Sztuczna inteligencja : Rozumienie języka naturalnego

(XV / XVI)

ĆWICZENIA

1. Sklasyfikuj każde z poniższych zdań jako niepoprawne składniowo, poprawne składniowo, ale bez znaczenia, znaczące, ale nieprawdziwe lub prawdziwe. Na którym etapie procesu zrozumienia każdy z tych problemów jest wykrywany?

Bezbarwne zielone pomysły śpią wściekle.

Owoce lecą jak banan.

Dogs The Bite Man a.

George Washington był piątym prezydentem USA.

To ćwiczenie jest łatwe.

Chcę być pod wodą w zacienionym ogrodzie ośmiornicy.

2. Omów struktury reprezentacyjne i wiedzę niezbędną do zrozumienia poniższych zdań.

Brązowy pies zjadł kość.

Przymocuj duże koło do osi za pomocą nakrętki sześciokątnej.

Mary podlała rośliny.

Duch jest chętny, ale ciało jest słabe.

Moje królestwo za konia!

3. Przeanalizuj każde z tych zdań, używając gramatyki „świata psów" z sekcji 15.2.1. Które z poniższych wyroków są nielegalne? Czemu?

Pies gryzie psa.

Duży pies gryzie mężczyznę.

Emma lubi chłopca.

Mężczyzna lubi.

Ugryź mężczyznę.

4. Rozszerz gramatykę świata psów, tak aby zawierała niedozwolone zdania w ćwiczeniu 3.

5. Przeanalizuj każde z tych zdań, korzystając z gramatyki kontekstowej z sekcji 15.2.3.

Mężczyźni lubią psa.

Pies gryzie mężczyznę.

6. Utwórz drzewo parsowania dla każdego z poniższych zdań. Będziesz musiał rozszerzyć nasze proste gramatyki o bardziej złożone konstrukcje językowe, takie jak przysłówki, przymiotniki i frazy przyimkowe. Jeśli zdanie ma więcej niż jedno parsowanie, zrób diagram wszystkich z nich i wyjaśnij semantyczne informacje, które zostaną użyte do wybrania parsowania.

Czas leci jak strzała, ale owoce lecą jak banan.

Tom dał Mary we wtorek dużą, czerwoną książkę.

Rozumowanie to sztuka, a nie nauka.

Błądzić jest rzeczą ludzką, przebaczać boskości.

7. Rozszerz gramatykę świata psów o przymiotniki we frazach rzeczownikowych. Upewnij się, że dopuszczasz nieokreśloną liczbę przymiotników. Wskazówka: użyj reguły rekurencyjnej adjective_list, która jest pusta lub zawiera przymiotnik, po którym następuje lista przymiotników. Zmapuj tę gramatykę na sieci przejściowe.

8. Dodaj następujące bezkontekstowe reguły gramatyczne do gramatyki świata psów w sekcji 15.2.1. Zmapuj wynikową gramatykę na sieci przejściowe.

zdanie ↔ noun_phrase verb_phrase prepositional_phrase

przyimek_fraza ↔ przyimek noun_phrase

przyimek ↔ z

przyimek ↔ do

przyimek ↔ on

9. Zdefiniuj parser ATN dla gramatyki świata psów z przymiotnikami (ćwiczenie 7) i wyrażeniami przyimkowymi (ćwiczenie 8).

10. Zdefiniuj pojęcia i relacje w grafach pojęciowych potrzebnych do przedstawienia znaczenia gramatyki Ćwiczenia 9. Zdefiniuj procedury budowania reprezentacji semantycznej z drzewa parsowania.

11. Rozszerz gramatykę kontekstualną w sekcji 15.2.3, aby sprawdzić zgodność semantyczną między podmiotem a czasownikiem. W szczególności mężczyźni nie powinni gryźć psów, chociaż psy mogą lubić lub gryźć mężczyzn. Wykonaj podobną modyfikację do gramatyki ATN.

12. Rozwiń gramatykę ATN w sekcji 15.2.4, aby uwzględnić pytania dotyczące kogo i jakie.

13. Opisz, jak można połączyć modele Markowa z sekcji 15.4 z bardziej symbolicznym podejściem do rozumienia języka z sekcji 15.1-15.3. Opisać, w jaki sposób modele stochastyczne można połączyć z tradycyjnymi systemami opartymi na wiedzy, rozdział 8.

14. Rozszerz przykład interfejsu bazy danych z sekcji 15.5.2, tak aby odpowiadał na pytania w formularzu „Ile zarabia Don Morrison?” Będziesz musiał rozszerzyć gramatykę, język reprezentacji i bazę wiedzy.

15. Weź poprzednie zadanie i ułóż jego słowa, w tym znaki interpunkcyjne, w przypadkowej kolejności.

16. Załóżmy, że menedżerowie są wymienieni w relacji pracownik_ wynagrodzenie z innymi pracownikami na przykładzie w sekcji 15.5.2. Rozszerz przykład, aby obsługiwał zapytania, takie jak „Znajdź pracownika, który zarabia więcej niż jego przełożony”.

17. W jaki sposób można połączyć podejścia stochastyczne z sekcji 15.4 z technikami analizy bazy danych, które opisano w sekcji 15.5.2.

18. Wykorzystanie podejścia stochastycznego do odkrywania wzorców w relacyjnej bazie danych jest ważnym obszarem obecnych badań, czasami nazywanym eksploracją danych (patrz Rozdział 10.3). Jak można wykorzystać tę pracę do udzielenia odpowiedzi na pytania, takie jak te postawione w sekcji 15.5.2, dotyczące relacyjnych baz danych?

19. Jako projekt zbuduj system wyodrębniania informacji dla jakiejś domeny wiedzy do wykorzystania w sieci WWW. Sugestie znajdują się w sekcji 15.5.3.

20. Z materiałów pomocniczych weźmy strukturę Prologu bezkontekstowych i kontekstowych parserów i przymiotników, przysłówków i zdań zdaniowych do gramatyki.

21. Z materiałów pomocniczych tej książki, przyjmij do gramatyki strukturę probabilistycznego parsera bezkontekstowego Prologu oraz przymiotników, przysłówków i zdań zdaniowych.

ROZUMIENIE JĘZYKA NATURALNEGO

15.0 Problem rozumienia języka naturalnego

Komunikowanie się za pomocą języka naturalnego, czy to w formie tekstu, czy mowy, w dużej mierze zależy od naszych umiejętności językowych, znajomości interesującej nas dziedziny i oczekiwań w tej dziedzinie. Zrozumienie języka to nie tylko przekazywanie słów: wymaga również wnioskowania o celach, wiedzy i założeniach mówcy, a także o kontekście interakcji. Wdrożenie programu rozumienia języka naturalnego wymaga, abyśmy przedstawiali wiedzę i oczekiwania dotyczące domeny oraz skutecznie je uzasadniali. Musimy wziąć pod uwagę takie kwestie, jak brak monotoniczności, weryfikacja przekonań, metafora, planowanie, uczenie się i praktyczne zawiłości interakcji międzyludzkich. Ale to są główne problemy samej sztucznej inteligencji! Rozważmy jako przykład następujące wersety z Sonetu XVIII Szekspira:

Czy mam cię porównać do letniego dnia?

Jesteś piękniejszy i bardziej umiarkowany:

Ostry wiatr wstrząsa ukochanymi pąkami maja,

A letnia dzierżawa ma zbyt krótki termin:

Nie możemy zrozumieć tych linii poprzez uproszczone, dosłowne potraktowanie znaczenia. Zamiast tego musimy zająć się takimi kwestiami, jak:

1. Jakie były intencje Szekspira na piśmie? Musimy poznać ludzką miłość i jej społeczne konwencje, aby zacząć rozumieć te kwestie. A może Szekspir próbował po prostu dostać coś do swojego wydawcy, żeby mu zapłacić?

2. Dlaczego Szekspir porównał swoją ukochaną do letniego dnia? Czy chodzi mu o to, że ma 24 godziny i może powodować oparzenia słoneczne, czy też sprawia, że czuje ciepło i piękno lata?

3. Jakich wniosków wymaga ten fragment? Zamierzone znaczenie Szekspira nie znajduje się bezpośrednio w tekście; należy ją wywnioskować za pomocą metafor, analogii i wiedzy podstawowej. Na przykład, jak możemy zinterpretować odniesienia do gwałtownych wiatrów i krótkiego lata jako lamentujące nad krótkością ludzkiego życia i miłości?

4. Jak metafora kształtuje nasze rozumienie? Słowa nie są jedynie odniesieniami do wyraźnych przedmiotów, takich jak klocki na stole: znaczenie wiersza polega na selektywnym przypisywaniu ukochanemu właściwości letniego dnia. Które właściwości są przypisywane, a które nie, a przede wszystkim dlaczego niektóre właściwości są ważne podczas gdy inni są ignorowani?

5. Czy komputerowy system zamiany tekstu na mowę musi wiedzieć coś na temat pentametru jambicznego? Jak komputer mógłby podsumować, o czym „jest" ten wiersz, lub w inteligentny sposób wydobyć to z korpusu poezji?

Nie możemy po prostu połączyć ze sobą słownikowych znaczeń słów Szekspira i nazwać wyniku zrozumieniem. Zamiast tego musimy zastosować złożony proces wychwytywania wzorców słów, analizowania zdań, konstruowania reprezentacji znaczeń semantycznych i interpretowania tych znaczeń w świetle naszej wiedzy o dziedzinie problemowej. Nasz drugi przykład to część reklamy internetowej na stanowisko wydziału informatyki. Wydział Informatyki Uniwersytetu Nowego Meksyku. . . prowadzi poszukiwania w celu obsadzenia dwóch etatów. Jesteśmy zainteresowani zatrudnieniem osób z zainteresowaniami: Oprogramowanie, w tym narzędzia analityczne, projektowe i programistyczne. . . Systemy, w tym architektura, kompilatory, sieci. . .

... Kandydaci musieli mieć doktorat w.. . Wydział posiada uznane na całym świecie programy badawcze z zakresu obliczeń adaptacyjnych, sztucznej inteligencji,. . . i cieszy się ścisłą współpracą badawczą z Santa Fe Institute i kilkoma krajowymi laboratoriami. . .

Przy zrozumieniu tego ogłoszenia o pracę pojawia się kilka pytań:

1. Skąd czytelnik może wiedzieć, że ta reklama dotyczy stanowiska naukowego, skoro wyraźnie podano tylko „tytuł stażu"? Na jak długo ludzie są zatrudniani na podstawie umowy o pracę?

2. Jakie oprogramowanie i narzędzia programowe są potrzebne do pracy w środowisku uniwersyteckim, gdy żadne z nich nie zostało wyraźnie wymienione? Cobol, Prolog, UML? Aby zrozumieć te oczekiwania, potrzebowałaby dużej wiedzy na temat nauczania i badań uniwersyteckich.

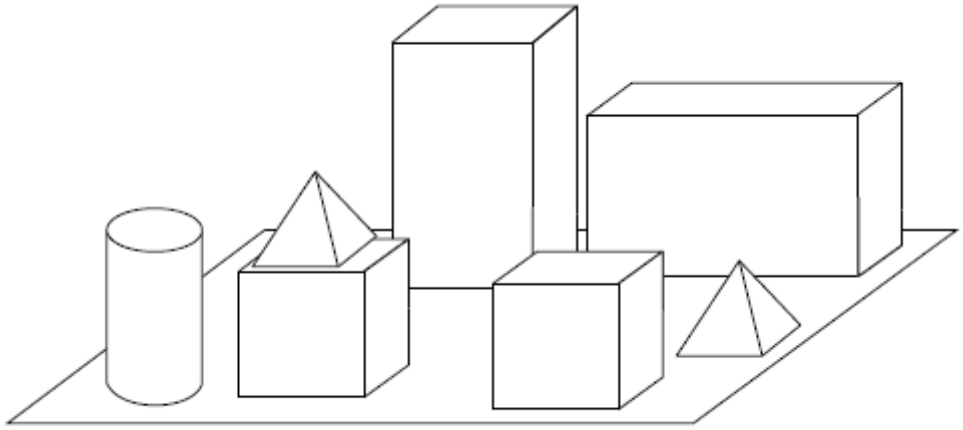
3. Co wspólnego z ogłoszeniem o pracy na uczelni mają uznane na arenie międzynarodowej programy i współpraca z interesującymi instytucjami?

4. Jak komputer mógłby podsumować, o czym jest ta reklama? Co musi wiedzieć, aby w inteligentny sposób pobrać z sieci to ogłoszenie dla doktoranta poszukującego pracy?

Istnieją (przynajmniej!) Trzy główne kwestie związane ze zrozumieniem języka. Po pierwsze, zakłada się dużą ilość wiedzy ludzkiej. Akty językowe opisują relacje w często złożonym świecie. Znajomość tych relacji musi być częścią każdego zrozumienia system. Po drugie, język jest oparty na wzorcach: fonemy są składnikami słów, a słowa tworzą frazy i zdania. Porządki fonemów, słów i zdań nie są przypadkowe. Komunikacja jest niemożliwa bez raczej ograniczonego użycia tych komponentów. Wreszcie akty językowe są wytworem agentów, czy to człowieka, czy komputera. Agenci są osadzeni w złożonych środowiskach o wymiarze indywidualnym i socjologicznym. Akty językowe są celowe.

Ta Część zawiera wprowadzenie do problemów rozumienia języka naturalnego i technik obliczeniowych opracowanych w celu ich rozwiązania. Chociaż w tym rozdziale skupiamy się przede wszystkim na rozumieniu tekstu, systemy rozumienia mowy i generowania również muszą rozwiązać

te problemy, a także dodatkowe trudności związane z rozpoznawaniem i ujednoznacznianiem słów osadzonych w określonym kontekście. Wczesne programy AI, ze względu na wiedzę wymaganą do zrozumienia nieograniczonego języka, poczyniły postęp, ograniczając się do mikroświatów, ograniczonych zastosowań która wymagała minimalnej wiedzy o domenie. Jednym z najwcześniejszych programów stosujących to podejście był SHRDLU Terry'ego Winograda, który mógł rozmawiać o świecie bloków składającym się z bloków o różnych kształtach i kolorach oraz dłoni do poruszania nimi, jak na rysunku 1.



SHRDLU może odpowiadać na zapytania w języku angielskim, takie jak „Co jest na czerwonym bloku?" „Jaki kształt ma niebieski klocek na stole?" lub „Umieść zieloną piramidę na czerwonej cegle". Może obsługiwać odniesienia do zaimków, takie jak „Czy istnieje czerwony blok? Podnieś to." Mógłby nawet zrozumieć elipsy, takie jak „Jakiego koloru jest blok na niebieskiej cegle? Kształt?" Ze względu na prostotę świata bloków udało się zapewnić systemowi odpowiednią wiedzę. Ponieważ świat bloków nie obejmował trudniejszych problemów zdroworozsądkowego rozumowania, takich jak zrozumienie czasu, przyczynowości, możliwości czy przekonań, techniki przedstawiania tej wiedzy były stosunkowo proste. Pomimo swojej ograniczonej dziedziny, SHRDLU dostarczył model integracji składni i semantyki oraz wykazał, że program z wystarczającą znajomością dziedziny dyskursu może komunikować się w sposób znaczący w języku naturalnym. Uzupełnieniem opisanego właśnie składnika rozumienia języka wymagającego dużej wiedzy jest modelowanie wzorców i oczekiwań samych wyrażeń językowych. Łańcuchy Markowa oferują potężne narzędzie do wychwytywania tych prawidłowości. Na przykład w używaniu języka przedimki i przymiotniki zazwyczaj poprzedzają rzeczowniki, a nie następują po nich, a niektóre rzeczowniki i czasowniki występują razem. Modele Markowa mogą również przechwytywać relacje między wzorcami językowymi a światami, które opisują. W sekcji 15.1 przedstawiamy analizę wysokiego poziomu rozumienia języka. Sekcja 15.2 przedstawia analizę składniową; sekcja 15.3 łączy składnię i semantykę przy użyciu rozszerzonego analizowania sieci przejść. Sekcja 15.4 przedstawia stochastyczne podejście do wychwytywania prawidłowości w wyrażeniach językowych. Wreszcie, w sekcji 15.5 rozważymy kilka zastosowań, w których przydatne są programy do rozumienia języka naturalnego: odpowiadanie na pytania, dostęp do informacji w bazach danych, zapytania internetowe i podsumowanie tekstu.

15.1 Dekonstruowanie języka: analiza

Język jest skomplikowanym zjawiskiem, obejmującym procesy tak zróżnicowane, jak rozpoznawanie dźwięków lub drukowanych liter, analiza składniowa, wysokiego poziomu wnioskowania semantyczne,

a nawet przekazywanie treści emocjonalnych poprzez rytm i fleksję. Aby sobie z tym poradzić złożoność, lingwiści opisali różne poziomy analizy języka naturalnego:

1. Prozodia zajmuje się rytmem i intonacją języka. Ten poziom analizy jest trudny do sformalizowania i często zaniedbywany; jednak jego znaczenie jest ewidentne w potężnym wpływie poezji lub pieśni religijnych, a także w roli odgrywanej przez rytm w dziecięcej zabawce słownej i gaworzeniu niemowląt.

2. Fonologia bada dźwięki, które są łączone w język. Ta gałąź językoznawstwa jest ważna dla komputerowego rozpoznawania i generowania mowy.

3. Morfologia dotyczy składników (morfemów), z których składają się słowa. Obejmują one zasady rządzące tworzeniem słów, takie jak wpływ przedrostków (un-, non-, anti- itp.) I sufiksów (-ing, -ly itp.), Które modyfikują znaczenie słów rdzeniowych. Analiza morfologiczna jest ważna przy określaniu roli słowa w zdaniu, w tym jego czasu, liczby i części mowy.

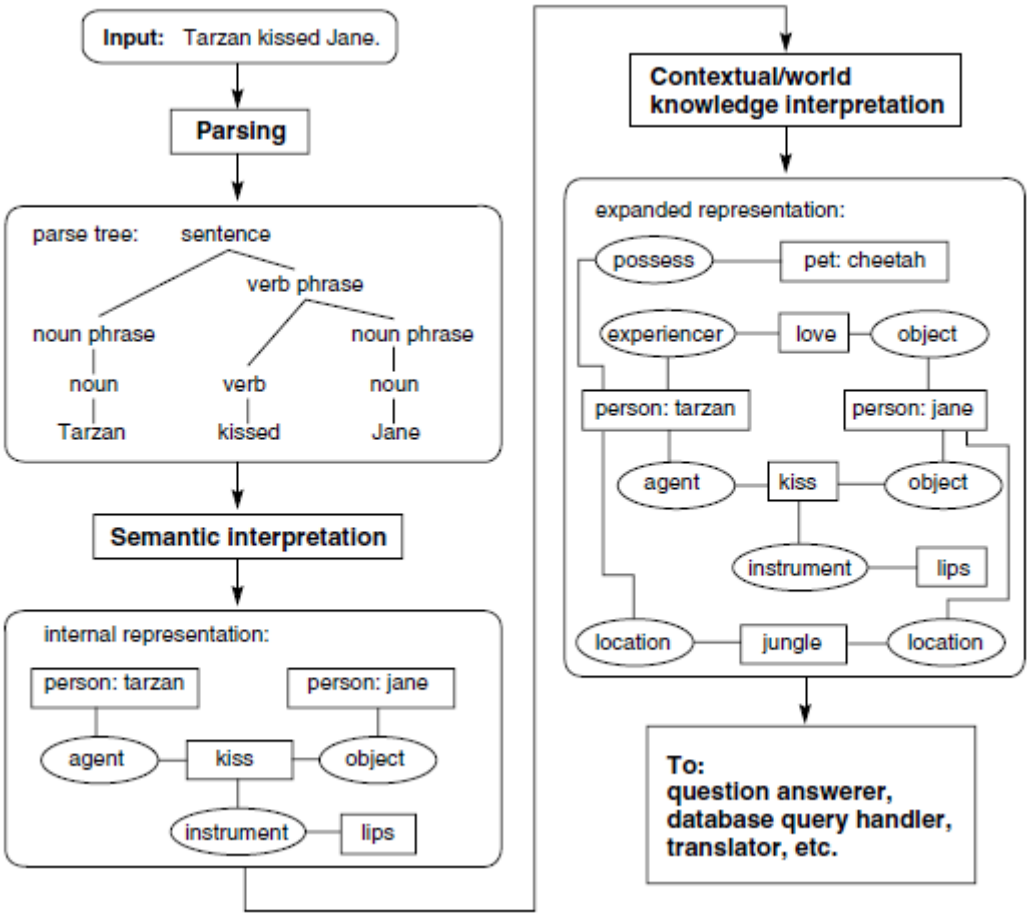
4. Składnia bada zasady łączenia słów w frazy prawne i zdania oraz wykorzystanie tych reguł do analizowania i generowania zdań. Jest to najlepiej sformalizowany, a tym samym najskuteczniejszy zautomatyzowany element analizy językowej.

5. Semantyka rozważa znaczenie słów, fraz i zdań oraz sposoby przekazywania znaczenia w wyrażeniach języka naturalnego.

6. Pragmatyka to badanie sposobów używania języka i jego wpływu na słuchacza. Na przykład pragmatyka odnosi się do przyczyny, dla której „Tak" jest zwykle niewłaściwą odpowiedzią na pytanie „Czy masz zegarek?"

7. Wiedza o świecie obejmuje wiedzę o świecie fizycznym, świecie ludzkich interakcji społecznych oraz roli celów i intencji w komunikacji. Ta ogólna wiedza jest niezbędna do zrozumienia pełnego znaczenia tekstu lub rozmowy.

Chociaż te poziomy analizy wydają się naturalne i są poparte dowodami psychologicznymi, są one do pewnego stopnia sztucznymi podziałami, które zostały narzucone językowi. Wszystkie te elementy intensywnie oddziałują na siebie, a nawet niskie intonacje i wariacje rytmiczne mają wpływ na znaczenie wypowiedzi, na przykład użycie sarkazmu. Ta interakcja jest ewidentna w związku między składnią a semantyką i chociaż pewien podział wzdłuż tych linii wydaje się niezbędny, dokładna granica jest trudna do scharakteryzowania. Na przykład zdania takie jak „Oni jedzą jabłka" mają wiele analiz, rozwiązane tylko przez zwrócenie uwagi na znaczenie w kontekście. Składnia wpływa również na semantykę, o czym świadczy rola struktury frazy w interpretacji znaczenia zdania. Chociaż dokładna natura rozróżnienia między składnią a semantyką jest często dyskutowana, zarówno dowody psychologiczne, jak i ich użyteczność w zarządzaniu złożonością problemu przemawiają za jego zachowaniem. Te głębsze kwestie dotyczące rozumienia i interpretacji języka omówimy ponownie w rozdziale 16. Chociaż specyficzna organizacja programów rozumienia języka naturalnego różni się w zależności od różnych filozofii i zastosowań - np. Interfejs dla bazy danych, system automatycznego tłumaczenia, program do rozumienia historii - wszyscy muszą przetłumaczyć oryginał zdania w wewnętrzną reprezentację jego znaczenia. Zasadniczo rozumienie języka naturalnego w oparciu o symbole przebiega według etapów przedstawionych na rysunku 2.



Pierwszym etapem jest parsowanie, które polega na analizie składniowej struktury zdań. Przetwarzanie obu weryfikuje, czy zdania są poprawnie sformułowane pod względem składniowym, a także określa strukturę językową. Identyfikując główne relacje językowe, takie jak podmiot – czasownik, czasownik – dopełnienie i rzeczownik – modyfikator, analizator składni zapewnia ramy dla interpretacji semantycznej. Jest to często przedstawiane jako drzewo parsowania. Parser języka wykorzystuje znajomość składni języka, morfologii i pewnej semantyki. Drugi etap to interpretacja semantyczna, w wyniku której powstaje przedstawienie znaczenia tekstu. Na rysunku 15.2 przedstawiono to jako wykres koncepcyjny. Inne powszechnie używane reprezentacje obejmują zależności koncepcyjne, ramki i reprezentacje oparte na logice.

Interpretacja semantyczna wykorzystuje wiedzę o znaczeniu słów i strukturze językowej, np. Rolach przypadków rzeczowników czy przechodniości czasowników. Na rysunku 2 program wykorzystał znajomość znaczenia pocałunku, aby dodać domyślną wartość ust dla instrumentu całowania. Na tym etapie wykonywane są również kontrole spójności semantycznej. Na przykład definicja czasownika pocałunek może zawierać ograniczenia, że przedmiotem jest osoba, jeśli agent jest osobą, to znaczy Tarzan całuje Jane i (normalnie) nie całuje Cheetah. Na trzecim etapie struktury z bazy wiedzy są dodawane do wewnętrznej reprezentacji zdania, aby uzyskać rozszerzoną reprezentację znaczenia zdania. To dodaje niezbędnej wiedzy o świecie wymaganej do pełnego zrozumienia, na przykład faktów, że Tarzan kocha Jane, że Jane i Tarzan żyją w dżungli oraz że Cheetah jest zwierzęciem Tarzana. Ta wynikowa struktura przedstawia znaczenie tekstu w języku naturalnym i jest wykorzystywana przez system do dalszego przetwarzania. Na przykład w interfejsie bazy danych rozszerzona struktura łączyłaby reprezentację znaczenia zapytania z wiedzą o organizacji bazy danych. Można to następnie przetłumaczyć na odpowiednie zapytanie w języku bazy danych. W programie rozumienia historii ta

rozszerzona struktura reprezentowałaby znaczenie historii i służyłaby do udzielania odpowiedzi na jej pytania (zobacz omówienie skryptów w rozdziale 7 i podsumowanie tekstu w sekcji 15.5.3). Etapy te istnieją w większości (nie probabilistycznych) systemów rozumienia języka naturalnego, chociaż mogą, ale nie muszą, odpowiadać odrębnym modułom oprogramowania. Na przykład wiele programów nie tworzy jawnego drzewa analizy, ale bezpośrednio generuje wewnętrzną reprezentację semantyczną. Niemniej jednak drzewo jest implicite w analizie zdania. Parsowanie przyrostowe (Allen 1987) jest powszechnie stosowaną techniką, w której fragment reprezentacji wewnętrznej jest tworzony, gdy tylko zostanie przeanalizowana znaczna część zdania. Te fragmenty są łączone w kompletną strukturę w miarę postępu analizy. Te fragmenty są również używane do rozwiązywania niejednoznaczności i kierowania parserem.

15.2 Składnia

15.2.1 Specyfikacja i analiza z użyciem gramatyki bezkontekstowej

Część 3 wprowadził użycie reguł przepisywania w celu określenia gramatyki. Poniższe zasady definiują gramatykę prostych zdań przechodnich, takich jak „Mężczyzna lubi psa”. Zasady są ponumerowane w celach informacyjnych.

1. sentence

↔ noun_phrase verb_phrase

2. noun_phrase

↔ rzeczownik

3. noun_phrase

↔ artykuł rzeczownik

4. verb_phrase

↔ czasownik

5. verb_phrase

↔ czasownik noun_phrase

6. artykuł

↔ a

7. artykuł

↔

8. rzeczownik

↔ człowiek

9. rzeczownik

↔ pies

10. czasownik

↔ polubień

11. czasownik

↔ ukąszenia

Reguły od 6 do 11 mają angielskie słowa po prawej stronie; reguły te tworzą słownik słów, które mogą występować w zdaniach. Te słowa są końcami gramatyki i określają leksykon języka. Terminy opisujące pojęcia lingwistyczne wyższego poziomu (zdanie, fraza_ rzeczownika itp.) Nazywane są nieterminalami. Nieterminale pojawiają się w tym kroju. Zwróć uwagę, że terminale nie pojawiają się po lewej stronie żadnej reguły. Zdaniem prawnym jest dowolny ciąg terminali, które można wyprowadzić za pomocą tych reguł. Derywacja zaczyna się od nieterminalnego zdania symbolu i tworzy ciąg terminali poprzez serię podstawień określonych regułami gramatyki. Legalna zmiana zastępuje symbol, który pasuje do lewej strony reguły, na symbole po prawej stronie tej reguły. Na pośrednich etapach wyprowadzania łańcuch może zawierać zarówno terminale, jak i nieterminale i jest nazywany formą zdaniową. Wyprowadzenie zdania „Mężczyzna gryzie psa" daje:

STRING: ZASTOSUJ REGUŁĘ NR

sentence 1

noun_phrase verb_phrase 3

article noun verb_phrase 7

The noun verb_phrase 8

The man verb_phrase 5

The man verb noun_phrase 11

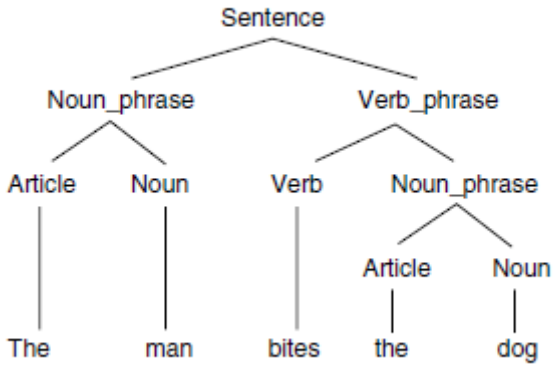
The man bites noun_phrase 3

The man bites article noun 7

The man bites the noun 9

The man bites the dog

To jest przykład wyprowadzenia odgórnego: zaczyna się od symbolu zdania i przechodzi do ciągu terminali. Derywacja oddolna zaczyna się od ciągu terminali i zastępuje wzorce po prawej stronie wzorami z lewej strony, kończąc się, gdy pozostaje tylko symbol zdania. Wyprowadzenie można przedstawić jako strukturę drzewa, znaną jako drzewo analizy, w której każdy węzeł jest symbolem z zestawu reguł gramatyki. Węzły wewnętrzne drzewa są nieterminalami; każdy węzeł i jego dzieci odpowiadają, odpowiednio, lewej i prawej stronie reguły w gramatyce. Węzły liści to terminale, a symbol zdania to korzeń drzewa. Drzewo parsowania „Mężczyzna gryzie psa" pokazano na rysunku 3.



Istnienie drzewa derywacji lub parsowania nie tylko świadczy o tym, że zdanie jest legalne w gramatyce, ale także określa jego strukturę. Struktura fraz w gramatyce definiuje głębszą organizację językową języka. Na przykład rozbicie zdania na noun_phrase i verb_phrase określa relację między czynnością a jej agentem. Ta struktura frazy odgrywa zasadniczą rolę w interpretacji semantycznej, definiując etapy pośrednie w derywacji, na których może mieć miejsce przetwarzanie semantyczne. Parsowanie to problem konstruowania wyprowadzenia lub drzewa parsowania dla ciągu wejściowego z formalnej definicji gramatyki. Algorytmy analizujące dzielą się na dwie klasy: parsery odgórne, które rozpoczynają się od symbolu zdania najwyższego poziomu i próbują zbudować drzewo, którego liście pasują do zdania docelowego, oraz parsery oddolne, które rozpoczynają się od słów w zdaniu (terminale) i spróbuj znaleźć serię redukcji, które dają symbol zdania. Jedną z trudności, która może zwiększyć złożoność problemu analizy, jest określenie, która z kilku potencjalnie mających zastosowanie reguł powinna być używana na dowolnym etapie wyprowadzania. W przypadku złego wyboru parser może nie rozpoznać wyroku prawnego. Na przykład, próbując przeanalizować zdanie „Pies gryzie” w sposób oddolny, zasady 7, 9, a 11 tworzy czasownik rzeczownik rzeczownik. W tym momencie błędne zastosowanie reguły 2 spowodowałoby wytworzenie czasownika przedimek rzeczownik_phrase; nie można tego sprowadzić do symbolu zdania. Zamiast tego parser powinien zastosować regułę 3. Podobne problemy mogą wystąpić w analiza odgórna. Problem z wyborem prawidłowej reguły na dowolnym etapie parsowania jest rozwiązywany albo przez zezwolenie parserowi na ustawienie wskaźników powrotu i powrót do sytuacji problemowej, jeśli dokonano niewłaściwego wyboru, albo przez użycie funkcji look-ahead do sprawdzenia strumień wejściowy dla funkcji pomagających określić właściwą regułę do zastosowania. W obu przypadkach musimy zadbać o kontrolę złożoności wykonania, gwarantując jednocześnie poprawną analizę. Odwrotnym problemem jest generowanie lub tworzenie wyroków prawnych z wewnętrznej reprezentacji. Generowanie rozpoczyna się od przedstawienia pewnych znaczących treści (takich jak sieć semantyczna lub graf zależności pojęciowych) i konstruuje poprawne gramatycznie zdanie, które przekazuje to znaczenie. Generowanie to nie tylko odwrotność analizy; napotyka wyjątkowe trudności i wymaga oddzielnych metodologii. W materiałach pomocniczych prezentujemy bezkontekstowe i kontekstowe parsery zejścia rekursywnego. Ponieważ parsowanie jest szczególnie ważne w przetwarzaniu języków programowania, a także języka naturalnego, naukowcy opracowali szereg różnych algorytmów analizowania, w tym zarówno strategie odgórne, jak i oddolne. Chociaż pełne omówienie algorytmów analizowania wykracza poza zakres tego rozdziału, rozważymy kilka podejść do analizowania bardziej szczegółowo. W następnej sekcji pokażemy parser Earley, ważny parser wielomianowy oparty na programowaniu dynamicznym. W sekcji 15.3 wprowadzamy ograniczenia semantyczne do parserów z parserami sieci przejściowej. Te parsery nie są wystarczająco wydajne do

semantycznej analizy języka naturalnego, ale stanowią podstawę dla rozszerzonych sieci przejściowych, które okazały się użytecznym i potężnym narzędziem w pracy z językiem naturalnym.

15.2.2 The Earley Parser: programowanie dynamiczne ponownie

Programowanie dynamiczne (DP) zostało pierwotnie zaproponowane przez Richarda Bellmana i przedstawione z kilkoma przykładami. Pomysł jest prosty: rozwiązując złożony problem, który można podzielić na wiele powiązanych ze sobą podproblemów, należy zapisać generowane rozwiązania częściowe, aby można je było ponownie wykorzystać w ciągłym procesie rozwiązywania. To podejście jest czasami nazywane zapamiętywaniem wyników podproblemu w celu ponownego wykorzystania. Istnieje wiele przykładów programowania dynamicznego w dopasowywaniu wzorców, na przykład przy określaniu miary różnicy między dwoma ciągami bitów lub znaków. Ogólna różnica między strunami będzie funkcją różnic między ich określonymi składnikami. Przykładem tego jest moduł sprawdzania pisowni sugerujący słowa, które są „bliskie" błędnie napisanego słowa. Innym przykładem DP jest rozpoznawanie słów w rozumieniu mowy jako funkcji możliwych fonemów ze strumienia wejściowego. Ponieważ fonemy są rozpoznawane (z powiązanymi z nimi prawdopodobieństwami), najbardziej odpowiednie słowo jest często funkcją połączonych połączonych miar probabilistycznych poszczególnych telefonów. W tej sekcji używamy DP do pokazania parser Earleya określającego, czy ciągi słów tworzą zdania poprawne składniowo. Nasz pseudokod został zaadaptowany z tego autorstwa Jurafsky'ego i Martina. Algorytmy analizowania z sekcji 15.2.1 są często implementowane z przeszukiwaniem rekurencyjnym, przeszukiwania najpierw w głąb i od lewej do prawej możliwych akceptowalnych struktur składniowych. To wyszukiwanie podejście może oznaczać, że wiele z możliwych do zaakceptowania częściowych analiz pierwszych (najbardziej po lewej) składników ciągu jest wielokrotnie odtwarzanych. Ta ponowna weryfikacja wczesnych rozwiązań częściowych w ramach pełnej struktury analizy jest wynikiem późniejszych wymagań dotyczących wycofywania wyszukiwanie i może stać się wykładniczo kosztowne i wymagać większych analiz. Programowanie dynamiczne zapewnia wydajną alternatywę, w której częściowe analizy, po wygenerowaniu, są zapisywane do ponownego wykorzystania w ogólnej końcowej analizie ciągu słów. Utworzono pierwszy parser oparty na DP przez Earley.

Zapamiętywanie i pary kropek

Podczas analizowania za pomocą algorytmu Earleya zapamiętywanie rozwiązań częściowych (częściowe analizy) odbywa się za pomocą struktury danych zwanej wykresem. Dlatego podejście Earley do analizy składniowej jest często nazywane analizowaniem wykresów. Wykres jest generowany za pomocą kropkowanych reguł gramatycznych. Kropkowana reguła gramatyki zapewnia reprezentację, która wskazuje na wykresie stan procesu analizowania w dowolnym momencie. Każda reguła z kropkami należy do jednej z trzech kategorii, w zależności od tego, czy pozycja kropki znajduje się na początku, gdzieś w środku, czy na końcu prawej strony reguły gramatycznej, RHS. Te trzy kategorie nazywamy odpowiednio początkowymi, częściowymi lub zakończonymi etapami analizy:

Wstępna prognoza: Symbol →. ●RHS_unseen

Częściowa analiza: Symbol → RHS_seen ●RHS_unseen

Ukończono analizę: Symbol → RHS_seen ●

Ponadto istnieje naturalna zgodność między stanami zawierającymi różne reguły z kropkami a krawędziami drzewa (-ów) analizy utworzonej przez analizę. Rozważ następującą bardzo prostą gramatykę, w której symbole terminali są otoczone cudzysłowami, jak w „mary":

Zdanie → Rzeczownik Czasownik

Rzeczownik → „mary”

Czasownik → „działa”

Gdy wykonujemy analizę z góry na dół tego zdania od lewej do prawej, powstaje następująca sekwencja stanów:

Zdanie →. ●Rzeczownik Czasownik przewidzieć: rzeczownik, po którym następuje czasownik

Rzeczownik → ● mary przewiduje: mary

Rzeczownik → mary ●zeskanowano: mary

Zdanie → Rzeczownik ●Czasownik ukończony: rzeczownik; przewidzieć: czasownik

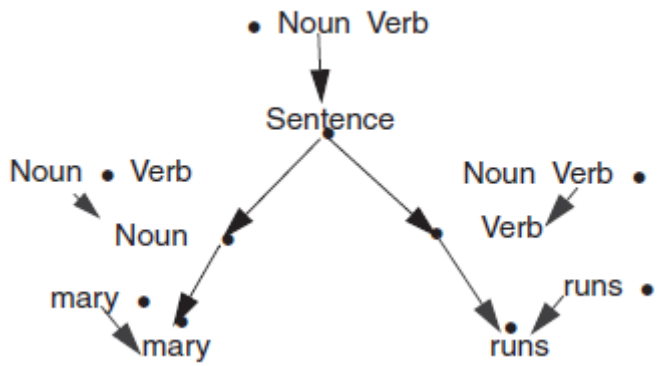
Czasownik → ●przebiegi przewidywać: przebiegi

Czasownik → działa ● skanowano: przebiegi

Zdanie → Rzeczownik Czasownik ● zakończone: Czasownik zakończone: zdanie

Zwróć uwagę, że skanowanie i wykonywanie procedur deterministycznie daje wynik. Procedura przewidywania opisuje możliwe reguły analizy, które można zastosować do bieżącej sytuacji. Skanowanie i przewidywanie tworzą stany w drzewie analizy przedstawionym na rysunku 4. Algorytm Earleya działa na zasadzie odgórnych i od lewej do prawej przewidywań dotyczących analizowania danych wejściowych. Każda prognoza jest zapisywana jako stan zawierający wszystkie istotne informacje o prognozie, gdzie kluczowym składnikiem każdego stanu jest reguła kropkowa. (Drugi komponent implementacji zostanie przedstawiony w następnej sekcji). Wszystkie predykcje generowane po zbadaniu określonego słowa z wejścia są zbiorczo nazywane zestawem stanów. Dla danego zdania wejściowego zawierającego n słów, od w1 do wn, generowanych jest łącznie n + 1 zestawów stanów: [S_0, S_1,…, S_n]. Zestaw stanu początkowego S_0 zawiera przewidywania, które są dokonywane przed sprawdzeniem jakichkolwiek słów wejściowych, S_1 zawiera prognozy wykonane po zbadaniu w_1 i tak dalej. Cały zbiór zestawów stanów nazywamy wykresem utworzonym przez parser.

Rysunek 4 ilustruje zależność między generowaniem zestawu stanów a sprawdzaniem słów wejściowych.



Chociaż tradycyjnie zestawy stanów, które składają się na każdy składnik parsowania, nazywane są zbiorami stanów, kolejność generowania tych stanów jest ważna. Dlatego każdy składnik wykresu nazwiemy listą stanów i opiszemy go jako [State1, State2,…, Staten]. Działa to również dobrze z implementacją Prologu w materiałach pomocniczych, gdzie listy stanów będą utrzymywane jako listy

Prologu. Na koniec opisujemy każdy stan z listy stanów jako sekwencję określonych symboli ujętych w nawiasy, na przykład ($ → ● S).

Rozważmy teraz algorytm Earley analizujący proste zdania wykonywane przez Mary, używając powyższej gramatyki. Algorytm rozpoczyna się od utworzenia fikcyjnego stanu początkowego ($ → ●S), czyli pierwszego elementu listy stanów S0. Stan ten reprezentuje przewidywanie, że ciąg wejściowy może być analizowany jako zdanie i jest wstawiany do S0 przed sprawdzeniem jakichkolwiek słów wejściowych. Pomyślna analiza tworzy ostateczną listę stanów Sn, która zawiera stan ($ → S ●). Zaczynając od S0, parser wykonuje pętlę, w której każdy stan Si na bieżącej liście stanów jest sprawdzany w kolejności i używany do generowania nowych stanów. Każdy nowy stan jest generowany przez jedną z trzech procedur nazywanych predyktorem, skanerem i zakończeniem. Odpowiednią procedurę określa reguła z kropkami w stanie S, a konkretnie symbol gramatyki (jeśli występuje) następujący po kropce w regule. W naszym przykładzie pierwszy badany stan zawiera regułę ($ → ● S). Ponieważ po kropce występuje symbol S, stan ten „oczekuje", że wystąpi instancja S jako następna na wejściu. Ponieważ S jest nieterminalnym symbolem gramatyki, procedura predykcyjna generuje wszystkie stany odpowiadające możliwemu parsowi S. W tym przypadku, ponieważ istnieje tylko jedna alternatywa dla S, a mianowicie, że S → Rzeczownik Czasownik, tylko jeden stan (S → ● Rzeczownik Czasownik), jest dodawany do S0. Ponieważ stan ten oczekuje części mowy, oznaczonej przez nieterminalny symbol Rzeczownik następujący po kropce, algorytm sprawdza następne słowo wejściowe, aby zweryfikować to przewidywanie. Odbywa się to przez procedurę skanera, a ponieważ następne słowo pasuje do predykcji, mary jest rzeczywiście rzeczownikiem, skaner generuje nowy stan zapisujący dopasowanie: (Rzeczownik → mary ●). Ponieważ ten stan zależy od słowa wejściowego w1, staje się pierwszym stanem w liście stanów S1, a nie dodawanie do S0. W tym miejscu wykres zawierający dwie listy stanów wygląda następująco, gdzie po każdym stanie nazywamy procedurę, która go wygenerowała:

S0: [($ → ● S), fikcyjny stan początkowy

(S → ● Rzeczownik Czasownik)] predyktor

S1: [(rzeczownik → mary ●)] Skaner

Każdy stan na liście stanów S0 został już przetworzony, więc algorytm przechodzi do S1 i rozważa stan (Rzeczownik → Mary ●). Ponieważ jest to stan ukończony, stosowana jest procedura kompletna. Dla każdego stanu oczekującego rzeczownika, to znaczy, że ma ● Wzorzec rzeczownika, element uzupełniający generuje nowy stan, który rejestruje odkrycie rzeczownika, przesuwając kropkę nad symbol rzeczownika. W tym przypadku wypełniacz tworzy stan (S → ● Rzeczownik Czasownik) w S0 i generuje nowy stan (S → Rzeczownik ●Czasownik) na liście S1. Stan ten oczekuje części mowy, co powoduje, że skaner bada następne słowo wejściowe w2. Ponieważ w2 jest czasownikiem, skaner generuje stan (czasownik → działa ●) i dodaje go do S2, co daje następujący wykres:

S0: [($ → ●S), start

(S → ● Rzeczownik Czasownik)] predyktor

S1: [(rzeczownik → mary ●), Scanner

(S → Rzeczownik ●Verb)] completeer

S2: [(Verb → run ●)] Skaner

Przetwarzając nowy stan S2, wypełniacz przesuwa kropkę w (S → Rzeczownik ● Verb) w celu wytworzenia (S → Rzeczownik Czasownik ●), Z którego kompletujący generuje stan ($ → S ●)

Oznaczający pomyślną analizę zdania. Ostateczny wykres dla biegów Mary, z trzema listami stanów, to:

S0: [($ → ● S), start

(S → ● Rzeczownik Czasownik)] predyktor

S1: [(rzeczownik → mary ●), Scanner

(S → Rzeczownik ●Verb)] completeer

S2: [(Verb → run ●)] Skaner

(S → Rzeczownik Czasownik ●), completeer

($ → S ●.)] Dopełniacz

15.2.2.2 Algorytm Earleya: pseudokod

Aby przedstawić obliczeniowo listy stanów utworzone przez powyższe reguły par z kropkami, tworzymy indeksy pokazujące, jaka część prawej strony reguły gramatycznej została przeanalizowana. Najpierw opisujemy tę reprezentację, a następnie oferujemy pseudokod do jej implementacji w algorytmie Earley. Każdy stan na liście stanów jest uzupełniony o indeks wskazujący, jak daleko przetworzono strumień wejściowy. W ten sposób rozszerzamy każdy opis stanu do reprezentacji (reguła kropkowana [i, j]), w której para [i, j] oznacza, ile prawej strony, RHS, reguły gramatyki zostało zobaczone lub przeanalizowane do chwili obecnej . Po prawej stronie przeanalizowanej reguły, która zawiera zero lub więcej widocznych i niewidocznych elementów wskazanych przez ●, mamy (A → Seen ● Unseen, [i, j]), gdzie i jest początkiem Seen, a j jest pozycją ● w sekwencji słów. Teraz dodajemy indeksy do omówionych wcześniej stanów analizy dla zdania, które wykonuje mary: ($ →● S, [0, 0]) utworzone przez predyktor, i = j = 0, nic nie zostało przeanalizowane (Rzeczownik → mary●, [0,1]) skaner widzi w1 między indeksami słów 0 i 1 (S → Rzeczownik ● Verb, [0,1]) uzupełniający widział Rzeczownik (mary) między 0 a 1 (S → Rzeczownik Czasownik ●., [0,2]) Completeer widział zdanie S między 0 a 2 .W ten sposób wzorzec indeksowania stanu przedstawia wyniki uzyskane przez każdy z trzech generatorów stanu przy użyciu reguł kropkowych wraz z indeksem słowa wi. Podsumowując, trzy procedury generowania stanów listy stanów to: predyktor generujący stany z indeksem [j, j] przechodzący do wykresu [j], skaner uwzględniający słowo wj + 1 w celu wygenerowania stanów indeksowanych przez [j, j + 1] do wykresu [j + 1] i kompletator działający na regułach z indeksem [i, j], i <j, dodając wpis stanu do wykresu [j]. Zauważ, że stan z reguły kropkowanej [i, j] zawsze trafia do wykresu listy stanów [j]. Zatem listy stanów zawierają chart [0], ..., chart [n] dla zdania zawierającego n słów. Teraz, gdy przedstawiliśmy schemat indeksowania do reprezentacji wykresu, podajemy pseudokod dla parsera Earley.

function EARLEY-PARSE(words, grammar) returns chart

begin

chart := empty

ADDTOCHART(($ →● S, [0, 0]), chart[0]) % dummy start state

for i from 0 to LENGTH(words) do

for each state in chart[i] do

if rule_rhs(state) = … ●A … and A is not a part of speech

```
then PREDICTOR(state)
else if rule_rhs(state) = … ● L … % L is part of speech
then SCANNER(state)
else COMLETER(state) % rule_rhs = RHS .
end
procedure PREDICTOR((A → … ● B …, [i, j]))
begin
for each (B → RHS) in grammar do
ADDTOCHART((B→ ● RHS, [j, j]), chart[j])
end
procedure SCANNER((A → … ● L …, [i, j]))
begin
if (L → word[j]) is_in grammar
then ADDTOCHART((L→ word[j] ●, [j, j + 1]), chart[j + 1])
end
procedure COMPLETER((B → … ●, [j, k]))
begin
for each (A → … ● B …, [i, j]) in chart[j] do
ADDTOCHART((A→ … B ● …, [i, k]), chart[k])
end
procedure ADDTOCHART(state, state-list)
begin
if state is not in state-list
then ADDTOEND(state, state-list)
end
```

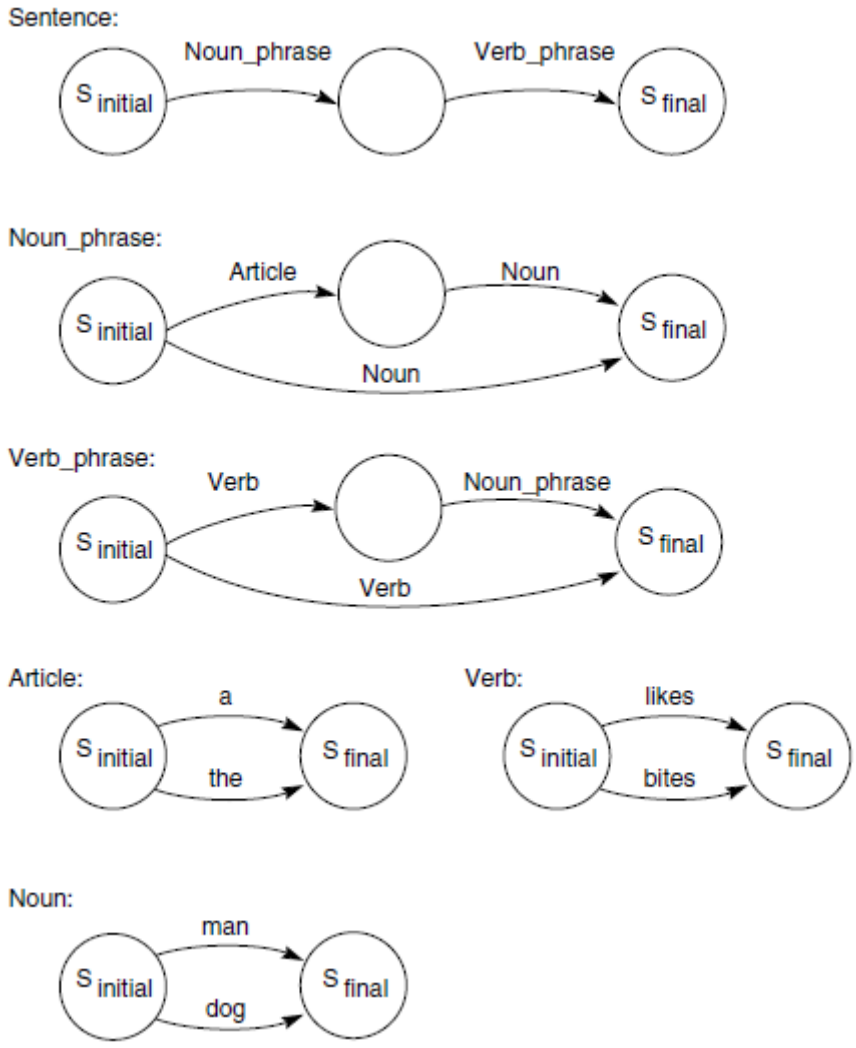
Nasz pierwszy przykład, analiza Earley zdania, które prowadzi Mary, miała być prosta, ale ilustracyjna, z bardzo szczegółową prezentacją list stanów i ich indeksów. bardziej złożone - i niejednoznaczne - zdanie, John zwany Mary z Denver jest przedstawiony w materiałach pomocniczych wraz z kodem do implementacji parsowania przy użyciu algorytmu Earley zarówno w języku Prolog, jak i Java. Dwuznaczność tego zdania znajduje odzwierciedlenie w dwóch różnych analizach, które odzwierciedlają tę niejednoznaczność. Wyboru jednej z tych analiz dokonuje się przez uruchomienie wstecznego składnika parsera Earleya opartego na programowaniu dynamicznym.

15.3 Parsery i semantyka sieci przejścia

Do tej pory zaoferowaliśmy reprezentacje i algorytmy, które wspierają syntaktyczną analizę składniową opartą na symbolach. Analiza relacji składniowych, nawet jeśli ogranicza się do analizy kontekstowej (np. Zgodność rzeczownik-czasownik), nie uwzględnia relacji semantycznych. W tej sekcji przedstawiamy sieci przejściowe, aby rozwiązać ten problem.

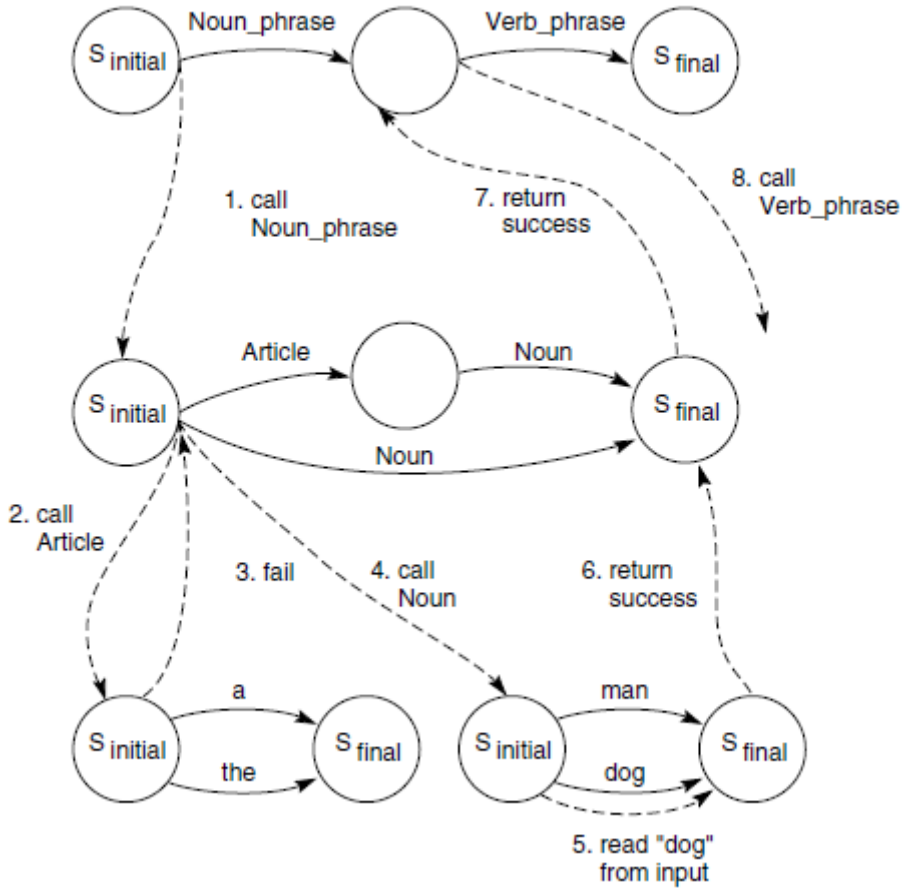
15.3.1 Parsery sieci przejściowej

Parser sieci przejściowej reprezentuje gramatykę jako zbiór maszyn skończonych lub sieci przejściowych. Każda sieć odpowiada pojedynczemu nieterminalowi w gramatyce. Łuki są oznaczone symbolami końcowymi lub nieterminalnymi. Każda ścieżka w sieci, stan początkowy do stanu końcowego, odpowiada pewnej regule dla tego nieterminala; sekwencja etykiet łuków na ścieżce to sekwencja symboli po prawej stronie reguły. Gramatyka sekcji 15.2.1 jest reprezentowana przez sieci z rysunku 5.



Gdy istnieje więcej niż jedna reguła dla nieterminala, odpowiednia sieć ma wiele ścieżek od początku do celu, np. Reguły noun_phrase ↔ noun i noun_phrase ↔ rzeczownik są przechwytywane przez alternatywne ścieżki w sieci noun_phrase. Znalezienie udanego przejścia przez sieć dla nieterminala odpowiada zastąpieniu tego nieterminala prawą stroną reguły gramatycznej. Na przykład, aby

przeanalizować zdanie, parser sieci przejściowej musi znaleźć przejście przez sieć zdań. Rozpoczyna się w stanie początkowym (Sinitial) i przyjmuje przejście noun_phrase, a następnie przejście verb_phrase, aby osiągnąć stan końcowy (Sfinal). Jest to równoważne zamianie oryginalnego symbolu zdania na noun_phrase verb_phrase. Aby przejść przez łuk, parser sprawdza jego etykietę. Jeśli etykieta jest symbolem terminala, parser sprawdza strumień wejściowy, aby zobaczyć, czy następne słowo pasuje do etykiety łuku. Jeśli nie pasuje, przejście nie może zostać wykonane. Jeśli łuk jest oznaczony nie terminalem symbol, parser wyszukuje sieć dla tego nieterminala i rekurencyjnie próbuje znaleźć ścieżkę przez nią. Jeśli parser nie znajdzie ścieżki w tej sieci, nie można przejść przez łuk najwyższego poziomu. Powoduje to, że parser cofa się i próbuje użyć innej ścieżki przez sieć. W ten sposób parser próbuje znaleźć ścieżkę w sieci zdań; jeśli się powiedzie, ciąg wejściowy jest zdaniem prawnym w gramatyce. Rozważmy proste zdanie Ukąszenia psa, przeanalizowane i zilustrowane na rysunku 6:



1. Parser zaczyna od sieci zdań i próbuje poruszać się po łuku oznaczonym noun_phrase. Aby to zrobić, pobiera sieć dla noun_phrase.

2. W sieci noun_phrase parser najpierw próbuje przejść do zaznaczonego artykułu. To powoduje, że rozgałęzia się do sieci w celu uzyskania artykułu.

3. Nie można znaleźć ścieżki do węzła końcowego sieci artykułów, ponieważ pierwsze słowo „Dog” nie pasuje do żadnej z etykiet łuków. Parser kończy się niepowodzeniem i wraca do sieci noun_phrase.

4. Parser próbuje podążać za łukiem oznaczonym rzeczownikiem w sieci noun_phrase i rozgałęzia się do sieci dla rzeczownika.

5. Parser pomyślnie przechodzi przez łuk oznaczony „pies”, ponieważ odpowiada to pierwszemu słowu strumienia wejściowego.

6. Sieć rzeczowników zwraca sukces. Pozwala to na przejście łuku oznaczonego rzeczownikiem w sieci noun_phrase do stanu końcowego.

7. Sieć noun_phrase zwraca sukces do sieci najwyższego poziomu, umożliwiając przejście łuku oznaczonego noun_phrase.

8. Sekwencja podobnych kroków jest wykonywana w celu przeanalizowania części zdania verb_phrase

Pseudokod parsera przejściowego znajduje się poniżej. Jest definiowany za pomocą dwóch wzajemnie rekurencyjnych funkcji, parsowania i przejścia. Parse przyjmuje symbol gramatyki jako argument: jeśli symbol jest terminalem, parse sprawdza go względem następnego słowa w strumieniu wejściowym. Jeśli jest to nieterminal, analiza składniowa pobiera sieć przejściową skojarzoną z symbolem i wywołuje przejście w celu znalezienia ścieżki w sieci. Aby przeanalizować zdanie, wywołaj parse (zdanie).

```
function parse(grammar_symbol);

begin

save pointer to current location in input stream;

case

grammar_symbol is a terminal:

if grammar_symbol matches the next word in the input stream

then return (success)

else begin

reset input stream;

return (failure)

end;

grammar_symbol is a nonterminal:

begin

retrieve the transition network labeled by grammar symbol;

state := start state of network;

if transition(state) returns success

then return (success)

else begin

reset input stream;

return (failure)

end

end
```

```
end

end.

function transition (current_state);

begin

case

current_state is a final state:

return (success)

current_state is not a final state:

while there are unexamined transitions out of current_state

do begin

grammar_symbol := the label on the next unexamined transition;

if parse(grammar_symbol) returns (success)

then begin

next_state := state at end of the transition;

if transition(next_state) returns success;

then return (success)

end

end

return (failure)

end

end.
```

Przejście przyjmuje stan w sieci przejściowej jako argument i próbuje znaleźć ścieżkę przez tę sieć w sposób głęboki. Ponieważ parser może popełnić błąd i wymagać śledzenia wstecznego, parse zachowuje wskaźnik do bieżącej lokalizacji w strumieniu wejściowym. Pozwala to na zresetowanie strumienia wejściowego do tej lokalizacji w przypadku cofnięcia się parsera. Ten parser sieci przejściowej określa, czy zdanie jest poprawne gramatycznie, ale nie tworzy drzewa parsowania. Można to osiągnąć poprzez zwrócenie przez funkcje poddrzewa drzewa parsowania zamiast sukcesu symbolu. Modyfikacje, które mogłyby to osiągnąć, to:

1. Za każdym razem, gdy parse funkcji jest wywoływana z symbolem terminala jako argumentem i ten terminal pasuje do następnego symbolu wejścia, zwraca drzewo składające się z pojedynczego węzła-liścia oznaczonego tym symbolem.

2. Kiedy parse jest wywoływana z nieterminalnym, grammar_symbol, wywołuje przejście. Jeśli przejście się powiedzie, zwraca uporządkowany zestaw poddrzew (opisanych poniżej). Parse łączy je w drzewo, którego korzeniem jest grammar_symbol i którego dziećmi są poddrzewa zwrócone przez przejście.

3. Podczas wyszukiwania ścieżki w sieci, wywołania przejścia analizują etykiety każdego łuku. Po pomyślnym przeprowadzeniu analizy składni zwraca drzewo reprezentujące analizę tego symbolu. Przejście zapisuje te poddrzewa w uporządkowanym zestawie i po znalezieniu ścieżki w sieci zwraca uporządkowany zestaw drzew parsowania odpowiadający sekwencji etykiet łuków na ścieżce.

15.3.2 Hierarchia Chomsky'ego i gramatyki kontekstualne

W sekcji 15.3.1 zdefiniowaliśmy niewielki podzbiór języka angielskiego przy użyciu gramatyki bezkontekstowej. Gramatyka bezkontekstowa pozwala regułom mieć tylko jeden nieterminal po lewej stronie. W konsekwencji regułę można zastosować do każdego wystąpienia tego symbolu, niezależnie od kontekstu. Chociaż gramatyki bezkontekstowe okazały się potężnym narzędziem do definiowania języków programowania i innych formalizmów w informatyce, istnieją powody, by sądzić, że same w sobie nie są wystarczająco potężne, aby reprezentować reguły składni języka naturalnego. Na przykład zastanówmy się, co się stanie, jeśli dodamy liczbę pojedynczą oraz rzeczowniki i czasowniki w liczbie mnogiej do gramatyki sekcji 15.2.1:

rzeczownik ↔ mężczyzn

rzeczownik ↔ psy

czasownik ↔ ukąszenia

czasownik ↔ jak

Wynikowa gramatyka przeanalizuje zdania, takie jak „Psy lubią mężczyzn”, ale akceptuje również „A mężczyzna gryzie psy”. Parser akceptuje te zdania, ponieważ reguły nie używają kontekstu do określenia, kiedy należy skoordynować liczbę pojedynczą i mnogą. Reguła definiująca zdanie jako noun_phrase, po którym następuje verb_phrase, nie wymaga, aby rzeczownik i czasownik zgadzały się co do liczby lub żeby przedimki zgadzały się z rzeczownikami. Języki wolne od Conext można rozszerzyć, aby poradzić sobie z takimi sytuacjami, ale bardziej naturalne podejście polega na tym, że grammer jest wrażliwy na kontekst, gdzie komponenty drzewa parsowania są zaprojektowane tak, aby wzajemnie się ograniczać. Chomsky (1965) jako pierwszy zaproponował świat hierarchicznych i coraz potężniejszych gramofonów (Hopcroft i Ullman 1979). Na dole tej hierarchii znajduje się klasa języków regularnych, których gramatykę można zdefiniować za pomocą maszyny skończonej, sekcja 3.1. Zwykłe języki mają wiele zastosowań w informatyce, ale nie są wystarczająco potężne, aby reprezentować składnię większości języków programowania. Języki bezkontekstowe znajdują się ponad zwykłymi językami w hierarchii Chomsky'ego. Języki bezkontekstowe są definiowane za pomocą reguł przepisywania, takich jak w sekcji 15.2.1; reguły bezkontekstowe mogą mieć tylko jeden symbol nieterminalny po lewej stronie. Parsery sieci przejściowej są w stanie przeanalizować klasę języków bezkontekstowych. Warto zauważyć, że jeśli nie pozwolimy na rekurencję w parserze sieci przejściowej, tj. Łuki mogą być etykietowane tylko symbolami terminala, a przejścia nie mogą „wywoływać” innej sieci, to klasa języków, które mogą być tak zdefiniowane, odpowiada zwykłej wyrażenia. Tak więc języki regularne stanowią właściwy podzbiór języków bezkontekstowych. Języki kontekstowe stanowią właściwy nadzbiór języków bezkontekstowych. Są one definiowane za pomocą gramatyki kontekstowej, która zezwala na więcej niż jeden symbol po lewej stronie reguły i umożliwia zdefiniowanie kontekstu, w którym można zastosować tę regułę. Zapewnia to spełnienie globalnych ograniczeń, takich jak zgodność liczby i inne kontrole semantyczne. Jedynym ograniczeniem reguł gramatycznych zależnych od kontekstu jest to, że prawa strona jest co najmniej tak długa, jak lewa (Hopcroft i Ullman 1979). Czwarta klasa, tworząca nadzbiór języków zależnych od kontekstu, to klasa języków rekurencyjnie wyliczalnych. Rekurencyjnie wyliczalne języki mogą być definiowane przy użyciu nieograniczonych reguł produkcji; ponieważ te reguły są mniej ograniczone niż reguły kontekstowe,

rekurencyjnie wyliczalne języki są właściwym nadzbiorem języków kontekstowych. Ta klasa nie jest interesująca w definiowaniu składni języka naturalnego, chociaż jest ważna w teorii informatyki. Pozostała część tej sekcji skupia się na języku angielskim jako języku kontekstowym. Prosta gramatyka bezkontekstowa dla zdań z formy rzeczownik czasownik, która wymusza zgodność liczbową między przedimkiem i rzeczownikiem oraz podmiotem i czasownikiem, jest dana przez:

zdanie ↔ noun_phrase verb_phrase

noun_phrase ↔ numer artykułu rzeczownik

noun_phrase ↔ liczba rzeczownik

liczba ↔ liczba pojedyncza

liczba ↔ liczba mnoga

artykuł liczba pojedyncza ↔ a liczba pojedyncza

artykuł liczba pojedyncza ↔ liczba pojedyncza

artykuł liczba mnoga ↔ pewna liczba mnoga

artykuł w liczbie mnogiej ↔ liczba mnoga

rzeczownik w liczbie pojedynczej ↔ pies w liczbie pojedynczej

rzeczownik w liczbie pojedynczej ↔ człowiek w liczbie pojedynczej

rzeczownik ↔ mężczyźni liczba mnoga

rzeczownik w liczbie mnogiej ↔ psy w liczbie mnogiej

czasownik w liczbie pojedynczej_fraza ↔ czasownik w liczbie pojedynczej

czasownik w liczbie mnogiej_fraza ↔ czasownik w liczbie mnogiej

czasownik w liczbie pojedynczej ↔ ugryzienia

czasownik w liczbie pojedynczej ↔ lubi

czasownik w liczbie mnogiej ↔ zgryz

czasownik w liczbie mnogiej ↔ podobny

W tej gramatyce nieterminale liczby pojedynczej i mnogiej oferują ograniczenia do określenia, kiedy można zastosować różne reguły przedimka, rzeczownika i czasownika_fraza, zapewniając zgodność liczb. Wyprowadzenie zdania „Psy gryzą” przy użyciu tej gramatyki jest podane przez:

zdanie.

noun_phrase verb_phrase.

artykuł liczba mnoga rzeczownik czasownik_fraza.

Rzeczownik w liczbie mnogiej verb_phrase.

Psy w liczbie mnogiej verb_phrase.

Psy w liczbie mnogiej.

Psy gryzą.

Podobnie, możemy użyć gramatyki kontekstowej do sprawdzania zgodności semantycznej. Na przykład moglibyśmy zabronić wykonywania zdań, takich jak „Mężczyzna gryzie psa", dodając do gramatyki nieterminal „act_of_biting". Ten nieterminal można by zaznaczyć w regułach, aby zapobiec sytuacji, w której w zdaniu zawierającym „ugryzienia" słowo „człowiek" jest przedmiotem.

Chociaż gramatyki kontekstowe mogą definiować struktury językowe, których nie można uchwycić za pomocą gramatyk bezkontekstowych, mają szereg wad przy projektowaniu praktycznych parserów:

1. Gramatyki kontekstowe drastycznie zwiększają liczbę reguł i nieterminali w gramatyce. Wyobraź sobie złożoność gramatyki kontekstowej, która zawierałaby liczbę, osobę i wszystkie inne formy porozumienia wymagane w języku angielskim.

2. Zasłaniają strukturę wyrażeń języka, który jest tak jasno przedstawiony w zasadach bezkontekstowych.

3. Próbując poradzić sobie z bardziej skomplikowanymi sprawdzeniami zgodności i spójności semantycznej w samej gramatyce, tracą wiele korzyści z oddzielenia składniowych i semantycznych składników języka.

4. Gramatyki kontekstualne nie rozwiązują problemu budowania semantycznej reprezentacji znaczenia tekstu. Parser, który po prostu akceptuje lub odrzuca zdania, nie jest wystarczający; musi zwracać użyteczną reprezentację znaczenia semantycznego zdania.

W materiałach pomocniczych do tej książki przedstawiamy zarówno parsery bezkontekstowe, jak i kontekstowe, a także generatory zdań. Reżim kontroli dla tych programów Prologu to rekurencyjne zejście najpierw w głąb. Następnie przeanalizujemy rozszerzone sieci przejściowe (ATN), rozszerzenie sieci przejściowych, które mogą definiować języki kontekstowe, ale ma kilka zalet w porównaniu z gramatyką kontekstową w projektowaniu parserów.

15.3.3 Semantyka: parsery sieci rozszerzonego przejścia

Alternatywą dla gramatyki kontekstualnej jest zachowanie prostszej struktury bezkontekstowych reguł gramatycznych, ale rozszerzenie tych reguł o dołączone procedury, które wykonują niezbędne testy kontekstowe. Te procedury są wykonywane, gdy reguła jest wywoływana podczas analizowania. Zamiast używać gramatyki do opisu takich pojęć, jak liczba, czas i osoba, przedstawiamy je jako cechy dołączone do terminali i nieterminali gramatyki. Procedury dołączone do reguł gramatyki mają dostęp do tych funkcji w celu przypisania wartości i wykonania niezbędnych testów. Gramatyki, które wykorzystują rozszerzenia gramatyk bezkontekstowych do implementacji wrażliwości kontekstowej, obejmują gramatykę rozszerzonej struktury fraz, rozszerzenia gramatyki logicznej i rozszerzoną sieć przejść (ATN). W tej sekcji przedstawiamy parsowanie ATN i zarys projektu prostego parsera ATN dla zdań dotyczących „świata psów" przedstawionych w sekcji 15.2.1. Zajmujemy się pierwszymi dwoma krokami przedstawionymi na rysunku 2: utworzeniem drzewa parsowania i jego wykorzystaniem do skonstruowania reprezentacji znaczenia zdania. W tym przykładzie używamy grafów koncepcyjnych, chociaż parsery ATN mogą być również używane z reprezentacjami opartymi na sieci semantycznej, skrypcie, ramce lub logice. Rozszerzone sieci przejściowe rozszerzają sieci przejściowe, umożliwiając dołączanie procedur do łuków sieci. Parser ATN wykonuje te dołączone procedury podczas przechodzenia przez łuki. Procedury mogą przypisywać wartości cechom gramatycznym i wykonywać testy, powodując niepowodzenie przejścia, jeśli określone warunki (takie jak zgodność liczb) nie są spełnione. Te procedury również konstruują drzewo parsowania, które jest używane do generowania wewnętrznej semantycznej reprezentacji znaczenia zdania. Reprezentujemy zarówno terminale, jak i

nieterminale jako identyfikatory (np. Czasownik, fraza_ rzeczownika) z dołączonymi funkcjami. Na przykład słowo jest opisywane za pomocą jego korzenia morfologicznego, wraz z cechami jego części mowy, liczby, osoby itp. Nieterminale w gramatyce są podobnie opisane. Fraza rzeczownikowa jest opisywana przez rodzajnik, rzeczownik, liczbę i osobę. Zarówno terminale, jak i nieterminale mogą być reprezentowane za pomocą struktur przypominających ramkę z nazwanymi gniazdami i wartościami. Wartości tych szczelin określają cechy gramatyczne lub wskaźniki do innych struktur. Na przykład pierwsza sekcja ramki zdania zawiera wskaźnik do definicji wyrażenia rzeczownikowego. Rysunek 7 pokazuje ramki dla nieterminali zdania, noun_phrase i verb_phrase w naszej prostej gramatyce.

Sentence
Noun phrase:
Verb phrase:

Noun phrase
Determiner:
Noun:
Number:

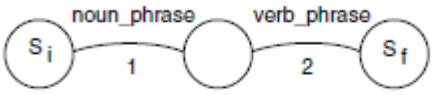
Verb phrase
Verb:
Number:
Object:

Poszczególne słowa są przedstawiane za pomocą podobnych struktur. Każde słowo w słowniku jest zdefiniowane ramką, która określa jego część mowy (rodzajnik, rzeczownik itp.), Jego rdzeń morfologiczny i istotne cechy gramatyczne. W naszym przykładzie sprawdzamy tylko zgodność numeru i rejestrujemy tylko tę funkcję. Bardziej wyrafinowane gramatyki wskazują osobę i inne cechy. Te hasła słownikowe mogą również wskazywać na koncepcyjną definicję grafową znaczenia słowa do użycia w interpretacji semantycznej. Kompletny słownik naszej gramatyki przedstawiono na rysunku 8.

Word	Definition
a	PART_OF_SPEECH: article ROOT: a NUMBER: singular
bite	PART_OF_SPEECH: verb ROOT: bite NUMBER: plural
bites	PART_OF_SPEECH: verb ROOT: bite NUMBER: singular
dog	PART_OF_SPEECH: noun ROOT: dog NUMBER: singular
dogs	PART_OF_SPEECH: noun ROOT: dog NUMBER: plural
like	PART_OF_SPEECH: verb ROOT: like NUMBER: plural
likes	PART_OF_SPEECH: verb ROOT: like NUMBER: singular
man	PART_OF_SPEECH: noun ROOT: man NUMBER: singular
men	PART_OF_SPEECH: noun ROOT: man NUMBER: plural
the	PART_OF_SPEECH: article ROOT: the NUMBER: plural or singular

Rysunek 9 przedstawia ATN dla naszej gramatyki, z pseudokodowymi opisami testów wykonanych na każdym łuku.

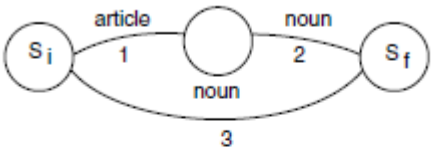
sentence:




```
function sentence-1;
  begin
    NOUN_PHRASE := structure returned by
          noun_phrase network;

    SENTENCE.SUBJECT := NOUN_PHRASE;
  end.
```

```
function sentence-2;
  begin
    VERB_PHRASE := structure returned by
          verb_phrase network;

    if NOUN_PHRASE.NUMBER =
                              VERB_PHRASE.NUMBER
      then begin
        SENTENCE.VERB_PHRASE := VERB_PHRASE;
        return SENTENCE
      end
      else fail
  end.
```

noun_phrase:

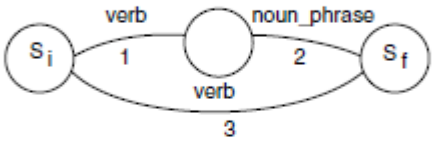



```
function noun_phrase-1;
  begin
    ARTICLE := definition frame for next word of input;

    if ARTICLE.PART_OF_SPEECH=article
      then NOUN_PHRASE.DETERMINER := ARTICLE
      else fail
  end.
```

```
function noun_phrase-2;
  begin
    NOUN := definition frame for next word of input;

    if NOUN.PART_OF_SPEECH=noun and
          NOUN.NUMBER agrees with
              NOUN_PHRASE.DETERMINER.NUMBER
      then begin
        NOUN_PHRASE.NOUN := NOUN;
        NOUN_PHRASE.NUMBER := NOUN.NUMBER
        return NOUN_PHRASE
      end
      else fail
  end.
```

```
function noun_phrase-3
  begin
    NOUN := definition frame for next word of input;

    if NOUN.PART_OF_SPEECH=noun
      then begin
        NOUN_PHRASE.DETERMINER := unspecified;
        NOUN_PHRASE.NOUN := NOUN
        NOUN_PHRASE.NUMBER := NOUN.NUMBER
      end
      else fail
  end.
```

verb_phrase:



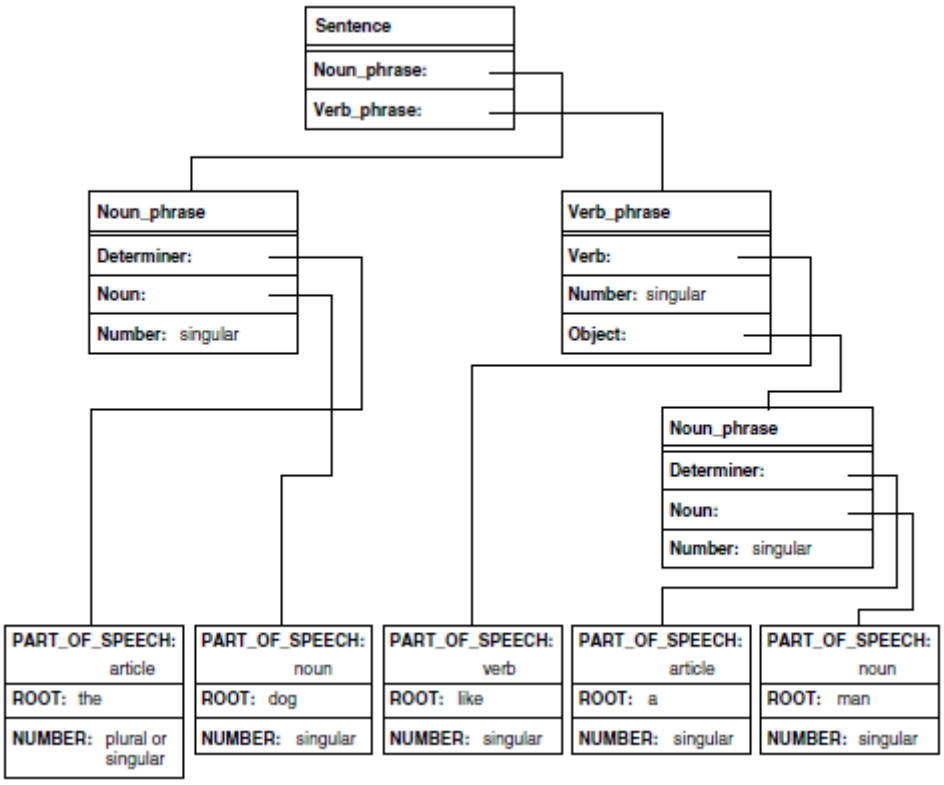

```
function verb_phrase-1
  begin
    VERB := definition frame for next word of input;

    if VERB.PART_OF_SPEECH=verb
      then begin
        VERB_PHRASE.VERB := VERB;
        VERB_PHRASE.NUMBER := VERB.NUMBER;
      end;
  end.
```

```
function verb_phrase-2
  begin
    NOUN_PHRASE := structure returned by
                                  noun_phrase network;

    VERB_PHRASE.OBJECT := NOUN_PHRASE;
    return VERB_PHRASE
  end.
```

```
function verb_phrase-3
  begin
    VERB := definition frame for next word of input;

    if VERB.PART_OF_SPEECH=verb
      then begin
        VERB_PHRASE.VERB := VERB;
        VERB_PHRASE.NUMBER := VERB.NUMBER;
        VERB_PHRASE.OBJECT := unspecified;
        return VERB_PHRASE;
      end;
  end.
```

Łuki są oznaczone zarówno nieterminalami gramatyki, jak i liczbami; liczby te służą do wskazania funkcji przypisanej do każdego łuku. Funkcje te muszą działać pomyślnie, aby przejść po łuku. Kiedy parser wywołuje sieć w celu uzyskania nieterminala, tworzy nową ramkę dla tego nieterminala. Na przykład po wejściu do sieci noun_phrase tworzy nową ramkę noun_phrase. Szczeliny w ramce są wypełnione funkcjami dla tej sieci. Szczelinom tym można przypisać wartości funkcji gramatycznych lub wskaźniki do składników struktury składniowej (np. Verb_phrase może składać się z czasownika i rzeczownik_phrase). Po osiągnięciu stanu końcowego sieć zwraca tę strukturę. Kiedy sieć przechodzi przez łuki oznaczone rzeczownikiem, przedimkiem i czasownikiem, odczytuje następne słowo ze strumienia wejściowego i pobiera definicję tego słowa ze słownika. Jeśli słowo nie jest oczekiwaną częścią mowy, reguła zawodzi; w przeciwnym razie ramką definicji jest zwrócony. Na rysunku 9 ramki i szczeliny są oznaczone przy użyciu notacji Frame.Slot; np. miejsce na numer ramki czasownika jest wskazywane przez VERB.NUMBER. W miarę postępu analizy każda funkcja buduje i zwraca ramkę opisującą powiązaną strukturę składniową. Ta struktura zawiera wskaźniki do struktur zwracanych przez sieci niższego poziomu. Funkcja zdania najwyższego poziomu zwraca strukturę zdania reprezentującą drzewo analizy dla danych wejściowych. Ta struktura jest przekazywana do

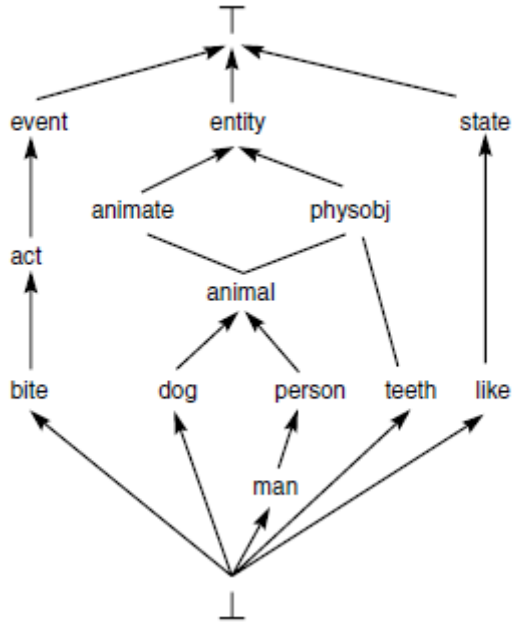
interpretera semantycznego. Rysunek 10 przedstawia drzewo analizy, które jest zwracane dla zdania „Pies lubi człowieka”.



Następna faza przetwarzania języka naturalnego obejmuje drzewo parsowania, takie jak na rysunku 10, i konstruuje semantyczną reprezentację wiedzy dziedzinowej i treści znaczeniowej zdania.

15.3.4 Łączenie wiedzy składniowej i semantycznej z sieciami ATN

Interpreter semantyczny konstruuje reprezentację znaczenia ciągu wejściowego, rozpoczynając od korzenia lub węzła zdania i przechodząc przez drzewo analizy. W każdym węźle rekurencyjnie interpretuje elementy potomne tego węzła i łączy wyniki w jeden wykres pojęciowy; ten wykres jest przekazywany w górę drzewa. Na przykład interpreter semantyczny buduje reprezentację verb_phrase, rekurencyjnie budując reprezentacje elementów potomnych węzła, czasownika i noun_phrase, i łącząc je w celu utworzenia interpretacji fraza czasownikowa. Jest to przekazywane do węzła zdań i łączone z reprezentacją podmiotu. Rekursja zatrzymuje się na terminalach drzewa parsowania. Niektóre z nich, takie jak rzeczowniki, czasowniki i przymiotniki, powodują pobieranie pojęć z bazy wiedzy. Inne, takie jak artykuły, nie odpowiadają bezpośrednio pojęciom w bazie wiedzy, ale kwalifikują inne pojęcia na wykresie. Tłumacz semantyczny w naszym przykładzie korzysta z bazy wiedzy dotyczącej „świata psów”. Pojęcia w bazie wiedzy obejmują obiekty pies i człowiek oraz czynności takie jak i gryzienie. Pojęcia te są opisane w hierarchii typów na rysunku 11.



Oprócz pojęć musimy zdefiniować relacje, które będą używane na naszych grafach koncepcyjnych. W tym przykładzie używamy następujących pojęć: agent łączy akt z pojęciem typu animate. agent definiuje relację pomiędzy akcją a animowanym obiektem, który ją wywołuje. Doświadczający łączy stan z koncepcją typu animowanego. Określa relację między stanem psychicznym a jego doświadczającym.

instrument łączy czynność z bytem i definiuje instrument używany w czynności.

obiekt łączy zdarzenie lub stan z bytem i reprezentuje relację czasownik-obiekt.

part łączy pojęcia typu physobj i definiuje relację między całością a częścią.

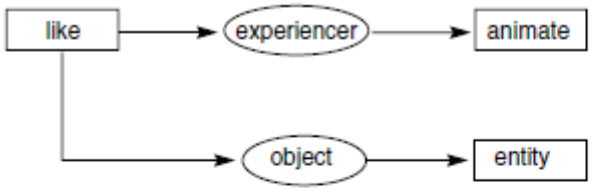
Czasownik odgrywa szczególnie ważną rolę w budowaniu interpretacji, ponieważ określa relacje między podmiotem, przedmiotem i innymi składnikami zdania. Reprezentujemy każdy czasownik za pomocą ramki wielkości liter, która określa:

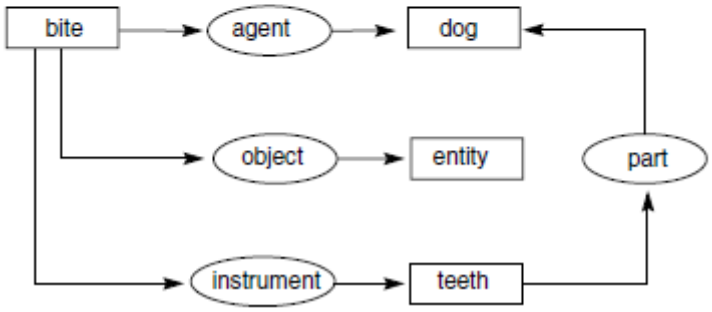
1. Relacje językowe (agent, przedmiot, instrument itd.) Właściwe dla danego czasownika. Na przykład czasowniki przechodnie mają dopełnienie; czasowniki nieprzechodnie nie.

2. Ograniczenia wartości, które mogą być przypisane do dowolnego komponentu ramy obudowy. Na przykład w ramce przypadku dla czasownika „ugryzienia” stwierdziliśmy, że agent musi być w typie pies. Powoduje to odrzucenie wyrażenia „Człowiek gryzie psa” jako niepoprawne semantycznie.

3. Wartości domyślne na elementach ramy obudowy. W ramce „zgryz” mamy domyślną wartość zębów dla pojęcia powiązanego z relacją instrumentu.

Ramki przypadków dla czasowników like i bite pojawiają się na rysunku 12.





Definiujemy działania, które budują reprezentację semantyczną z regułami lub procedurami dla każdego potencjalnego węzła w drzewie parsowania. Reguły dla naszego przykładu są opisane jako procedury pseudokodu. W każdej procedurze, jeśli określone łączenie lub inny test nie powiedzie się, interpretacja ta jest odrzucana jako semantycznie niepoprawna:

procedure sentence;

begin

call noun_phrase to get a representation of the subject;

call verb_phrase to get a representation of the verb_phrase;

using join and restrict, bind the noun concept returned for the subject to

the agent of the graph for the verb_phrase

end.

procedure noun_phrase;

begin

call noun to get a representation of the noun;

case

the article is indefinite and number singular: the noun concept is generic;

the article is definite and number singular: bind marker to noun concept;

number is plural: indicate that the noun concept is plural

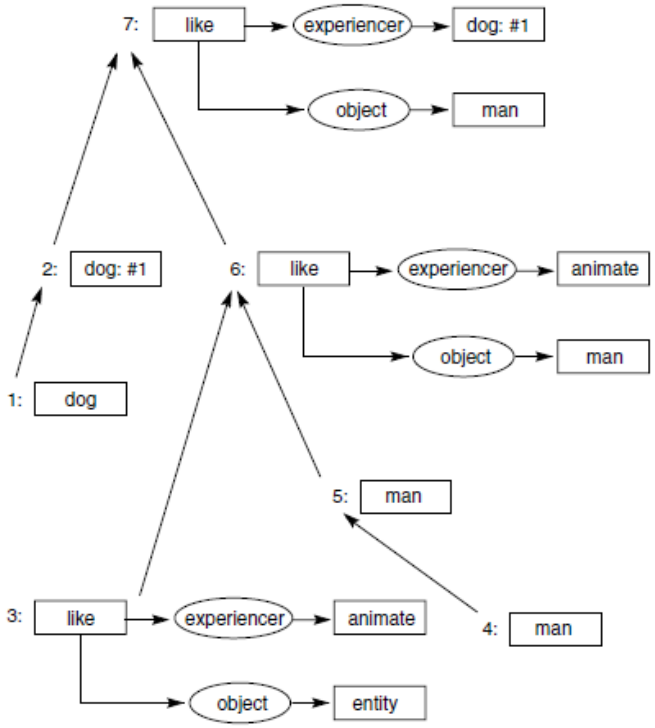
end case

end.

procedure verb_phrase;

```
begin
call verb to get a representation of the verb;
if the verb has an object
then begin
call noun_phrase to get a representation of the object;
using join and restrict, bind concept for object to object of the verb
end
end.
procedure verb;
begin
retrieve the case frame for the verb
end.
procedure noun;
begin
retrieve the concept for the noun
end.
```

Przedimki nie odpowiadają pojęciom w bazie wiedzy, ale określają, czy ich pojęcie rzeczownika jest ogólne czy specyficzne. Nie omawialiśmy reprezentacji pojęć w liczbie mnogiej; odnieś się do Sowy (1984), aby je traktować jako grafy koncepcyjne. Korzystając z tych procedur, wraz z hierarchią pojęć na rysunku 11 i ramkami przypadków z rysunku 12, śledzimy działania interpretera semantycznego w budowaniu semantycznej reprezentacji zdania „Pies lubi człowieka” z drzewa parsowania na rysunku 10 . Ten ślad pojawia się na rysunku 13. Działania podjęte w śladzie to (liczby odnoszą się do rysunku 13):



1. Zaczynając od węzła zdania, nazwij zdanie.

2. zdanie wywołuje noun_phrase.

3. noun_phrase wywołuje rzeczownik.

4. rzeczownik zwraca pojęcie psa rzeczownika (1 na rysunku 13).

5. Ponieważ przedimek jest określony, noun_phrase wiąże indywidualny znacznik z pojęciem (2) i zwraca to pojęcie do zdania.

6. zdanie wywołuje verb_phrase.

7. verb_phrase wywołuje czasownik, który pobiera ramkę wielkości liter dla like (3).

8. verb_phrase wywołuje noun_phrase, które następnie wywołuje rzeczownik, aby pobrać koncepcję man (4).

9. Ponieważ przedimek jest nieokreślony, noun_phrase pozostawia to pojęcie rodzajowe (5).

10. Procedura verb_phrase ogranicza pojęcie bytu w ramce przypadku i łączy je z pojęciem człowieka (6). Ta struktura powraca do zdania.

11. zdanie łączy koncepcję psa: # 1 z węzłem doświadczającym ramki przypadku (7).

Ten wykres pojęciowy przedstawia znaczenie zdania.

Generowanie języka jest pokrewnym problemem, którego dotyczą programy rozumienia języka naturalnego. Generowanie zdań angielskich wymaga skonstruowania semantycznie poprawnego wyniku z wewnętrznej reprezentacji znaczenia. Na przykład agent relacja wskazuje na relację podmiot-czasownik między dwoma pojęciami. Proste podejścia pozwalają na podłączenie odpowiednich słów do zapisanych szablonów zdań. Te szablony są wzorcami zdań i fragmentów, takich jak frazy rzeczownikowe i wyrażenia przyimkowe. Dane wyjściowe są konstruowane, przechodząc po grafie

pojęciowym i łącząc te fragmenty. Bardziej wyrafinowane podejścia do generowania języka wykorzystują gramatykę transformacyjną, aby odwzorować znaczenie na szereg możliwych zdań (Winograd 1972, Allen 1987). W materiałach pomocniczych do tej książki zbudowaliśmy w Prologu pełny semantyczny parser sieci z rekursywnym zstępowaniem. Ten parser używa operatorów grafów, takich jak łączenie i ograniczanie, w celu ograniczenia semantycznej reprezentacji sieci dołączonej do węzłów-liści drzewa parsowania. W części 15.5 pokażemy, jak program może budować wewnętrzne reprezentacje zjawisk językowych. Reprezentacje te są używane przez programy na wiele sposobów, w zależności od aplikacji, z których kilka przedstawiamy. Ale najpierw, w sekcji 15.4, przedstawiamy stochastyczne podejścia do uchwycenia wzorców i prawidłowości języka.

15.4 Stochastyczne narzędzia do rozumienia języka

W rozdziale 15.1 omówiliśmy naturalny rozkład konstrukcji zdań, który wspierał rozumienie języka. W rozdziałach 2 i 3 zauważyliśmy, że semantyczne i wymagające wiedzy aspekty języka mogą być wspierane przez struktury reprezentacyjne na poziomie słów i zdań. W tej sekcji przedstawiamy narzędzia stochastyczne, które na tych samych poziomach wspierają opartą na wzorcach analizę struktur umożliwiających zrozumienie języka.

15.4.1 Wprowadzenie: techniki statystyczne w analizie języka

Statystyczne techniki językowe to metody, które pojawiają się, gdy postrzegamy język naturalny jako proces losowy. W języku potocznym losowość sugeruje brak struktury, definicji lub zrozumienia. Jednak postrzeganie języka naturalnego jako procesu losowego uogólnia deterministyczny punkt widzenia. Oznacza to, że techniki statystyczne (lub stochastyczne) mogą dokładnie modelować zarówno te części języka, które są dobrze zdefiniowane, jak i te, które rzeczywiście mają pewien stopień losowości. Postrzeganie języka jako procesu losowego pozwala nam przedefiniować wiele podstawowych problemów w rozumieniu języka naturalnego w rygorystyczny, matematyczny sposób. Ciekawym ćwiczeniem jest na przykład zrobienie kilku zdań, wypowiedzenie poprzedniego akapitu z kropkami i nawiasami oraz wydrukowanie tych samych słów i symboli językowych uporządkowanych przez generator liczb losowych. Wynik nie będzie miał większego sensu. Warto zauważyć (Jurafsky i Martin 2008), że te same ograniczenia oparte na wzorcach działają na wielu poziomach analizy językowej, w tym na wzorach akustycznych, kombinacjach fonemicznych, analizie struktury gramatycznej i tak dalej. Jako przykład użycia narzędzi stochastycznych, najpierw rozważymy problem tagowania części mowy. Większość ludzi zna ten problem z gramatyki. Chcemy oznaczyć każde słowo w zdaniu jako rzeczownik, czasownik, przyimek, przymiotnik i tak dalej. Ponadto, jeśli słowo jest czasownikiem, możemy chcieć wiedzieć, czy jest czynne, bierne, przechodnie czy nieprzechodnie. Jeśli słowo jest rzeczownikiem, czy jest w liczbie pojedynczej czy mnogiej i tak dalej. Trudność pojawia się w przypadku słów takich jak „swing". Jeśli mówimy „huśtawka na ganku", huśtawka jest rzeczownikiem, ale jeśli mówimy „huśtawka na piłce", huśtawka jest czasownikiem. Przedstawiamy kolejny cytat z Picassa wraz z odpowiednią częścią etykiet mowy:

Art is a lie that lets us see the truth.

Rzeczownik Czasownik Artykuł Rzeczownik Zaimek Czasownik Zaimek Czasownik Artykuł Rzeczownik

Aby rozpocząć naszą analizę, najpierw formalnie zdefiniujemy problem. Mamy zestaw słów w naszym języku $S_w = \{w_1, ..., w_n\}$, na przykład {a, aardvark, ..., zygota} oraz zestaw części mowy lub znaczników $S_t = \{t_1,. .., t_m\}$. Zdanie zawierające n słów jest ciągiem n zmiennych losowych W_1, W_2, ... W_n. Nazywa się to zmiennymi losowymi, ponieważ z pewnym prawdopodobieństwem mogą przyjmować dowolną wartość w S_w. Znaczniki T_1, T_2, ..., T_n są również sekwencją zmiennych losowych. Wartość, jaką przyjmuje T_i, będzie oznaczona t_i, a wartość W_i to wi. Chcemy znaleźć sekwencję wartości dla tych

tagów, co jest najbardziej prawdopodobne, biorąc pod uwagę słowa w zdaniu. Formalnie chcemy wybrać $t_1, ..., t_n$, aby zmaksymalizować:

$p (T_1 = t_1, ..., T_n = t_n \mid W_1 = w_1, ..., W_n = w_n)$

Przypomnijmy, że p (X | Y) oznacza prawdopodobieństwo X przy założeniu, że Y wystąpiło. Zwyczajowo odrzuca się odniesienie do zmiennych losowych i po prostu pisze:

$p (t_1, ..., t_n \mid w_1, ..., w_n)$ równanie 1

Zauważ, że gdybyśmy dokładnie znali ten rozkład prawdopodobieństwa i mielibyśmy wystarczająco dużo czasu, aby zmaksymalizować wszystkie możliwe zestawy znaczników, zawsze otrzymalibyśmy najlepszy możliwy zestaw znaczników dla rozważanych słów. Ponadto, jeśli naprawdę istniałaby tylko jedna poprawna sekwencja znaczników dla każdego zdania, pomysł, który mógł zaakceptować twój nauczyciel gramatyki, ta probabilistyczna technika zawsze znajdowałaby tę poprawną sekwencję! Zatem prawdopodobieństwo prawidłowej sekwencji wyniesie 1, a dla wszystkich pozostałych ciągów 0. To właśnie mieliśmy na myśli, mówiąc, że statystyczny punkt widzenia może uogólniać deterministyczny. W rzeczywistości, z powodu ograniczonej przestrzeni dyskowej, danych i czasu, nie możemy zastosować tej techniki i musimy wymyślić jakiś rodzaj przybliżenia. Reszta tej sekcji dotyczy coraz lepszych sposobów przybliżania równania 1. Po pierwsze, zauważ, że możemy przepisać równanie 1 w bardziej użyteczny sposób:

$$p(t_1, ..., t_n \mid w_1, ..., w_n) = p(t_1, ..., t_n , w_1, ..., w_n) / p(w_1, ..., w_n)$$

a ponieważ maksymalizujemy to, wybierając $t_1, ..., t_n$, możemy uprościć równanie 1 do:

$$p(t_1, ..., t_n, w_1,..., w_n) =$$

$$p(t_1)p(w_1 \mid t_1)p(t_2 \mid t_1,w_1) ... p(t_n \mid w_1, ..., w_n , t_1, ..., t_{n-1}) =$$

$$\prod_{i=1}^{n} p(t_i \mid t_1, ..., t_{i-1}, w_1, ..., w_{i-1}) \, p(w_i \mid t_1, ..., t_{i-1}, w_1, ..., w_{i-1}) \qquad \text{equation 2}$$

Zauważ, że równanie 2 jest równoważne równaniu 1.

15.4.2 Podejście modelowe Markowa

W praktyce, jak widzieliśmy wcześniej w naszej dyskusji na temat rozdziałów 9.3 i 13, maksymalizacja równań z prawdopodobieństwami uwarunkowanymi wieloma innymi zmiennymi losowymi, takimi jak te, które znajdujemy w równaniu 2., jest zwykle złożonym zadaniem: po pierwsze , to jest trudne do przechowywania prawdopodobieństwa zmiennej losowej uwarunkowanej wieloma innymi zmiennymi losowymi, ponieważ liczba możliwych prawdopodobieństw rośnie wykładniczo wraz z liczbą zmiennych warunkujących. Po drugie, nawet gdybyśmy mogli przechowywać wszystkie wartości prawdopodobieństwa, często trudno jest oszacować ich wartości. Szacowanie jest zwykle wykonywane empirycznie, licząc liczbę wystąpień zdarzenia w ręcznie oznaczonym korpusie treningowym, a zatem, jeśli zdarzenie występuje tylko kilka razy w tym zbiorze uczącym, nie uzyskamy dobrego oszacowania jego prawdopodobieństwa. Oznacza to, że łatwiej jest oszacować p (kot | the) niż p (kot | Pies gonił), ponieważ będzie mniej wystąpień tego ostatniego w zestawie szkoleniowym. Wreszcie, znalezienie łańcucha znaczników, który maksymalizuje struktury takie jak równanie 2, zajęłoby zbyt dużo czasu, co zostanie pokazane poniżej.

Najpierw wykonujemy użyteczne przybliżenia równania 2. Pierwsza zgrubna próba to:

$p(t_i \mid t_1, ..., t_{i-1}, w_1, ..., w_{i-1})$ podejścia $p(t_i \mid t_{i-1})$ i $p(w_i \mid t_1, ..., t_{i-1}, w_1, ..., w_{i-1})$ zbliża się do $p(w_i \mid t_i)$.

Są to założenia Markowa pierwszego rzędu, gdzie zakłada się, że rozważana rzecz obecna jest zależna tylko od rzeczy bezpośrednio poprzedzającej, tj. Jest niezależna od rzeczy z bardziej odległej przeszłości. Podłączając te przybliżenia z powrotem do równania 2, otrzymujemy

$$\prod_{I=1}^{n} p(t_i \mid t_{i-1}) \, p(w_i \mid t_i) \qquad \text{equation 3}$$

Równanie 3 jest proste w obsłudze, ponieważ jego prawdopodobieństwa można łatwo oszacować i zapisać. Przypomnijmy, że równanie 3 jest tylko oszacowaniem $P(t_1, ..., t_n \mid w_1, ..., w_n)$ i nadal musimy je zmaksymalizować, wybierając tagi, tj. $t_1, ..., t_n$. Na szczęście istnieje algorytm programowania dynamicznego (DP) zwany algorytmem Viterbiego który pozwoli nam to zrobić. Algorytm Viterbiego oblicza prawdopodobieństwo wystąpienia sekwencji znaczników t2 dla każdego słowa w zdaniu, gdzie t jest liczbą możliwych tagów. W przypadku konkretnego kroku rozważane sekwencje znaczników mają następującą postać:

artykuł artykuł {best tail}

artykuł czasownik {best tail}

...

artykuł rzeczownik {best tail}

...

...

artykuł rzeczownikowy {best tail}

...

rzeczownik rzeczownik {best tail}

gdzie {best tail} jest najbardziej prawdopodobną sekwencją tagów znalezionych dynamicznie dla ostatnich n - 2 słów dla danego znacznika n - 1. W tabeli znajduje się wpis dla każdej możliwej wartości numeru tagu n - 1 i tagu liczba n (stąd mamy sekwencje znaczników t^2). Na każdym kroku algorytm znajduje maksymalne prawdopodobieństwo i dodaje jeden znacznik do każdej najlepszej sekwencji ogona. Algorytm ten gwarantuje znalezienie sekwencji znaczników, która maksymalizuje równanie 3 i działa w $O(t^2 s)$, gdzie t jest liczbą znaczników, a s jest liczbą słów w zdaniu. Jeśli $p(t_i)$ jest uwarunkowane na ostatnich n znacznikach, a nie na ostatnich dwóch, algorytm Viterbiego przyjmie $O(t^n s)$. W ten sposób widzimy, dlaczego warunkowanie zbyt wielu przeszłych zmiennych wydłuża czas potrzebny do znalezienia wartości maksymalizującej. Na szczęście przybliżenia użyte w równaniu 3 działają dobrze. Z około 200 możliwymi tagami i dużym zestawem uczącym do szacowania prawdopodobieństwa, tagger używający tych metod jest dokładny w około 97%, co jest zbliżone do ludzkiej dokładności. Zaskakująca dokładność przybliżenia Markowa wraz z jego prostotą sprawia, że jest ono przydatne w wielu zastosowaniach. Na przykład większość systemów rozpoznawania mowy wykorzystuje tzw. Model trygramowy, aby dostarczyć systemowi pewnej „wiedzy gramatycznej" do przewidywania słów, które wypowiedział użytkownik. Model trygramowy to prosty model Markowa, który szacuje prawdopodobieństwo aktualnego słowa uwarunkowane dwoma poprzednimi słowami. Wykorzystuje

algorytm Viterbiego i inne właśnie opisane techniki. Aby uzyskać więcej informacji na temat tej i pokrewnych technik, zobacz Jurasky i Martin.

15.4.3 Podejście drzewa decyzyjnego

Oczywistym problemem związanym z podejściem Markowa jest to, że uwzględnia ono tylko kontekst lokalny. Jeśli zamiast oznaczać słowa prostymi częściami mowy, chcemy np. Zidentyfikować agenta, zidentyfikować przedmiot lub zdecydować, czy czasowniki są aktywne, czy pasywne, to wymagany jest bogatszy kontekst. Poniższe zdanie ilustruje ten problem: Polityka ogłoszona w grudniu przez Prezydenta gwarantuje niższe podatki. W rzeczywistości prezydent jest agentem, ale program wykorzystujący model Markowa prawdopodobnie zidentyfikowałby politykę jako agenta i ogłosił jako aktywny czasownik. Możemy sobie wyobrazić, że program byłby lepszy w probabilistycznym wyborze agenta tego typu zdania, gdyby można zadawać pytania typu: „Czy obecny rzeczownik jest nieożywiony?" lub „Czy słowo obok występuje kilka słów przed rozważanym rzeczownikiem?" Przypomnij sobie, że problem tagowania jest równoważny maksymalizowaniu równania 2, tj.

$$\prod_{i=1}^{n} p(t_i \mid t_1, \ldots, t_{i-1}, w_1, \ldots, w_{i-1})\, p(w_i \mid t_1, \ldots, t_{i-1}, w_1, \ldots, w_{i-1}).$$

Teoretycznie rozważenie szerszego kontekstu wymaga po prostu znalezienia lepszych szacunków dla tych prawdopodobieństw. Sugeruje to, że moglibyśmy chcieć użyć odpowiedzi na powyższe pytania gramatyczne, aby sprecyzować prawdopodobieństwa. Istnieje kilka sposobów rozwiązania tego problemu. Po pierwsze, możemy połączyć podejście Markowa z technikami analizy przedstawionymi w pierwszych trzech sekcjach tego rozdziału. Druga metoda pozwala nam znaleźć prawdopodobieństwa uwarunkowane pytaniami tak lub nie za pomocą algorytmu ID3 lub innego równoważnego algorytmu. Drzewa ID3 mają tę dodatkową zaletę, że z bardzo dużego zestawu możliwych pytań wybiorą tylko te, które są dobre w precyzowaniu szacunków prawdopodobieństwa. W przypadku bardziej skomplikowanych zadań przetwarzania języka naturalnego, takich jak parsowanie, drzewa oparte na ID3 są często preferowane zamiast modeli Markowa. Następnie opiszemy, jak używać ID3 do konstruowania drzewa decyzyjnego używanego podczas analizowania. Przypomnij sobie, że w powyższej sekcji zadaliśmy pytanie „Czy obecny rzeczownik jest nieożywiony?" Takie pytania możemy zadawać tylko wtedy, gdy wiemy, które słowa są ożywione, a które nieożywione. W rzeczywistości istnieje zautomatyzowana technika, która może przypisywać nam słowa do tego typu zajęć. Technika ta nazywa się wzajemnym grupowaniem informacji. Informacje o wzajemnej wymianie między dwiema zmiennymi losowymi X i Y są zdefiniowane w następujący sposób:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)}$$

Aby dokonać wzajemnego grupowania informacji na podstawie słownika słów, zaczynamy od umieszczenia każdego słowa ze słownika w odrębnym zestawie. Na każdym kroku obliczamy średnią wzajemną informację między zbiorami za pomocą bigramu, czyli modelu następnego słowa, i wybierane jest połączenie dwóch zestawów słów, co minimalizuje utratę średniej wzajemnej informacji dla wszystkich klas. Na przykład, jeśli początkowo mamy słowa cat, kitten, run i green, to w pierwszym kroku algorytmu mamy zestawy:

{kot} {kotek} {bieg} {zielony}.

Jest prawdopodobne, że prawdopodobieństwo wystąpienia następnego słowa, biorąc pod uwagę, że poprzednie słowo było kotem, jest prawie równe prawdopodobieństwu następnego słowa, biorąc pod uwagę, że poprzednie słowo było kociakiem. Innymi słowy:

p (eats | cat) jest mniej więcej takie samo jak p (eats | kitten)

p (miauczy | kot) jest mniej więcej tym samym, co p (miauczy | kotek)

Tak więc, jeśli pozwolimy X1, X2, Y1 i Y2 być zmiennymi losowymi takimi, że:

X1 = {{kot}, {kotek}, {bieg}, {zielony}}

Y1 = słowo następujące po X1

X2 = {{kot, kotek}, {bieg}, {zielony}}

Y2 = słowo następujące po X2,

wtedy wzajemne informacje między X2 i Y2 są niewiele mniejsze niż wzajemne informacje między X1 i Y1, więc kot i kociak prawdopodobnie zostaną połączone. Jeśli będziemy kontynuować tę procedurę, dopóki nie połączymy wszystkich możliwych klas, otrzymamy drzewo binarne. Następnie można przypisać kody bitowe do słów w oparciu o gałęzie pobrane w drzewie, które docierają do węzła-liścia, w którym znajduje się to słowo. Odzwierciedla to semantyczne znaczenie tego słowa. Na przykład:

cat = 01100011

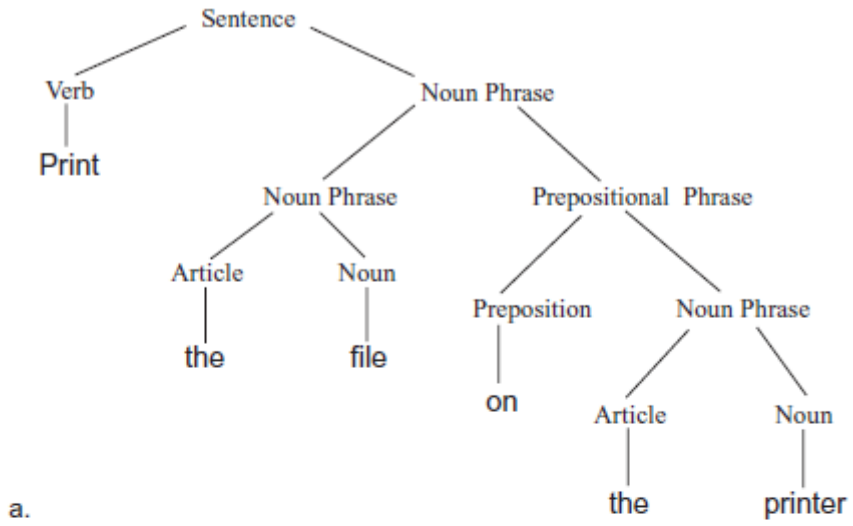
kociak = 01100010

Co więcej, możemy stwierdzić, że słowa „podobne do rzeczowników" to wszystkie te słowa, które mają 1 w lewym bicie, a słowami, które najprawdopodobniej reprezentują przedmioty nieożywione, mogą być te, których trzecim bitem jest 1.

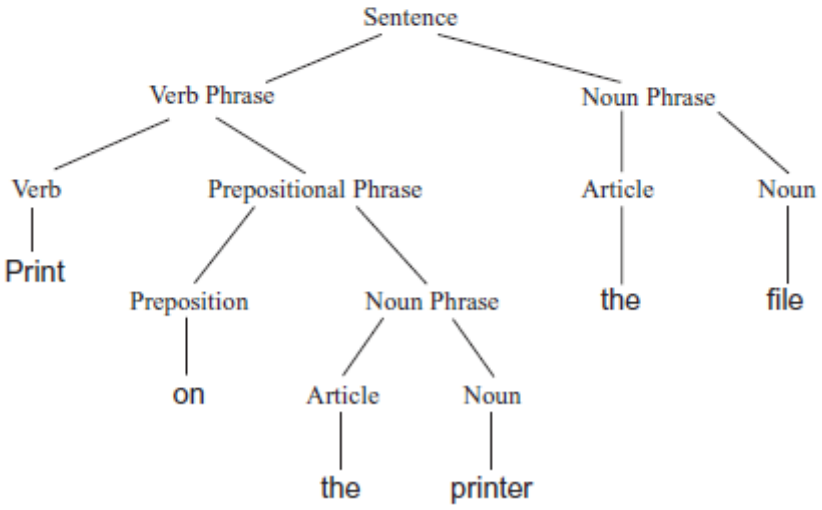
To nowe kodowanie słów ze słownika umożliwia bardziej efektywne zadawanie pytań przez parser. Zwróć uwagę, że grupowanie nie bierze pod uwagę kontekstu, więc nawet jeśli „książka" może być skupiona jako słowo „podobne do rzeczownika", będziemy chcieli, aby nasz model oznaczył go jako czasownik, gdy zostanie znaleziony w wyrażeniu „książka lot."

15.4.4 Probabilistyczne podejścia do analizy

Techniki stochastyczne były już stosowane w wielu dziedzinach lingwistyki komputerowej i nadal istnieje wiele możliwości zastosowania ich w obszarach, które oparły się tradycyjnym, symbolicznym podejściom. Zastosowanie metod statystycznych w parsowaniu było po raz pierwszy motywowane problemem niejednoznaczności. Niejednoznaczność wynika z faktu, że często istnieje kilka możliwych parsów dla danego zdania i musimy wybrać, która pars może być najlepsza. Na przykład zdanie Drukuj plik na drukarce można przeanalizować za pomocą jednego z dwóch drzew przedstawionych na rysunku 14.



a.



b.

W takich sytuacjach same reguły gramatyczne nie wystarczą, aby wybrać poprawną analizę. W przypadku Drukuj plik na drukarce musimy wziąć pod uwagę pewne informacje dotyczące kontekstu i semantyki. W rzeczywistości podstawowym zastosowaniem technik stochastycznych w dziedzinie analizowania jest pomoc w rozwiązywaniu niejednoznaczności. W obecnym przykładzie możemy użyć tego samego narzędzia, które jest używane w części znakowania mowy, algorytmu ID3. ID3 pomaga nam w przewidywaniu prawdopodobieństwa, że parsowanie jest poprawne na podstawie semantycznych pytań dotyczących zdania. W przypadku, gdy w zdaniu występuje pewna niejednoznaczność składniowa, możemy to wybrać parse, która ma największe prawdopodobieństwo, że będzie poprawna. Jak zwykle ta technika wymaga dużego korpusu szkoleniowego zdań wraz z ich poprawnymi analizami. Ostatnio ludzie ze społeczności zajmujących się modelowaniem statystycznego języka naturalnego stali się bardziej ambitni i próbowali używać technik statystycznych bez gramatyki do analizowania. Chociaż szczegóły analizy bez gramatyki wykraczają poza zakres tej książki, wystarczy powiedzieć, że jest to bardziej związane z rozpoznawaniem wzorców niż z tradycyjnymi technikami analizowania opisanymi wcześniej w tej części. Analiza gramatyczna była całkiem udana. W eksperymentach porównujących tradycyjny parser oparty na gramatyce z parserem bez gramatyki, zajmującym się analizowaniem tego samego zestawu zdań, parser gramatyczny uzyskał wynik, używając popularnej miary, w nawiasach 69%, a parser bez gramatyki 78%. Wyniki te są dobre, choć

nie wybitne. Co ważniejsze, gramatyka w tradycyjnym parserze została skrupulatnie opracowana przez wyszkolonego językoznawcę w ciągu około dziesięciu lat, podczas gdy parser bez gramatyki zasadniczo nie używał zakodowanych na stałe informacji językowych, a jedynie wyrafinowane modele matematyczne, które mogłyby wywnioskować potrzebne informacje ze szkolenia. dane. Więcej informacji na temat parsowania bez gramatyki i pokrewnych zagadnień można znaleźć w Manning i Schutze. Rozumienie mowy, konwersja mowy na tekst i rozpoznawanie pisma ręcznego to trzy kolejne obszary, które mają długą historię stosowania stochastycznych metod modelowania języka. Najczęściej stosowaną metodą statystyczną w tych obszarach jest model trygramowy do przewidywania następnego słowa. Siła tego modelu tkwi w jego prostocie: przewiduje on następne słowo na podstawie dwóch ostatnich słów. Ostatnio w społeczności języków statystycznych podjęto prace nad utrzymaniem prostoty i łatwości użycia tego modelu, przy jednoczesnym uwzględnieniu ograniczeń gramatycznych i zależności z większej odległości. To nowe podejście wykorzystuje tak zwane trygramy gramatyczne. Trygramy gramatyczne są oparte na podstawowych skojarzeniach między parami słów (tj. Podmiot-czasownik, rodzajnik-rzeczownik i czasownik-przedmiot). Łącznie te skojarzenia nazywane są gramatyką linków. Gramatyka linków jest znacznie prostsza i łatwiejsza do skonstruowania niż tradycyjne gramatyki używane przez językoznawców i dobrze działa z metodami probabilistycznymi. Berger i inni opisują program statystyczny Candide, który tłumaczy tekst francuski na tekst angielski. Kandyd używa zarówno statystyki, jak i teorii informacji, aby opracować model prawdopodobieństwa procesu tłumaczenia. Szkoli tylko na dużym korpusie francuskim i Angielskie pary zdań i dają wyniki porównywalne, aw niektórych przypadkach lepsze niż Systran, komercyjny program tłumaczeniowy. Szczególnie interesujący jest fakt, że system Candide nie wykonuje tradycyjnej analizy w procesie tłumaczenia. Zamiast używać trygramów gramatycznych i gramatyk linków, o których właśnie wspomniałem.

15.4.5 Probabilistyczne parsery bezkontekstowe

W tej sekcji przedstawiamy dwa różne probabilistyczne parsery bezkontekstowe, oparte na strukturze i leksykalizowane. Parser oparty na strukturze używa miar prawdopodobieństwa dla każdej występującej reguły analizy gramatycznej. Prawdopodobieństwo każdej reguły parsowania jest funkcją tej reguły w połączeniu z prawdopodobieństwami jej składników. Demonstrujemy probabilistyczny parser oparty na strukturze, rozszerzając zestaw bezkontekstowych reguł gramatycznych z sekcji 15.2 o miary prawdopodobieństwa:

1. zdanie ↔ rzeczownik_fraza czasownik_fraza p (s) = p (r1) p (np) p (vp)

2. noun_phrase ↔ rzeczownik p (np) = p (r2) p (rzecz.)

3. noun_phrase ↔ rzeczownik przedimka p (np) = p (r3) p (przedimek) p (rzeczownik)

4. verb_phrase ↔ czasownik p (vp) = p (r4) p (czasownik)

5. verb_phrase ↔ czasownik rzeczownik_phrase p (vp) = p (r5) p (czasownik) p (np)

6. artykuł ↔ a p (artykuł) = p (a)

7. artykuł ↔ p (artykuł) = p (the)

8. rzeczownik ↔ mężczyzna p (rzecz.) = P (mężczyzna)

9. rzeczownik ↔ dog p (rzecz.) = P (dog)

10. czasownik ↔ lubi p (rzeczownik) = p (lubi)

11. czasownik ↔ ukąszenia p (rzecz.) = P (ugryzienia)

Miary prawdopodobieństwa dla poszczególnych słów można wyliczyć z prawdopodobieństwa wystąpienia w określonym korpusie zdań angielskich. Miarę prawdopodobieństwa dla każdej reguły gramatycznej, p(ri), można określić na podstawie tego, jak często ta reguła gramatyczna występuje w zdaniach korpusu języka. Oczywiście przykład tutaj jest prosty - ma na celu scharakteryzowanie tego typu probabilistycznego parsera bezkontekstowego. Drugi przykład, probabilistyczny, leksykalizowany bezkontekstowy parser, również zostanie zademonstrowany poprzez rozszerzenie reguł gramatycznych z sekcji 15.2. W tej sytuacji chcemy wziąć pod uwagę wiele aspektów wyroku. Po pierwsze, możemy mieć prawdopodobieństwa użycia języka w bigramie lub trygramie. Ponownie będziemy zależni od odpowiedniego korpusu językowego, aby uzyskać miary bigramów lub trygramów. Podobnie jak w naszym poprzednim przykładzie, użyjemy miar prawdopodobieństwa, uzyskanych z korpusu języka, dla wystąpienia każdej z reguł analizy. Na koniec będziemy mieli probabilistyczne miary poszczególnych rzeczowników i czasowników występujących razem. Na przykład, jeśli rzeczownikiem podmiotu jest pies. prawdopodobieństwo, że czasownik będzie ukąszony. Zwróć uwagę, że ten środek, nawet jeśli pochodzi z korpusu języka, może również wymusić zgodność rzeczownika-czasownika. Nasz parser nazywamy leksykalizacją, ponieważ jesteśmy w stanie skupić się na prawdopodobieństwach występowania określonych wzorców słów. Dla uproszczenia przedstawiamy nasz parser tylko dla bigramów:

1. sentence ↔ noun_phrase(noun) verb_phrase(verb)

p(sentence) = p(r1) p(np) p(vp) p(verb | noun)

2. noun_phrase ↔ noun

p(np) = p(r2) p(noun)

3. noun_phrase ↔ article noun

p(np) = p(r3) p(article) p(noun) p(noun | article)

4. verb_phrase ↔ verb

p(vp) = p(r4) p(verb)

5. verb_phrase ↔ verb noun_phrase

p(vp) = p(r5) p(verb) p(noun_phrase(noun)) p(noun |verb)

6. article ↔ a p(article) = p(a)

7. article ↔ the p(article) = p(the)

8. noun ↔ man p(noun) = p(man)

. p(man | a)

p(man | the)

etc.

p(likes | man)

etc.

Każdy z dwóch probabilistycznych parserów tej sekcji został zbudowany w Prologu w materiałach pomocniczych do tego rozdziału. Istnieje wiele innych obszarów, w których rygorystyczne techniki

modelowania języka stochastycznego nie zostały jeszcze wypróbowane, ale mogą przynieść użyteczne wyniki. Jednym z potencjalnych zastosowań jest ekstrakcja informacji, czyli problem uzyskania określonej ilości konkretnych informacji z tekstu pisanego. Po drugie, można dopasować wzorce w mowie dźwięcznej, aby połączyć je z koncepcjami określonej bazy wiedzy. Wreszcie jest oczywiście wyszukiwanie w sieci semantycznej. Więcej szczegółów na temat stochastycznych podejść do przetwarzania języka naturalnego można znaleźć w Jurafsky i Martin oraz Manning i Schutze.

15.5 Aplikacje języka naturalnego

15.5.1 Rozumienie historii i odpowiadanie na pytania

Ciekawym testem na technologię rozumienia języka naturalnego jest napisanie programu, który potrafi czytać opowiadanie lub inny fragment tekstu w języku naturalnym i odpowiadać na pytania na jego temat. W rozdziale Części 7 omówiliśmy niektóre kwestie reprezentacyjne związane ze zrozumieniem historii, w tym znaczenie połączenia wiedzy podstawowej z wyraźną treścią tekstu. Jak pokazano na rysunku 2, program może to osiągnąć, wykonując połączenia sieciowe między semantyczną interpretacją danych wejściowych a koncepcyjnymi strukturami grafów w bazie wiedzy. Bardziej wyrafinowane reprezentacje, takie jak skrypty, mogą modelować bardziej złożone sytuacje obejmujące zdarzenia zachodzące w czasie. Gdy program zbuduje rozszerzoną reprezentację tekstu, może inteligentnie odpowiadać na pytania o to, co przeczytał. Program analizuje pytanie w wewnętrzną reprezentację i dopasowuje to zapytanie do rozszerzonej reprezentacji historii. Rozważmy przykład z rysunku 2. Program odczytał zdanie „Tarzan pocałował Jane” i zbudował rozszerzoną reprezentację. Załóżmy, że pytamy program „Kto kocha Jane?” Analizując pytanie, pytający, kto, co, dlaczego itp. Wskazuje na intencję pytania. Kto pyta o agenta akcji; jakie pytania dotyczą przedmiotu działania; jak pytania dotyczą środków, za pomocą których czynność została wykonana i tak dalej. Pytanie „Kto kocha Jane?” tworzy wykres z rysunku 15. Węzeł agenta na wykresie jest oznaczony? aby wskazać, że jest to cel pytania. Ta struktura jest następnie łączona z rozszerzoną reprezentacją tekstu. Pojęcie, które wiąże się z osobą:? pojęcie na wykresie zapytań to odpowiedź na pytanie: „Tarzan kocha Jane”.

15.5.2 Interfejs bazy danych

Głównym wąskim gardłem w projektowaniu programów rozumienia języka naturalnego jest zdobycie wystarczającej wiedzy na temat domeny dyskursu. Obecna technologia jest ograniczona do wąskich dziedzin z dobrze zdefiniowaną semantyką. Obszarem zastosowań spełniającym te kryteria jest rozwój interfejsów w języku naturalnym dla baz danych. Chociaż bazy danych przechowują ogromne ilości informacji, informacje te są bardzo regularne i mają wąski zakres; ponadto semantyka bazy danych jest dobrze zdefiniowana. Te funkcje, wraz z użytecznością bazy danych, która może akceptować zapytania w języku naturalnym, sprawia, że interfejsy baz danych są ważnym zastosowaniem technologii rozumienia języka naturalnego. Zadaniem frontendu bazy danych jest przetłumaczenie pytania w języku naturalnym na dobrze sformułowane zapytanie w języku bazy danych. Na przykład przy użyciu bazy danych SQL język jako cel (Ullman 1982), interfejs języka naturalnego tłumaczyłby pytanie „Kto zatrudnił Johna Smitha?” do zapytania:

SELECT MANAGER

FROM MANAGER_OF_HIRE

WHERE EMPLOYEE = John Smith

Wykonując to tłumaczenie, program musi zrobić coś więcej niż tylko przetłumaczyć oryginalne zapytanie; musi również zdecydować, gdzie szukać w bazie danych (relacja MANAGER_OF_HIRE), nazwa pola, do którego ma być dostęp (MANAGER), oraz ograniczenia zapytania (PRACOWNIK = „Jan

Kowalski"). Żadna z tych informacji nie znajdowała się w pierwotnym pytaniu; został znaleziony w bazie wiedzy, która wiedziała o organizacji bazy danych i znaczeniu potencjalnych pytań. Relacyjna baza danych organizuje dane w relacjach między domenami jednostek. Załóżmy na przykład, że tworzymy bazę danych pracowników i chcielibyśmy uzyskać dostęp do wynagrodzenia każdego pracownika oraz menedżera, który ją zatrudnił. Ta baza danych składałaby się z trzech domen lub zbiorów jednostek: zbioru menedżerów, zbioru pracowników i zbioru wynagrodzeń. Moglibyśmy uporządkować te dane w dwie relacje, wynagrodzenie_pracownika, które odnosi się do pracownika i jej pensji, oraz manager_of_hire, które odnosi się do pracownika i jego kierownika. W relacyjnej bazie danych relacje są zwykle wyświetlane jako tabele, które wyliczają wystąpienia relacji. Kolumny tabel są często nazywane; nazwy te nazywane są atrybutami relacji. Rysunek 16 przedstawia tabele dla relacji wynagrodzenie_pracownika i wynagrodzenia_zaawansowanego_w hrabstwie.

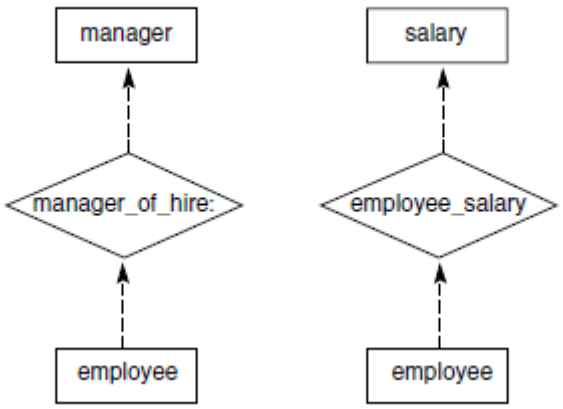
manager_of_hire:

employee	manager
John Smith	Jane Martinez
Alex Barrero	Ed Angel
Don Morrison	Jane Martinez
Jan Claus	Ed Angel
Anne Cable	Bob Veroff

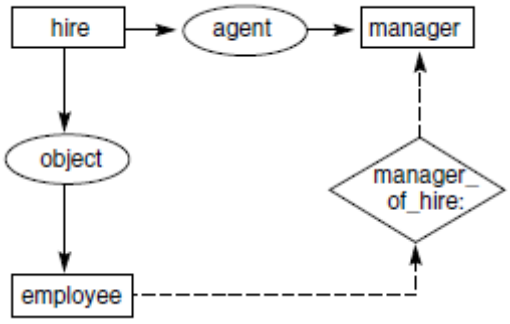
employee_salary:

employee	salary
John Smith	$35,000.00
Alex Barrero	$42,000.00
Don Morrison	$50,000.00
Jan Claus	$40,000.00
Anne Cable	$45,000.00

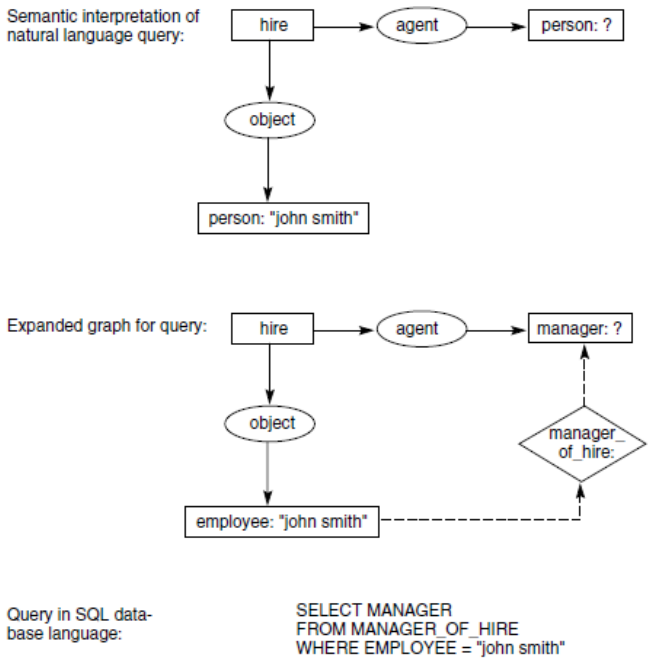
Manager_of_hire ma dwa atrybuty: pracownika i kierownika. Wartości relacji są parami pracowników i menedżerów. Jeśli założymy, że pracownicy mają unikalne imię i nazwisko, menedżera i pensję, to nazwisko pracownika może służyć jako klucz zarówno do wynagrodzenia, jak i do atrybutów menedżera. Atrybut jest kluczem do innego atrybutu, jeśli jednoznacznie określa wartość elementów innego atrybutu. Prawidłowe zapytanie wskazuje atrybut docelowy i określa wartość lub zestaw ograniczeń; baza danych zwraca określone wartości atrybutu docelowego. Zależności między kluczami i innymi atrybutami możemy wskazać graficznie na wiele sposobów, w tym na diagramach relacji encja-jednostka i diagramach przepływu danych . Oba te podejścia wyświetlają mapowanie kluczy na atrybuty przy użyciu skierowanych wykresów. Grafy pojęciowe możemy rozszerzyć o diagramy tych zależności . Relacja bazy danych definiująca odwzorowanie jest oznaczona rombem, na którym znajduje się nazwa relacji. Atrybuty relacji są wyrażone jako pojęcia na grafie pojęciowym, a kierunek strzałek wskazuje przyporządkowanie kluczy do innych atrybutów. Wykresy relacji jednostka-jednostka dla relacji wynagrodzenie_pracownik i kierownik_w_hire można zobaczyć na rysunku 17.



Podczas tłumaczenia z języka angielskiego na formalne zapytanie musimy określić rekord zawierający odpowiedź, pole tego rekordu, które ma zostać zwrócone, oraz wartości kluczy określających to pole. Zamiast tłumaczyć bezpośrednio z języka angielskiego na język bazy danych, najpierw tłumaczymy na bardziej wyrazisty język, taki jak grafy pojęciowe. Jest to konieczne, ponieważ wiele zapytań w języku angielskim jest niejednoznacznych lub wymaga dodatkowej interpretacji w celu utworzenia poprawnie sformułowanego zapytania do bazy danych. Użycie bardziej wyrazistej reprezentacji język pomaga w tym procesie. Interfejs użytkownika języka naturalnego analizuje i interpretuje zapytanie w postaci grafu pojęciowego, jak opisano wcześniej w tym rozdziale. Następnie łączy ten wykres z informacjami w bazie wiedzy za pomocą operacji łączenia i ograniczania. W tym przykładzie chcemy obsługiwać zapytania, takie jak „Kto zatrudnił Jana Kowalskiego?" lub „Ile zarabia Jan Kowalski?" Dla każdego potencjalnego zapytania przechowujemy wykres, który definiuje jego czasownik, role przypadków dla tego czasownika oraz wszelkie istotne diagramy relacji encji dla pytania. Rysunek 18 przedstawia wpis w bazie wiedzy dla czasownika „zatrudniać".



Interpreter semantyczny tworzy wykres zapytania użytkownika i łączy ten wykres z odpowiednim wpisem w bazie wiedzy. Jeśli istnieje dołączony wykres relacji encji, który odwzorowuje klucze w celu pytania, program może użyć tego wykresu relacji encji do utworzenia zapytania do bazy danych. Rysunek 19 przedstawia wykres zapytań dla pytania „Kto zatrudnił Johna Smitha?" oraz wynik połączenia tego z wpisem do bazy wiedzy z rysunku 18.



Pokazuje również zapytanie SQL utworzone na podstawie tego wykresu. Zwróć uwagę, że nazwa odpowiedniego rekordu, pole docelowe i klucz zapytania nie zostały określone w zapytaniu w języku naturalnym. Zostały one wywiedzione z bazy wiedzy. Na rysunku 18 agent i obiekt pierwotnego zapytania były znane tylko jako typ person. Aby połączyć je z wpisem do bazy wiedzy do wynajęcia, były one najpierw ograniczone odpowiednio do typów menedżerów i pracowników. Hierarchia typów może zatem służyć do sprawdzania typów w możliwych rozwiązaniach pierwotnego zapytania. Gdyby John Smith nie był typem pracownika, pytanie byłoby niepoprawne i program mógłby to wykryć. Po zbudowaniu rozszerzonego wykresu zapytania program sprawdza koncepcję docelową oznaczoną znakiem? I ustala, że relacja manager_of_hire mapowała klucz na tę koncepcję. Ponieważ klucz jest powiązany z wartością john smith, pytanie było poprawne i program utworzyłby prawidłowe zapytanie do bazy danych. Tłumaczenie tego wykresu relacji encji na język SQL lub inny dowolny inny język baz danych ogólnego przeznaczenia jest proste. Chociaż ten przykład jest uproszczony, ilustruje użycie podejścia opartego na wiedzy do budowania frontonu bazy danych języka naturalnego. Idee w naszym przykładzie są wyrażone na grafach pojęciowych, ale równie dobrze można je odwzorować na inne reprezentacje, takie jak ramy lub formalizmy oparte na logice predykatów.

15.5.3 System pozyskiwania i podsumowywania informacji w sieci WWW

Sieć WWW oferuje wiele ekscytujących wyzwań, a także interesujące możliwości badań nad sztuczną inteligencją i rozumieniem języka naturalnego. Jedną z największych jest potrzeba zaprojektowania inteligentnego oprogramowania, które potrafi podsumować „ciekawe” materiały internetowe. W rzeczywistości problem ten składa się z dwóch części: lokalizowania potencjalnie interesujących informacji w Internecie, a następnie wydobywania (lub wydobywania) krytycznych składników tych informacji. Chociaż pierwszy problem związany z lokalizacją potencjalnie interesujących informacji jest krytyczny), teraz rozważymy podejście do dopasowywania i wypełniania szablonów w celu rozwiązania drugiego problemu. Po zlokalizowaniu informacji, być może przez dopasowanie słów kluczowych lub użycie bardziej wyrafinowanej wyszukiwarki, system wyodrębniania informacji przyjmuje jako dane wejściowe ten nieograniczony tekst, a następnie podsumowuje go w odniesieniu do wcześniej określonej dziedziny lub interesującego tematu. Znajduje użyteczne informacje o domenie i koduje ten formularz informacyjny odpowiedni do raportowania użytkownikowi lub do zapełniania

ustrukturyzowanej bazy danych. W przeciwieństwie do dogłębnego systemu rozumienia języka naturalnego, systemy ekstrakcji informacji przeglądają tekst, aby znaleźć odpowiednie sekcje, a następnie koncentrują się tylko na przetwarzaniu tych sekcji. Proponujemy podejście oparte na szablonach, podobne do technik ekstrakcji proponowanych przez kilku badaczy. Przegląd naszego systemu ekstrakcji informacji przedstawiono na rysunkach 20 i 21. Załóżmy, że chcemy znaleźć i podsumować na stronie WWW informacje dotyczące stanowisk wydziałowych na poziomie uniwersytetu informatycznego. Rysunek 20 przedstawia docelowe ogłoszenie o pracę, a także przedstawia szablon informacji, które chcemy, aby nasze oprogramowanie wyodrębniło z tego i podobnych ogłoszeń o pracę

Przykładowe ogłoszenie o pracę w informatyce (fragment):

The Department of Computer Science of the University of New Mexico. . . is conducting

a search to fill two tenure-track positions. We are interested in hiring people with research

interests in:

Software, including analysis, design, and development tools. . .

Systems, including architecture, compilers, networks. . .

...

Candidates must have completed a doctorate in. . .

The department has internationally recognized research programs in adaptive computation,

artificial intelligence, . . . and enjoys strong research collaborations with the Santa Fe

Institute and several national laboratories. ...

Przykładowy częściowo wypełniony szablon:

Employer: Department of Computer Science, University of New Mexico

Location City: Albuquerque

Location State: NM 87131

Job Description: Tenure track faculty

Job Qualifications: PhD in . . .

Skills Required: software, systems, . . .

Platform Experience: . . .

About the Employer: (text attached)

1. Tekst: Katedra Informatyki Uniwersytetu

of New Mexico prowadzi poszukiwania, aby zapełnić dwa

pozycje na torze. Jesteśmy zainteresowani wypełnieniem

na stanowiskach adiunkta. . .

2.Tokenizacja i tagowanie: / det Department / rzeczownik / prep ...

3. Analiza zdań: Dział / podmiot przeprowadza / wyszukiwanie czasownika / obj

Komputer / rzeczownik ...

4. Wydobycie: Pracodawca: Katedra Informatyki

Opis stanowiska: Pozycja na torze zatrudnienia ...

5. Łączenie: pozycja toru pracy = wydział

Nowy Meksyk = NM ...

6. Generowanie szablonu: jak na rysunku 19

We wczesnych próbach ekstrakcji informacji systemy języka naturalnego różniły się pod względem podejść. Z jednej strony systemy przetwarzały tekst przy użyciu tradycyjnych narzędzi: pełne rozbicie składniowe każdego zdania, któremu towarzyszyła szczegółowa analiza semantyczna. Często następowało przetwarzanie na poziomie dyskursu. Z drugiej strony systemy wykorzystywały techniki dopasowywania słów kluczowych z niewielką lub żadną wiedzą lub analizą na poziomie językowym. Jednak wraz z budową i oceną większej liczby systemów ograniczenia tych ekstremalnych podejść do ekstrakcji informacji stały się oczywiste. Bardziej nowoczesne podejście do ekstrakcji informacji, zaadaptowane z Cardie, zostało odzwierciedlone na rysunkach 20 i 21. Chociaż szczegóły tej architektury mogą się różnić w różnych aplikacjach, rysunki wskazują główne funkcje wykonywane podczas ekstrakcji. Najpierw każde zdanie „interesującej" witryny internetowej jest tokenizowane i tagowane. W tym celu można użyć taggera stochastycznego przedstawionego w sekcji 15.4. Etap analizy zdań, który następuje i przeprowadza analizę, tworzy następnie grupy rzeczowników, czasowniki, frazy przyimkowe i inne konstrukcje gramatyczne. Następnie faza wyodrębniania znajduje i określa semantyczne encje istotne dla tematu ekstrakcji. W naszym przykładzie będzie to identyfikacja nazwy pracodawcy, lokalizacji stanowiska wydziału, wymagań dotyczących stanowiska (wykształcenie, umiejętności obsługi komputera itp.), Czasu rozpoczęcia spotkania itp. Faza ekstrakcji jest pierwszym całkowicie specyficznym dla domeny etapem proces. Podczas ekstrakcji system identyfikuje określone relacje między odpowiednimi składnikami tekstu. W naszym przykładzie Wydział Informatyki jest postrzegany jako pracodawca, a lokalizacja jest postrzegana jako Uniwersytet Nowego Meksyku. Faza łączenia musi obejmować takie kwestie, jak odniesienie do synonimów i rozwiązanie anafory. Przykładami synonimów są teuretrack stanowisko i pozycja na wydziale, a także Nowy Meksyk i NM. Rezolucja Anafora łączy Wydział Informatyki w pierwszym zdaniu z nami, to jest przedmiotem drugiego zdania. Wnioski na poziomie dyskursu dokonane podczas scalania wspomagają fazę generowania szablonu, która określa liczbę odrębnych relacji w tekście, odwzorowuje te wyodrębnione fragmenty informacji na każde pole szablonu i tworzy ostateczny szablon wyjściowy. Pomimo ostatnich postępów, obecne systemy ekstrakcji informacji nadal mają problemy. Po pierwsze, dokładność i niezawodność tych systemów można znacznie poprawić, ponieważ błędy w ekstrakcji wydają się wynikać z raczej płytkiego zrozumienia znaczenia (semantyki) tekstu wejściowego. Po drugie, zbudowanie systemu ekstrakcji informacji w nowej domenie może być trudne i czasochłonne. Oba te problemy są związane z dziedzinowym charakterem zadania ekstrakcji. Proces wyodrębniania informacji poprawia się, jeśli źródła wiedzy językowej są dostrojone do określonej domeny, ale ręczne modyfikowanie wiedzy językowej specyficznej dla domeny jest zarówno trudne, jak i podatne na błędy.

Niemniej jednak istnieje obecnie szereg interesujących aplikacji. Glasgow i inni zbudowali system wspomagający ubezpieczycieli w analizie wniosków dotyczących ubezpieczeń na życie. Soderland i inni) zbudowali system do wyodrębniania objawów, ustaleń fizycznych, wyników testów i diagnoz z

dokumentacji medycznej pacjenta w celu przetworzenia ubezpieczenia. Istnieją programy do analizy artykułów prasowych w celu znalezienia i podsumowania wspólnych przedsięwzięć biznesowych, systemy, które automatycznie klasyfikują dokumenty prawne oraz programy, które wyodrębniają szczegółowe informacje z komputerowych ofert pracy.

15.5.4 Wykorzystanie algorytmów uczenia się do uogólniania wyodrębnionych informacji

Inna aplikacja łączy kilka pomysłów przedstawionych w tym rozdziale, a także algorytmy z uczenia maszynowego (sekcja 10.3). Cardie i Mooney oraz Nahm i Mooney zasugerowali, że informacje wyodrębnione z tekstu można uogólniać algorytmy uczenia maszynowego, a wynik ponownie wykorzystany w zadaniu ekstrakcji informacji. Podejście jest proste. Gotowe lub nawet częściowo wypełnione szablony podsumowania tekstu, jak pokazano na przykład na rysunku 20, są zbierane z odpowiednich witryn internetowych. Wynikowe informacje z szablonu są następnie przechowywane w relacyjnej bazie danych, w której algorytmy uczące się, takie jak ID3 lub C4.5, są używane do wyodrębniania drzew decyzyjnych, które, mogą odzwierciedlać relacje reguł ukryte w zbiorach danych. (Technikę tę nazwaliśmy eksploracją danych). Mooney i jego koledzy proponują, aby te nowo odkryte relacje można było następnie wykorzystać do udoskonalenia oryginalnych szablonów i struktur wiedzy wykorzystywanych do ekstrakcji informacji. Przykłady tego typu informacji, które można znaleźć na podstawie analizy podań o pracę w informatyce w sekcji 15.5.3, mogą obejmować: jeśli stanowisko jest wydziałem informatyki, to doświadczenie na określonej platformie komputerowej nie jest wymagane; jeśli uniwersytety zatrudniają wykładowców, wymagane jest doświadczenie badawcze, itp. Więcej szczegółów na temat tego podejścia do generalizacji informacji można znaleźć w Nahm i Mooney. Istnieje wiele ekscytujących problemów związanych z lingwistyką komputerową, których rozwiązanie nie jest naszym celem. Kończymy trzema otwartymi kwestiami. Po pierwsze, jaka jest kluczowa jednostka dla rozumienia języka mówionego? Podejmowano różne próby odpowiedzi na to pytanie, począwszy od pełnej analizy zdania w języku angielskim, przez próby rozpoznawania pojedynczych słów, po skupienie się na poszczególnych telefonach. Czy to możliwe, że skupienie się na sylabie jest kluczem do zrozumienia ludzkiego języka? Ile słowa musimy usłyszeć, zanim zostanie rozpoznane? Jaka jest optymalna ziarnistość do analizy mowy dźwięcznej (Greenberg 1999)? Po drugie, czy dostęp do pojęć w bazie wiedzy poprzez rozpoznawanie dźwięku może prowadzić do dalszych uwarunkowań późniejszej interpretacji języka? Załóżmy, że sylaba okaże się użyteczną jednostką do analizy języka. Załóżmy, że sylaby bigramów są przydatne do sugerowania słów mówcy. Czy te słowa można odwzorować w pojęciach bazy wiedzy, a powiązaną wiedzę można wykorzystać do ulepszenia interpretacji kolejnych sylab? Osoba dzwoniąca do organizatora podróży korzystającego z komputera może rozpoznać sylaby sugerujące nazwę miasta, a komputer może wtedy przewidzieć pytania związane z odwiedzaniem tego miasta. Wreszcie długoterminowym celem przetwarzania języka naturalnego jest stworzenie sieci semantycznej; jak to się stanie? Czy to się stanie? Czy istnieje metodologia, według której można powiedzieć, że komputery „coś rozumieją”? Czy też zawsze będą istnieć oparte na wzorcach „sztuczki” oparte na wyszukiwaniu, które są wystarczająco dobre, aby „przekonać” ludzi, że komputer rozumie (ponownie test Turinga)? Czy to „wystarczająco dobre” podejście nie jest tym, co my, ludzie, i tak robimy dla siebie nawzajem? Ostatni rozdział kontynuuje tę i powiązane dyskusje