semestr 16Z

Karol Niedzielewski
Jakub Pankiewicz
Kinga Staszkiewicz

Odgadywanie części mowy i formy gramatycznej na podstawie sąsiedztwa wyrazu

Kluczowe decyzje projektowe

Naszym zadaniem było napisanie programu, który będzie w stanie odgadnąć jaką częścią mowy i w jakiej formie gramatycznej jest dany wyraz w zdaniu dzięki wykorzystaniu reguł wnioskowania. W projekcie wykorzystujemy analizator morfologiczny "Morfologik", dzięki któremu otrzymujemy informację o tym, jakimi częściami mowy w jakich formach gramatycznych może być dany wyraz. Napisany przez nas program ma na celu ujednoznaczniać przypadki, gdy do wyrazu przypisana jest więcej niż jedna część mowy lub forma gramatyczna.

Projekt został zrealizowany w języku C++.

Z racji tego, że język polski jest językiem trudnym[1] i z wieloma wyjątkami, musieliśmy przyjąć pewne założenia, które umożliwiły nam realizację zadania w terminie.

Główne założenia:

- program zawiera reguły napisane dla zdań prostych i równoważników zdań, które mogą, ale niekoniecznie muszą działać dla zdań złożonych, zwłaszcza podrzędnie;
- nie bierzemy pod uwagę interpunkcji, gdyż zdarzają się sytuacje, w których mogą one występować, ale nie muszą, bądź występują w różnych miejscach w zależności od kontekstu, którego nie analizujemy;
- analizowane zdanie nie zawierają skrótów - do sprawdzania czy w danym miejscu zdania może występować skrót niezbędna byłaby znajomość kontekstu i znaczenia, a nie tylko formy gramatycznej;
- analizowane zdania napisane są prozą, bez nadmiernej poetyki (takiej jak np. *"Dziś piękność twą w całej ozdobie / Widzę i opisuję"* - przestawiony szyk zdania);
- analizowane zdania nie zawierają wyliczeń, tj. nie występuje w nich sytuacja, w której w zdaniu następują bezpośrednio po sobie więcej niż dwa rzeczowniki w mianowniku (gdyż przykładowo zdanie *"Żyły piły miały smykały"* w zależności od umieszczenia przecinka może składać się z trzech rzeczowników i czasownika, bądź kolejno rzeczownika, czasownika, rzeczownika i czasownika, a nawet rzeczownika, rzeczownika, czasownika i rzeczownika);

[1] W większości istniejących rankingów, język polski jest klasyfikowany w ścisłej czołówce najtrudniejszych języków świata ze względu na dużą liczbę wyjątków i nieregularności.

- jeżeli wyraz nie jest nazwą własną, to niezależnie od jego miejsca w zdaniu musi zostać napisany małą literą (Morfologik nie rozpoznaje bowiem słów pospolitych napisanych wielką literą, a przedmiotem naszego projektu nie było decydowanie czy dany wyraz rozpoczyna się wielką literą z powodu znajdowania się na początku zdania, czy z powodu bycia nazwą własną);
- grupa podmiot-orzeczenie nie jest rozdzielona (w przeciwnym wypadku zbyt trudnym zadaniem byłoby ustalenie, które ze słów są ze sobą w takiej relacji)

Przyjęliśmy, iż w naszym projekcie wykorzystamy mechanizm wnioskowania wstecz - dla założonego właściwego dla danego sąsiedztwa wyrazów znaczenia słowa sprawdzamy zgodność ze znaczeniami pozostałych słów. Jeśli następuje zgodność - otrzymaliśmy rozwiązanie, jeśli nie, przyjmujemy za właściwe kolejne znaczenie i powtarzamy algorytm do skutku.

Z przyczyn technicznych (tj. braku znanej nam możliwości zakończenia działania programu Morfologik w sposób inny niż ręczne przerwanie procesu[2]) nie udało nam się w pełni "wcielić" Morfologika do naszego programu. Mimo wszystko, z powodzeniem wykorzystujemy Morfologika w osobnym programie (który kończy się wraz z "siłowym" zakończeniem działania Morfologika), dzięki któremu zapisujemy wynik działania Morfologika do pliku.

Instrukcja dla użytkownika

W celu odgadnięcia części mowy i formy gramatycznej wyrazów w zdaniu należy uruchomić program *konwerter.exe*. W otworzonej konsoli należy wpisać, wyraz po wyrazie, zdanie, które ma zostać przeanalizowane. Należy pamiętać, aby zaczynać wielką literą jedynie słowa, które w kontekście zdania są nazwami własnymi. Po każdym wpisanym wyrazie należy wcisnąć *enter*. Kiedy wpisane zostaną wszystkie słowa tworzące zdanie należy zamknąć okno i uruchomić program *analizator.exe*, upewniając się najpierw, że oba programy znajdują się w tej samej lokalizacji. Program ten zapisze odgadnięte części mowy i formy gramatyczne wyrazów do pliku oraz wyświetli je w konsoli. W przypadku jakiś problemów, tj. uszkodzonego pliku, niemożności odganięcia form, etc. wyświetlony zostanie komunikat o błędzie.

W celu kompilacji pogramu *konwerter* należy skompilować plik *Converter.cpp*, natomiast w celu kompilacji programu *analizator* należy skompilować pliki *main.cpp*, *Meaning.cpp*, *Word.cpp*, *Sentence.cpp* z podlinkowanymi plikami nagłówkowymi *Meaning.h*, *Word.h* oraz *Sentence.h*.

Opis struktury programu

Program *konwerter* służy jedynie do odpalenia programu Morfologik oraz przekierowania jego wyjścia do pliku, nie ma zatem potrzeby opisywania jego struktury, gdyż jest ona nieskomplikowana.

[2] Wysłaliśmy nawet maila w tej sprawie do twórców Morfologika, jednak nie otrzymaliśmy odpowiedzi.

Natomiast jeśli chodzi o program właściwy, tj. *analizator*, to zaimplementowano w nim trzy klasy: *Meaning*, *Word* oraz *Sentence*, na obiektach których wykonywane są wszelkie operacje mające na celu odganięcia poprawnej części mowy i formy gramatycznej.

- klasa *Meaning* zawiera pola symbolizujące część mowy i atrybuty formy gramatycznej danego znaczenia, takie jak *partofspeech_*, *number_*, *grammarcase_*, *gender_*, itp. oraz metody, tzw. *gettery* i *settery* do wymienionych pól;
- klasa *Word* zawiera pola *word_*, *meanings_* oraz *position_*, które zawierają odpowiednio słowo podane w zdaniu, listę obiektów typu *Meaning* oznaczającą wszystkie możliwe znaczenia danego słowa oraz pozycję słowa w zdaniu; dostępne metody to także tzw. *gettery* i *settery* do wymienionych pól
- klasa *Sentence* zawiera pola *listOfWords_* oraz *chosenMeanings*, które zawierają odpowiednio listę słów występujących w zdaniu oraz tablicę z wybranym właściwymi w stosunku do całego zdania znaczeniami słowa;
  metody tej klasy można podzielić na trzy grupy:
  - → metody pozwalające na odczyt z pliku słów wraz z możliwymi znaczeniami uzyskanymi za pomocą programu Morfologik oraz zapisanie tych danych do obiektu klasy *Sentence*;
  - → metody pozwalające na analizę obiektu klasy *Sentence*, w tym funkcja rekurencyjna (schemat działania na ostatniej stronie) porównująca znaczenia dwóch sąsiednich słów oraz funkcje-reguły, które wywoływane są dla danej pary znaczeń pod warunkiem bycia daną częścią mowy (przykładowy kod funkcji-reguły wraz z omówieniem na przedostatniej stronie)
  - → metoda pozwalająca na zapis do pliku wyniku działania programu

Wnioski

Napisanie programu odgadującego część mowy i formę gramatyczną na podstawie sąsiedztwa wyrazu okazało się trudniejsze, niż mogłoby się wydawać. Problemem nie było jednak zaimplementowanie logiki (w tym rekurencyjnej funkcji wnioskowania czy funkcji-reguł), lecz wymyślenie reguł, które należy zapisać.

Język polski jest skomplikowany i przeciętnemu studentowi Politechniki z trudnością przychodzi analizowanie zasad rządzących jego gramatyką, nawet mimo przyjętych, upraszczających zadanie założeń. Na pierwszy rzut oka wydaje się, że w gramatyce języka polskiego więcej jest wyjątków, niż reguł. Jakby tego było mało, część mowy i forma gramatyczna w wielu przypadkach zależy od kontekstu. Przykładowo zdanie "*Granie w Tatrach.*" może w zależności od kontekstu mieć na pierwszym miejscu rzeczownik, bądź czasownik. Problemy związane z kontekstem występują w wielu innych przypadkach[3].

[3] Zdanie "*Kasia lubi piec ciasteczka*" dla przeciętnego człowieka oznacza, że Kasia lubi czynność pieczenia ciasteczek, jednak pod względem gramatycznym może ono również oznaczać, że Kasia lubi piec, który należy do bliżej nieokreślonego ciasteczka. Zdanie to raczej nie ma sensu jeśli chodzi o znaczenie, aczkolwiek jest poprawne gramatycznie. Wystarczy zamienić chociażby słowo *piec* na *lukier*, aby otrzymać zdanie o sensownym znaczeniu a identycznej strukturze gramatycznej.

Na potrzeby projektu zapisaliśmy podstawowe reguły, które udało nam się ustalić na podstawie własnych analiz zdań wstępujących w języku polskim oraz w wyniku poszukiwań w internecie. Służą one do rozpoznawania prostych, niezbyt skomplikowanych zdań.

Napisanie programu rozpoznającego część mowy i fromę gramatyczną na podstawie sąsiedztwa wyrazów dla dowolnego zdania byłoby zajęciem wymagającym znacznie większej znajomości języka polskiego oraz znacznie dłuższego czasu przeznaczonego na realizację zadania.

Reguły zaimplementowane w programie

- *verbVSverb* - reguła stwierdza zgodność dwóch następujących po sobie czasowników tylko jeżeli pierwszy z nich jest formą czasu przyszłego czasownika *być* => czas przyszły złożony; w przeciwnym przypadku stwierdza niezgodność;
  Dziewczyna będzie robić kawę.
  Dziewczyna jadła spała.
- *substVSverb* - reguła stwierdza zgodność rzeczownika i następującego po nim czasownika tylko jeśli zgadza się rodzaj, liczba i przypadek obu słów; w przeciwnym przypadku stwierdza niezgodność;
  Kot spoczywa w pokoju.
  Kot spoczywają w pokoju.
- *prepVSall* - reguła stwierdza zgodność zaimka i następującego po nim słowa tylko jeżeli następujące słowo nie jest czasownikiem oraz zgadza się przypadek obu słów (w Morfologiku bowiem w przypadku zaimków jako przypadek podany jest przypadek, w jakim mogą występować słowa po tym zaimku); w przeciwnym przypadku stwierdza niezgodność;
  Dzieci siedzą w szafie.
  Dzieci siedzą w szafa.
- *allVSprep* - reguła zawsze stwierdza zgodność słowa i następującego po nim zaimka;
  Robił w drewnie.
  Dziura w drewnie.
  Jest ładnie za oknem.
- *substVSadj* - reguła stwierdza zgodność rzeczownika i następującego po nim przymiotnika bądź imiesłowu przymiotnikowego tylko jeśli występuje jedna z sytuacji:
  - rzeczownik jest w dowolnym przypadku, przymiotnik jest w dopełniaczu
    Prezent/prezentu/prezentowi/prezent/prezentem/prezencie ładnego chłopca.
  - rzeczownik jest w bierniku, przymiotnik jest w celowniku
    Daję kota ładnej dziewczynie.
  - rzeczownik jest w celowniku, przymiotnik jest w bierniku
    Daję dziewczynie ładnego kota.

  w przeciwnym przypadku stwierdza niezgodność;
  Daję kocie ładna dziewczyna.
  Daję kot dziewczyna.

- *adjVSall* - reguła stwierdza zgodność przymiotnika i następującego po nim słowa tylko jeśli słowo to nie jest czasownikiem lecz rzeczownikiem, przymiotnikiem bądź imiesłowem przymiotnikowym oraz zgadza się rodzaj, liczba i przypadek obu słów; w przeciwnym przypadku stwierdza niezgodność;

  Ładna dziewczyna.
  Ładna wysoka dziewczyna.
  Ładna wysokiej dziewczyna.
- *substVSsubst* - reguła stwierdza zgodność dwóch następujących po sobie rzeczowników w takich samych przypadkach jak w funkcji *substVSadj*; w przeciwnym przypadku stwierdza niezgodność;
- *verbVSsubstAdj* - reguła stwierdza zgodność czasownika i następującego po nim rzeczownika lub przymiotnika jeśli:
  - czasownik jest formą czasownika *być*, rzeczownik/przymiotnik jest w mianowniku albo narzędniku
  - czasownik nie jest formą czasownika *być*, rzeczownik/przymiotnik nie jest w mianowniku ani w wołaczu

  w przeciwnym przypadku stwierdza niezgodność;
- *gerVSall* - reguła stwierdza zgodność rzeczownika odsłownego i następującego po nim słowa jeśli następujący po nim wyraz nie jest w mianowniku ani w wołaczu; w przeciwnym przypadku stwierdza niezgodność;

  Robienie (dobrego) ciasta.
  Robienie ciasto.
- jeśli para znaczeń nie pasuje do żadnej z powyższych reguł, zgodność stwierdzana jest jeśli zgadza się rodzaj, liczba i przypadek obu słów; przy czym jeśli dane słowo nie ma jakiegoś atrybutu, przyjmuje się je za zgodne pod tym względem z każdym słowem; w przeciwnym przypadku stwierdza niezgodność;

Podział pracy przy realizacji projektu

Wspólnie:
- szkielet programu: pliki nagłówkowe, podstawowe pola i metody klas;
- tworzenie bazy potencjalnie problematycznych słów;
- przygotowanie koncepcji działania programu, wybór metody wnioskującej;
- testowanie działania programu;

Karol Niedzielewski:
- tworzenie reguł ogólnych;
- dostosowanie kodu do systemu Linux;
- dodanie możliwości uruchamiania programu z parametrami;
- obsługa wyjątków;

Jakub Pankiewicz:
- uruchamianie Morfologika z poziomu C++, zapis wyjścia z Morfologika do pliku;
- tworzenie szczegółowych reguł;
- implementacja reguł;
- tworzenie zbioru zdań do testowania działania programu;

Kinga Staszkiewicz:

- tworzenie szczegółowych reguł;
- implementacja reguł i funkcji rekurencyjnej;
- szczegółowa implementacja pól i metod klas;
- tworzenie dokumentacji;

Przykładowe funkcje-reguły oraz zasada działania funkcji użytej rekurencyjnej

```
1.  //rule for preposition before another words
2.  //return 3 - meanings are matching; return 0 - not matching;
3.  int Sentence::prepVSall(Meaning& m1,Meaning& m2)
4.  {
5.      //grammar case of preposition is pointing the grammar case of the word that's following it
6.      if (compareGrammarCases(m1.getGrammarCase(),m2.getGrammarCase())==1)
7.          return 3;
8.      else
9.          return 0;
10. }
```

Powyższa funkcja jest funkcją-regułą dla sytuacji, w której jako pierwsze z dwóch analizowanych słów następuje przyimek, zaś po nim inna część mowy. Przyimek po przeanalizowaniu przez program Morfologik ma podany jako przypadek możliwy przypadek słowa występującego po nim. W związku z tym, jeśli nie następuje zgodność przypadków, stwierdzamy niezgodność.

```
1.  //rule for verb following another verb
2.  //return 3 - meanings are matching; return 0 - not matching;
3.  int Sentence::verbVSverb(Meaning& m1,  Meaning& m2)
4.  {
5.      //one verb can follow another only in complex future tense
6.      if (!m1.getFuture().compare("będzie"))
7.          return 3;
8.      else
9.          return 0;
10. }
```

Powyższa funkcja jest funkcją-regułą dla sytuacji, w której po sobie następują dwa czasowniki. W języku polskim, na tyle na ile udało się nam ustalić, sytacja taka występuje jedynie w czasie przyszłym złożonym[4]. W związku z tym, jeśli pierwszy z analizowanych czasowników nie jest formą przyszłą czasownika "być", stwierdzamy niezgodność.

[4] Oprócz zdań złożonych, których zgodnie z założeniami nie analizujemy. Np. "*Dziewczyna spała, śniła i marzyła*."

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Start: *przyjmujemy pierwsze znaczenie pierwszego słowa jako poprawne oraz pierwsze znaczenie drugiego słowa jako potencjalnie poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 1: *porównujemy znaczenie drugiego słowa ze znaczeniem słowa pierwszego - jeśli następuje niezgodność odrzucamy dane znaczenie słowa drugiego;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 2: *przyjmujemy drugie znaczenie słowa drugiego jako potencjalnie poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 3: *porównujemy znaczenie drugiego słowa ze znaczeniem słowa pierwszego - jeśli następuje zgodność uznajemy je jako poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 4: *przyjmujemy pierwsze znaczenie słowa trzeciego jako potencjalnie poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 5: : *porównujemy znaczenie trzeciego słowa ze znaczeniem słowa pierwszego - jeśli następuje niezgodność odrzucamy dane znaczenie słowa trzeciego; słowo trzecie nie ma innych znaczeń, zatem cofamy nasze rozważania do wyboru innego znaczenia słowa drugiego;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 6: *przyjmujemy trzecie znaczenie słowa drugiego jako potencjalnie poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 7: *porównujemy znaczenie drugiego słowa ze znaczeniem słowa pierwszego - jeśli następuje zgodność uznajemy je jako poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 8: *przyjmujemy pierwsze znaczenie słowa trzeciego jako potencjalnie poprawne;*

słowo 1	słowo 2	słowo 3	…
znaczenie1	znaczenie1	znaczenie1	
znaczenie2	znaczenie2		
	znaczenie3		
	znaczenie4		

Krok 9: *porównujemy znaczenie drugiego słowa ze znaczeniem słowa pierwszego - jeśli następuje zgodność uznajemy je jako poprawne;*
i tak dalej dla kolejnych słów w zdan