Parsery wykorzystywane w analizie języka naturalnego

1. Link Grammar Parser
2. Part of Speech Tagging
3. PCFG's
4. HPCFG's
5. Parser Charniak'a
6. LoPar
7. MiniPar

Autorzy: *Tomasz Masternak, Adam Łączyński*

Link Grammar Parser

- **Link Grammar Parser** - jest parserem języka angielskiego wykorzystującym gramatykę łączeń (ang. *Link Grammar*)

- **Gramatyka łączeń** – bazuje na zjawisku zwanym *planarnością*, które jest cechą większości języków naturalnych. Polega ono na tym, że jeśli poprowadzi się łuki pomiędzy powiązanymi ze sobą słowami, to łuki te nie przecinają się. Łuki obrazują relację pomiędzy parą połączonych słów – od grupowania wyrazów w pary wywodzi się nazwa gramatyki.

Gramatyka łączeń

- **Klasyczna gramatyka łączeń** składa się ze zbioru słów, które są terminalnymi symbolami gramatyki, z których każde posiada tzw. łączniki. Wyróżniamy dwa rodzaje łączników, tj.:
  - lewostronne
  - prawostronne

  Ponadto każdy łącznik posiada swoją etykietę, kierunek oraz słowo, z którym jest związany.

Kiedy zdanie należy do gramatyki łączeń?

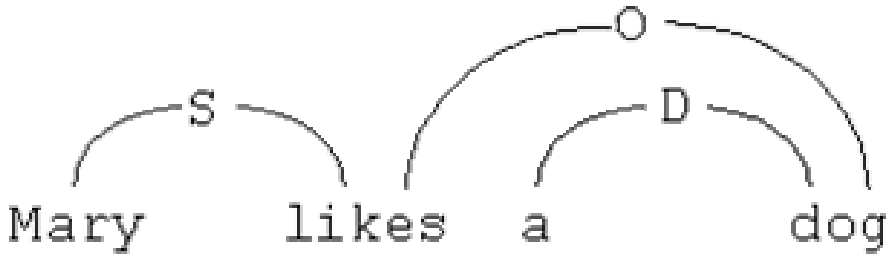
- Ciąg słów jest zdaniem języka opisywanego przez gramatykę łączeń, jeśli można poprowadzić łuki pomiędzy poszczególnymi wyrazami zdania w ten sposób, że spełnione są następujące warunki:

  1. łuki poprowadzone ponad słowami **nie przecinają się**
  2. łuki tworzą **graf spójny** tzn. od dowolnego punku wyrazu w ciągu istnieje ścieżka do dowolnie innego wyrazu w ciągu przy założeniu, że łuki nie są skierowane
  3. dla każdego słowa występującego w zdaniu spełnione są wymagania łączeniowe
  4. **łukiem** nazywamy połączenie dwóch łączników, lewostronnego i prawostronnego o tych samych etykietach

- Gramatyka łączeń reprezentowana jest przez słownik, który zawiera słowa oraz przyporządkowane mu wymagania łączeniowe.

Przykład – graficzna reprezentacja

Mary likes a dog



Graficzna struktura zdania wraz z połączeniami

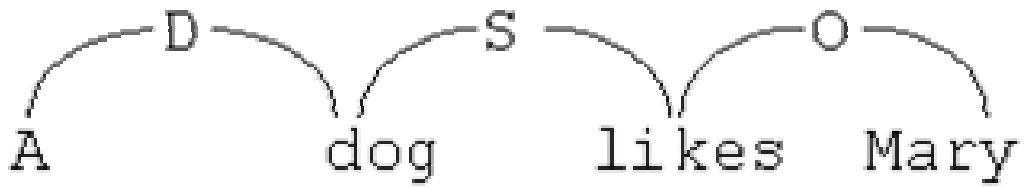
Każdy łuk posiada etykietę, w tym przykładzie znaczenie jest następujące:

- **O** – łączy orzeczenie z wyrazem prawej strony orzeczenia pełniącego najczęściej funkcję dopełnienia
- **S** – łączy orzeczenie z wyrazem stojącym po lewej stronie orzeczenia pełniącego funkcje podmiotu
- **D** – łączy przedimek z rzeczownikiem, którego on dotyczy

Graficzna reprezentacja cd.

- Na podstawie tych samych słów możemy stworzyć zdanie:

A dog likes Mary



- Zestaw łączników dla powyższych zdań jest inny, jest to możliwe gdyż z każdym zwykle związane jest wiele zestawów łączników.

Zestaw łączników jako formuła logiczna

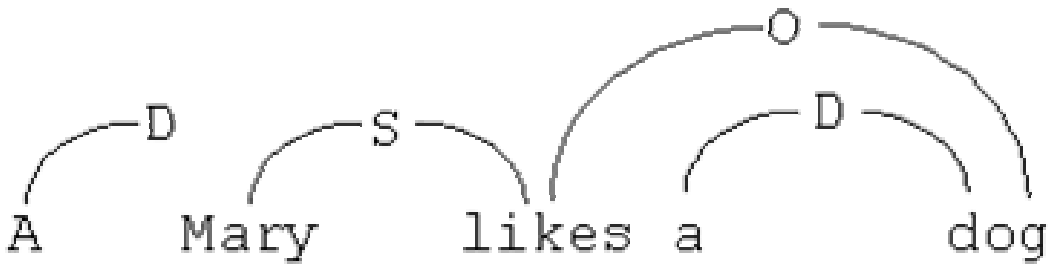
- Łączniki mogą być zapisane jako zestawy łączników lub jako formuła logiczna. Dla powyższego przykładu (*Mary likes a dog*) mamy następujące formuły logiczne:

Słowo	Lewostronny	Prawostronny	Formuła logiczna
a		D	D+
dog	D	S	D- And S+ Or O- And D-
	O D		
likes	S	O	S- And O+
Mary	O		O- Or S+
		S	

- Znak ”-” reprezentuje łącznik lewostronny natomiast znak ”+” łącznik prawostronny.

Zdanie niepoprawne gramatycznie

- Wymagania łączeniowe dla danego słowa są spełnione, jeśli w zdaniu dla wybranego zestawu łączników wzięły one wszystkie udział w budowie łuków z pozostałymi wyrazami. Poniżej znajduje się przykład zdania niepoprawnego gramatycznie:

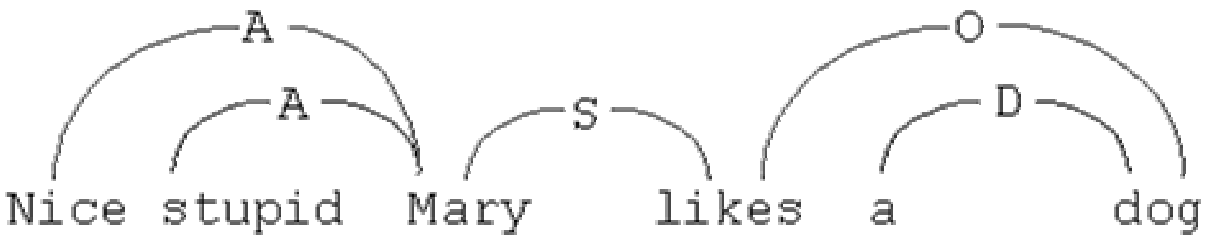


- Analiza powyższego przykładu uświadamia, że łączniki muszą mieć swoją hierarchię związaną z odległością słowa, z którym są one połączone. Łączniki o wyższej hierarchii łączą się ze słowami bardziej oddalonymi, niż łączniki o niższej hierarchii. Na pokazanym przykładzie dla słowa *dog* łącznik O ma wyższą hierarchię niż łącznik typu D.

Modyfikatory typu łącznika

- W języku angielskim są przypadki, że wyraz może się łączyć z innymi wyrazami poprzez ten sam typ łącznika, przy czym z góry nie można określić ilości tych łączników. Taki typ łącznika nazywa się **multiłącznikem** i wyróżniony jest poprzez przedrostek ”@”. Oznacza on, iż łącznik musi wystąpić co najmniej raz, np:

```
Mary: @A- & S+
```



- W gramatyce łączeń za pomocą typów łączników można sprawdzić zgodność liczby podmiotu i orzeczenia oraz zgodność przedimka z rzeczownikiem. Odbywa się to poprzez dodanie przyrostka do łącznika. Np. łącznik typu Ss oznacza łącznik typu S i liczbę pojedynczą.

Wady gramatyki łączeń

- Ponieważ każde słowo wymaga oddzielnej definicji łączeń, pojawienie się na wejściu nowego słowa wymaga zdefiniowania dla niego zbioru łączeń – zadanie trudne bez obszernej wiedzy dotyczącej gramatyki łączeń

Link Grammar Parser cd.

- Link Grammar Parser dostaje zdanie wejściowe, dla którego tworzy poprawne połączenia i drukuje ja w następującej formie:

Mary likes a dog

```
          +---Os---+
   +--Ss-+     +-Ds+
   |     |     |   |
Mary likes.v a dog.n

Constituent tree:

(S (NP Mary)
   (VP likes
       (NP a dog)))
```

- Parser dla pojedynczych zdań: http://www.link.cs.cmu.edu/link/submit-sentence-4.html

The Phrase Parser

- Link Grammar Parser zawiera **The Phrase Parser,** który na podstawie wygenerowanej struktury graficznej wyszczególnia grupy wyrazów takie jak:
  - S: clause:
    ”[S He said [S he left when [S he saw the dog [S I bought]]]]”
  - NP: noun phrase (fraza rzeczownikowa), ”I saw [NP the dog]”
  - VP: verb phrase (fraza czasownikowa),
    "The dog [VP will [VP run in the park]]“
  - PP: prepositional phrase (fraza przyimkowa)
    "The dog ran [PP in the park]”

```
(S  (NP Mary)
    (VP likes
        (NP a dog)))
```

Link Grammar Parser cd.

- Parser oznacza rozpoznane słowa przyrostkami takimi jak:

  ".n" - rzczownik

  “.v" - czasownik

  ".a" - przymiotnik

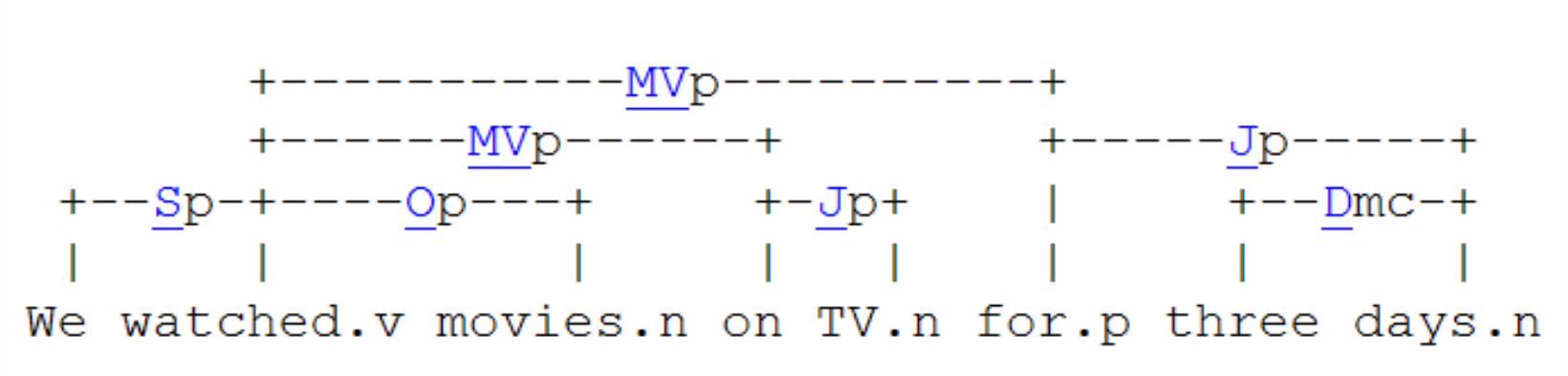
  ".e" - przysłówek

  “.[?]” – słowo nie występuje w słowniku

  [word] – słowo nie zostało dopasowane, tzw. null link

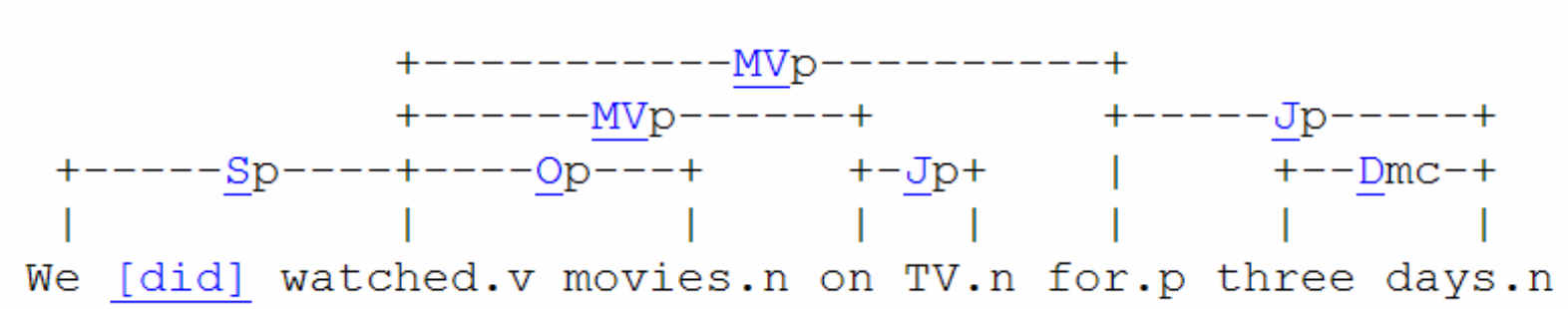
Link Grammar Parser, przykłady

We watched movies on TV for three days

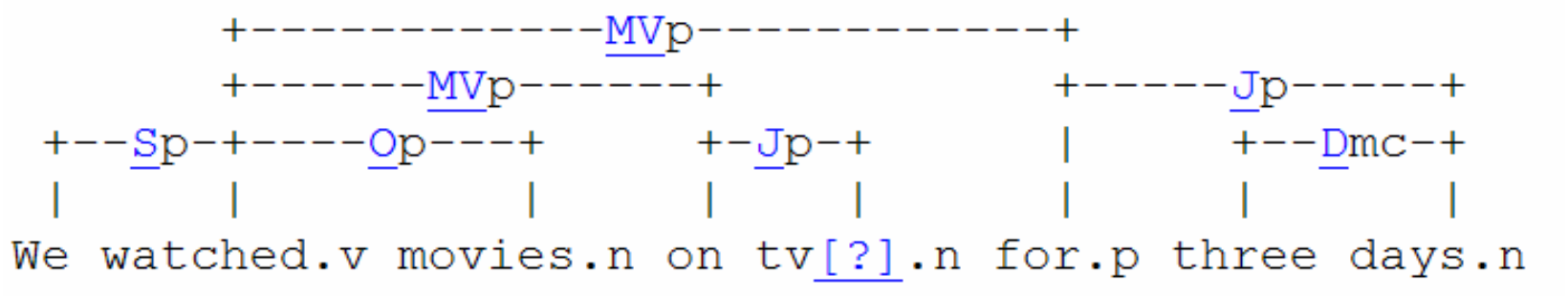


Link Grammar Parser, przykłady z błędami

*We **did** watched movies on TV for three days*



*We watched movies on **tv** for three days*



Niektóre własności Link Grammar Parser

- Wielkość liter: parser rozpoznaje czy wyraz powinien zaczynać się z duże czy małej litery (przykładowo imiona)
- Wyrazy połączone myślnikiem: chociaż nie wszystkie wyrazy składające się z 2 członów połączonych myślnikiem są przechowywane w słowniku parser akceptuje je
- Obsługa wartości liczbowych
- Obsługa interpunkcji oraz nawiasów
- Idiomy: można definiować idiomy łącząc wyrazy tworzące znakiem podkreślenia, przykładowo:

```
a_la_mode: A+ or B-;
```

Link Grammar Parser Podsumowanie

1. Link Grammar Parser napisany jest w języku C
2. Wykorzystanie go zewnętrznych aplikacjach jest bardzo łatwe
3. Słownik w który zaopatrzony jest parser zawiera ponad 60,000 wyrazów
4. Parser omija te części zdanie, które nie są dla niego zrozumiałe i analizuje dalsze część (oznaczając pominięte wyrazy nawiasami [])
5. Może inteligentnie wnioskować z kontekstu o kategorii nieznanego mu słowa
6. Przystosowany jest do parsingu zdań zapisanych przy użyciu *newspaper-style language*, a więc styl *conversational style* nie będzie poprawnie parsowany

http://www.link.cs.cmu.edu/link/

Metody statystyczne przetwarzania języka naturalnego

- Part of Speech Tagging.
- PCFG’s – Probabilistic Context Free Grammars.
- HPCFG’s – Head Lexicalized Probabilistic Context Free Grammars.

Charakterystyka wymienionych metod

- Part of Speech Tagging – jest to przetwarzanie języka naturalnego polegające na określaniu jaką częścią mowy jest każdy wyraz w aktualnie przetwarzanym zdaniu.
- Parsing z użyciem PCFG’s – wykorzystanie rozszerzonych gramatyk bezkontekstowych w celu usunięcia nie jednoznaczności charakterystycznych dla CFG.
- Parsing z użyciem HPCFG’s – rozszerzenie PCFG o dodatkowe informacje.

Part of Speech Tagging

- O co chodzi?

The	can	will	rust.
det	modal-verb	**modal-verb**	noun
	noun	noun	**verb**
	verb	verb	

frequency

Wiemy jakimi częściami mowy mogą być wyrazy, które przetwarzamy. Ponadto wiemy w jakiej roli najczęściej występują. Posiadając taką wiedzę chcemy tak dobrać tagi dla każdego wyrazu, aby odpowiadały rzeczywistej roli jaką on pełni.

Part of Speech Tagging cd.

- Propozycja rozwiązania

  Można zaproponować następujący sposób rozwiązania tego problemu. Mianowicie wybieramy dla każdego wyrazu to tagowanie które jest najbardziej prawdopodobne.

- Problem z proponowanym rozwiązaniem

  Oczywiście taki algorytm nie zdaje w praktyce egzaminu chociaż, jego poprawność dla losowej próbki języka angielskiego wynosi ok. 90% (wynika to z faktu, iż dużo słów posiada tylko jedno tagowanie ☺).

  Jednakże algorytm ten nie jest całkowicie bezużyteczny ponieważ jego matematyczna interpretacja podpowiada zmodyfikowane podejście, które daje dużo lepsze wyniki.

Part of Speech Tagging cd.

- Co naiwny algorytm robi tak naprawdę?

Skoro wiemy jak często występuje każde z tagowań to możemy mówić o prawdopodobieństwie wyboru poszczególnych tagów po natrafieniu na konkrety wyraz. Niech:

$p(t \mid w_i)$ prawd. wyboru tagu t dla i-tego wyrazu zdania w.

W takim przypadku nasz naiwny algorytm rozwiązuje następujące zagadnienie:

$$\arg\max_{t=1..n} \prod_{i=1}^{n} p(t_i \mid w_i)$$

tzn. w taki sposób dobiera tagi do wczytanych słów, aby iloczyn prawdopodobieństw był największy.

Part of Speech Tagging cd.

- Co jest nie tak?

Główny problem polega na tym, że nasze podejście nie uwzględnia pewnego istotnego faktu. Tagowanie aktualnego słowa nie zależy tylko i wyłącznie od niego ale także od kontekstu w jakim się znajduje.

Następujące spostrzeżenie nasuwa taką modyfikacje zagadnienia aby kontekst uwzględnić. Mianowicie rozwiązać zagadnienie następujące.

$$\arg\max_{t=1..n} \prod_{i} p(t_i \mid t_{i-1})\, p(w_i \mid t_i)$$

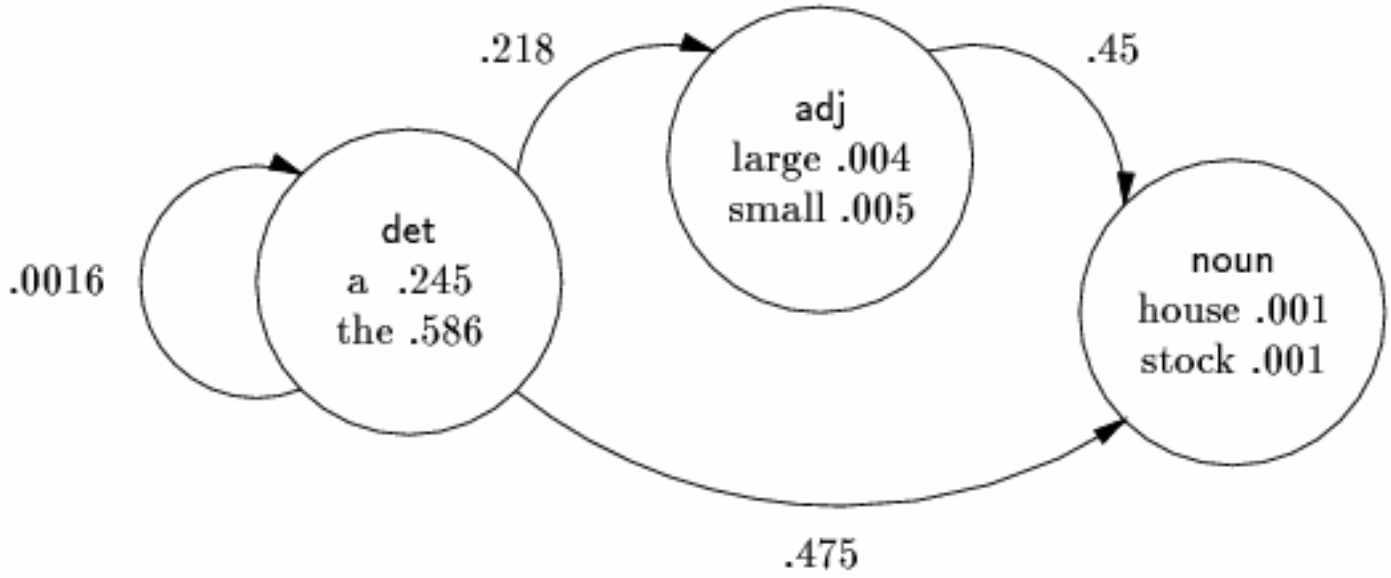
Uwzględniamy więc lewy kontekst aktualnego tag'u. Drugi człon nie wygląda zbyt intuicyjnie ale jest matematycznie poprawny.

Part of Speech Tagging cd.

- Hidden Markov Model

  Problem, który teraz otrzymaliśmy może zostać wyrażony w sposób równoważy przy wykorzystaniu HMM.

- Co to jest HMM?



Part of Speech Tagging cd.

- Jak możemy wyrazić nasz problem przy użyciu HMM?

  Nasze zagadnienie może zostać wyrażone w sposób następujący:

  Posiadając HMM dla przetwarzanego języka oraz traktując przetwarzane zdanie jako wyjście HMM znajdź najbardziej prawdopodobną trasę przejścia przez HMM.

  Zagadnienie to nie jest banalne można jednak je rozwiązać w korzystając z programowania dynamicznego, algorytm Viterbi'ego.

  W krótkim zarysie wygląda on tak, że najpierw wyliczamy dla każdego stanu prawdopodobieństwo dojścia do niego przez najbardziej prawdopodobną ścieżkę od długości k, a następnie na sukcesywnie wydłużamy długość ścieżki, aż osiągnie długość naszego zdania wejściowego.

Part of Speech Tagging cd.

- Jakie taggery, są dostępne?

  Istnieje dużo tagerów, nie wszystkie wykorzystują algorytm Viterbi'ego. Często są one częścią parserów języków natrualnych. Wśród dostępnych można wymienić:

  – TNT
    - Rozwijany na uniwersytecie Saarland,
    - Napisany w C pod Soarisa,
    - Możliwy do wykorzystania z każdym językiem pod warunkiem, że posiadamy korpus,
    - Dostępny na stornie: http://www.coli.uni-saarland.de/~thorsten/tnt/
  – LoPar
    - Zostanie omówiony w dalszej części prezentacji,
    - Tagger jest jedną z funkcji tego parsera

Probabilistic Context Free Grammars

- Dlaczego probabilistic?

  Okazuję się, iż możliwe jest wykorzystanie gramatyk bezkontekstowych w celu parsowania języków naturalnych. Korzyści w stosunku do tagger'ow są oczywiste ponieważ na wyjściu parsera otrzymujemy drzewo wyprowadzenia syntaktycznego przetwarzanej formy zdaniowej. Niestety CFG okazują się za słabe aby mogły być wykorzystywane w niezmienionej postaci.

  Rozważmy przykładową gramatykę oraz możliwe drzewa wyprowadzenia dla formy zdaniowej „Salespeople sold the dog biscuits".

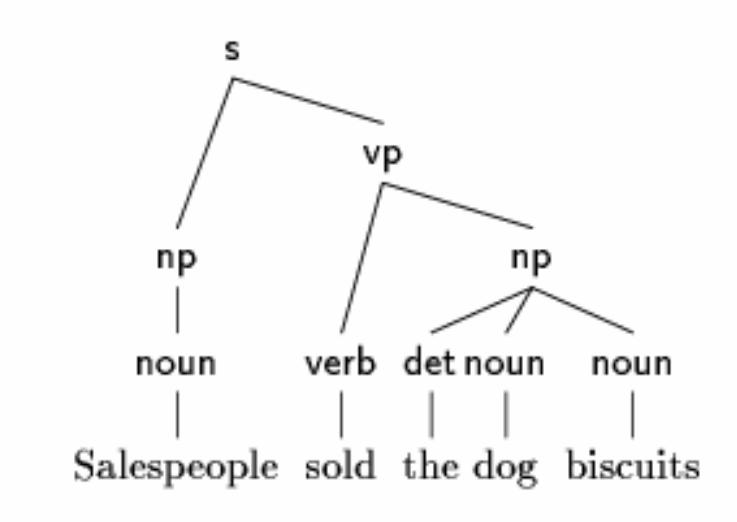
  Gramatyka:

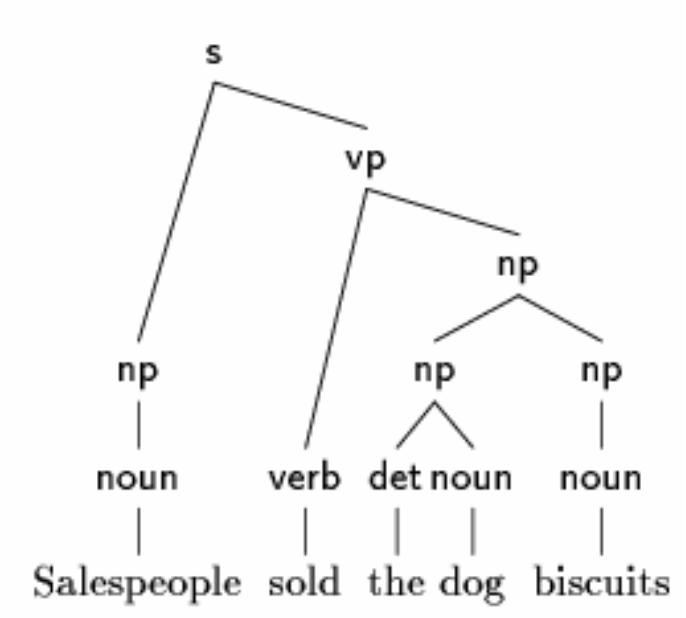
  s -> np vp

  np -> noun | det noun | np np | det noun noun

  vp -> verb np | verb np np

Probabilistic Context Free Grammars cd.





Probabilistic Context Free Grammars cd.

Jak wydać na poprzednim slajdzie otrzymujemy pewne wyprowadzenia, którym nie da się przypisać żadnego znaczenia. Jest to oczywiście problem. Aby go rozwiązać wprowadzono PCFG. Pomysł polega na tym aby każdej produkcji nadać pewne prawdopodobieństwo jej zastosowania, a następnie generować to drzewo wyprowadzenia, które jest najbardziej prawdopodobne.

Nasza przykładowa gramatyka mogłaby wyglądać następująco:

s	→	np vp	(1.0)	np	→	det noun	(0.5)
vp	→	verb np	(0.8)	np	→	noun	(0.3)
vp	→	verb np np	(0.2)	np	→	det noun noun	(0.15)
				np	→	np np	(0.05)

Probabilistic Context Free Grammars cd.

PCFG a CFG

- Mamy te same produkcje co w gramatyce początkowej,
- Z każdą produkcją związane jest prawdopodobieństwo jej zastosowania,
- Dla każdego nieterminala suma prawd. jego produkcji wynosi 1.
- Ponadto dla każdego drzewa wyprowadzenia określa się jego prawdopodobieństwo korzystając ze wzoru:

$$P(T) = \prod_{r} P(r)^{F(r)}$$

Gdzie T to drzewo wyprowadzenia, r to zastosowana produkcja, P(r) to prawdopodobieństwo zastosowania tej produkcji, a F(r) to ilość wystąpień danej produkcji w drzewie wyprowadzenia.

Probabilistic Context Free Grammars cd.

Jak działają parsery PCFG?

- Do tworzenia drzewa wyprowadzenia używa się dwóch algorytmów mianowicie algorytm CYK oraz algorytm Earley'a. Umożliwiają one znalezienie wszystkich drzew wyprowadzeń jak również tego najbardziej prawdopodobnego w czasie n^3, gdzie $n = |w|$.
- Tworzenie gramatyki potrzebnej do parsowania jest trudniejsze. Problemy na które natrafiamy to np. jakie produkcje umieścić w gramatyce, tak aby zdanie miało jakieś drzewo wyprowadzenia oraz jak przypisać prawdopodobieństwa do konkretnych produkcji. Na szczęście istnieje pewien sprytny sposób. Korzysta on z tzw. tree-banks. Są to ręcznie przetworzone zdania języka naturalnego wraz z ich poprawnymi drzewami parsingu. Postępujemy następująco: jeśli napotkamy nową produkcję to dodajemy ją do gramatyki, ponadto na podstawie częstotliwości wystąpień danej produkcji liczymy jej prawdopodobieństwo.

Head lexicalized Probabilistic Context Free Grammars

Okazuje się, iż można poprawić dokładność parsowania poprzez wprowadzenie dalszych zmian do gramatyki. HPCFG są to PCFG w których oprócz prawdopodobieństwa w każdej produkcji ustalamy który symbol po prawej stronie jest tzw. „lexical head”, a oznaczamy to apostrofem.

Intuicyjnie „head” jest to najważniejszy element w danej produkcji. Dla symbolu terminalnego jego głową jest on sam. W drzewie wyprowadzenia głowy wyliczane są bottom-up.

Algorytmy parsing’u dla gramatyk HPCFG działają podobnie jak dla PCFG uwzględniając dodatkowo wpływ wyznaczonych głów na prawdopodobieństwo danego wyprowadzenia.

Parser Charniak’a:

- Lexicalized PCFG’s,
- Dla Unix’a,
- Napisany w C++,
- Dostępny na stronie: ftp://ftp.cs.brown.edu/pub/nlparser/
- Obsługuje język Angelski oraz język Chiński,
- Dostajemy go z modulem umożliwiającym uczenie (znajduje się on w katalogu Train), jeśli mamy jakiś Bank-Tree (standardowo nie jest dostarczany z parserem) to możemy przeprowadzić proces treningu naszego parsera tzn. sprawić aby na podstawie banku drzew obliczył prawdopodobieństwa potrzebne do swojego działania,
- Program uruchamiamy poleceniem parseIt <kat. stat.>, standardowo w katalogu Data/En znajdują się prawd. dla języka angielskiego,
- Opcja –N powoduje wyliczenie n najbardziej prawdopodobnych drzew parkingu,

Parser Charniak’a:

- Parser pobiera dane ze standardowego wejścia i wypisuje je na standardowe wyjście. Kolejne wiersze muszą być ograniczone tagami <s> </s>
- Dla zdania <s> (``He'll work at the factory.'') </s>

  Parser zwróci nam następujące drzewo wyprowadzenia

```
(S1 (PRN (-LRB- -LRB-)
            (S (`` ``)
               (NP (PRP He))
               (VP (MD 'll)
                     (VP (VB work) (PP (IN at) (NP (DT the) (NN factory)))))
               (. .)
               ('' ''))
            (-RRB- -RRB-)))
```

LoPar

- Universtiy of Stuttgart:
  http://www.ims.uni-stuttgart.de/tcl/SOFTWARE/LoPar-en.html
- Napisany w Javie,
- Parsowanie CFG,
- Parsowanie PCFG oraz HPCFG,
- Trening PCFG oraz HPCFG,
- Tagowanie metodą Viterbie’go,
- …

LoPar

Opis każdej gramatyki składa się z następujących plików:

- .gram – opis produkcji gramatyki
- .lex – opis symboli leksykalnych
- .start – opis możliwych symboli początkowych
- .oc – opis postępowania z nieznanymi wyrazami

LoPar

Dane	Nazwa Pliku	Przykład
gramatyka	.gram	10 NP DET N’
leksemy	.lex	sleeps V 1 sleep
symbole początkowe	.start	S 1
wyrazy nieznane	.oc	V 10

LoPar daje możliwość tworzenia swoich gramatyk oraz określania częstotliwości na podstawie danych ćwiczebnych podobnie jak parser Charniak’a.

LoPar

Dla przykładowego wejścia:

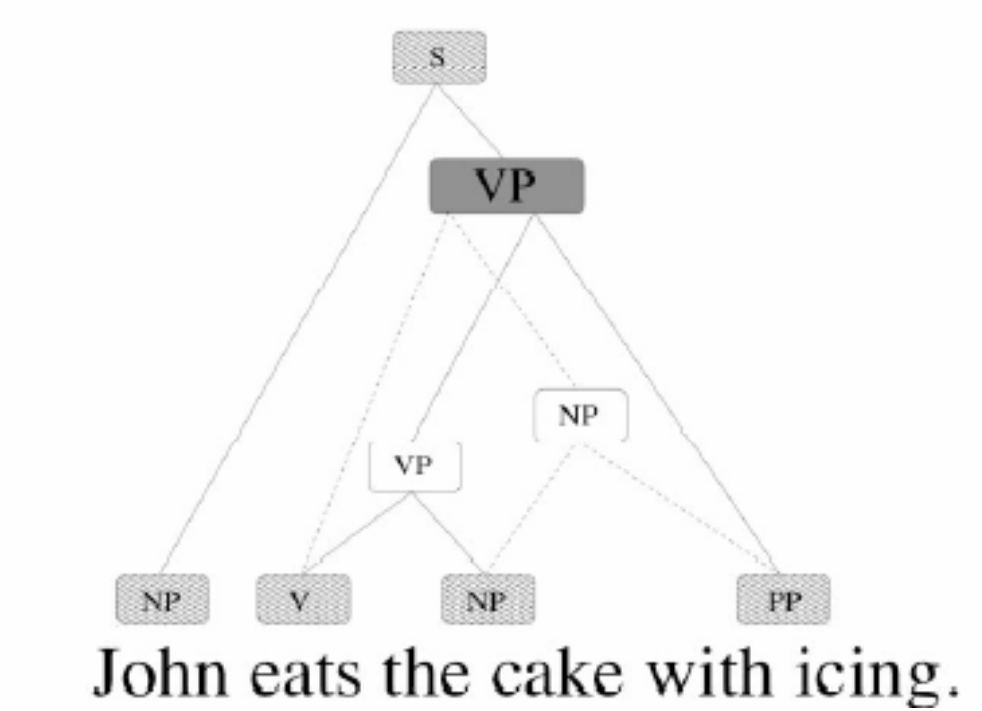
„The girl eats the cake with the strawbeeries”

Otrzymujemy:

S 0 8 4 1 4 %%
NP 0 2 1 2 3 %%
DET 0 1 **The** %%
N 1 2 **girl** %%
VP 2 8 6 5 6 % 8 5 7 10 % 9 15 10 %%
V 2 3 **eats** %%
NP 3 8 2 7 10 %%
NP 3 5 1 8 9 %%
DET 3 4 **the** %%
N 4 5 **cake** %%
PP 5 8 3 11 12 %%
P 5 6 **with** %%
NP 6 8 1 13 14 %%
DET 6 7 **the** %%
N 7 8 **strawbeeries** %%
VP 2 5 6 5 7 %%%

LoPar

Ręczna interpretacja nie jest potrzebna ponieważ LoPar dostarczony jest z programami do wizualizacji znalezionych drzew parsingu tj. vfs lub viewtree.



Parser MiniPar

- **MiniPar** - to parser języka angielskiego przedstawiający relacje gramatyczne występujące w zdaniu bazując na relacjach zależności

- **Relacja zależności** - jest to asymetryczna relacja binarna pomiędzy słowem zwanym głową (***head***), a drugim słowem zwanym modyfikatorem (***modifier***). Gramatyki zależności reprezentują zdanie jako zbiór relacji zależności.

- Wyrazy w zdaniu mogą mieć kilka modyfikatorów, ale każdy wyraz może modyfikować, co najwyżej jeden inny wyraz. Relacje zależności umożliwiają reprezentacje zdania jako **drzewa zależności**.

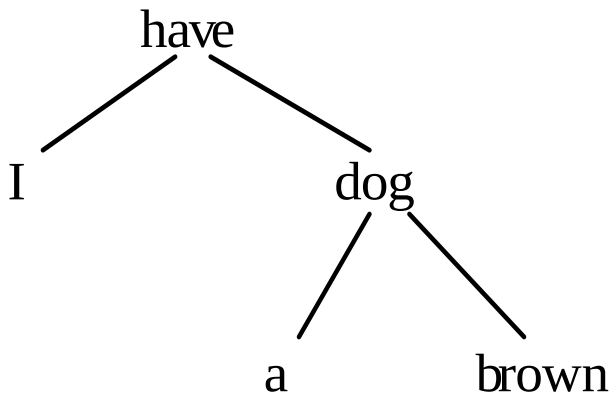
MiniPar – drzewo zależności

- W węzłach drzewa zależności zapisane są poszczególne słowa wchodzące w skład zdania. Struktura drzewa odzwierciedla zależności pomiędzy tymi słowami w ten sposób, że węzły podrzędne odpowiadają modyfikatorom węzła nadrzędnego. Korzeń drzewa nie modyfikuje innych wyrazów, jest on nazywamy głową zdania

- Elementem głównym jest orzeczenie, a pozostałe wyrazy stanowią jego modyfikatory (lub modyfikatory modyfikatorów).

- Przykładowe drzewo zależności dla zdania:

I have a brown dog



MiniPar cd.

- MiniPar reprezentuje gramatykę jako sieć, w której węzły reprezentują kategorie gramatyczne, powiązania między węzłami reprezentują rodzaj relacji zależności. Gramatyka ta zawiera 35 węzłów i 59 połączeń. MiniPar wyposażony jest w leksykon zawierający około 130000 pozycji.

- Parser ten tworzy wszystkie możliwe drzewa zależności dla zdania wejściowego i spośród nich wybiera jedno o najlepszym dopasowaniu. Najlepsze dopasowanie wybierane jest na podstawie informacji statystycznych zawartych w korpusie będącym częścią składowa parsera.

MiniPar cd.

- Przykładem korpus jest korpus ”Susanne” będący podzbiorem korpusu ”The Brown Corpus of American English”. Zawiera on drzewa zależności dla 64 z 500 tekstów ”The Brown Corpus of American English”.

- Parser MiniPar wydobywa około 80% do 87% relacji syntaktycznych obecnych w tekście, z czego około 89% relacji syntaktycznych wskazanych jest poprawnie.

- http://www.cs.ualberta.ca/~lindek/minipar.htm
- http://www.cfilt.iitb.ac.in/archives/minipar_evaluation.pdf

Bibliografia

- http://www.link.cs.cmu.edu/link/
- strony man dla LoPar
- „LoPar Desing and Implementation” Helmut Schmid
- „Statistical Techniques for Natural Language Parsing” Eugene Charniak,
- http://www.ai.kaist.ac.kr/~jkim/cs570-2003/lecture-tp/HMM.ppt
- http://www.cs.ualberta.ca/~lindek/minipar.htm
- http://www.cfilt.iitb.ac.in/archives/minipar_evaluation.pdf