Parsery języków naturalnych

Arkadiusz Świerczek **Piotr Włodek**

Parsery języka naturalnego należą do rozległej dziedziny informatyki zwanej **NLP (Natural Language Processing)**.

Główne zadania tej dziedziny to:

wspomaganie prac z tekstami, tworzenie słowników, programów nauczających języki obce, dialog z komputerem w języku naturalnym (ludzkim), pisanym i mówionym;

analiza sensu/gramatyki istniejących tekstów, streszczenia;

komentowanie tekstu, wydobywanie informacji – szukarki!

Tłumaczenie maszynowe, analiza i synteza mowy.

Historia

1950, Wczesne wysiłki: identyfikacji słów kluczowych, prymitywne programy do tłumaczenia maszynowego.

1960, Usprawnienia: baza danych, indeksacja, dopasowanie do wzorców wypowiedzi, podstawowe analizy gramatyczne.

1970, Reprezentacja logiczna, analiza semantyczna, rola kontekstu, rozszerzone sieci semantyczne ATM.

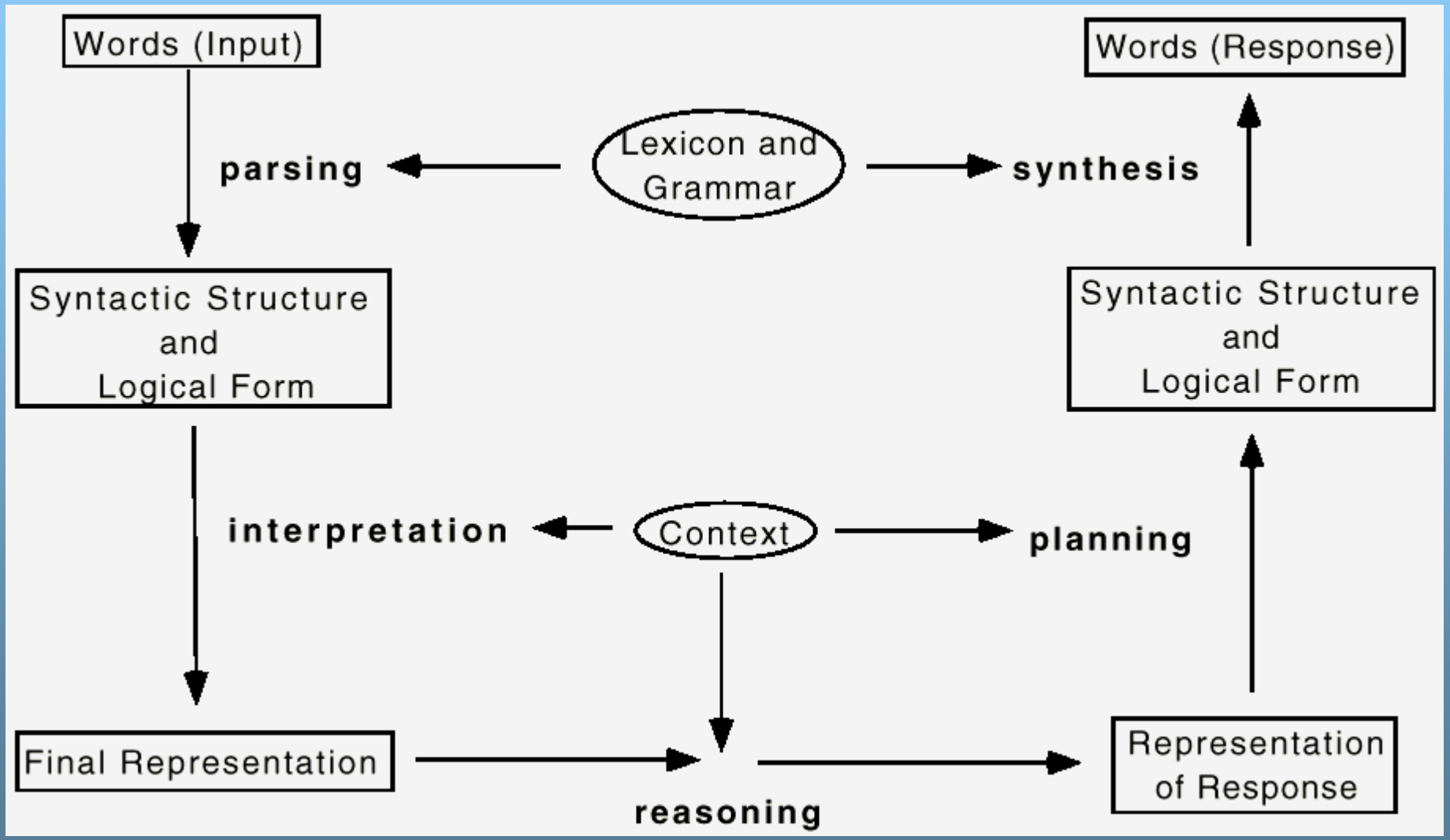
1975, Systemy wykorzystujące bazy wiedzy, wnioskujące, rozumujące.

1980, Programowanie logiczne, gramatyki ograniczeń, modele planowania do rozumienia i generacji tekstu.

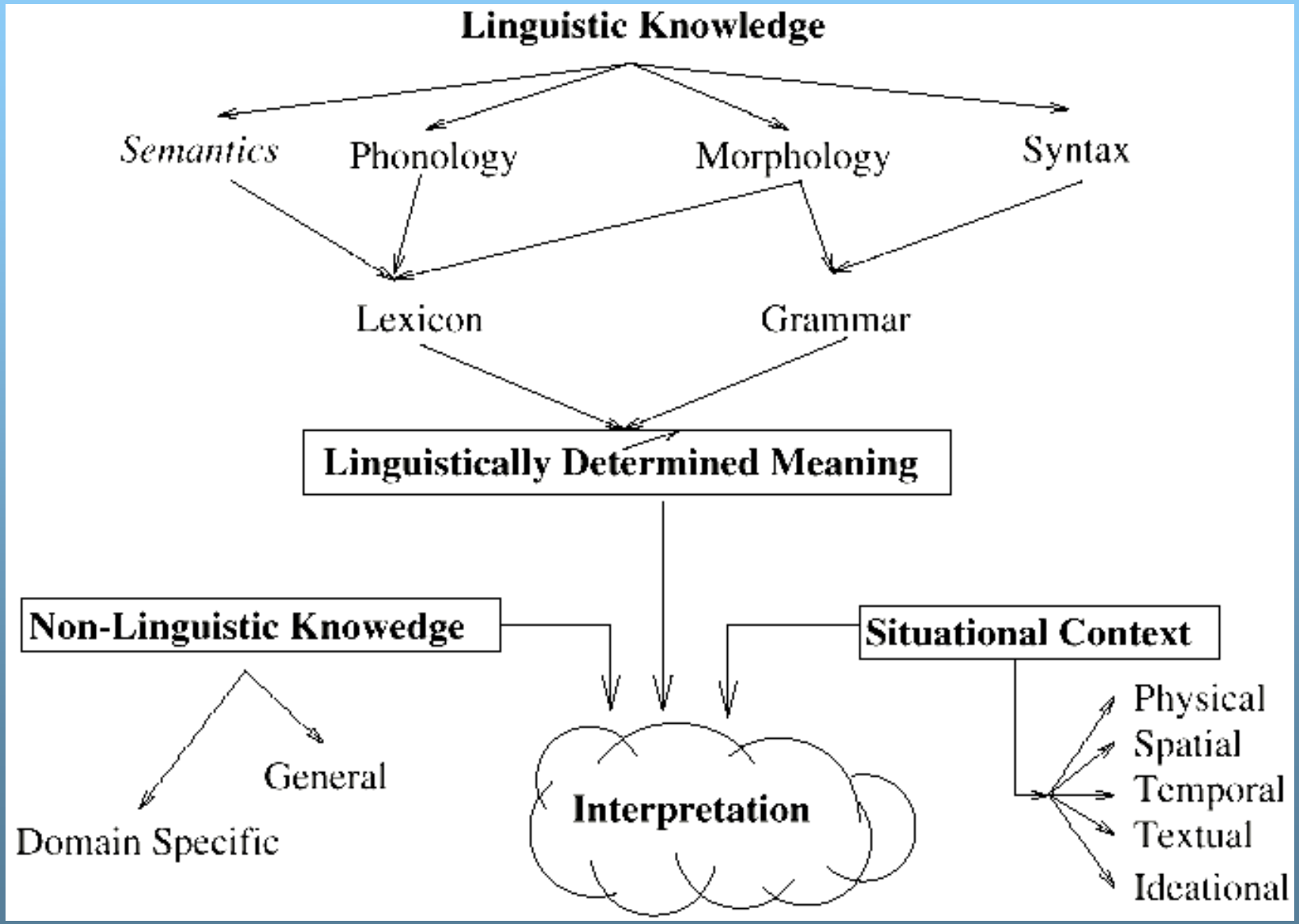
1990, Formalizacja sieci semantycznych, modele statystyczne, metody koneksjonistyczne, integracja mowy i tekstu, obszerne językowe bazy danych, pragmatyka języka.

2000, Gramatyki probabilistyczne (data oriented parsing)

Architektura systemów NLP



Język i wiedza



Kroki analizy

- **Fonologiczna** – fonetyka zajmuje się strukturą dźwięków; fonemy, formantami – istnieje międzynarodowy alfabet fonetyczny; fonologia zajmuje się organizacją fonemów w wypowiedziach.
- **Morfologiczna** – struktura słów, fleksja, rdzenie, leksykony wykorzystujące regularności języka na poziomie słów.
- ***Syntaktyczna*** *– konstrukcja zdań, rozbiór gramatyczny -> gramatyki i parsery*
- **Semantyczna** – sens, jednoznaczność, wybór słów, zależnie od kontekstu; markery semantyczne i organizacja leksykonu.
- **Pragmatyczna** – sposób użycia wyrazów, relacje zaimki-fraza; relacje między frazami, relacje przyczynowe, rozumienie całości, założenia.

Alan Turing test

To test który pozwala odpowiedzieć na niegdyś zadane przez Turinga pytanie: "Can a Machine Think?" Zakłada, że jeśli odpowiedzi maszyny będą nierozróżnialne od ludzkich, można nazwać maszynę myślącą. Co roku odbywa się międzynarodowy konkurs o nagrodę Loebner'a w dziedzinie sztucznej inteligencji polegający na imitowaniu człowieka przez systemy NLP w rozmowie tekstowej z sędziami, którzy oceniają poziom 'człowieczeństwa' systemu. Główna nagroda - $100 000 – przypada systemowi, który oszuka sędziów.

CORPUS

Gromadzenie wiedzy na temat języków – corpus (duża kolekcja pisanego bądź mówionego języka do badań lingwistycznych)

Próba usystematyzowania całego języka. Systemy takie na ogół wykorzystują gotowe teksty pisane – literackie, prasowe itp. lub mówione. Systematyzacja składa się z kilku etapów (na przykładzie Penn TreeBank):

- oznaczanie przez automatyczne systemy części języka (POS – part of speech) poszczególnych wyrazów (np. noun, verb itp.) Proces zwany czasem tagowaniem
- weryfikacja przez pracowników
- analiza syntaktyczna i oznaczenie części zdań przez parser
- ponowna weryfikacja danych przez pracowników

Oto przykładowe corpus'y języka angielskiego:

- BNC (British National Corpus)
- Brown Corpus
- Penn Treebank
- Susanne Corpus

W naszym referacie omówione zostaną parsery:

- Link Parser
  - http://www.link.cs.cmu.edu/
- MiniPar
  - http://www.cs.ualberta.ca/~lindek/minipar.htm
- parser Charniak’a
  - ftp://ftp.cs.brown.edu/pub/nlpparser

Link Grammar Parser

- Link Grammar Parser 4.0 jest parserem języka angielskiego opartym na Link Grammar
- Dokonuje rozbioru gramatycznego podanego zdania etykietując powiązane wyrazy
- Został napisany w C.
- Dostępny jest na wszystkie platformy gdzie dostępny jest kompilator C, w szczególności dla Windows i Unix
- Można korzystać z wersji zamieszczonej na stronie WWW
- Dostępne są nakładki graficzne dla wygodniejszego korzystania
- Projekt oparty jest na licencji umożliwiającej wykorzystanie komercyjne

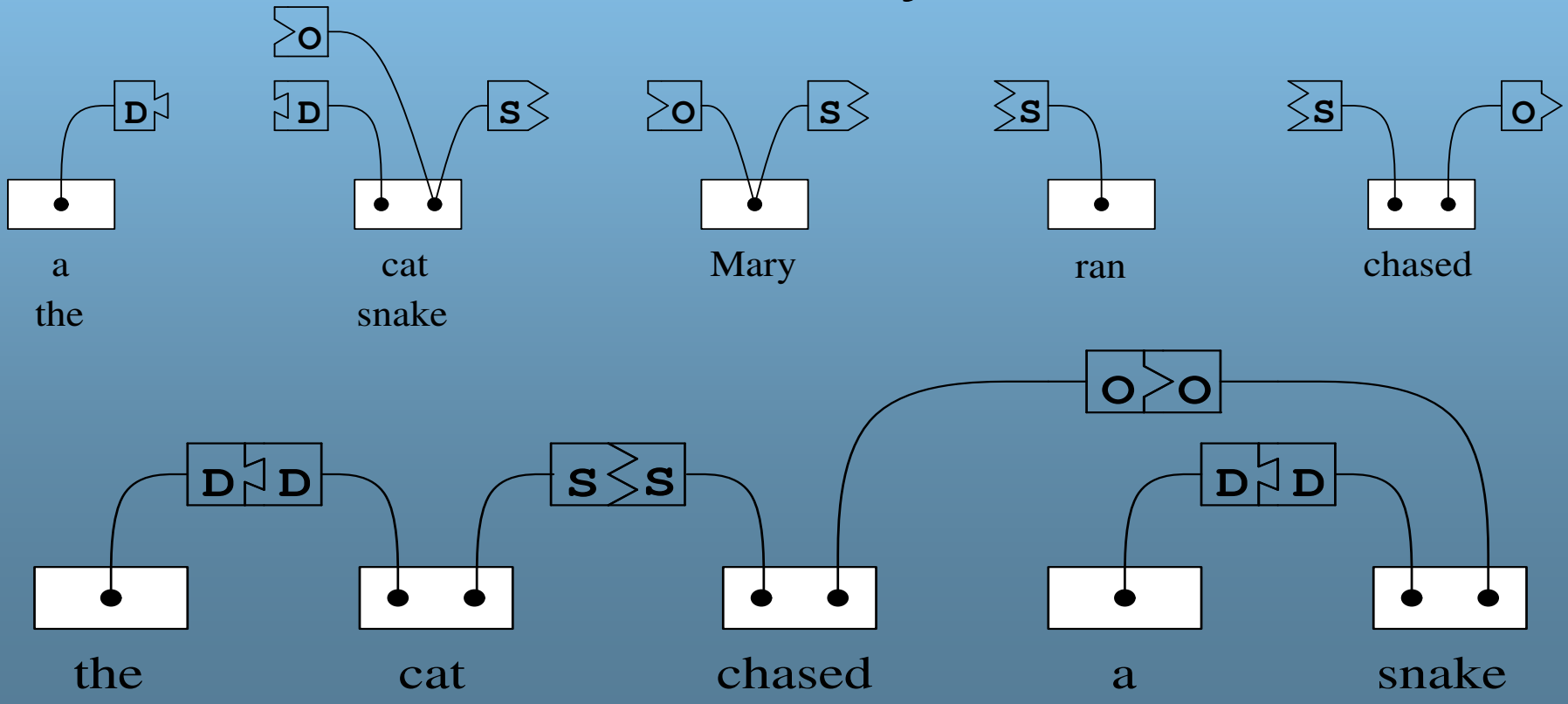
Link Grammar – idea i notacja

Słowa to bloki z których wychodzą konektory różnych rodzajów w dwie strony – lewo i prawo. Konektor wiążący od lewej (-) może być przyłączony do konektora tego samego typu wiążącego od prawej (+). Dwa konektory połączone ze sobą tworzą link. Wszystkie słowa mają reguły opisujące ich konektory i to jak mogą się one łączyć z innymi. Poprawne zdanie to takie w którym żadne reguły konektorów nie są naruszone.

Link Grammar – idea i notacja

- Przykładowe słowa:

  a, the, cat, snake, Mary, ran, chased



Word rules

Wszystkie konektory posiadają nazwy pisane z dużej litery, są zapisane w słowniku w którym każdy wyraz może się pojawić dokładnie raz. Przykład:

```
blah: A+;
```

Oznacza to, że jeśli użyjemy słowa `blah` w jakimś zdaniu, musi ono formować ”A” link, czyli po jego prawej stronie musi stać wyraz posiadający konektor A-.

Przykład cd:

```
blah: A+ & B+;
```

Oznacza to, że `blah` musi tworzyć ”A” link po prawej stronie oraz ”B” link także po prawej stronie. Ważna jest kolejność, a więc słowo tworzące ”A” link musi wystąpić przed słowem tworzącym ”B” link. Dzieje się tak dla tego ze konektory są skierowane w tym samym kierunku. Gdyby były skierowane w przeciwnym kierunku, kolejność mogłaby być dowolna.

Niektóre konektory mogą być opcjonalne, zapisuje się je wtedy pomiędzy ‘{‘ i ‘}’. Przykład:

```
blah: A+ & {B+};
```

Oznacza to że `blah` musi tworzyć ”A” link po prawej i może dodatkowo tworzyć ”B” link. Nazwę konektora można poprzedzić @ co będzie oznaczać, że dany konektor może wystąpić więcej niż jeden raz.

Przykład:

```
blah: A+ & {@B+};
```

Oznacza to że `blah` musi tworzyć ”A” link po prawej i może dodatkowo tworzyć po prawej dowolną ilość ”B” linków.

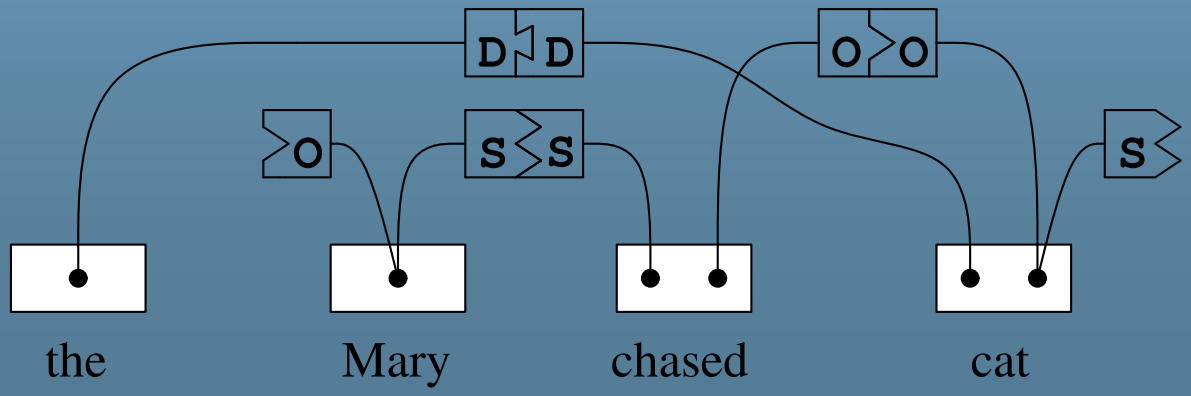
Connector subscripts

Konektory mogą się łączyć jeżeli są tego samego typu. Mogą także formować linki wykorzystując connector subscripts. CS to napis złożony z małych liter następujący po nazwie konektora, np. `"Sap+"`. Ten konektor może łączyć się z `"S-"`, `"Sa-"` oraz `"Sap-"` ale nie z `"Sb-"` czy `"Spa-"`.

Global rules

Poza zasadami dla słów określającymi konektory, definiowanymi w słowniku wyróżniamy dwie globalne zasady które muszą być spełnione przez parsowane zdanie.

- żadne linki nie mogą się krzyżować
- wszystkie wyrazy w zdaniu muszą być niebezpośrednio ze sobą połączone



Link Parser - uruchomienie

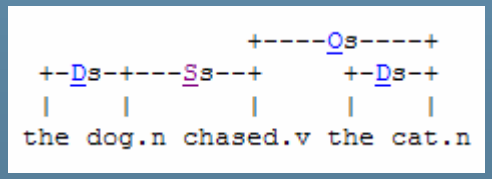
- Interfejs tekstowy – daje dodatkowe możliwośći
- Interfejs graficzny – nakładka, zwiększa wygodę używania
- Interfejs WWW – dostępny pod adresem http://www.link.cs.cmu.edu/link/submit-sentence-4.html

Przykład działania

Parsowane zdanie: *The black dog has gone.*

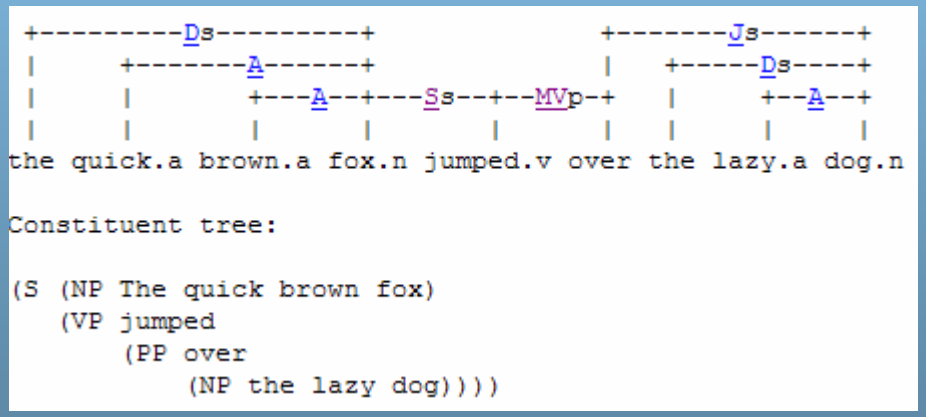
```
++++Time                                                0.00 seconds (464.12 total)
Found 1 linkage (1 with no P.P. violations)
  Unique linkage. cost vector = (UNUSED=0 DIS=0 AND=0 LEN=8)

 +-----Ds-----+
 |     +---A--+--Ss-+--PP--+
 |     |      |     |      |
the black.a dog.n has.v gone.v

Constituent tree:

(S (NP The black dog)
   (VP has
       (VP gone)))
```

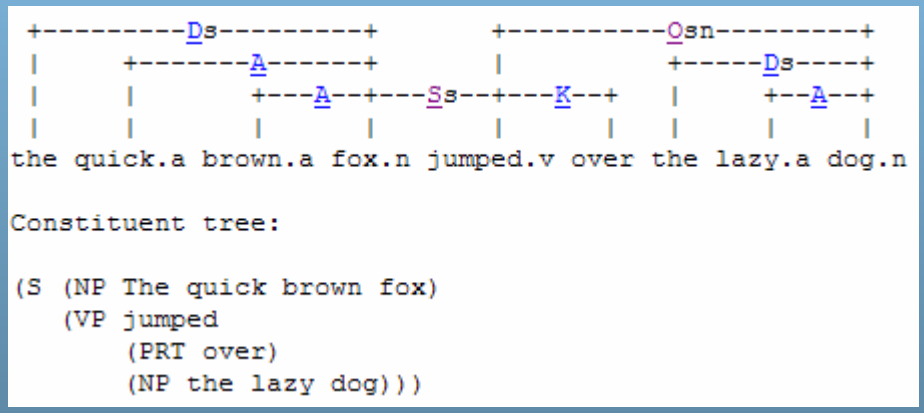
Parsowane zdanie: *The dog chased the cat*.



Przykład działania cd.

- Parsowane zdanie: *The quick brown fox jumped over the lazy dog.*





Podstawowe rodzaje linków

- **D** – łączy determiners z rzeczownikami
  np.: ***some*** birds, ***the*** dog, ***a*** cat
- **A** – łączy przymiotniki z rzeczownikami
- **S** – łączy podmiot z czasownikiem
- **PP** – łączy czasowniki formułujące *past participle*
- **O** – łączy czasowniki z obiektami
- **MV** – łączy czasowniki z przysłówkami, wyrażeniami czasowymi, itp.
- **J** – łączy przyimki ze słowami przez nie określanymi

Najważniejsze pliki

- Słownik – w nim zdefiniowane są wszystkie reguły dla wyrazów i powiązanych z nimi konektorów. Pojedynczy wyraz może w słowniku wystąpić dokładnie raz, dlatego też wprowadza się *word subscripts* określające rodzaj słowa np.: *run.v* i *run.n*. W słowniku są zdefiniowane także makra. Służą one do zapisywania skomplikowanych wyrażeń dla konektorów i późniejszego wielokrotnego wykorzystania.

Najważniejsze pliki cd.

- Pliki z wyrazami – w nich zapisane jest całe słownictwo. Pliki zawierające wyrazy mogą być logicznie podzielone ze względu na rodzaj słów w nich zawartych. Pliki wykorzystywane są w makrach do definiowania konektorów dla całych grup wyrazów.

MiniPar

MiniPar

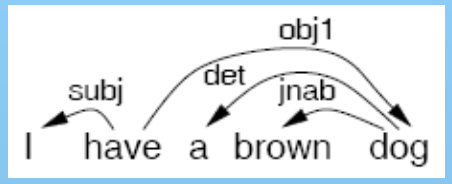
Jest oparty o tzw. dependency grammar (gramatyka zależności). Gramatyka taka przedstawia powiązania między słowami w postaci binarnej niesymetrycznej relacji zależności.

- Head (governor, parent) – „głowa” relacji. Każdy wyraz może być „głową” wielu relacji

- Modifier (dependent, daughter) to drugi element relacji. Każde słowo może ‘modyfikować’ tylko co najwyżej jedno inne słowo. Słowo, które nie modyfikuje żadnego innego to korzeń (root) drzewa wszystkich relacji

- Category – to kategoria językowa, do której należy słowo

- Type – typ relacji



Modifier	Category	Head	Type
I	N	< have	subj
have	V		
a	Det	< dog	spec
brown	Adj	< dog	adjn
dog	N	> have	comp

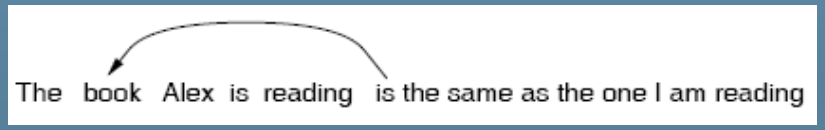
Na bazie tych relacji konstruuje się drzewo zależności. Przykład obok.

Wiedzę syntaktyczną o danym języku można zapisać jako tablicę zależności, która określa, jaka kategoria słów (POS – part of speech) może być w danej relacji z inną.

Ze względu na praktyczną nieefektywność takiego zapisu danych w parsingu tablicę taką przekształca się do tablicy parsingu. Tablica parsingu jest używana na podobnej zasadzie jak tablica w parserze shift/reduce, czyli dla [stan parsera, kategoria słowa wejściowego] określa, jaka operacja ma zostać podjęta.

Znak < w polu Head oznacza, że głową jest pierwsze na prawo wystąpienie słowa z pola Head. >> oznaczałoby, że słowo Head jest drugim na lewo wystąpieniem danego słowa z pola Head.

Przykładowo, zapis pokazanej poniżej relacji między ‘book’ i ‘is’ w polu Head to „<< is”.



[PRED chased V

[< SUBJ dog N

[< DET the Det]

[< ATTR big Adj]]

[> DIROBJ cat N

[< DET the Det]]]

Oto przykładowe zdanie przeparsowane za pomocą MiniPar:

The above evaluation ignores the labels assigned to the dependency relationships

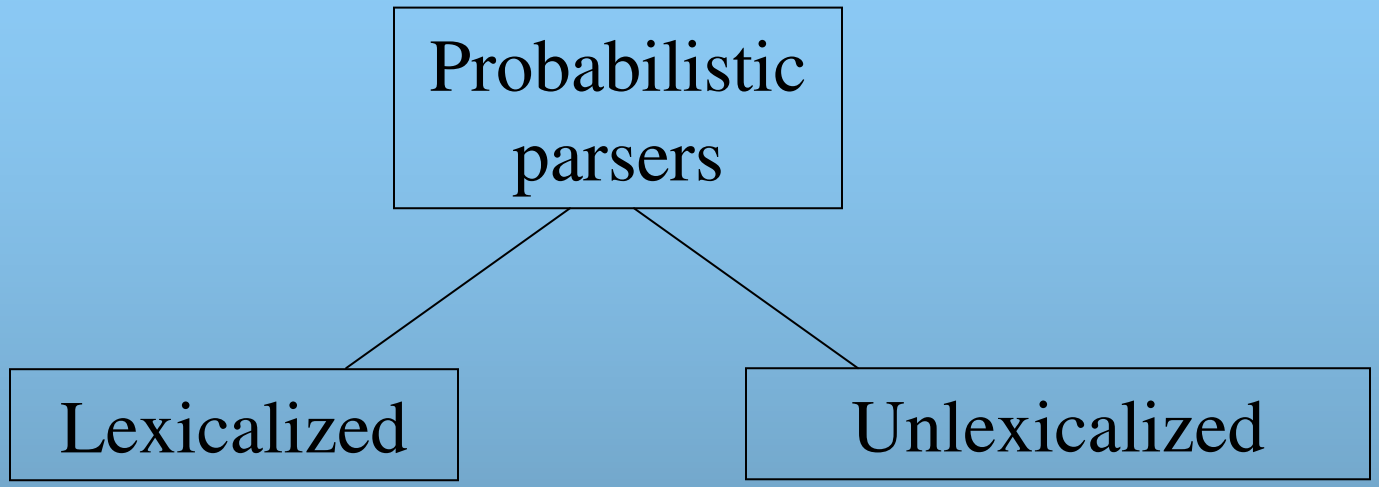
label	word	base	wtype	ref	pos	gov	antecedent
E0		fin	C	*			
1	The	~	Det	3	det	evaluation	
2	above	~	A	3	mod	evaluation	
3	evaluation	~	N	4	s	ignore	
4	ignores	ignore	V	E0	i	fin	
E2		evaluation	N	4	subj	ignore	3
5	the	~	Det	6	det	label	
6	labels	label	N	4	obj	ignore	
7	assigned	assign	V	6	vrel	label	
E3		label	N	7	obj	assign	6
8	to	~	Prep	7	mod	assign	
9	the	~	Det	11	det	relationship	
10	dependency	~	N	11	nn	relationship	
11	relationships	relationship	N	8	pcomp-n	to	

Oto link do parsera online
http://141.225.14.229/Minipar/WebForm1.aspx

Twórca MiniPar – Dekang Lin – przeprowadził testy skuteczności swojego parsera w porównaniu z frazami z Susanne Corpus. Parser wykazał się 89% precyzji w znajdowaniu właściwych relacji.

Charniak's Parser

Probabilistic Parsers



- Acts on individual words
- Exhaustive lexicon and rules – not so easy to build
- Data sparsity problem rampant

- Acts on word categories
- Easier to build
- Computational costs low
- Data sparsity problem is less rampant

Obliczanie prawdopodobieństw w prostym parserze

- Prawdopodobieństwo użycia produkcji r(c) : to prawdopodobieństwo rozwinięcia składnika c z użyciem produkcji r(c) w porównaniu do innych możliwych produkcji

- Prawdopodobieństwo użycia ciągu produkcji π: obliczane jako iloczyn prawdopodobieństw wszystkich użytych produkcji r(c)

$$p(s,\pi) = \prod_{c} p(r(c))$$

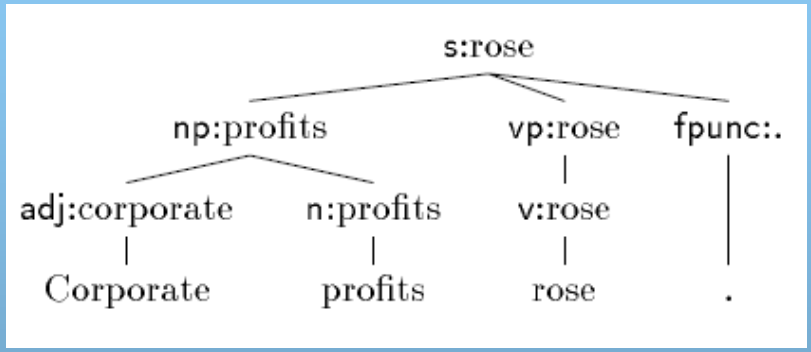
sentence

parse

Multiplying the rule probabilities

Oto przykład prostego drzewa wywodu dla zdania:

Corporate profits rose.

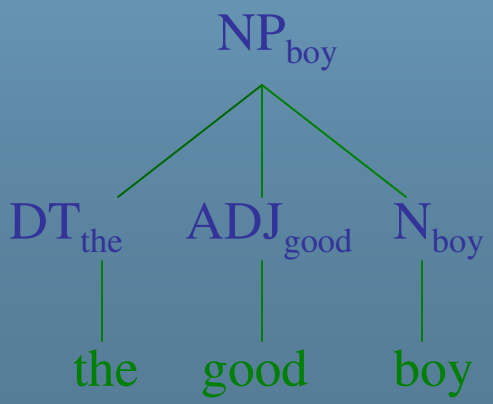


W każdym węźle nie będącym liściem drzewa oprócz nieterminala zapisany jest head, czyli główne słowo tej produkcji, np. dla *vp* głową jest *rose*.

Znajdowanie head ma bardzo istotne znaczenie dla parsera Charniaka, gdyż head wpływa na obliczanie prawdopodobieństw danych produkcji.

Rozszerzenie - Leksykalizacja

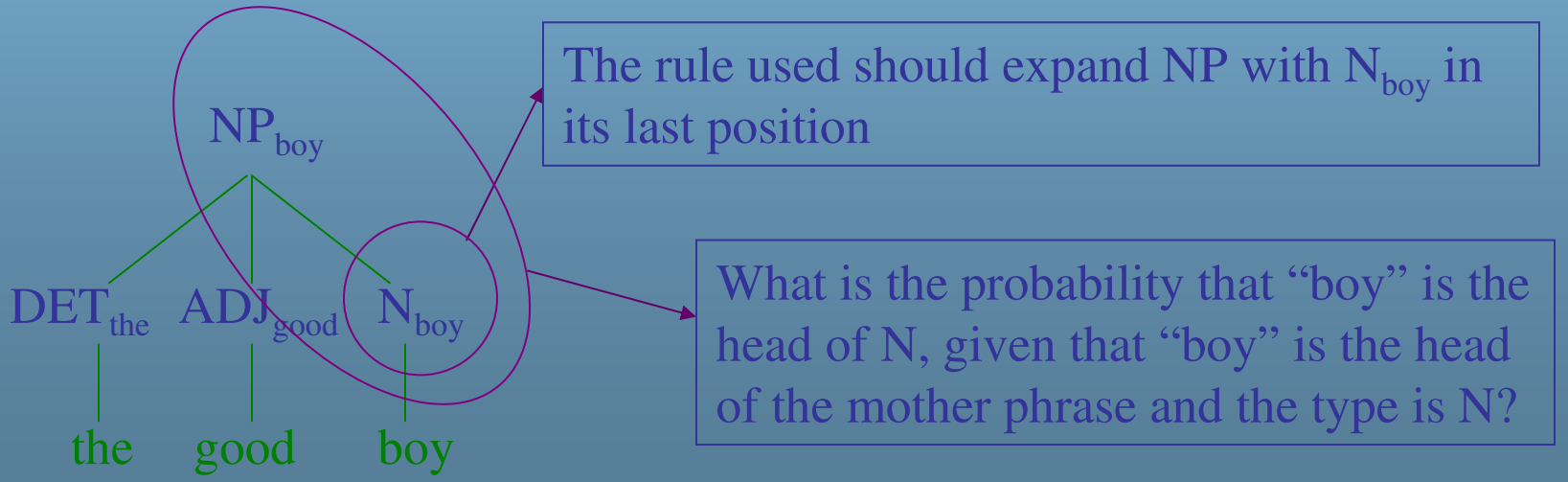
- Zamiast rozważać POS (części mowy) - rozważajmy konkretne wyrazy
- Problemem może być ‘sparsowalność’, gdyż nie wszystkie kombinacje słów jesteśmy w stanie wprowadzić do reguł parsera
- Rozwiązanie:
  - Użycie head – najważniejszego słowa frazy
  - Użycie produkcji biorąc pod uwagę jedynie zawartość head zamiast wszystkich słów produkcji



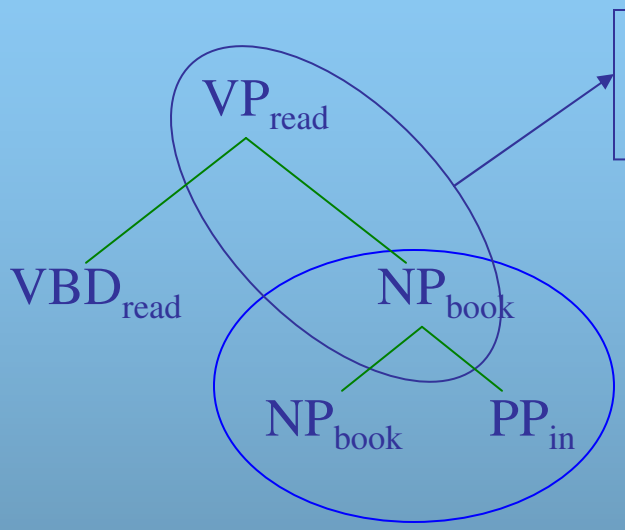
- “the good boy” may not be in corpus.
- But, “boy” will be there.
- Only combinations to be considered – heads of phrases and subphrases : (boy, the), (boy, good) & (boy, boy)

Potrzebne dane statystyczne

- P(rule | head) : Reguła (produkcja) użyta dla frazy opiera się również na głowie frazy – prawdopodobieństwo warunkowe – „użycia reguły ‘rule’ pod warunkiem że głową frazy jest ‘head’”
- P(head | head of mother phrase, type of subphrase) głowa frazy jest powiązana z głową frazy nadrzędnej oraz typem frazy



Przykład



Find P(head of phrase = “book” |
head of mother phrase = “read” & type of phrase is NP)

Find P(Rule is NP →NP PP|
head of phrase is “book”)

- P(sentence, parse): obliczane jako iloczyn tych dwóch prawdopodobieństw we wszystkich użytych produkcjach
  - P(rule | head)
  - P(head | head of mother phrase, type of subphrase)

„Trenowanie” parsera

- korzystamy z pewnego corpus’a zawierającego zbiór przeparsowanych fraz danego języka
- wczytujemy bezkontekstową gramatykę z corpus’a
- przetwarzamy zdania z corpus’a aby obliczyć odpowiednie prawdopodobieństwa produkcji

Parser deterministyczny a probabilistyczny

- Deterministyczny parser dokonuje domyślnych wyborów podczas parsingu
- Probabilistyczny parser dokonuje wyborów na podstawie prawdopodobieństw

- Deterministyczny parser zwraca pierwszy znaleziony poprawny ciąg produkcji
- Probabilistyczny parser zwraca najbardziej prawdopodobny wynik spośród wszystkich poprawnych wyników

- Deterministyczny parser nie rozwiązuje problemu syntaktycznej niejednoznaczności
- Probabilistyczny parser używa statystyk do rozwiązania tego problemu