

Gramatyki bezkontekstowe i parsing

Paweł Rychlikowski

Instytut Informatyki UWr

11 grudnia 2018

- W roku 1992 powstała **Formalna gramatyka języka polskiego**, będąca programem w Prologu, wykorzystującym DCG. Autorem był Marek Świdzinski, który wcześniej opisywał gramatykę polską pisząc na przykład książki dla polonistów.
- Świgr bazuje na tej gramatyce, ale ze „znaczącymi modyfikacjami”. Korzysta również ze słownika walencyjnego (powstałego w 1998)
- Obecna wersja pochodzi z roku 2013.

Czy można inaczej?

Czy można inaczej?

- Losowanie zdań jest raczej dobrym pomysłem (nie da się stworzyć treebanku z całego korpusu).

Czy można inaczej?

- Losowanie zdań jest raczej dobrym pomysłem (nie da się stworzyć treebanku z całego korpusu).
- Czy można korzystać z gramatyki, jeżeli znacząco obniża to koszty tworzenia treebanku? (gramatyka może sugerować coś lingwistom, ale może to nie problem?)

Czy można inaczej?

- Losowanie zdań jest raczej dobrym pomysłem (nie da się stworzyć treebanku z całego korpusu).
- Czy można korzystać z gramatyki, jeżeli znacząco obniża to koszty tworzenia treebanku? (gramatyka może sugerować coś lingwistom, ale może to nie problem?)
- O kształcie drzewa nie powinien decydować jeden człowiek. Typowy protokół: 3 lingwistów, jak 2 się kłóci, to trzeci decyduje.

Czy można inaczej?

- Losowanie zdań jest raczej dobrym pomysłem (nie da się stworzyć treebanku z całego korpusu).
- Czy można korzystać z gramatyki, jeżeli znacząco obniża to koszty tworzenia treebanku? (gramatyka może sugerować coś lingwistom, ale może to nie problem?)
- O kształcie drzewa nie powinien decydować jeden człowiek. Typowy protokół: 3 lingwistów, jak 2 się kłóci, to trzeci decyduje.

Uwaga

Jedynymi odrzuconymi wypowiedzeniami powinny być te, które zostaną zgodnie uznane za niepoprawne.

O projekcyjności zdań

O projekcyjności zdań

Zagadka

Co jest nie tak w zdaniu wpłynąłem na suchego przestwór oceanu?

Zagadka

Co jest nie tak w zdaniu **wpłynąłem na suchego przestwór oceanu**?

- Fraza **suchego oceanu** jest nieciągła w tekście!

Zagadka

Co jest nie tak w zdaniu **wpłynąłem na suchego przestwór oceanu**?

- Fraza **suchego oceanu** jest nieciągła w tekście!
- Pamiętamy: **babuleńka koziółki** miała **dwa rogate**

Zagadka

Co jest nie tak w zdaniu **wpłynąłem na suchego przestwór oceanu**?

- Fraza **suchego oceanu** jest nieciągła w tekście!
- Pamiętamy: **babuleńka koziołki miała dwa rogate**
- Wnioskowanie o języku polskim na przykładzie Składnicy prowadzi do błędnych wniosków o „naturalności” powyższych konstrukcji!

- Składnicę można pobrać np. w formacie Tiger XML (dobrze udokumentowanym)
- Istnieje możliwość przeszukiwania on-line (<http://treebank.nlp.ipipan.waw.pl>)
- Za chwilę zobaczymy jak używać składnicy w bibliotece NLTK

Prezentacja Składnicy. (2)

Programy: `skladnica_demo2.py` `skladnica_demo1.py`

- Wspomniana wyszukiwarka prezentuje przykładowy interfejs wyszukiwania w bankach drzew.
- Zawiera elementy znane nam z NKJP, na przykład [base=rozmawiać]
- Można pytać o relacje w drzewie, na przykład znajdować frazy typu A, zawierające podfrazę typu B, a nie zawierający podfrazy typu C.

Przykłady zapytań

- `[cat=fno] >* [cat=zdanie]` fraza nominalna zawierająca gdzieś tam zdanie, np.:

gniew i agresja przeciwko tym, którzy zniszczyli tak pięknie do tej pory uporządkowany świat

- `[cat=fno] >* [cat=zdanie]` fraza nominalna zawierająca gdzieś tam zdanie, np.:
gniew i agresja przeciwko tym, którzy zniszczyli tak pięknie do tej pory uporządkowany świat
- Faza nominalna zawierająca jako podfrazę bezpośrednią frazę przymiotnikową i dodatkowo zawierającą słowo „kobieta”:
#z: `[cat=fno] > [cat=fpt] & (#z >* [base=kobieta])`

krótkiej historii uczestnictwa kobiet w życiu publicznym

Starsza kobieta, ubrana w czarny przeciwdeszczowy płaszcz z kapturem

Słownik walencyjny

Zbiór typów (albo, jak powiedzieliby lingwiści, ram walencyjnych) dla różnych słów.

Słownik walencyjny

Zbiór typów (albo, jak powiedzieliby lingwiści, ram walencyjnych) dla różnych słów.

Projekt Walenty stanowi duży, publicznie dostępny zbiór takich danych

- Opracowany w Zakładzie Inżynierii Lingwistycznej IPI PAN
- Zawiera 58585 informacji o 15866 słów.
- Tworzony (częściowo) automatycznie, weryfikowany przez lingwistów.

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Walenty to wie i dodaje:

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Walenty to wie i dodaje:

- Nie było upału, **straszyło** deszczem, ale przez weekend nie padało.

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Walenty to wie i dodaje:

- Nie było upału, **straszyło** deszczem, ale przez weekend nie padało.
- Jeżeli **straszysz** się, że nie jesteś nic wart, (...)

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy!**

Walenty to wie i dodaje:

- Nie było upału, **straszyło** deszczem, ale przez weekend nie padało.
- Jeżeli **straszysz** się, że nie jesteś nic wart, (...)
- Byśmy sami **straszyli** się nadciągającą apokalipsą.

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Walenty to wie i dodaje:

- Nie było upału, **straszyło** deszczem, ale przez weekend nie padało.
- Jeżeli **straszysz** się, że nie jesteś nic wart, (...)
- Byśmy sami **straszyli** się nadciągającą apokalipsą.
- Bo ci zrobię pokrzywkę - **straszy** Pająk.

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Walenty to wie i dodaje:

- Nie było upału, **straszyło** deszczem, ale przez weekend nie padało.
- Jeżeli **straszysz** się, że nie jesteś nic wart, (...)
- Byśmy sami **straszyli** się nadciągającą apokalipsą.
- Bo ci zrobię pokrzywkę - **straszy** Pająk.
- Jedne duchy **straszą** na zamku, drugie kąpią się w jeziorze, (...)

Eksperyment walencyjny

Jakie ramy walencyjne ma czasownik **straszyć** (ja wymyśliłem takie):

- Starszy brat namiętnie **straszył** swoją siostrę.
- Kangurzyca **straszyła** Maleństwo, że jak będzie robiło takie miny jak Prosiaczek, to mu zostanie.
- W tym domu **straszy**!

Walenty to wie i dodaje:

- Nie było upału, **straszyło** deszczem, ale przez weekend nie padało.
- Jeżeli **straszysz** się, że nie jesteś nic wart, (...)
- Byśmy sami **straszyli** się nadciągającą apokalipsą.
- Bo ci zrobię pokrzywkę - **straszy** Pająk.
- Jedne duchy **straszą** na zamku, drugie kąpią się w jeziorze, (...)

- Walenty jest dystrybuowany w stosunkowo wygodnym otwartym formacie tekstowym,
- Jest też dołączony do niego plik pdf, z drukowalnym (hmm...) słownikiem walencyjnym, wraz z przykładowymi zdaniami.
- Na stronie **14795** znajdziemy czasownik **programować** wraz z przymiotnikiem **programowalny**.

- Treebank jest zapisem działania pewnej gramatyki (być może hipotetycznej).
- Mając bank drzew, możemy łatwo otrzymać gramatykę (jak?)

Treebank i gramatyka (2)

Gramatyka z banku drzew

Dla drzewa postaci:

(S (NP (A mały) (N piesek)) (V skacze) (PP (Prep po) (N polu))))

możemy wydedukować produkcje:

Treebank i gramatyka (2)

Gramatyka z banku drzew

Dla drzewa postaci:

(S (NP (A mały) (N piesek)) (V skacze) (PP (Prep po) (N polu))))

możemy wydedukować produkcje:

- $S \rightarrow NP\ V\ PP$
- $NP \rightarrow A\ N$
- $PP \rightarrow Prep\ N$
- ...

Treebank i gramatyka (2)

Gramatyka z banku drzew

Dla drzewa postaci:

(S (NP (A mały) (N piesek)) (V skacze) (PP (Prep po) (N polu))))

możemy wydedukować produkcje:

- $S \rightarrow NP\ V\ PP$
- $NP \rightarrow A\ N$
- $PP \rightarrow Prep\ N$
- ...

Uwaga

Tak otrzymana gramatyka będzie generowała wszystkie zdania z Treebanku, z dokładnie takimi rozbiorami (i wiele innych zdań, i wiele innych rozbiorów dla istniejących zdań).

- Otrzymywanie gramatyki ze zdań może być całkiem praktycznym pomysłem, daje bowiem możliwość ocenienia naturalności pewnych konstrukcji (naturalne są takie, które zdarzają się częściej niż inne)
- Będziemy o tym mówić omawiając **probabilistyczne gramatyki bezkontekstowe** i ich warianty.

O wielości drzew rozbioru

Wspominaliśmy w kilku miejscach o tym, że pewne gramatyki generują wiele (setki?) drzew rozbioru. Popatrzmy na przykładowe zdanie:

Show me the meal on Flight UA 386 from San Francisco to Denver.

O wielości drzew rozbioru

Wspominaliśmy w kilku miejscach o tym, że pewne gramatyki generują wiele (setki?) drzew rozbioru. Popatrzmy na przykładowe zdanie:

Show me the meal on Flight UA 386 from San Francisco to Denver.

W zdaniu mamy 3 przyimki

O wielości drzew rozbioru

Rozważamy zdanie

Show me the meal on Flight UA 386 from San Francisco to Denver.

W gramatyce mamy (m.in.) produkcje:

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$PP \rightarrow Prep NP$

(...)

O wielości drzew rozbioru

Rozważamy zdanie

Show me the meal on Flight UA 386 from San Francisco to Denver.

W gramatyce mamy (m.in.) produkcje:

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$PP \rightarrow Prep NP$

(...)

Poprawny rozbiór to (tylko węzły NP):

O wielości drzew rozbioru

Rozważamy zdanie

Show me the meal on Flight UA 386 from San Francisco to Denver.

W gramatyce mamy (m.in.) produkcje:

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$PP \rightarrow Prep NP$

(...)

Poprawny rozbiór to (tylko węzły NP):

(Show me (the meal on (np Flight UA 386 from (np San Francisco) to Denver)))

O wielości drzew rozbioru

Rozważamy zdanie

Show me the meal on Flight UA 386 from San Francisco to Denver.

W gramatyce mamy (m.in.) produkcje:

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$PP \rightarrow Prep NP$

(...)

Poprawny rozbiór to (tylko węzły NP):

(Show me (the meal on (np Flight UA 386 from (np San Francisco) to Denver)))

W niepoprawnych rozbiorach pojawiają się takie frazy, jak:

O wielości drzew rozbioru

Rozważamy zdanie

Show me the meal on Flight UA 386 from San Francisco to Denver.

W gramatyce mamy (m.in.) produkcje:

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$PP \rightarrow Prep NP$

(...)

Poprawny rozbiór to (tylko węzły NP):

(Show me (the meal on (np Flight UA 386 from (np San Francisco) to Denver)))

W niepoprawnych rozbiorach pojawiają się takie frazy, jak:

- the meal on Flight UA 386
- San Francisco to Denver

O wielości drzew rozbioru

Inny przykład na wielość rozbioru:

*Wczoraj spotkałem dyrektora zjednoczenia zakładów
produkcji parówek.*

z gramatyką, zawierającą produkcje:

$np:gen \rightarrow np:gen \ np:gen$

O wielości drzew rozbioru

Inny przykład na wielość rozbioru:

*Wczoraj spotkałem dyrektora zjednoczenia zakładów
produkcji parówek.*

z gramatyką, zawierającą produkcje:

$np:gen \rightarrow np:gen \ np:gen$

Uwaga

CFG używane przez informatyków do opisów języków programowania są zwykle jednoznaczne (czy zawsze?), bo chcemy mieć jedno drzewo rozbioru programu, opisujące jego składnię i pośrednio semantykę. Dla języków naturalnych tak się nie da!

Czasem warto wiedzieć, który wyraz jest głównym wyrazem frazy.
Przykładowo:

krótkiej historii uczestnictwa kobiet w życiu publicznym
Starsza kobieta, ubrana w czarny przeciwdeszczowy
płaszcz z kapturem
Druga pod względem częstości
Mijali właśnie pierwszą z brzegu zagrodę
przez otwarte okno

Czasem warto wiedzieć, który wyraz jest głównym wyrazem frazy.
Przykładowo:

*krótkiej **historii** uczestnictwa kobiet w życiu publicznym*
Starsza kobieta, ubrana w czarny przeciwdeszczowy
płaszcz z kapturem
Druga pod względem częstości
Mijali właśnie pierwszą z brzegu zagrodę
przez otwarte okno

Czasem warto wiedzieć, który wyraz jest głównym wyrazem frazy.
Przykładowo:

*krótkiej **historii** uczestnictwa kobiet w życiu publicznym*
*Starsza **kobieta**, ubrana w czarny przeciwdeszczowy*
płaszcz z kapturem
Druga pod względem częstości
Mijali właśnie pierwszą z brzegu zagrodę
przez otwarte okno

Czasem warto wiedzieć, który wyraz jest głównym wyrazem frazy.
Przykładowo:

*krótkiej **historii** uczestnictwa kobiet w życiu publicznym*
*Starsza **kobieta**, ubrana w czarny przeciwdeszczowy*
płaszcz z kapturem
Druga *pod względem częstości*
Mijali właśnie pierwszą z brzegu zagrodę.
przez otwarte okno

Czasem warto wiedzieć, który wyraz jest głównym wyrazem frazy.
Przykładowo:

*krótkiej **historii** uczestnictwa kobiet w życiu publicznym*
*Starsza **kobieta**, ubrana w czarny przeciwdeszczowy*
płaszcz z kapturem
Druga pod względem częstości
Mijali właśnie pierwszą z brzegu zagrodę.
przez otwarte okno

Czasem warto wiedzieć, który wyraz jest głównym wyrazem frazy.
Przykładowo:

*krótkiej **historii** uczestnictwa kobiet w życiu publicznym*
*Starsza **kobieta**, ubrana w czarny przeciwdeszczowy*
płaszcz z kapturem
Druga pod względem częstości
Mijali właśnie pierwszą z brzegu zagrodę.
przez otwarte okno

Head i wielość rozbiorów

Parametr head może pomagać w wybieraniu właściwych rozbiorów.
Co niepokojącego występuje w rozbiorze:

*((dyrektor ((zjednoczenia (zakładów produkcji)))
parówek)*

Head i wielość rozbiorów

Parametr head może pomagać w wybieraniu właściwych rozbiorów.
Co niepokojącego występuje w rozbiorze:

((dyrektor ((zjednoczenia (zakładów produkcji)))
parówek)

Odpowiedź

Pojawia się nienaturalne połączenie **zjednoczenie** – **parówka**,
zamiast naturalnego **produkcja** – **parówka**.

Postać normalna Chomsky'ego

- Jedną z dwóch powszechnie używanych postaci normalnych gramatyki jest **Postać normalna Chomsky'ego (CNF)**

Postać normalna Chomsky'ego

- Jedną z dwóch powszechnie używanych postaci normalnych gramatyki jest **Postać normalna Chomsky'ego (CNF)**
- Mamy tylko dwa rodzaje produkcji:
 - $A \rightarrow BC$, gdzie A, B, C są symbolami nieterminalnymi
 - $A \rightarrow w$, gdzie A jest symbolem terminalnym, w jest symbolem terminalnym

Postać normalna Chomsky'ego

- Jedną z dwóch powszechnie używanych postaci normalnych gramatyki jest **Postać normalna Chomsky'ego (CNF)**
- Mamy tylko dwa rodzaje produkcji:
 - $A \rightarrow BC$, gdzie A, B, C są symbolami nieterminalnymi
 - $A \rightarrow w$, gdzie A jest symbolem terminalnym, w jest symbolem terminalnym

Part of speech

Produkcje $A \rightarrow w$ można traktować jako produkcje definiujące słownik morfosyntaktyczny.

Sprowadzanie do postaci normalnej

- Każdą gramatykę bezkontekstową opisującą język bez słowa pustego da się sprowadzić do postaci normalnej Chomskiego.
- Oczywiście drzewa będą wyglądać inaczej, ale język akceptowany przez gramatykę będzie dokładnie taki sam.

Sprowadzanie do postaci normalnej

- Każdą gramatykę bezkontekstową opisującą język bez słowa pustego da się sprowadzić do postaci normalnej Chomskiego.
- Oczywiście drzewa będą wyglądać inaczej, ale język akceptowany przez gramatykę będzie dokładnie taki sam.

Jak wygląda procedura?

- Produkcje zawierające po prawej stronie symbole terminalne i nieterminalne zamieniamy na produkcje zawierające same nieterminalne, wprowadzając (być może) dodatkowe symbole nieterminalne i nowe produkcje postaci $A \rightarrow w$.

Algorytm sprowadzanie do PNC

- Produkcje zawierające po prawej stronie symbole terminalne i nieterminalne zamieniamy na produkcje zawierające same nieterminalne, wprowadzając (być może) dodatkowe symbole nieterminalne i nowe produkcje postaci $A \rightarrow w$.
- Za pomocą dodatkowych symboli nieterminalnych „skracamy” prawe strony produkcji, by miały długość 2.

- Produkcje zawierające po prawej stronie symbole terminalne i nieterminalne zamieniamy na produkcje zawierające same nieterminalne, wprowadzając (być może) dodatkowe symbole nieterminalne i nowe produkcje postaci $A \rightarrow w$.
- Za pomocą dodatkowych symboli nieterminalnych „skracamy” prawe strony produkcji, by miały długość 2.

Przykład

Produkcję $A \rightarrow BCDE$ zamieniamy na 2 produkcje:

- $A \rightarrow BX_{CDE}$
- $X_{CDE} \rightarrow CDE$

- Dynamiczny algorytm parsingu dla gramatyk bezkontekstowych w postaci normalnej Chomskiego, sprawdzający, czy dane słowo należy do języka generowanego przez gramatykę.

Algorytm Cocke–Younger–Kasami

- Dynamiczny algorytm parsingu dla gramatyk bezkontekstowych w postaci normalnej Chomskiego, sprawdzający, czy dane słowo należy do języka generowanego przez gramatykę.
- Zakładamy, że mamy dane zdanie s .

- Dynamiczny algorytm parsingu dla gramatyk bezkontekstowych w postaci normalnej Chomskiego, sprawdzający, czy dane słowo należy do języka generowanego przez gramatykę.
- Zakładamy, że mamy dane zdanie s .
- Dla każdego i, j będziemy próbowali obliczać zbiór symboli nieterminalnych, takich że:

$$U(i, j) = \{X \mid X \Rightarrow^* s[i : j]\}$$

- Dynamiczny algorytm parsingu dla gramatyk bezkontekstowych w postaci normalnej Chomskiego, sprawdzający, czy dane słowo należy do języka generowanego przez gramatykę.
- Zakładamy, że mamy dane zdanie s .
- Dla każdego i, j będziemy próbowali obliczać zbiór symboli nieterminalnych, takich że:

$$U(i, j) = \{X \mid X \Rightarrow^* s[i : j]\}$$

- Pytanie o należenie to sprawdzenie, czy...

- Dynamiczny algorytm parsingu dla gramatyk bezkontekstowych w postaci normalnej Chomskiego, sprawdzający, czy dane słowo należy do języka generowanego przez gramatykę.
- Zakładamy, że mamy dane zdanie s .
- Dla każdego i, j będziemy próbowali obliczać zbiór symboli nieterminalnych, takich że:

$$U(i, j) = \{X \mid X \Rightarrow^* s[i : j]\}$$

- Pytanie o należenie to sprawdzenie, czy... $S \in U[i : \text{len}(s)]$

Algorytm CYK (2)

- Jeżeli $j = i + 1$, wówczas $U[i,i+1]$ jest równe:

Algorytm CYK (2)

- Jeżeli $j = i + 1$, wówczas $U[i, i+1]$ jest równe:

$$\{X | (X \rightarrow s[i]) \in P\}$$

gdzie P jest zbiorem produkcji

Algorytm CYK (2)

- Jeżeli $j = i + 1$, wówczas $U[i, i+1]$ jest równe:

$$\{X | (X \rightarrow s[i]) \in P\}$$

gdzie P jest zbiorem produkcji

- Dla $j - i > 1$ zakładamy, że mamy policzone U dla mniejszych rozpiętości i

Algorytm CYK (2)

- Jeżeli $j = i + 1$, wówczas $U[i, i+1]$ jest równe:

$$\{X | (X \rightarrow s[i]) \in P\}$$

gdzie P jest zbiorem produkcji

- Dla $j - i > 1$ zakładamy, że mamy policzone U dla mniejszych rozpiętości i

$$U(i, j) = \bigcup_{k=1, \dots, j} \{X | (X \rightarrow AB) \in P \wedge A \in U[i, k] \wedge B \in U[k, j]\}$$

Algorytm CYK (2)

- Jeżeli $j = i + 1$, wówczas $U[i, i+1]$ jest równe:

$$\{X | (X \rightarrow s[i]) \in P\}$$

gdzie P jest zbiorem produkcji

- Dla $j - i > 1$ zakładamy, że mamy policzone U dla mniejszych rozpiętości i

$$U(i, j) = \bigcup_{k=1, \dots, j} \{X | (X \rightarrow AB) \in P \wedge A \in U[i, k] \wedge B \in U[k, j]\}$$

Uwaga

Na przyszłym wykładzie zajmiemy się innymi algorytmami parsingu.

- Mamy dwie pętle zewnętrzne po i oraz j oraz pętlę wewnętrzną po k sprawdzającą różne podziały
- Do tego pętla po regułach gramatyki

Złożoność CYK

- Mamy dwie pętle zewnętrzne po i oraz j oraz pętlę wewnętrzną po k sprawdzającą różne podziały
- Do tego pętla po regułach gramatyki

Złożoność CYK

Biorąc to pod uwagę otrzymujemy $\Theta(n^3 \cdot |G|)$

Złożoność CYK

- Mamy dwie pętle zewnętrzne po i oraz j oraz pętlę wewnętrzną po k sprawdzającą różne podziały
- Do tego pętla po regułach gramatyki

Złożoność CYK

Biorąc to pod uwagę otrzymujemy $\Theta(n^3 \cdot |G|)$

Uwaga

Gramatyka może mieć kilka tysięcy produkcji, zdanie – kilkanaście wyrazów, czyli wydaje się, że to trochę duża złożoność. Ale z drugiej strony w praktyce nie przeglądalibyśmy wszystkich produkcji (dlaczego?)

$U[i,j]$ mówi o różnych możliwych interpretacjach fragmentu $s[i:j]$.
Możemy się spodziewać, że tych interpretacji nie będzie bardzo dużo.

$U[i,j]$ mówi o różnych możliwych interpretacjach fragmentu $s[i:j]$.
Możemy się spodziewać, że tych interpretacji nie będzie bardzo dużo.

Przykładowo ciąg **duży stół** to może być:

$U[i,j]$ mówi o różnych możliwych interpretacjach fragmentu $s[i:j]$. Możemy się spodziewać, że tych interpretacji nie będzie bardzo dużo.

Przykładowo ciąg **duży stół** to może być:

- Fraza nominalna w mianowniku
- Fraza nominalna w bierniku
- Przekleństwo, takie jak „kurka wodna” (???)
- ???

$U[i,j]$ mówi o różnych możliwych interpretacjach fragmentu $s[i:j]$. Możemy się spodziewać, że tych interpretacji nie będzie bardzo dużo.

Przykładowo ciąg **duży stół** to może być:

- Fraza nominalna w mianowniku
- Fraza nominalna w bierniku
- Przekleństwo, takie jak „kurka wodna” (???)
- ???

Zatem może nam się bardziej opłacać przejrzeć wszystkie par nieterminali z A i B (a nie iteracja po gramatyce).

- Zaczyna od symbolu startowego, wybierając jego rozwinięcie (na zajęciach z Prologa powiedzielibyśmy „niedeterministycznie zgadując”)

- Zaczyna od symbolu startowego, wybierając jego rozwinięcie (na zajęciach z Prologa powiedzielibyśmy „niedeterministycznie zgadując”)
- Rekurencyjnie sprawdza kolejne symbole z wybranej produkcji.

- Zaczyna od symbolu startowego, wybierając jego rozwinięcie (na zajęciach z Prologa powiedzielibyśmy „niedeterministycznie zgadując”)
- Rekurencyjnie sprawdza kolejne symbole z wybranej produkcji.
- Potencjalny nawrót zdarza się w momencie konfrontacji symbolu terminalnego w gramatyce z wyrazem w zdaniu.

- Zaczyna od symbolu startowego, wybierając jego rozwinięcie (na zajęciach z Prologa powiedzielibyśmy „niedeterministycznie zgadując”)
- Rekurencyjnie sprawdza kolejne symbole z wybranej produkcji.
- Potencjalny nawrót zdarza się w momencie konfrontacji symbolu terminalnego w gramatyce z wyrazem w zdaniu.


```
import nltk  
nltk.app.rdparser()
```

Top Down vs Bottom Up

Różne algorytmy mogą działać w stylu TopDown albo Bottom Up.

Top Down vs Bottom Up

Różne algorytmy mogą działać w stylu TopDown albo Bottom Up.

Top down

Zaczynamy od symbolu startowego, próbujemy znaleźć drzewo rozbioru dla całego zdania. Nasze poszukiwania są ukierunkowane na sparsowanie zdania.

Top Down vs Bottom Up

Różne algorytmy mogą działać w stylu TopDown albo Bottom Up.

Top down

Zaczynamy od symbolu startowego, próbujemy znaleźć drzewo rozbioru dla całego zdania. Nasze poszukiwania są ukierunkowane na sparsowanie zdania.

Bottom up

Zaczynamy od pojedynczych wyrazów i próbujemy je łączyć w coraz większe struktury. Nawet, jak nie znajdziemy rozbioru dla zdania, to możemy znaleźć różne mniejsze, użyteczne rozbioru dla fraz (na przykład nominalnych).

Jakie gramatyki lubi RDP?

- Na pewno nie lubi lewostronnie rekurencyjnej! (bo się zapętla)
- Raczej fajne jest, jak może szybko „nawrócić”, jak widać że nic z danego rozbioru nie będzie.
- Czyli dobrze jest, żeby możliwie szybko pojawił się symbol terminalny.

Jakie gramatyki lubi RDP?

- Na pewno nie lubi lewostronnie rekurencyjnej! (bo się zapętla)
- Raczej fajne jest, jak może szybko „nawrócić”, jak widać że nic z danego rozbioru nie będzie.
- Czyli dobrze jest, żeby możliwie szybko pojawił się symbol terminalny.

Idealna produkcja

Największą radość sprawimy parserowi RDP, dając mu regułę postaci: $A \rightarrow wBCDE$, gdzie w jest symbolem terminalnym.

- Wszystkie produkcje mają postać $A \rightarrow wA_1 \dots A_n$ (n może być równe 0)
- Możemy założyć, że język nie zawiera słowa pustego (tak jak w postaci normalnej Chomsky'ego)

Postać normalna Greibach

- Wszystkie produkcje mają postać $A \rightarrow wA_1 \dots A_n$ (n może być równe 0)
- Możemy założyć, że język nie zawiera słowa pustego (tak jak w postaci normalnej Chomsky'ego)

Uwaga

Każdą gramatykę da się przekształcić do postaci normalnej Greibach przy zachowaniu akceptowanego języka.

Wyznaczmy postaci normalne dla następującej gramatyki (zielone są problematyczne):

$$NP \rightarrow Adj\ NP$$
$$NP \rightarrow NP\ Adj_2$$
$$Adj \rightarrow mały \mid głupi$$
$$Adj_2 \rightarrow brunatny \mid polarny$$
$$NP \rightarrow miś$$
$$NP \rightarrow NP\ i\ NP$$

Wyznaczmy postaci normalne dla następującej gramatyki (zielone są problematyczne):

$NP \rightarrow Adj\ NP$

$NP \rightarrow NP\ Adj_2$

$Adj \rightarrow mały \mid głupi$

$Adj_2 \rightarrow brunatny \mid polarny$

$NP \rightarrow miś$

$NP \rightarrow NP\ i\ NP$

- Parsing standardowego DCG przypomina parsing RDP (backtracking nie wymaga implementacji, bo jest realizowany przez mechanizm wykonywania Prologa)

- Parsing standardowego DCG przypomina parsing RDP (backtracking nie wymaga implementacji, bo jest realizowany przez mechanizm wykonywania Prologa)
- Jak inaczej napisać parsing TopDown w Prologu?

- Parsing standardowego DCG przypomina parsing RDP (backtracking nie wymaga implementacji, bo jest realizowany przez mechanizm wykonywania Prologa)
- Jak inaczej napisać parsing TopDown w Prologu?
- Widząc regułę postaci $A \rightarrow BCD$ możemy generować wszystkie „sensowne” podziały parsowanego tekstu na 3 części i rekurencyjnie parsować każdy z nich.

Przykłady na sensowne podziały:

Przykłady na sensowne podziały:

- Jeżeli wiemy, że żaden symbol nie generuje pustego ciągu, wówczas możemy rozpatrywać tylko podziały, w których każdy tekst ma długość niezerową.

Przykłady na sensowne podziały:

- Jeżeli wiemy, że żaden symbol nie generuje pustego ciągu, wówczas możemy rozpatrywać tylko podziały, w których każdy tekst ma długość niezerową.
- Możemy też znać i uwzględniać inne więzy nałożone na długości, na przykład że PP ma długość większą niż 2.

Przykłady na sensowne podziały:

- Jeżeli wiemy, że żaden symbol nie generuje pustego ciągu, wówczas możemy rozpatrywać tylko podziały, w których każdy tekst ma długość niezerową.
- Możemy też znać i uwzględniać inne więzy nałożone na długości, na przykład że PP ma długość większą niż 2.
- Jak na przykład C jest wyrazem (np. spójnikiem), to możemy to uwzględnić na etapie podziału

Przykłady na sensowne podziały:

- Jeżeli wiemy, że żaden symbol nie generuje pustego ciągu, wówczas możemy rozpatrywać tylko podziały, w których każdy tekst ma długość niezerową.
- Możemy też znać i uwzględniać inne więzy nałożone na długości, na przykład że PP ma długość większą niż 2.
- Jak na przykład C jest wyrazem (np. spójnikiem), to możemy to uwzględnić na etapie podziału
- Pewną wiedzę może nam dać również analiza tagów, przykładowo: NP w dopełniaczu powinna zawierać rzeczownik w dopełniaczu.

Algorytm Shift-Reduce

Innym algorytmem parsowania, który przedstawia dość istotne idee jest Shift-Reduce

Innym algorytmem parsowania, który przedstawia dość istotne idee jest Shift-Reduce

- Mamy w algorytmie dwie struktury:
 - Zbiór drzew (las), dla sparsowanych fragmentów

Innym algorytmem parsowania, który przedstawia dość istotne idee jest Shift-Reduce

- Mamy w algorytmie dwie struktury:
 - Zbiór drzew (las), dla sparsowanych fragmentów
 - Listę L zawierającą niesparsowany sufiks zdania

Innym algorytmem parsowania, który przedstawia dość istotne idee jest Shift-Reduce

- Mamy w algorytmie dwie struktury:
 - Zbiór drzew (las), dla sparsowanych fragmentów
 - Listę L zawierającą niesparsowany sufix zdania
- Operacja Shift oznacza przesunięcie wyrazu z listy i utworzenie jednowęzłowego drzewka z tym wyrazem

Innym algorytmem parsowania, który przedstawia dość istotne idee jest Shift-Reduce

- Mamy w algorytmie dwie struktury:
 - Zbiór drzew (las), dla sparsowanych fragmentów
 - Listę L zawierającą niesparsowany sufiks zdania
- Operacja Shift oznacza przesunięcie wyrazu z listy i utworzenie jednowęzłowego drzewka z tym wyrazem
- Operacja Reduce oznacza połączenie (zgodne z gramatyką) pewnej liczby drzew w nowe drzewo

Gramatyka do prezentacji algorytmu SR

Interesuje nas gramatyka, która umożliwia sparsowanie zdania:

The dog saw a man in the park

Gramatyka do prezentacji algorytmu SR

Interesuje nas gramatyka, która umożliwi sparsowanie zdania:

The dog saw a man in the park

Gramatyka:

NP -> Det N

Det -> a | the

N -> dog | man

PP -> Prep NP

VP -> NP V NP PP

V -> saw | read | killed

Algorytm SR w działaniu

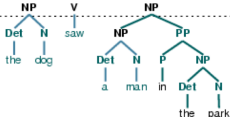
1. Initial state

Stack	Remaining Text
	the dog saw a man in the park

3. After reduce shift reduce

Stack	Remaining Text
Det N the dog	saw a man in the park


5. After building a complex NP

Stack	Remaining Text
	

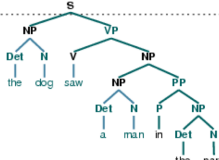
2. After one shift

Stack	Remaining Text
the	dog saw a man in the park

4. After recognizing the second NP

Stack	Remaining Text
	in the park

6. Built a complete parse tree

Stack	Remaining Text
	

- Nie mówiliśmy nic o nawracaniu, prosta implementacja SR parsera może nie znaleźć rozbioru!

- Nie mówiliśmy nic o nawracaniu, prosta implementacja SR parsera może nie znaleźć rozbioru!
- Można zaimplementować jakąś strategię decydowania o S i R.
Na przykład:

- Nie mówiliśmy nic o nawracaniu, prosta implementacja SR parsera może nie znaleźć rozbioru!
- Można zaimplementować jakąś strategię decydowania o S i R.
Na przykład:
 - Preferować reduce, jeżeli możliwe

- Nie mówiliśmy nic o nawracaniu, prosta implementacja SR parsera może nie znaleźć rozbioru!
- Można zaimplementować jakąś strategię decydowania o S i R.
Na przykład:
 - Preferować reduce, jeżeli możliwe
 - Mając do wyboru wiele redukcji wybierać tę, która „usunie” więcej drzew

- Nie mówiliśmy nic o nawracaniu, prosta implementacja SR parsera może nie znaleźć rozbioru!
- Można zaimplementować jakąś strategię decydowania o S i R. Na przykład:
 - Preferować reduce, jeżeli możliwe
 - Mając do wyboru wiele redukcji wybierać tę, która „usunie” więcej drzew

Uwaga

Jest dużo możliwości „uczenia” algorytmu parsingu: wynikiem tego uczenia miałyby być metoda wyboru S/R w zależności od danych. Mając rozbiór bowiem wiemy, dokładnie, jak powinien działać optymalny S/R parser, możemy zatem uczyć takiej strategii. Jak?

Algorytm Earleya

- Dynamiczny algorytm parsingu dla gramatyk w dowolnej postaci.
- Łatwo go uogólniać i opracowywać różne warianty
- Zaczniemy od podstawowej konstrukcji

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta \ [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Interpretacja

Każdy z powyższych napisów jest zdaniem, mówiącym:

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Interpretacja

Każdy z powyższych napisów jest zdaniem, mówiącym:

- W miejscu a próbuję znaleźć fragment zdania pasujący do nieterminala A ,

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta \ [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Interpretacja

Każdy z powyższych napisów jest zdaniem, mówiącym:

- W miejscu a próbuję znaleźć fragment zdania pasujący do nieterminala A ,
- używając do tego produkcji $A \rightarrow \alpha\beta$,

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta \ [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Interpretacja

Każdy z powyższych napisów jest zdaniem, mówiącym:

- W miejscu a próbuję znaleźć fragment zdania pasujący do nieterminala A ,
- używając do tego produkcji $A \rightarrow \alpha\beta$,
- udało mi się dojść do miejsca b

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta \ [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Interpretacja

Każdy z powyższych napisów jest zdaniem, mówiącym:

- W miejscu a próbuję znaleźć fragment zdania pasujący do nieterminala A ,
- używając do tego produkcji $A \rightarrow \alpha\beta$,
- udało mi się dojść do miejsca b
- zjadając część produkcji przed znakiem •

Struktura danych w Algorytmie Earleya

Algorytm Earleya (AE) tworzy w każdym z $N + 1$ punktów zdania o długości N tablicę informacji, wyglądających następująco:

$$A \rightarrow \alpha \bullet \beta [a:b]$$

gdzie $A \rightarrow \alpha\beta$ jest produkcją gramatyki, natomiast $0 \leq a \leq b \leq N + 1$, są pozycjami w tekście (jak w Pythonie)

Interpretacja

Każdy z powyższych napisów jest zdaniem, mówiącym:

- W miejscu a próbuję znaleźć fragment zdania pasujący do nieterminala A ,
- używając do tego produkcji $A \rightarrow \alpha\beta$,
- udało mi się dojść do miejsca b
- zjadając część produkcji przed znakiem •

- $S \rightarrow \bullet NP VP$ [0:0]
Zaczynamy analizę zdania.

- $S \rightarrow \bullet NP VP$ [0:0]
Zaczynamy analizę zdania.
- $NP \rightarrow AP \bullet NP$ [5:7]
Pierwsze 2 wyrazy na pozycji piątej z sukcesem sparsoowaliśmy jako frazę przymiotnikową, próbujemy dalej znaleźć frazę rzeczownikową.

- $S \rightarrow \bullet NP VP$ [0:0]
Zaczynamy analizę zdania.
- $NP \rightarrow AP \bullet NP$ [5:7]
Pierwsze 2 wyrazy na pozycji piątej z sukcesem sparsoowaliśmy jako frazę przymiotnikową, próbujemy dalej znaleźć frazę rzeczownikową.
- $N \rightarrow w \bullet$ [k:k+1]
Na pozycji k znajduje się słowo w

- $S \rightarrow \bullet NP VP$ [0:0]
Zaczynamy analizę zdania.
- $NP \rightarrow AP \bullet NP$ [5:7]
Pierwsze 2 wyrazy na pozycji piątej z sukcesem sparsoowaliśmy jako frazę przymiotnikową, próbujemy dalej znaleźć frazę rzeczownikową.
- $N \rightarrow w \bullet$ [k:k+1]
Na pozycji k znajduje się słowo w
- $S \rightarrow NP VP \bullet$ [0:10]
Sparsowaliśmy z sukcesem zdanie o długości 10.

- Dla każdej produkcji $S \rightarrow \alpha$ dodaje element do tablicy $\text{Chart}[0]$

- Dla każdej produkcji $S \rightarrow \alpha$ dodaje element do tablicy Chart[0]
- Elementem tym jest oczywiście:

$$S \rightarrow \bullet \alpha [0:0]$$

Dla każdego i i dla każdego elementu w $\text{Chart}[i]$ zastosuj jedną z 3 procedur

- Scanner (jeżeli stan jest niekompletny i mamy do przetworzenia POS)
- Predictor (jeżeli stan jest niekompletny i mamy do przetworzenie nie POS)
- Completer (jeżeli stan jest kompletny)

Definicja

Stan jest **kompletny** jeżeli na końcu jest • (czyli przetworzyliśmy wszystko).

- Stanem jest $A \rightarrow \alpha \bullet B\beta$ $[i:j]$
- B to POS słowa na pozycji j
- Czyli możemy je skonsumować i przesunąć się o 1 krok
- Dodajemy do $\text{Chart}[j+1]$ stan

$$B \rightarrow \text{word}[j] \bullet [j:j+1]$$

- Podobnie jak Scanner, ale następnym symbolem jest nieterminal, zatem musimy rozpocząć jego analizę.
- Stanem jest $A \rightarrow \alpha \bullet B\beta$ $[i:j]$
- Dla każdej produkcji $B \rightarrow \gamma$ dodamy do $\text{Chart}[j]$
- stan $B \rightarrow \bullet\gamma$ $[j:j]$

- Przetwarza reguły, które są już zakończone przesuając kropkę w innych stanach.
- Stanem jest $B \rightarrow \gamma \bullet [j:k]$
- Dla każdego stanu

$$A \rightarrow \alpha \bullet B\beta [i:j]$$

w Chart[j] dodaj do Chart[k] stan

$$A \rightarrow \alpha B \bullet \beta [i:k]$$