Integracja parsera zależnościowego z parserem kategorialnym

Wojciech Jaworski

Instytut Podstaw Informatyki
Polskiej Akademii Nauk

2 marca 2017

Podział na zdania

- Wymagania względem reprezentacji podziału na zdania:
  - reprezentacja podziału na
    - ★ tokeny,
    - ★ zdania,
    - ★ akapity

    i całego tekstu,
  - reprezentacja zdania w różnych stadiach przetwarzania:
    - ★ sekwencja znaków,
    - ★ graf tokenów,
    - ★ skompresowany las drzew zależnościowych, ...
  - reprezentacja mowy niezależnej i zagnieżdżonych zdań,
  - reprezentacja niejednoznacznego podziału na zdania,
  - reprezentacja formatów akceptowanych/generowanych przez różne parsery.

Konwersja drzew zależnościowych

- Korzystam z tego, że parsowanie w gramatykach kategorialnych jest dowodzeniem twierdzeń w logice.
- Logika intuicjonistyczna
  - zbiór przesłanek
  - przesłanki są łączne, przemienne, można je wielokrotnie używać w dowodzie
- Liniowa logika intuicjonistyczna
  - multizbiór przesłanek
  - przesłanki są łączne, przemienne, każdą z nich trzeba użyć dokładnie jeden raz w dowodzie
  - *glue semantics* w LFG

Konwersja drzew zależnościowych

- Niekomutatywna liniowa logika intuicjonistyczna
  - lista przesłanek
  - przesłanki są łączne, nieprzemienne, każdą z nich trzeba użyć dokładnie jeden raz w dowodzie
  - Logical Categorial Grammar
- Logika **NL**, M. Moortgat „Categorial Type Logics”
  - drzewo przesłanek
  - przesłanki nie są łączne, nie są przemienne, każdą z nich trzeba użyć dokładnie jeden raz w dowodzie
  - konwerter drzew zależnościowych

Przykładowy wywód gramatyczny

- Wywód w LCG:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Ala}}{np} \quad \dfrac{\text{ma}}{(ip\backslash np)/np} \quad \dfrac{\dfrac{\text{rudego}}{adjp} \quad \dfrac{\text{kota}}{np\backslash adjp}}{np}}{ip\backslash np}}{ip} \quad \dfrac{\text{.}}{s\backslash ip}}{s}$$

- Drzewo zależnościowe reprezentuję jako nawiasowanie tokenów:

((Ala (ma (rudego kota))) .)

- Konwersja drzewa zależnościowego:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{((Ala}}{np} \quad \dfrac{\text{(ma}}{(ip\backslash np)/np} \quad \dfrac{\dfrac{\text{(rudego}}{adjp} \quad \dfrac{\text{kota)))}}{np\backslash adjp}}{np}}{ip\backslash np}}{ip} \quad \dfrac{\text{.)}}{s\backslash ip}}{s}$$

Implementacja

- Niewielkie zmiany w parserze:
  - algorytm CYK zostaje zastąpiony spacerem po drzewie,
  - reguły formalizmu kategorialnego pozostają niezmienione,
  - leksykon gramatyki pozostaje niezmieniony.
- Zaimplementowane nawiasowanie wskazuje funktor oraz listy lewych i prawych argumentów:

$$\dfrac{\dfrac{\dfrac{[[\phi,\text{Ala},\phi],}{np}\quad\dfrac{\dfrac{\text{ma}}{(ip\backslash np)/np}\quad\dfrac{\dfrac{[[\phi,\text{rudego},\phi],}{adjp}\quad\dfrac{\text{kota},\phi]],}{np\backslash adjp}}{np}}{ip\backslash np}}{ip}\quad\dfrac{.,\phi}{s\backslash ip}}{s}$$

- ϕ oznacza listę pustą

Konstrukcje nieciągłe

- Nawiasowanie odzwierciedla strukturę drzewa zależnościowego,
  - w którym podrzędniki podzielone są na lewe i prawe
  - oraz posortowane względem kolejności wystąpienia w tekście.
- W przypadku konstrukcji nieciągłych występujących np. w zdaniu:

  Rudego Ala ma kota.

  nawiasowanie zmieni kolejność tokenów w zdaniu eliminując nieciągłość

  [[ϕ,Ala,ϕ], ma [[ϕ,rudego,ϕ], kota,ϕ]], .,ϕ

Konstrukcje nieuwzględnione w leksykonie

- Aby parsować konstrukcje nieuwzględnione w leksykonie gramatyki kategorialnej
  - do każdego typu dodawane są wielokrotne, opcjonalne argumenty
  - pasujące do dowolnego symbolu nie będącego funktorem.

$$\dfrac{\dfrac{\text{ma}}{(ip\{/?X,\backslash?X\}\backslash np)/np}\quad\dfrac{\dfrac{\dfrac{[[\phi,\text{blabla},\phi],}{unk\{/?X,\backslash?X\}}}{unk\{\backslash?X\}}}{unk}\quad\dfrac{\text{kota},\phi]}{np\{/?X,\backslash?X\}}}{np\{\backslash?X\}}}{np}}{ip\{/?X,\backslash?X\}\backslash np}$$

Zmiana powiązań między wierzchołkami w drzewie zależnościowym

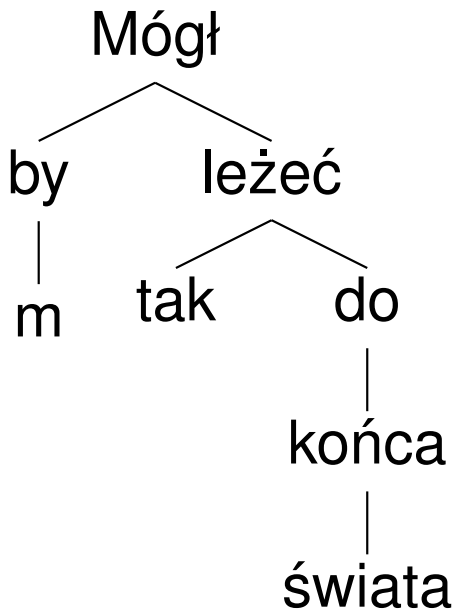
- Aby parsowanie powiodło się, drzewo zależnościowe na wejściu musi mieć strukturę zgodną z kolejnością aplikacji wierzchołków w gramatyce kategorialnej.
- W *Składnicy Zależnościowej* i w *Krzakach* struktura ta dla pewnych konstrukcji składniowych jest inna od oczekiwanej przez parser kategorialny.
- Dlatego przed parsowaniem drzewo zależnościowe poddajemy szeregowi lokalnych modyfikacji.

Tryb przypuszczający

- Czasownik w trybie przypuszczającym opisany jest za pomocą wpisu:

$$ip\{|\ by, |\ aglt, \ldots\}$$

- Zaś drzewo zależnościowe ma postać:



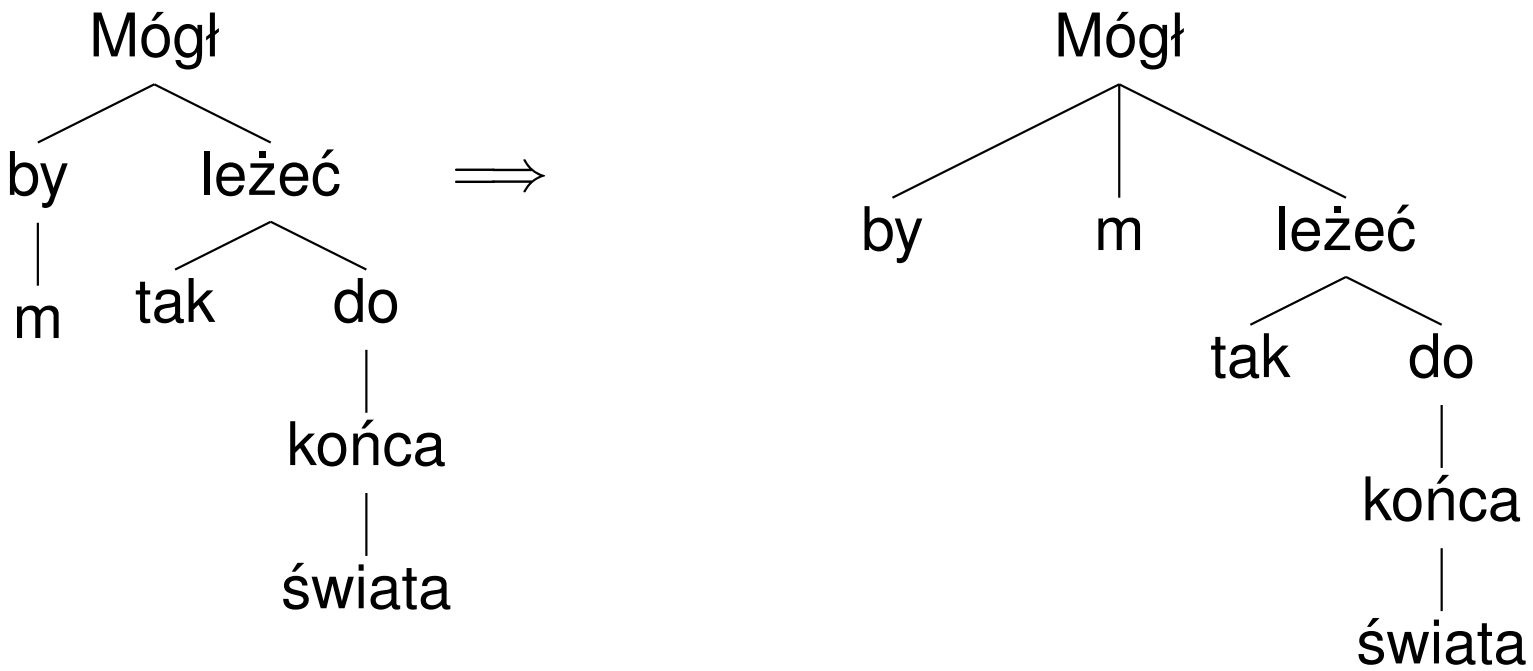
- Aby sparsować drzewo trzeba uczynić „m” bezpośrednim podrzędnikiem „Mógł”

Tryb przypuszczający

- Czasownik w trybie przypuszczającym opisany jest za pomocą wpisu:

$$ip\{|\ by, |\ aglt, \dots\}$$

- Zaś drzewo zależnościowe ma postać:



- Aby sparsować drzewo trzeba uczynić „m” bezpośrednim podrzędnikiem „Mógł”

Pozostałe modyfikacje

- usunięcie kropek i przecinków
- zmiana dowiązań myślników przy mowie niezależnej
- rozwiązywanie skrótów
- zmiana struktury wieloczłonowej koordynacji
- przesunięcie spójników podrzędnych, zaimków pytajnych i względnych powyżej czasownika.

Koncepcja przetwarzania skompresowanych lasów generowanych przez parser *Świgra*

- Skompresowany las jest to skierowany graf acykliczny mający wierzchołki:
  - multiplikatywne — wiążące frazę z jej składowymi,
  - addytywne — wyrażające możliwe alternatywne rozbiory składniowe fragmentu zdania.
- Wierzchołki multiplikatywne będą przetwarzane tak jak wierzchołki w drzewach zależnościowych.
- Wierzchołki addytywne będą przetwarzane za pomocą aktualnego algorytmu kompresującego niejednoznaczność dla parsera kategorialnego.
- Przeprowadzenie lokalnych zmian układu wierzchołków będzie wymagać rozkompresowania fragmentu lasu przed parsowaniem.

Algorytm MateParser

- Opis działania:
  - Wypełnia tablicę zgodnie z działaniem algorytmu CYK trzymając w każdym polu do 10 alternatywnych poddrzew
  - Przy wypełnianiu kolejnego pola rozpatrywane są osobno przypadki, gdy lewy węzeł jest nadrzędnikiem, i gdy jest nim prawy węzeł.
  - Algorytm rozpatruje każdą parę kandydatów na ojca i syna i wybiera 10 najlepiej ocenionych zgodnie z wagą wyliczoną na podstawie atrybutów.
  - Obsługuje nieciągłości za pomocą przegrupowywania wierzchołków z ograniczeniem poziomu nieciągłości w drzewie.
- Cele reimplementacji:
  - dezambiguacja lasów generowanych przez parser kategorialny.
  - łączenie sparsowanych fragmentów zdań, dla których parser kategorialny nie generuje pełnego rozbioru

Metoda realizacji

- Kluczowym elementem parsera zależnościowego jest model, który określa wagę powiązania między parą wierzchołków.
- Wagę tą można wykorzystać w różnych algorytmach i do różnych celów.
- Dezambiguację można wykonać uzupełniając algorytm MateParser o uwzględnianie więzów wprowadzanych przez parser kategorialny.
  - Jest to rozwiązanie analogiczne do proponowanego przy integracji ze Świgrą.
- Uzupełnianie drzew rozbioru można zrealizować podobnie,
  - z założeniem, że korzenie sparsowanych fragmentów mogą być synami wierzchołków należących do innych fragmentów.

Dziękuję za uwagę!