajka. Pokud analyzátor nezvládne některá ze slov rozpoznat, program jednoduše vypíše chybu a ukončí se; v opačném případě pokračujeme v analýze seznamu různých morfologických kategorií u každého slova.

Při syntaktické analýze v prvním kroku převedeme jednotlivé složky věty na neterminály. Poté tyto neterminály přidáme do prvního řádku CKY tabulky a následně generujeme vyšší patra tabulky přidáváním neterminálů z pravidel gramatiky. Tato část odpovídá prezentované variantě CKY algoritmu z přednášky.

Pokud na některý z neterminálů nejvyššího řádku tabulky míří startovní pravidlo S, vypíše se, že analýza byla úspěšná a že danou větu lze vygenerovat pomocí naší gramatiky. V režimu podrobného výpisu se uživateli zobrazí i použitá tabulka.

Příklady

Nerozpoznané slovo morfologickým analyzátorem:

```
$ echo 'Mám rád želvy.' | ./cky.py -v
Lexical analysis...
Error: the word 'rád' is not recognized by the morphological analyser Ajka.
```

Neúspěšně analyzovaná věta:

```
$ echo 'Věta tato smysl nedávat.' | ./cky.py -v
Lexical analysis...
Věta: k1gFnSc1
tato: k3gNnPc1, k3gNnPc4, k3gFnSc1, k1gMnSc5
smysl: k1gInSc1, k1gInSc4
nedávat: k5eNaImF
.: .
```

```
Loading the grammar...
Performing the analysis...
Věta   tato   smysl   nedávat   .
[Xk1gFnSc1]   [Xk1gMnSc5]   [Xk1gInSc4, Xk1gInSc1]   [Xk5eNaImF]   [XDOT]
[]   []   []   [Xk5eNaImF]
[]   []   []
[]   []
[]
The sentence is NOT OK according to our Czech grammar.
```

Úspěšně analyzovaná věta:

```
$ echo 'Věta k vyzkoušení.' | ./cky.py -v
Lexical analysis...
Věta: k1gFnSc1
k: k7c3
vyzkoušení: k5eAaPmNgInS, k5eAaPmNgMnS, k2eAgMnSc1d1, k2eAgInSc1d1,
k2eAgInSc4d1
.: .

Loading the grammar...
Performing the analysis...
Věta   k   vyzkoušení   .
[Xk1gFnSc1]   [Xk7c3]   [Xk5eAaPmNgInS, Xk2eAgMnSc1d1, Xk2eAgInSc1d1,
Xk5eAaPmNgMnS, Xk2eAgInSc4d1]   [XDOT]
[Xk1gFnSc1]   []   [Xk5eAaPmNgInS, Xk2eAgMnSc1d1, Xk2eAgInSc1d1, Xk2eAgInSc4d1,
Xk5eAaPmNgMnS]
[Xk1gFnSc1]   []
[Xk1gFnSc1]
The sentence is OK according to our Czech grammar.
```

Závěr

Tento způsob analýzy je dosti neefektivní, protože vyžaduje gramatiku v CNF tvaru, kterou není jednoduché vygenerovat efektivně. Přesto však funguje a knihovna NLTK se ukázala jako vhodný základ pro psaní algoritmů na zpracování přirozeného jazyka.