Adam Przepiórkowski

# Korpus IPI PAN
## wersja wstępna

INSTYTUT PODSTAW INFORMATYKI PAN
WARSZAWA 2004

Instytut Podstaw Informatyki
Polska Akademia Nauk
ul. Ordona 21
01-237 Warszawa

Copyright © 2004 by Adam Przepiórkowski

ISBN 83-910948-8-X

# Spis treści

# 1

# Wstęp



## 1.1. Korpus IPI PAN

Niniejsza publikacja jest wynikiem projektu KBN numer 7 T11C 043 20 realizowanego w Instytucie Podstaw Informatyki PAN (IPI PAN) od kwietnia 2001 do marca 2004 oraz badań statutowych prowadzonych w IPI PAN. Jej celem jest prezentacja stworzonego w ramach tego projektu i załączonego na płycie CD-ROM Korpusu IPI PAN oraz narzędzi do jego przeszukiwania. Opis ten powinien pozwolić na efektywne korzystanie z korpusu i z przeszukiwarki.

Korpus IPI PAN jest pierwszym publicznie dostępnym korpusem języka polskiego, w pełnym znaczeniu słowa *korpus*: jest to duży, liczący ponad 100 mln. pozycji zbiór tekstów polskich, częściowo reprezentatywny, znakowany lingwistycznie (morfosyntaktycznie), stworzony zgodnie ze współczesnymi standardami i praktykami tworzenia dużych korpusów tekstów.

Korpusy takie istnieją dla wielu języków, nie tylko europejskich, i są szeroko wykorzystywane w przetwarzaniu języka naturalnego, w leksykografii i w innych działach lingwistyki. W wielu krajach stworzenie korpusu uważa się za swoisty obowiązek wobec języka ojczystego. Z tego właśnie względu tego rodzaju wielkie korpusy zwane są często „narodowymi", by przytoczyć choćby dwa znane przykłady Brytyjskiego Korpusu Narodowego (`http://www.hcu.ox.ac.uk/BNC/`), czy Czeskiego Korpusu Narodowego (`http://ucnk.ff.cuni.cz/`). Jedynym publicznie dostępnym lingwistycznie anotowanym korpusem języka polskiego był dotychczas

stworzony w latach sześćdziesiątych korpus „Słownika frekwencyjnego polszczyzny współczesnej" (Kurcz i in., 1974, 1990) liczący pół miliona słów.

W Polsce dotychczasowe prace korpusowe były bardzo rozproszone, zaś ich wyniki są dostępne publicznie tylko w niewielkim stopniu. Prace takie były i są prowadzone m.in. w Warszawie (wydawnictwo PWN), Krakowie (Instytut Języka Polskiego PAN; IJP PAN), Łodzi (Uniwersytet Łódzki) i we Wrocławiu (Uniwersytet Wrocławski). Publicznie dostępne efekty tych prac to nie znakowane lingwistycznie próbki korpusu PWN: próbka dostępna na stronie `http://korpus.pwn.pl/`, licząca niecałe 2 mln. słów, oraz czterokrotnie większa próbka dołączana na płycie CD-ROM do wersji luksusowej „Uniwersalnego słownika języka polskiego", a także sprzedawana przez Uniwersytet Łódzki próbka licząca 10 mln. słów. Celem niniejszego projektu było wypełnienie tej luki i w konsekwencji przetarcie drogi dla zastosowań metod statystycznych w przetwarzaniu języka polskiego.

Binarna postać korpusu zamieszczona na dołączonej płycie CD-ROM, dostępna za pomocą przeszukiwarki Poliqarp, również znajdującej się na płycie CD-ROM, przeznaczona jest przede wszystkim dla lingwistów oraz innych świadomych użytkowników języka polskiego. Do zastosowań informatycznych zapewne znacznie bardziej przydatna będzie wersja tekstowa korpusu, udostępniana bezpośrednio przez IPI PAN (zapytania dotyczące formy i warunków udostępnienia wersji źródłowej korpusu należy kierować na adres `info@korpus.pl` lub `adamp@ipipan.waw.pl`).

Niniejszą wersję korpusu i narzędzi nazywamy *wersją wstępną*, gdyż jesteśmy w pełni świadomi niedoskonałości zarówno obecnej wersji korpusu, jak i załączonych na płycie CD-ROM narzędzi. Przy tak ogromnej ilości tekstu i w ramach ograniczonych zasobów, jakimi niniejszy projekt dysponował, nie była możliwa pełna weryfikacja efektów konwersji tekstów na format XML, wyników znakowania morfosyntaktycznego oraz informacji o pochodzeniu utworów półautomatycznie przypisanych poszczególnym tekstom. Szczególnie te ostatnie informacje — tzw. metadane — należy uznać za bardzo niepełne. Korpus IPI PAN w obecnej postaci jest, w pełnym znaczeniu tego pojęcia, „oportunistyczny": korpus ten zawiera najróżniejsze teksty w dosyć przypadkowych proporcjach i nie rości sobie pretensji do miana korpusu reprezentatywnego. Wydzielenie takiego reprezentatywnego podkorpusu oraz dodanie pełnej charakterystyki

tematycznej zawartości korpusu przewidziane jest w następnym etapie rozwoju korpusu.

Również załączone narzędzie do przeszukiwania korpusu — program Poliqarp — choć posiada wiele cech pozytywnie wyróżniających je spośród podobnych dostępnych narzędzi (np. CQP, GCQP, Bonito), nie jest jeszcze narzędziem skończonym: efektywność przeszukiwania dużych korpusów (powyżej 50 mln. słów) pozostawia wiele do życzenia, brak też w obecnej wersji funkcji statystycznych, możliwości graficznego formułowania zapytań, czy też większego wpływu na format wyświetlanych wyników. Mamy nadzieję, że niedostatki te zostaną zniwelowane w przyszłych wersjach oprogramowania.

## 1.2. Podziękowania

Korpus IPI PAN i narzędzia do jego stworzenia i przeszukiwania powstały przede wszystkim w ramach grantu Komitetu Badań Naukowych, a także w ramach prac statutowych prowadzonych w Instytucie Podstaw Informatyki PAN — truizmem jest stwierdzenie, że bez wsparcia obu tych instytucji realizacja niniejszego projektu nie byłaby możliwa.

Do powodzenia tego projektu przyczyniły się życzliwość i często bezinteresowne poparcie wielu osób. Prof. Zygmunt Saloni (Uniwersytet Warmińsko-Mazurski w Olsztynie) i Marcin Woliński (IPI PAN) udostępnili na potrzeby projektu analizator morfologiczny Morfeusz. Prof. Januszowi S. Bieniowi (Uniwersytet Warszawski; UW) zawdzięczamy dostęp do znacznie uporządkowanej wersji korpusu „Słownika frekwencyjnego polszczyzny współczesnej", wspomnianego już powyżej. Dr Jan Hajič (Uniwersytet Karola w Pradze) zgodził się na wykorzystanie w niniejszym projekcie narzędzia DAUJC do ręcznej dezambiguacji znaczników morfosyntaktycznych, opracowanego przez Jiřiego Hanę. Prof. Włodzimierz Gruszczyński (UW) udostępnił wzorce odmiany rzeczowników polskich, które przyczyniły się do poprawy automatycznej anotacji morfosyntaktycznej korpusu. Prof. František Čermak (Instytut Czeskiego Korpusu Narodowego, Uniwersytet Karola w Pradze) zaprosił wykonawców niniejszego projektu do złożenia wizyty w siedzibie Czeskiego Korpusu Narodowego, dzięki czemu możliwe było skorzystanie z doświadczeń tego projektu.

Jednym z najtrudniejszych zadań w projekcie było pozyskanie odpowiednio dużej liczby tekstów i praw autorskich, pozwalających na wykorzystanie tych tekstów w publicznie dostępnym korpusie. Zadaniem tym kierował przede wszystkim dr Rafał L. Górski (Instytut Języka Polskiego PAN; IJP PAN), zaś niewątpliwie dużą rolę w pozyskaniu przychylności wydawców i autorów odegrały rekomendacje prof. Jerzego Bralczyka (UW), prof. Stanisława Gajdy (Komitet Językoznawstwa PAN) oraz prof. Ireneusza Bobrowskiego (IJP PAN). Lista osób, które pomogły nam dotrzeć do wydawców i autorów jest zbyt długa, by ją tutaj zamieszczać (ale por. strony WWW projektu).

Jako kierownik projektu chciałbym serdecznie podziękować wykonawcom projektu za ich pracę i zaangażowanie. Łukasz Dębowski (IPI PAN) nie tylko stworzył w ramach niniejszego projektu statystyczny dezambiguator interpretacji morfosyntaktycznych, ale także odgrywał znaczącą rolę w administrowaniu projektem. System znaczników morfosyntaktycznych zastosowany w Korpusie IPI PAN jest wynikiem wielu dyskusji, w których udział brali Łukasz Dębowski, Marcin Woliński, a także Elżbieta Hajnicz (IPI PAN) i Zygmunt Saloni. Marcin Woliński stworzył także narzędzie pozwalające na szybkie poszerzenie zakresu empirycznego analizatora morfologicznego używanego w projekcie. W pracach nad ręcznym ujednoznacznianiem korpusu treningowego dla automatycznego dezambiguatora uczestniczyły i cennymi uwagami służyły: Monika Czerepowicka (Uniwersytet Warmińsko-Mazurski), Dorota Lewandowska (UW), Hanna Maliszewska (UW), Marta Nazarczuk-Błońska, Marta Piasecka (UW), Ewa Wolska i Beata Wójtowicz (UW), zaś o jakość tych prac dbała Elżbieta Hajnicz. Elżbiecie Hajnicz, Danucie Przepiórkowskiej, Łukaszowi Dębowskiemu i Marcinowi Wolińskiemu dziękuję także za komentarze na temat pierwszej wersji niniejszej publikacji, zaś Marcinowi Wolińskiemu — za projekt graficzny tej publikacji i za pomoc typograficzną.

Bardzo pracochłonnym zadaniem było stworzenie programu do indeksowania i przeszukiwania korpusu. Głównymi autorami programu Poliqarp, którego funkcjonalność pod wieloma względami wykracza poza przyjętą normę, są Zygmunt Krynicki (Polsko-Japońska Wyższa Szkoła Technik Komputerowych) i Daniel Janus (UW).

Trudnym i niewdzięcznym zadaniem okazało się konwertowanie tekstów z przeróżnych formatów, w jakich były one przekazywane przez wydawnictwa i autorów, do jednolitego formatu XML, którego początkowa

wersja została opracowana przez dr. Piotra Bańskiego (UW). W tworzeniu programów do konwersji tekstów i w samej konwersji brali udział: Piotr Bański, Artur Gniadzik (UW), Paweł Savov (UW), Katarzyna Sokołowska (UW), Radosław Moszczyński (UW), Jakub Sikora (UW) i Jakub Jurkiewicz (UW).

Mam nadzieję, że także w przyszłości idea budowy i dalszego rozwoju lingwistycznie znakowanego korpusu języka polskiego spotka się z życzliwością i wsparciem wielu osób i instytucji.

# 2

# Wstępne przetwarzanie tekstów



Utwory wchodzące w skład Korpusu IPI PAN przebywają długą drogę od postaci, w jakiej zostały pozyskane od autora lub wydawnictwa, do postaci obsługiwanej przez program do przeszukiwania korpusu opisany w rozdziale 4. Niniejszy rozdział zawiera krótkie omówienie poszczególnych etapów konwersji tekstów w Korpusie IPI PAN.

## 2.1. Od tekstów wejściowych do formatu XML

Wszystkie utwory wchodzące w skład korpusu są konwertowane z formatu wejściowego, w jakim zostały przekazane, np. HTML, Word, RTF, PDF, WordPerfect, PageMager, LaTeX itp., do wspólnego formatu tekstowego. W niniejszym projekcie tym wspólnym formatem jest nieznacznie zmodyfikowany standard reprezentacji korpusów, *XML Corpus Encoding Standard* (XCES; Ide i in. 2000). XCES jest XML-ową wersją wcześniejszego SGML-owego standardu *Corpus Encoding Standard* (CES; Ide i in. 1996) opartego na schemacie *Text Encoding Initiative* (TEI). Także różne strony kodowe sprowadzane są do jednego uniwersalnego systemu reprezentacji znaków, a mianowicie do UTF-8.

Każdy utwór wchodzący w skład korpusu IPI PAN jest konwertowany do 3 plików XML[1] umieszczanych w osobnym katalogu:

[1] Wcześniejsze adaptacje standardu XCES do potrzeb niniejszego projektu opisane są w pracach Bański 2001, 2003.

— `header.xml`: plik zawierający metadane, tj. dane o autorze, wydawcy, tytule, dacie publikacji itp., a także informacje o procesie konwersji na XML i o dalszych zmianach w wynikach tej konwersji,
— `text.xml`: plik zawierający tekst, informacje strukturalne (rozdziały, akapity itp.), oraz pewne informacje o formatowaniu (krój czcionki itp.),
— `morph.xml`: plik zawierający tekst anotowany morfosyntaktycznie, podzielony na zdania, akapity oraz pewne wyższe jednostki tekstu.

W ramach niniejszego projektu stworzony został zestaw narzędzi do konwertowania tekstów z formatów Word, HTML i PDF do wstępnych wersji plików `header.xml` i `text.xml` — teksty w innych formatach są najpierw konwertowane do któregoś z tych trzech formatów za pomocą ogólnie dostępnych narzędzi lub są one przetwarzane indywidualnie. Oczywiście w pełni zautomatyzowane wydobycie informacji o pochodzeniu tekstu i o jego strukturze z tekstów w formacie zawierającym przede wszystkim informacje typograficzne, np. z tekstów w formacie Word lub PDF, jest niemożliwe, zachodzi więc konieczność weryfikacji i poprawy wyników tej automatycznej konwersji przez człowieka, w tym usunięcia dłuższych wtrętów obcojęzycznych i innych fragmentów dzieła nie stanowiących ciągłego tekstu w języku polskim. Poza takim usuwaniem fragmentów utworów, teksty nie są normalizowane: liczby pisane cyframi, w tym daty, nie są tłumaczone na formy wyrazowe, skróty nie są rozwijane, błędy w tekście nie są poprawiane.

Z powodu dużej liczby tekstów w korpusie konwertowaniem tekstów zajmowało się ponad pięć różnych osób o różnym stopniu wiedzy informatycznej i znajomości technik XML-owych. Dlatego też, mimo iż stworzona została kilkunastostronicowa instrukcja (Przepiórkowski, 2004) omawiająca docelowy format plików `header.xml`, pewne różnice w wynikach konwersji realizowanej przez różne osoby są nieuniknione.

Omawiany etap przetwarzania tekstów wejściowych jest najbardziej czasochłonnym i pracochłonnym etapem budowy korpusu, zaś jego rezultatem jest powstanie pliku `header.xml` oraz wstępnej wersji pliku `text.xml`. Pliki te są walidującymi się plikami XML zgodnymi z wersją standardu XCES (`xcesDoc.dtd` i `xheader.elt`) przyjętą w niniejszym projekcie.

## 2.2. Dalsze przetwarzanie formatu XML

Dalsze etapy przetwarzania tekstu są w pełni zautomatyzowane. Na podstawie wstępnej wersji pliku `text.xml` utworzony jest plik `morph.xml`, który nie zawiera szczegółowych informacji o strukturze logicznej utworu, ale tekst w nim zawarty jest podzielony na zdania i oznakowany morfosyntaktycznie (por. rozdział 3).

Podział na zdania odbywa się według prostego algorytmu, który dla każdego znaku interpunkcyjnego mogącego oznaczać koniec zdania, np. dla kropki, bada otoczenie tego znaku, w tym sprawdza, czy znak ten jest częścią skrótu, a jeżeli tak, to czy ten skrót może kończyć zdanie, czy następny segment zaczyna się wielką literą itp. Oczywiście nie wszystkie decyzje podjęte przez ten algorytm są prawidłowe. Jak pokazują poniższe zdania, niekiedy nawet pełne zrozumienie tekstu nie wystarcza, by w sposób jednoznaczny stwierdzić, czy dany znak interpunkcyjny sygnalizuje koniec zdania.

(2.1) Kiedy to się działo? W latach 40. Stany Zjednoczone włączyły się do wojny.
(2.2) Skorzystać z Yahoo! Marek i jego koledzy nie chcieli.

W wyniku znakowania morfosyntaktycznego tekst zostaje podzielony na segmenty, w przybliżeniu — słowa, którym przypisane zostają znaczniki określające ich formy podstawowe, klasy gramatyczne (tzw. części mowy) i wartości kategorii gramatycznych odpowiednich dla tych klas (na przykład wartość kategorii przypadka dla rzeczowników).

Samo znakowanie morfosyntaktyczne odbywa się w dwóch etapach. W pierwszym etapie, analizator morfologiczny dzieli tekst na segmenty i przypisuje im interpretacje, lecz nie określa, które z tych interpretacji są właściwe w danym kontekście. W niniejszym projekcie wykorzystany został analizator morfologiczny *Morfeusz* opracowany przez Marcina Wolińskiego na podstawie danych lingwistycznych dostarczonych przez Zygmunta Saloniego, przede wszystkim na podstawie bazy danych czasowników polskich (por. Saloni 2001) oraz słownika *a tergo* opublikowanego jako Tokarski 1993. Analizator ten jest nadal rozwijany, zaś wiele błędów w anotacji obecnej wersji Korpusu IPI PAN wynika z niedoskonałości obecnej wersji tego analizatora.

Na drugim etapie znakowania morfosyntaktycznego, spośród interpretacji zaproponowanych na pierwszym etapie wybierane są te, które wydają się właściwe w danym kontekście. Do takiej dezambiguacji interpretacji morfosyntaktycznych służy program opracowany przez Łukasza Dębowskiego i oparty na metodach statystycznych (por. Dębowski 2001, 2003, 2004). Poniższy przykładowy fragment pliku `morph.xml` odpowiada ciągowi *Porządek dzienny*[2].

```
<tok>
 <orth>Porządek</orth>
 <lex><base>porządek</base><ctag>subst:sg:acc:m3</ctag></lex>
 <lex disamb="1">
  <base>porządek</base><ctag>subst:sg:nom:m3</ctag>
 </lex>
</tok>
<tok>
 <orth>dzienny</orth>
 <lex><base>dzienny</base><ctag>adj:sg:acc:m3:pos</ctag></lex>
 <lex><base>dzienny</base><ctag>adj:sg:nom:m1:pos</ctag></lex>
 <lex><base>dzienny</base><ctag>adj:sg:nom:m2:pos</ctag></lex>
 <lex disamb="1">
  <base>dzienny</base><ctag>adj:sg:nom:m3:pos</ctag>
 </lex>
</tok>
```

Przykład ten pokazuje, że w pliku `morph.xml` zachowane są nie tylko interpretacje wybrane przez program ujednoznaczniający (por. `disamb="1"` powyżej), ale także inne interpretacje zaproponowane przez analizator morfologiczny.

Interpretacje obu form występujących w powyższym przykładzie zostały całkowicie ujednoznacznione (do liczby pojedynczej, mianownika, rodzaju męskiego rzeczowego), możliwa jest jednak sytuacja, gdy pełne ujednoznacznienie interpretacji musiałoby mieć charakter arbitralny, jak w przykładzie (2.3), gdzie nie jest możliwe określenie, czy forma *go* jest formą biernikową (jak w (2.4a)), czy też formą dopełniaczową (jak w (2.4b)), i jak w przykładzie (2.5), gdzie forma *pijaną* może mieć interpretację biernikową (przez analogię do (2.6a)) lub narzędnikową (por. (2.6b)).

(2.3) Pożądała go.

[2] Znaczenie napisów typu `subst:sg:acc:m3` zostało szczegółowo omówione w rozdziale 3.

(2.4) a. Pożądał ją.
b. Pożądał jej.
(2.5) Pamiętam ją pijaną.
(2.6) a. Pamiętam go pijanego.
b. Pamiętam go pijanym.

W takich wypadkach wszystkie poprawne w danym kontekście interpretacje (elementy `<lex>`) powinny zostać oznaczone jako ujednoznacznione (`disamb="1"`).

Rezultatem tego etapu przetwarzania jest utworzenie ostatecznej wersji pliku `text.xml`, w którym wszystkie elementy XML posiadają odpowiednie indeksy, oraz pliku `morph.xml`, zawierającego anotację morfosyntaktyczną i powiązanego z plikiem `text.xml` za pomocą odnośników do tych indeksów. Oba te pliki są poprawnymi plikami XML, zgodnymi z nieznacznie zmodyfikowanym standardem XCES (`xheader.elt` oraz, odpowiednio, `xcesDoc.dtd` i `xcesAna.dtd`).

## 2.3. Od formatu XML do postaci binarnej

Bezpośrednie przeszukiwanie plików XML utworzonych w poprzednich etapach byłoby niezwykle nieefektywne, dlatego też wszyskie pliki `header.xml` i `morph.xml` wchodzące w skład korpusu kompilowane są do postaci binarnej, składającej się z różnych indeksów umożliwiających programowi opisanemu w rozdziale 4 szybki dostęp do wyszukiwanych fragmentów tekstu. W procesie kompilacji ignorowana jest część informacji zawartych w nagłówkach, tj. w plikach `header.xml`, uwzględniana jest jednak informacja o tytule utworu, jego autorze, dacie publikacji (jeżeli taka informacja jest dostępna) itp., a także pełna informacja o znakowaniu morfosyntaktycznym, w tym informacja o wieloznacznościach.

Na płycie CD-ROM stanowiącej część niniejszej publikacji dostępna jest wyłącznie taka właśnie binarna postać korpusu.

# 3

# System znaczników morfosyntaktycznych



Korpus IPI PAN jest korpusem anotowanym morfosyntaktycznie. Oznacza to, że poszczególnym ciągom znaków (w przybliżeniu *słowom*) w korpusie przypisane zostały tzw. *znaczniki* interpretujące dane ciągi jako wykładniki tekstowe pewnych form wyrazowych. Takie interpretowalne ciągi znaków nazywać będziemy *segmentami*. Zastosowane w niniejszym korpusie zasady podziału tekstu na interpretowalne segmenty opisane zostały w punkcie 3.1.

Jeden (lub w określonych wypadkach więcej) spośród tych znaczników przypisanych danemu segmentowi wybrany jest przez automatyczny dezambiguator lub przez osobę ujednoznaczniającą wyniki działania analizatora morfologicznego jako ten odpowiedni w danym kontekście. Na przykład w wypadku segmentu *nie*, niezależnie od kontekstu, w jakim ten segment wystąpił, analizator morfologiczny przypisuje mu znacznik interpretujący *nie* jako partykułę negacji NIE oraz kilka różniących się licz-

bą i rodzajem znaczników interpretujących go jako poprzyimkową formę zaimka ON. Jeżeli segment ten jest częścią napisu *Janek nie przyszedł*, znacznik interpretujący *nie* jako partykułę negacji NIE zostanie wybrany jako właściwy w danym kontekście. Jeżeli natomiast segment ten jest fragmentem ciągu *Twoje koleżanki przyjdą, poczekaj na nie*, jako właściwa zostanie wybrana interpretacja tego segmentu jako mnogiej, żeńskiej, biernikowej, poprzyimkowej formy zaimka ON.

W niniejszym korpusie dostępne są oba rodzaje informacji: zarówno wszystkie interpretacje przypisane danemu segmentowi przez analizator morfologiczny, jak i te, które zostały wybrane jako właściwe w danym kontekście. Struktura znaczników morfosyntaktycznych omówiona została w punkcie 3.2, zaś przyjęty tutaj zestaw kategorii i klas gramatycznych szczegółowo opisują punkty 3.3 i 3.4.

Wiele rozwiązań opisanych w niniejszym rozdziale zostało zaczerpniętych z prac Zygmunta Saloniego i jego współpracowników (Saloni, 1976, 1977, 1981, 1988; Gruszczyński i Saloni, 1978; Bień i Saloni, 1982; Bień, 1991) lub jest tymi pracami inspirowane. Ostateczny zestaw znaczników morfosyntaktycznych oraz reguł segmentacji tekstów został opracowany przez Marcina Wolińskiego i autora niniejszej publikacji w wyniku wielu dyskusji, w których udział brali także Łukasz Dębowski, Elżbieta Hajnicz oraz — w końcowej fazie — Zygmunt Saloni. Poprzednie wersje tych rozwiązań zostały opisane i szczegółowo uzasadnione w pracach: Woliński i Przepiórkowski 2001, Przepiórkowski i Woliński 2003a,b, Woliński 2003 i Przepiórkowski 2003b, a także w instrukcji dla osób anotujących teksty w korpusie (Przepiórkowski i in., 2004).

## 3.1. Podstawowe zasady segmentacji tekstu

Segmentacja tekstu polega na podziale tekstu na ciągi znaków podlegające anotacji, czyli na *segmenty*. Zasady segmentacji tekstu są nierozerwalnie związane z systemem znaczników morfosyntaktycznych: inny tagset będzie użyty w wypadku segmentacji napisu *bał się* na dwa segmenty, *bał* i *się*, a inny w wypadku, gdy ciągowi *bał się* powinien zostać przypisany jeden znacznik; różne także będą tagsety w zależności od tego, czy ciąg *przyszlibyśmy* zostanie podzielony na segmenty *przyszli*, *by* i *śmy*, czy też zostanie on potraktowany jako jeden segment.

W niniejszym korpusie przyjęto jako nadrzędną zasadę, że segmenty:

— są ciągłe, tzn. składają się z ciągu bezpośrednio następujących po sobie znaków, oraz
— są rozłączne, tzn. nie jest dopuszczalna sytuacja, gdy pewien ciąg znaków jednocześnie należy do dwóch lub większej liczby segmentów.

Ta prosta i intuicyjnie oczywista zasada ma pewne być może nieintuicyjne konsekwencje. Jak pokazuje poniższy przykład, jedną z tych konsekwencji jest konieczność traktowania tzw. *czasowników zwrotnych* jako par segmentów składających się z właściwej formy czasownikowej oraz z segmentu *się*.

(3.1) Bo ja się naprawdę boję głośno roześmiać.

W przykładzie tym, ilustrującym tzw. haplologię zaimka zwrotnego *się* (Kupść, 1999), jeden ciąg *się* wydaje się być jednocześnie częścią czasownika zwrotnego BAĆ SIĘ i czasownika zwrotnego ROZEŚMIAĆ SIĘ. Wymaganie rozłączności segmentów nie pozwala jednak traktować *się* jako części dwóch różnych segmentów, *boję się* i *roześmiać się*, zaś wymaganie ciągłości segmentów nie pozwala na wyróżnienie w (3.1) ani segmentu *boję się*, ani segmentu *roześmiać się*, a zatem w tego typu wypadkach należy uznać *się* za odrębny segment. Skoro ciąg *się* jest odrębnym segmentem w przykładach typu (3.1), naturalne (i poparte *brzytwą Ockhama*) jest traktowanie *się* jako osobnego segmentu także i w innych wystąpieniach form czasowników zwrotnych.

Na podstawie podobnego rozumowania zastosowanego do poniższych przykładów, podzielone na mniejsze segmenty muszą być także tzw. *formy analityczne* czasowników, ciągi typu *po polsku* itp., gdyż w przeciwnym wypadku ciągi *będę*, *niech*, *po* itp. musiałyby należeć jednocześnie do dwóch segmentów: *będę szedł* i *będę śpiewał*, *niech przyjdzie* i *niech zaśpiewa*, *po polsku* i *po angielsku*.

(3.2) a. Będę długo szedł i śpiewał.
b. Niech no tylko przyjdzie i zaśpiewa!
(3.3) Mówię po polsku i angielsku.

Uogólniając tego typu przykłady, w niniejszym korpusie przyjęto zasadę, że segmenty nigdy nie są dłuższe niż *słowa* rozumiane jako maksymal-

ne ciągi znaków nie będących separatorami słów, gdzie *separatorami słów* są odstępy oraz znaki interpunkcyjne z wyłączeniem dywizu, kropki będącej częścią skrótu oraz apostrofu w formach takich jak *Chomsky'ego* i *(de) l'Hospitala*. Znaki interpunkcyjne będące separatorami słów traktowane są jako osobne segmenty.

Zwykle tak rozumiane słowa są segmentami, choć istnieją sytuacje, gdy — znowu w myśl zasady ciągłości i rozłączności segmentów — wydzielić należy segmenty krótsze od słów.

(3.4) a. Dawno nie śpiewała*m* i nie tańczyła*m*.
b. Dawno*m* nie śpiewała i nie tańczyła.
(3.5) a. Kiedyś zatańczył*by*m i zaśpiewał*by*m tam.
b. Kiedyś *by*m tam zaśpiewał i zatańczył.

Przykład (3.4) pokazuje, że tzw. formy aglutynacyjne leksemu BYĆ, czyli ruchome końcówki *-(e)m*, *-(e)ś*, *-(e)śmy*, *-(e)ście*, powinny być traktowane jako osobne segmenty. Podobnie, przykład (3.5) uzasadnia odrębne traktowanie partykuły *by*. Wszystkie wyjątki od zasady traktowania słów jako pojedynczych segmentów wymienione są poniżej.

— Jako odrębne segmenty traktowane są formy aglutynacyjne leksemu BYĆ, a zatem następujące słowa reprezentują po dwa segmenty: |*łgał*|*eś*|, |*długo*|*śmy*|, |*tak*|*em*|.
— Za odrębne segmenty uznane są partykuły *by*, *-ż(e)* i *-li*, a zatem następujące słowa reprezentują po kilka segmentów: |*przyszedł*|*by*|, |*napisała*|*by*|*m*|, |*chodź*|*że*|, |*potrzebował*|*że*|*by*|*ś*|, |*znasz*|*li*|.
— Odrębnym segmentem jest poprzyimkowa nieakcentowana forma zaimka *-ń*: |*do*|*ń*|, |*ze*|*ń*|.
— Dzielone na segmenty są niektóre słowa zawierające łącznik, a mianowicie:
  — słowa typu |*polsko*|-|*niemiecki*|,
  — podwójne nazwiska, np. |*Kowalska*|-|*Nowakowska*|,
  nie są natomiast dzielone skrótowce zawierające łącznik sygnalizujący odmianę, np. *PRL-u*.
— Dzielone na segmenty są także występujące na końcu zdania formy kończące się kropką, np. skróty typu *itd.*, *itp.*, liczby pisane cyframi w znaczeniu porządkowym i inicjały, np. |*itp*|.|, |*George*| |*W*|.| itp. Dzielenie

form z kropką kończących zdanie jest uzasadnione podwójną rolą kropki w takiej pozycji: jest ona częścią formy i jednocześnie sygnalizuje koniec zdania (jest to tzw. haplologia kropki; por. p. 3.5.1). W wypadku, gdy takie formy nie występują na końcu zdania, są one uznawane za pojedyncze segmenty.

Z powyższych zasad wynika, że segmentacja tekstu w (3.6) wygląda tak, jak to przedstawiono w (3.7).

(3.6) Pojechalibyśmy z Janem M. Rokitą i Janem Nowakiem-Jeziorańskim na sesję polsko-amerykańską, gdyby nas zaprosił George W. Byłaby to nasza już 2. doń podróż od czasów PRL-u, a może i 3., czy nawet 4.

(3.7) *Pojechali*|*by*|*śmy*| *z*| *Janem*| *M.*| *Rokitą*| *i*| *Janem*| *Nowakiem*|*-*|*Jeziorańskim*| *na*| *sesję*| *polsko*|*-*|*amerykańską*|*,*| *gdyby*| *nas*| *zaprosił*| *George*| *W*|*.*| *Była*|*by*| *to*| *nasza*| *już*| *2.*| *do*|*ń*| *podróż*| *od*| *czasów*| *PRL-u*|*,*| *a*| *może*| *i*| *3.*|*,*| *czy*| *nawet*| *4*|*.*

## 3.2. Struktura znaczników morfosyntaktycznych

Znaczniki określają formę podstawową (tzw. lemat) i charakterystykę morfoskładniową danego segmentu (tzw. *znacznik morfosyntaktyczny*; czasami terminu *znacznik* będziemy używać w tym nieco węższym znaczeniu). W wypadku segmentu będącego znakiem interpunkcyjnym za formę podstawową przyjmujemy ten sam znak i przypisujemy mu znacznik interp. W dalszej części niniejszego rozdziału skupimy się na systemie anotacji form wyrazowych.

Każdy znacznik morfosyntaktyczny jest ciągiem wartości rozdzielonych dwukropkami, np.: subst:sg:nom:m1 dla segmentu *chłopiec*. Pierwsza wartość, np. subst, określa *klasę gramatyczą* (por. p. 3.4), następne zaś, np. sg, nom i m1 — wartości odpowiednich dla tej klasy *kategorii gramatycznych* (por. p. 3.3). Tagset przyjęty w niniejszym korpusie jest zatem *tagsetem pozycyjnym*, podobnie jak np. tagset Czeskiego Korpusu Narodowego, czy też rodzina tagsetów opracowanych w ramach projektu Multext-East (Erjavec, 2001).

## 3.3. Kategorie gramatyczne

Poniższa tabela przedstawia repertuar kategorii gramatycznych używanych w Korpusie IPI PAN.

| **Liczba**: (2 wartości) | | |
|---|---|---|
| pojedyncza | sg | *oko* |
| mnoga | pl | *oczy* |
| **Przypadek**: (7 wartości) | | |
| mianownik | nom | *woda* |
| dopełniacz | gen | *wody* |
| celownik | dat | *wodzie* |
| biernik | acc | *wodę* |
| narzędnik | inst | *wodą* |
| miejscownik | loc | *wodzie* |
| wołacz | voc | *wodo* |
| **Rodzaj**: (5 wartości) | | |
| męski osobowy | m1 | *papież, kto, wujostwo* |
| męski zwierzęcy | m2 | *baranek, walc, babsztyl* |
| męski rzeczowy | m3 | *stół* |
| żeński | f | *stuła* |
| nijaki | n | *dziecko, okno, co, skrzypce, spodnie* |
| **Osoba**: (3 wartości) | | |
| pierwsza | pri | *bredzę* |
| druga | sec | *bredzisz* |
| trzecia | ter | *bredzi* |
| **Stopień**: (3 wartości) | | |
| równy | pos | *cudny* |
| wyższy | comp | *cudniejszy* |
| najwyższy | sup | *najcudniejszy* |

| **Aspekt**: (2 wartości) | | |
|---|---|---|
| niedokonany | imperf | *iść* |
| dokonany | perf | *zajść* |
| **Zanegowanie**: (2 wartości) | | |
| niezanegowana | aff | *pisanie, czytanego* |
| zanegowana | neg | *niepisanie, nieczytanego* |
| **Akcentowość**: (2 wartości) | | |
| akcentowana | akc | *jego, niego, tobie* |
| nieakcentowana | nakc | *go, -ń, ci* |
| **Poprzyimkowość**: (2 wartości) | | |
| poprzyimkowa | praep | *niego, -ń* |
| niepoprzyimkowa | npraep | *jego, go* |
| **Akomodacyjność**: (2 wartości) | | |
| uzgadniająca | congr | *dwaj, pięcioma* |
| rządząca | rec | *dwóch, dwu, pięciorgiem* |
| **Aglutynacyjność**: (2 wartości) | | |
| nieaglutynacyjna | nagl | *niósł* |
| aglutynacyjna | agl | *niosł-* |
| **Wokaliczność**: (2 wartości) | | |
| wokaliczna | wok | *-em* |
| niewokaliczna | nwok | *-m* |

Kategorie **liczby**, **przypadka**, **osoby** i **stopnia** rozumiane są tutaj tradycyjnie i nie wymagają komentarza.

Kategoria **rodzaju** jest rozumiana w sensie pracy Mańczak 1956, a zatem rodzaj rzeczownika jest niezależny od wartości liczby, przy czym w ustaleniu rodzaju tych rzeczowników, które posiadają liczbę pojedynczą[1], pomóc mogą następujące konteksty:

[1] Poprzednie wersje tagsetu IPI PAN przyjmowały zestaw dziewięciu rodzajów zapro-

(3.8) Widzę jednego ____ z tych, których lubię. m1
(3.9) Widzę jednego ____ z tych, które lubię. m2
(3.10) Widzę jeden ____. m3
(3.11) Widzę jedno ____. n
(3.12) Widzę jedną ____. f

Kategoria **aspektu** jest czysto słownikowa, tj. formy nie odmieniają się przez aspekt, a jedynie mogą mieć pewną ustaloną wartość tej kategorii, stałą dla wszystkich form danego czasownika.

Kategoria **zanegowania** przysługuje tym formom czasownikowym, w wypadku których prefiks *nie-* jest pisany łącznie, a zatem odróżnia formy *pisanie* i *niepisanie*, *napisany* i *nienapisany*, ale nie formy *pisać* i *nie pisać*.

Kategorie **akcentowości** i **poprzyimkowości** dotyczą tylko niektórych form zaimków osobowych, a w wypadku poprzyimkowości — tylko niektórych form zaimków trzecioosobowych.

Kategoria **akomodacyjności** jest właściwa dla wszystkich form liczebnikowych i ma wartość uzgadniającą wtedy i tylko wtedy, gdy dana forma liczebnikowa wiąże się z formą rzeczownikową o tej samej wartości przypadka. Kategoria ta jest szczegółowo omówiona w pracach Przepiórkowski 2003b i Woliński 2003.

Dwie ostatnie kategorie, **aglutynacyjność** i **wokaliczność** wynikają z dzielenia form typu *niosłem* i *niosłam* na segmenty typu *niosł*|*em*, *niosła*|*m*. Choć w większości wypadków pierwszy segment w takich słowach ma tę samą postać, co odpowiednia trzecioosobowa pojedyncza forma przeszła,

ponowany w pracy Saloni 1976. Z powodu ograniczeń obecnej wersji analizatora morfologicznego oraz wobec wątpliwości (por. Przepiórkowski i in. 2002 i Woliński 2001) co do niektórych szczegółowych rozwiązań dotyczących rodzajów przymnogich zaproponowanych w tejże pracy Saloni 1976, niniejsza wersja tagsetu przyjmuje bardziej konserwatywny zestaw pięciu rodzajów. Formy przymnogie typu *wujostwo* oznaczamy jako posiadające rodzaj męski osobowy, zaś formy przymnogie typu *skrzypce*, *sanie*, *pomyje* uznajemy arbitralnie (ale por. rozumowanie przedstawione w pracy Przepiórkowski 2003a) za nijakie. Poniższe konteksty mogą pomóc przypisać odpowiednie rodzaje rzeczownikom *plurale tantum*:

(i) ____ byli ważni. m1
(ii) ____ były ważne. n

np. *ja* |*szedł*|*em*| i *on* |*szedł*|, czasami formy te się różnią, np. *ja* |*niosł*|*em*| i *on* |*niósł*|. W wypadku takich różnic, forma łącząca się z aglutynantem (cząstką typu *-em*), np. forma *niosł-*, zostanie oznakowana jako aglutynacyjna, zaś forma występująca samodzielnie, np. forma *niósł*, zostanie oznakowana jako nieaglutynacyjna. Dodatkowo, kategoria wokaliczności odróżnia aglutynanty łączące się z formami kończącymi się spółgłoską (np. *-em*) od aglutynantów łączących się z formami kończącymi się samogłoską (np. *-m*).

W niniejszym tagsecie brak jest kategorii czasu, trybu i strony, gdyż są one właściwe jednostkom większym niż segmenty.

## 3.4. Klasy gramatyczne

Podstawowym pojęciem niniejszego tagsetu odpowiadającym tradycyjnemu pojęciu *części mowy* jest *klasa gramatyczna*. Terminu tego będziemy używali wymiennie z terminem *klasa fleksyjna*.

Zasięg tradycyjnych części mowy, takich jak czasownik, rzeczownik, liczebnik czy zaimek, jest nieostry i przez to kontrowersyjny: czy tzw. *odsłowniki*, tj. formy typu *picie* i *palenie*, to czasowniki (posiadają kategorię aspektu, są regularnie powiązane z formami czasownikowymi typu *pić* i *palić*), czy też rzeczowniki (odmieniają się przez przypadek, posiadają słownikową kategorię rodzaju)?, czy *piąty* to liczebnik (na to wskazuje semantyka), czy też przymiotnik (na to wskazuje odmiana)?, czy *taki* to zaimek (semantyka), czy przymiotnik (odmiana)?

W Korpusie IPI PAN przyjęto klasy gramatyczne bardziej szczegółowe i lepiej zdefiniowane niż tradycyjne części mowy. Klasy te oparte są na pojęciu *fleksemu*, zaproponowanym w pracach Bień 1991, 2004, będącym pojęciem węższym od terminu *leksem*.

### 3.4.1. Fleksemy

W wypadku leksemów można nieściśle powiedzieć, że dwie formy należą do tego samego leksemu wtedy i tylko wtedy, gdy znaczą to samo, z dokładnością do produktywnych różnic znaczenia wynikających z różnych wartości odpowiednich kategorii gramatycznych (np. kategorii licz-

by czy osoby), oraz gdy mają podobną postać morfologiczną[2], a więc na przykład formy *pięć*, *pięcioma* i *pięciokrotny* można by uznać za formy tego samego leksemu, podobnie jak formami tego samego leksemu są *wypije*, *wypić* i *wypito*. Natomiast w wypadku fleksemów, do tych warunków dochodzi jeszcze warunek tożsamości kategorii gramatycznych: dwie formy należą do tego samego fleksemu wtedy i tylko wtedy, gdy znaczą to samo, mają podobną postać morfologiczną oraz posiadają te same kategorie gramatyczne, a zatem do tego samego fleksemu należą formy osobowe czasownika, posiadające kategorie liczby, rodzaju i aspektu, np *wypije*, *wypijecie*, *wypijemy*, ale nie formy *wypić* czy *wypito*, które kategorii liczby i rodzaju nie posiadają.

Na mocy powyższego pierwszego przybliżenia pojęcia *fleksem*, formy typu *wypić* i *wypito* należałoby uznać za formy tego samego fleksemu: mają one regularnie powiązane znaczenia i formy tekstowe, oraz te same kategorie gramatyczne, a mianowicie kategorię aspektu. Jednak obie te formy posiadają tę samą wartość aspektu, a zatem kategorie gramatyczne tych form nie pozwalają na ich rozróżnienie. Sytuacja taka może mieć miejsce w wypadku tzw. form wariantywnych, np. *funkcji* i *funkcyj*, czy *HIT-u* i *HIT-a*, których rola składniowa w zdaniu jest taka sama. W wypadku form *wypić* i *wypito* mamy jednak do czynienia z formami, których wartości kategorii gramatycznych są identyczne, lecz dystrybucja składniowa jest zupełnie odmienna. W takich wypadkach zażądamy, by formy te należały do osobnych fleksemów, a zatem wyodrębnimy nieodmienny fleksem bezokolicznikowy zawierający formę *wypić* i nieodmienny fleksem zawierający formę *wypito*.

Kontynuując to rozumowanie, formy czasownika dokonanego WYPIĆ możemy pogrupować w następujące fleksemy:

— tzw. pseudoimiesłów, zawierający formy odmienne przez liczbę i rodzaj, ale nie przez osobę, m.in.: *wypił*, *wypili*, *wypiły*,
— fleksem zawierający formy czasu przyszłego, odmienne przez liczbę i osobę, ale nie rodzaj, m.in.: *wypiję*, *wypijemy*, *wypiją*,
— rozkaźnik, także zawiera formy odmienne przez liczbę i osobę, ale w sposób defektywny: *wypijmy*, *wypij*, *wypijcie*,

[2] Znane wyjątki od tego ostatniego wymagania to leksem ROK, do którego należą m.in. formy *rokiem* i *latami*, czy leksem CZŁOWIEK, do którego należą m.in. formy *człowiekiem* i *ludźmi*.

— trzy fleksemy nieodmienne, zawierające po jednej formie: bezokolicznik (*wypić*), bezosobnik (*wypito*), imiesłów przysłówkowy uprzedni (*wypiwszy*),
— odsłownik, zawierający formy odmienne przez liczbę (często tylko potencjalnie), przypadek i zanegowanie, i posiadające ustalony rodzaj (nijaki), m.in.: *wypicie*, *wypiciem*, *niewypiciu*,
— imiesłów przymiotnikowy bierny, zawierający formy odmienne przez liczbę, przypadek i rodzaj, m.in.: *wypity*, *wypite*, *wypitymi*.

Rozumując podobnie, w wypadku czasownika niedokonanego PIĆ wyodrębnić można następujące fleksemy:

— pseudoimiesłów,
— fleksem zawierający formy czasu teraźniejszego, m.in.: *piję*, *pijemy*, *pijecie*,
— rozkaźnik,
— trzy fleksemy nieodmienne: bezokolicznik, bezosobnik oraz imiesłów przysłówkowy współczesny (*pijąc*),
— odsłownik,
— imiesłów przymiotnikowy bierny,
— imiesłów przymiotnikowy czynny, także zawierający formy odmienne przez liczbę, przypadek i rodzaj, m.in.: *pijący*, *pijące*, *pijącymi*.

Inne fleksemy należy wyróżnić dla czasownika BYĆ: oprócz pseudoimiesłowu (*był*, *byli* itd.), form teraźniejszych (*jestem*, *jesteśmy*, *jesteście* itd.), rozkaźnika (*bądźmy*, *bądź*, *bądźcie*), bezokolicznika (*być*), imiesłowu przysłówkowego współczesnego (*będąc*), odsłownika (*bycie* itp.) oraz imiesłowu przymiotnikowego czynnego (*będący* itp.), także:

— fleksem zawierający formy czasu przyszłego, odmienne przez liczbę i osobę, m.in. *będę*, *będziecie*, oraz
— aglutynant, tj. fleksem zawierający formy typu *-em*, *-śmy* itp.

W znacznie mniejszą liczbę fleksemów można pogrupować formy przymiotnikowe. Ze względu na odmianę można je podzielić na fleksemy przymiotnikowe zawierające formy odmienne przez liczbę, przypadek, rodzaj i — nie zawsze — stopień, np. fleksem składający się z form *polski*, *polskiej*, *polskimi* itp., oraz fleksemy nieodmienne, zawierające formy typu *polsko* (jak w *polsko-niemiecki*) oraz *polsku* (jak w *po polsku*). Ponieważ formy

te nie różnią się pod względem fleksji (żadna nie posiada kategorii fleksyjnych), zaś różnią się dystrybucją, wyodrębniamy dwa przymiotnikowe fleksemy nieodmienne: przymiotnik przyprzymiotnikowy (*polsko*) oraz przymiotnik poprzyimkowy (*polsku*).

Wyróżniając fleksemy rzeczownikowe, zakładamy, że mają one ustalony rodzaj gramatyczny, a więc na przykład *fryzjer* i *fryzjerka* to formy dwóch różnych fleksemów. A zatem typowy fleksem rzeczownikowy, odmienny przez przypadek i liczbę, zawiera 14 form (dla dwóch wartości kategorii liczby i siedmiu wartości przypadka), lecz istnieją także fleksemy *plurale tantum*, bez form liczby pojedynczej, np. fleksemy WUJOSTWO, URODZINY i SPODNIE, oraz *singulare tantum*, bez form liczby mnogiej, np. fleksemy KTO i CO.

Pewien kłopot sprawiają tzw. formy deprecjatywne rzeczowników męskoosobowych, np. *profesory* w *Przyszły te głupie profesory i błota naniosły*. Jak odróżnić formy niedeprecjatywne typu *profesorowie* od form deprecjatywnych typu *profesory*? Jedna z możliwości to wprowadzenie kategorii deprecjatywności, która by te formy odróżniała. Rozwiązanie to prowadziłoby jednak do komplikacji na poziomie opisu uzgodnienia takich deprecjatywnych form męskoosobowych z niemęskoosobowymi formami przymiotników (*te*, *głupie*) i czasowników (*przyszły*, *naniosły*). Dlatego też w niniejszym tagsecie przyjęto inne rozwiązanie, polegające na wyodrębnieniu dla takich form deprecjatywnych osobnych fleksemów, które zawierają jedynie dwie formy mnogie rodzaju męskiego zwierzęcego, o tym samym kształcie tekstowym (np. *profesory*), różniące się wartością kategorii przypadka, tj. formę mianownikową i wołaczową.

Także formy liczebnikowe tworzą kilka różnych fleksemów:

— liczebnik główny zawiera formy typu *pięć*, *pięciu*, *pięcioma* — formy te są odmienne przez przypadek, rodzaj i akomodacyjność (defektywnie), lecz mają ustaloną liczbę (mnogą),
— liczebnik zbiorowy zawiera formy typu *pięcioro*, *pięciorgiem* — oprócz ustalonej liczby mnogiej, formy mają ustalony rodzaj nijaki i odmieniają się przez przypadek i akomodacyjność (defektywnie),
— formy typu *piąty*, *piąta*, *piątymi* itp. stanowią fleksem przymiotnikowy,
— także formy *pięciokrotny*, *pięciokrotnemu* itp. stanowią osobny fleksem przymiotnikowy.

Większość tradycyjnie rozumianych zaimków jest z morfosyntaktycznego punktu widzenia przymiotnikami (*taki*, *jakiś*, *który* itp.), rzeczownikami (*kto*, *coś* itp.) itd. Ze względu na odmianę warto jednak wyróżnić zaimek SIEBIE, którego formy wydają się posiadać wyłącznie kategorię przypadka, oraz formy zaimków osobowych, posiadających dosyć skomplikowany paradygmat.

Wśród fleksemów odmiennych wymienić tu jeszcze należy stopniowalne przysłówki oraz fleksemy zawierające formy typu *winien*, *winna*, *winniśmy* itp. — pozostałe fleksemy są fleksemami nieodmiennymi, czyli zawierają pojedyncze formy, np. fleksem ORAZ zawierający spójnik *oraz*, czy też fleksem NA zawierający przyimek *na*.

### 3.4.2. Klasy fleksyjne

O ile fleksemy są niepustymi i rozłącznymi zbiorami form wyrazowych o jednorodnej charakterystyce semantycznej, morfologicznej, morfosyntaktycznej i — w pewnym stopniu — dystrybucyjnej, o tyle *klasy fleksyjne* to niepuste, rozłączne, morfosyntaktycznie i — w pewnym stopniu — dystrybucyjnie jednorodne zbiory fleksemów.

Tabela na następnej stronie zawiera przybliżoną charakterystykę morfoskładniową wszystkich klas fleksyjnych przyjmowanych w niniejszym tagsecie. Symbol $\oplus$ oznacza, że dla danej klasy fleksyjnej dana kategoria gramatyczna jest morfologiczna (fleksemy należące to tej klasy zwykle „odmieniają się" przez tę kategorię), zaś symbol $\odot$ oznacza, że dana kategoria jest słownikowa (dla każdego fleksemu danej klasy wszystkie formy tego fleksemu mają tę samą wartość tej kategorii, choć być może są to potencjalnie różne wartości dla różnych fleksemów, jak w wypadku rodzaju rzeczowników).

Poniżej przedstawiamy bardziej szczegółową charakterystykę morfosyntaktyczną oraz, w wypadku niektórych klas, dystrybucyjną poszczególnych klas fleksyjnych.

| | liczba | przypadek | rodzaj | osoba | stopień | aspekt | zanegowanie | akcentowość | poprzyimkowość | akomodacyjność | aglutynacyjność | wokaliczność |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rzeczownik | ⊕ | ⊕ | ⊙ | | | | | | | | | |
| rzeczownik deprecjatywny | ⊙ | ⊕ | ⊙ | | | | | | | | | |
| liczebnik główny | ⊙ | ⊕ | ⊕ | | | | | | | ⊕ | | |
| liczebnik zbiorowy | ⊙ | ⊕ | ⊙ | | | | | | | ⊕ | | |
| przymiotnik | ⊕ | ⊕ | ⊕ | | ⊕ | | | | | | | |
| przymiotnik przyprzym. | | | | | | | | | | | | |
| przymiotnik poprzyim. | | | | | | | | | | | | |
| przysłówek | | | | | ⊕ | | | | | | | |
| zaimek nietrzecioosobowy | ⊙ | ⊕ | ⊕ | ⊙ | | | | ⊕ | | | | |
| zaimek trzecioosobowy | ⊕ | ⊕ | ⊕ | ⊙ | | | | ⊕ | ⊕ | | | |
| zaimek SIEBIE | | ⊕ | | | | | | | | | | |
| forma nieprzeszła | ⊕ | | | ⊕ | | ⊙ | | | | | | |
| forma przyszła BYĆ | ⊕ | | | ⊕ | | ⊙ | | | | | | |
| aglutynant BYĆ | ⊕ | | | ⊕ | | ⊙ | | | | | | ⊕ |
| pseudoimiesłów | ⊕ | | ⊕ | | | ⊙ | | | | | ⊕ | |
| rozkaźnik | ⊕ | | | ⊕ | | ⊙ | | | | | | |
| bezosobnik | | | | | | ⊙ | | | | | | |
| bezokolicznik | | | | | | ⊙ | | | | | | |
| im. przys. współczesny | | | | | | ⊙ | | | | | | |
| im. przys. uprzedni | | | | | | ⊙ | | | | | | |
| odsłownik | ⊕ | ⊕ | ⊙ | | | ⊙ | ⊕ | | | | | |
| im. przym. czynny | ⊕ | ⊕ | ⊕ | | | ⊙ | ⊕ | | | | | |
| im. przym. bierny | ⊕ | ⊕ | ⊕ | | | ⊙ | ⊕ | | | | | |
| winien | ⊕ | | ⊕ | | | ⊙ | | | | | | |
| predykatyw | | | | | | | | | | | | |
| przyimek | | ⊙ | | | | | | | | | | |
| spójnik | | | | | | | | | | | | |
| kublik | | | | | | | | | | | | |
| ciało obce nominalne | ⊕ | ⊕ | ⊙ | | | | | | | | | |
| ciało obce luźne | | | | | | | | | | | | |
| forma nierozpoznana | | | | | | | | | | | | |
| interpunkcja | | | | | | | | | | | | |

**rzeczownik** zawiera fleksemy odmienne przez liczbę i przypadek, o ustalonym rodzaju gramatycznym, nie posiadające kategorii osoby, np. WODA, PROFESOR, PIĘCIOKROTNOŚĆ; do klasy tej zaliczymy także fleksemy defektywne *plurale tantum* i *singulare tantum*, lecz nie fleksemy deprecjatywne,

**rzeczownik deprecjatywny** zawiera fleksemy deprecjatywne, czyli fleksemy o ustalonej liczbie (mnogiej) i o ustalonym rodzaju (męskim zwierzęcym), defektywnie odmienne przez przypadek (tylko formy mianownika i wołacza), np. PROFESORY, STUDENTY,

**liczebnik główny** zawiera fleksemy odmienne przez przypadek, rodzaj i akomodacyjność (odmiana przez akomodacyjność jest defektywna), o ustalonej wartości liczby (zwykle mnoga), a więc fleksemy takie jak PIĘĆ i WIELE, w tym fleksemy liczebników defektywnych typu TROCHĘ i DUŻO, o wartościach przypadka ograniczonych do mianownika, biernika i dopełniacza,

**liczebnik zbiorowy** zawiera fleksemy odmienne przez przypadek i — defektywnie — akomodacyjność, o ustalonej wartości liczby (mnoga) i rodzaju (zawsze nijaki), a więc fleksemy liczebników zbiorowych typu PIĘCIORO,

**przymiotnik** zawiera fleksemy odmienne co najmniej przez liczbę, przypadek i rodzaj, a także być może przez stopień, a więc fleksemy typu MIŁY, TECHNICZNY, TAKI, KTÓRY, PIĄTY, WIELOKROTNY i JEDEN,

**przymiotnik przyprzymiotnikowy** zawiera nieodmienne fleksemy odprzymiotnikowe typu POLSKO,

**przymiotnik poprzyimkowy** zawiera nieodmienne fleksemy odprzymiotnikowe typu POLSKU,

**przysłówek** zawiera fleksemy odmienne jedynie przez stopień (przysłówki stopniowalne, np. BARDZO, MIŁO) oraz fleksemy nieodmienne, niebędące przymiotnikami przyprzymiotnikowymi lub poprzyimkowymi, lecz o znaczeniu i postaci regularnie powiązanych z odpowiednimi przymiotnikami (niestopniowalne przysłówki odprzymiotnikowe, np. TECHNICZNIE CZY TAK),

**zaimek nietrzecioosobowy** zawiera cztery fleksemy odmienne przez przypadek i rodzaj: JA, MY, TY, WY, z których każdy ma określone liczbę i osobę, zaś niektóre formy fleksemów JA i TY są zróżnicowane ze względu na akcentowość,

**zaimek trzecioosobowy** zawiera jeden fleksem, ON, o ustalonej trzeciej osobie, odmienny przez liczbę, przypadek i rodzaj, z niektórymi formami zróżnicowanymi ze względu na akcentowość i poprzyimkowość,

**siebie** zawiera jeden fleksem, SIEBIE, odmienny przez przypadek, lecz bez formy mianownika i wołacza,

**forma nieprzeszła** zawiera fleksemy odmienne przez liczbę i osobę: przyszłe (o ustalonej dokonanej wartości aspektu) oraz teraźniejsze (o ustalonej niedokonanej wartości aspektu),

**forma przyszła czasownika BYĆ** zawiera jeden fleksem, składający się z form *będę*, *będziesz* itp.,

**aglutynant czasownika BYĆ** zawiera jeden fleksem, składający się z form *-m*, *-em*, *-śmy* itp.,

**pseudoimiesłów** zawiera fleksemy odmienne przez liczbę i rodzaj, o ustalonej wartości aspektu,

**rozkaźnik** zawiera fleksemy defektywnie odmienne przez liczbę i osobę (zawierające formy 1.os. l.p., 2.os. l.p. oraz 2.os. l.m.),

**bezosobnik** zawiera nieodmienne fleksemy na *-no*, *-to* (o ustalonym aspekcie),

**bezokolicznik** zawiera nieodmienne fleksemy bezokolicznikowe (o ustalonym aspekcie),

**imiesłów przysłówkowy współczesny** zawiera niedokonane imiesłowy przysłówkowe,

**imiesłów przysłówkowy uprzedni** zawiera dokonane imiesłowy przysłówkowe,

**odsłownik** zawiera fleksemy odmienne przez liczbę, przypadek i zanegowanie, o ustalonym rodzaju (zawsze nijaki) i aspekcie,

**imiesłów przymiotnikowy czynny** zawiera czynne imiesłowy przymiotnikowe odmienne przez liczbę, przypadek, rodzaj i zanegowanie, o ustalonej wartości aspektu (niedokonanej),

**imiesłów przymiotnikowy bierny** zawiera bierne imiesłowy przymiotnikowe odmienne przez liczbę, przypadek, rodzaj i zanegowanie, o ustalonej wartości aspektu,

**winien** zawiera fleksemy WINIEN, POWINIEN i RAD, odmienne przez liczbę i rodzaj, posiadające tylko analityczne formy czasu przeszłego i trybu warunkowego,

**predykatyw** zawiera nieodmienne syntetycznie fleksemy typu BRAK, TRZEBA, WARTO, ŻAL, SŁYCHAĆ, WIDAĆ itp., które odmieniają się wyłącznie analitycznie (np. *było warto, warto, warto by, będzie warto*),

**przyimek** zawiera nieodmienne fleksemy przyimkowe, posiadające ustaloną wartość kategorii przypadka, odpowiadającą rekcji przyimka[3], niełączące się z niepoprzyimkowymi formami zaimków: BEZ, BEZE, CO, DLA, DO, DOKOŁA, DOOKOŁA, DZIĘKI, KOŁO, KONTRA, KU, MIĘDZY, MIMO, NA, NAD, NADE, NAOKOŁO, NAPRZECIW, NAPRZECIWKO, O, OBOK, OD, ODE, ODNOŚNIE, OPRÓCZ, PER, PO, POD, PODE, PODCZAS, PODLE, PODŁUG, POMIĘDZY, POMIMO, PONAD, PONIŻEJ, POPRZEZ, POŚRODKU, POŚRÓD, POWYŻEJ, POZA, PRÓCZ, PRZECIW, PRZECIWKO, PRZED, PRZEDE, PRZEZ, PRZEZE, PRZY, SPOD, SPODE, SPOMIĘDZY, SPOŚRÓD, SPOZA, SPRZED, ŚRÓD, U, W, WE, WBREW, WEDLE, WEDŁUG, WEWNĄTRZ, WOBEC, WOKOŁO, WOKÓŁ, WSKUTEK, WŚRÓD, WZDŁUŻ, Z, ZA, ZE, ZAMIAST, ZEWNĄTRZ, ZNAD, ZNADE, ZZA,

**spójnik** zawiera nieodmienne fleksemy spójnikowe: A, ABY, ACZKOLWIEK, ALBO, ALBOWIEM, ALE, ALEŻ, ANI, ANIŻELI, AŻ, AŻEBY, BĄDŹ, BO, BOWIEM, BY, BYLE, CHOCIAŻ, CHOCIAŻBY, CHOĆ, CHOĆBY, CZY, CZYLI, DOPÓKI, DOPÓTY, GDY, GDYBY, GDYŻ, I, IM, IŻ, IŻBY, JAK, JAKBY, JAKKOLWIEK, JAKO, JAKOBY, JEDNAK, JEDNAKŻE, JEŚLI, JEŚLIBY, JEŻELI, KIEDY, LECZ, LUB, NATOMIAST, NI, NIM, NIŻ, ORAZ, PONIEWAŻ, PÓKI, PÓTY, PRZETO, SKORO, TAK, TEDY, TO, TOTEŻ, TYLKO, TYM, WIĘC, ZAMIAST, ZANIM, ZARÓWNO, ZAŚ, ZATEM, ŻE, ŻEBY,

**kublik** zawiera nieodmienne fleksemy nie mieszczące się w poprzednich kategoriach, a więc m.in. fleksemy JUŻ, NADER, ZBYT, SIĘ, NIE, BY, -LI, OJ, AHA itp.

Oprócz powyższych klas fleksyjnych, wprowadzone zostały cztery dodatkowe klasy pomocnicze:

**ciało obce nominalne** to klasa wtrętów obcojęzycznych, wzorów matematycznych lub chemicznych itp., znajdujących się w pozycji frazy rzeczownikowej, a więc o potencjalnie możliwej do określenia wartości liczby, przypadka i rodzaju,

**ciało obce luźne** to klasa wtrętów obcojęzycznych, wzorów matematycznych lub chemicznych itp. w pozycji innej niż rzeczownikowa,

[3] A zatem znaczenie kategorii przypadka w wypadku form przyimkowych jest skrajnie różne od znaczenia tej kategorii przy innych formach.

**forma nierozpoznana** to klasa fleksemów nierozpoznanych w wyniku automatycznej analizy morfologicznej,

**interpunkcja** klasa zawiera nieodmienne „fleksemy" interpunkcyjne zawierające formy takie jak *:*, *.*, *!* itp.

### 3.4.3. Formy podstawowe

Jak stwierdzono powyżej, znaczniki morfosyntaktyczne przypisywane poszczególnym segmentom zawierają nie tylko informację o klasie gramatycznej i odpowiednich kategoriach gramatycznych, ale także o formie podstawowej właściwej dla danej interpretacji segmentu. Jaka jednak powinna być forma podstawowa segmentu *idziemy*? Czy powinna to być jedna z form należących do tego samego fleksemu, co *idziemy*, a więc na przykład *idę*, czy też powinna to być tradycyjna forma podstawowa, a więc bezokolicznik *iść*, mimo iż należy on do innego fleksemu?

W niniejszym korpusie przyjęto rozwiązanie tradycyjne, a więc segmentom przypisywane są tradycyjne formy podstawowe, takie jak bezokolicznik czy też mianownik rodzaju męskiego liczby pojedynczej, nawet jeśli dana forma podstawowa nie należy do tego samego fleksemu, co znakowany segment.

Poniższa tabela zawiera informacje o formach podstawowych dla poszczególnych klas fleksyjnych, a także skróty nazw klas fleksyjnych używane w opisywanym korpusie.

| fleksem | skrót | forma podstawowa | przykład |
|---|---|---|---|
| rzeczownik | subst | mianownik liczby pojedynczej | *profesor* |
| rzeczownik deprecjatywny | depr | mianownik liczby pojedynczej rzeczownika | *profesor* |
| liczebnik główny | num | mianownik rodzaju męskiego rzeczowego | *pięć, dwa* |
| liczebnik zbiorowy | numcol | mianownik rodzaju męskiego rzeczowego liczebnika głównego | *pięć, dwa* |
| przymiotnik | adj | mianownik liczby pojedynczej rodzaju męskiego stopnia równego | *polski* |
| przymiotnik przyprzym. | adja | mianownik liczby pojedynczej rodzaju męskiego przymiotnika w stopniu równym | *polski* |

| | | | |
|---|---|---|---|
| przymiotnik poprzyimkowy | adjp | mianownik liczby pojedynczej rodzaju męskiego przymiotnika w stopniu równym | *polski* |
| przysłówek | adv | forma stopnia równego | *dobrze, bardzo* |
| zaimek nietrzecioosobowy | ppron12 | mianownik liczby pojedynczej | *ja* |
| zaimek trzecioosobowy | ppron3 | mianownik liczby pojedynczej | *on* |
| zaimek SIEBIE | siebie | biernik | *siebie* |
| forma nieprzeszła | fin | bezokolicznik | *czytać* |
| forma przyszła BYĆ | bedzie | bezokolicznik | *być* |
| aglutynant BYĆ | aglt | bezokolicznik | *być* |
| pseudoimiesłów | praet | bezokolicznik | *czytać* |
| rozkaźnik | impt | bezokolicznik | *czytać* |
| bezosobnik | imps | bezokolicznik | *czytać* |
| bezokolicznik | inf | bezokolicznik | *czytać* |
| im. przys. współczesny | pcon | bezokolicznik | *czytać* |
| im. przys. uprzedni | pant | bezokolicznik | *czytać* |
| odsłownik | ger | bezokolicznik | *czytać* |
| im. przym. czynny | pact | bezokolicznik | *czytać* |
| im. przym. bierny | ppas | bezokolicznik | *czytać* |
| winien | winien | forma męska liczby pojedynczej | *powinien, rad* |
| predykatyw | pred | jedyna forma tego fleksemu | *warto* |
| przyimek | prep | jedyna forma tego fleksemu | *na, przez, w* |
| spójnik | conj | jedyna forma tego fleksemu | *oraz* |
| kublik | qub | jedyna forma tego fleksemu | *nie, -że, się* |
| ciało obce nominalne | xxs | mianownik liczby pojedynczej | *de, l'Hospital* |
| ciało obce luźne | xxx | jedyna forma tego fleksemu | *bene* |
| forma nierozpoznana | ign | jedyna forma tego fleksemu | |
| interpunkcja | interp | jedyna forma tego fleksemu | *;, ., (, ]* |

## 3.5. Nietypowe segmenty języka pisanego

Anotacja morfosyntaktyczna tekstów pisanych wymaga podjęcia szeregu decyzji o segmentacji i znakowaniu ciągów znaków znajdujących się na pograniczu zainteresować lingwistów i typografów. Niniejszy punkt zawiera omówienie szczegółowych decyzji dotyczących pewnych klas takich ciągów typowych dla tekstów pisanych.

### 3.5.1. Haplologia kropki

Pewna klasa form języka naturalnego kończy się kropką, np.:

— skróty typu *np.*, *itp.*, a w przypadkach zależnych także *dr.*, *mgr.* itp.,
— liczby pisane cyframi w znaczeniu porządkowym,
— inicjały.

Nie jest jasne, jak traktować kropkę w takich formach, gdy występują one na końcu zdania i gdy kropka ta zdaje się pełnić także funkcję znaku interpunkcyjnego kończącego zdanie, np.:

(3.13) Działo się to w 1945 r.
(3.14) Czy to 3. pacjent? Nie, 2.
(3.15) Obecnego prezydenta Stanów Zjednoczonych zwą George W.

W niniejszym tagsecie przyjęte zostało konsekwentne choć nietradycyjne rozwiązanie polegające na traktowaniu kropki występującej na końcu zdania zawsze jako znaku interpunkcyjnego. A zatem w przykładach powyżej kropka jest traktowana jako osobny segment: *r*|.|, *2*|.| i *W*|.|.

Z drugiej strony, gdy kropka w takich formach nie pełni podwójnej funkcji, jest ona traktowana jako część większego segmentu, np. formy *r.*, *2.* i *W.* są pojedynczymi segmentami w poniższych zdaniach:

(3.16) Działo się to w 1945 r. lub później.
(3.17) Czy to 3. pacjent? Nie, to 2. pacjent.
(3.18) Obecny prezydent Stanów Zjednoczonych nazywa się George W. Bush.

Formami hasłowymi skrótów są też skróty: bez kropki w wypadku skrótów takich jak *wg*, *dr* i *mgr*, z kropką w wypadku skrótów takich jak

*hab.*, *itp.*, czy *np.*, zgodnie z regułami polskiej ortografii. W wypadku liczb porządkowych, formą hasłową jest ta sama liczba, ale zawsze z kropką (nawet jeżeli w tekście liczba wystąpiła bez kropki). Formami hasłowymi inicjałów są inicjały, także zawsze z kropką, np.:

(3.19) *Klawiatura, myszka itp. są wliczone w cenę komputera.*
— segment: *itp.*
— forma hasłowa: *itp.*
(3.20) *Wliczone w cenę komputera są klawiatura, myszka itp.*
— segment: *itp*
— forma hasłowa: *itp.*
(3.21) *Działo się to w 1945 r. lub później*
— segmenty: *1945*, *r.*
— formy hasłowe: *1945.*, *r.*
(3.22) *Działo się to w 1945 r.*
— segmenty: *1945*, *r*
— formy hasłowe: *1945.*, *r.*
(3.23) *To 3. pacjent.*
— segment: *3.*
— forma hasłowa: *3.*
(3.24) *Nie, to już 4.*
— segment: *4*
— forma hasłowa: *4.*
(3.25) *To George W. Bush.*
— segment: *W.*
— forma hasłowa: *W.*
(3.26) *Ale zwą go George W.*
— segment: *W*
— forma hasłowa: *W.*
(3.27) *Oto mgr Kwaśniewski.*
— segment: *mgr*
— forma hasłowa: *mgr*
(3.28) *Rozmawiałem z mgr. Kwaśniewskim.*
— segment: *mgr.*
— forma hasłowa: *mgr*

### 3.5.2. Skróty

Segmentacja i lematyzacja skrótów została omówiona w p. 3.5.1.

Znaczniki skrótów jednowyrazowych powinny odpowiadać znacznikom wyrazów, do których te skróty w tym kontekście się rozwijają. Na przykład, dla segmentu *mgr.* w zdaniu *Rozmawiałem z mgr. Kwaśniewskim* powinien to być znacznik subst:sg:inst:m1.

W wypadku skrótów wielowyrazowych, ich klasa fleksyjna jest ustalana na podstawie ich odmiany i dystrybucji, np.:

— kubliki: *itp.*, *itd.*, *np.*, *etc.*, *jw.*,
— przymiotniki: *tzw.*, *śp.*[4], *ww.*,
— rzeczowniki: *br.*, *cd.*,
— przyimki: *ds.* (prep:gen), *pt.* (prep:nom),
— formy nieprzeszłe czasownika: *cdn.*

### 3.5.3. Liczby

Forma hasłowa liczby zapisanej cyframi arabskimi lub rzymskimi to ta sama liczba, być może z dodaną kropką dla liczb odpowiadających liczebnikom porządkowym (por. p. 3.5.1).

Jeżeli liczba odpowiada frazie liczebnikowej porządkowej, to jest ona znakowana jako przymiotnik, tak jak liczebniki porządkowe, zaś jeżeli odpowiada liczebnikowi głównemu — jako liczebnik główny, przy czym liczba *1* jest przymiotnikiem, podobnie jak forma *jednego*, np.:

(3.29) Dałem to *21.* pacjentowi. adj:sg:dat:m1:pos
(3.30) Dałem to *21* pacjentom. num:pl:dat:m1:congr
(3.31) Dałem to *1.* pacjentowi. adj:sg:dat:m1:pos
(3.32) Dałem to *1* pacjentowi. adj:sg:dat:m1:pos
(3.33) *0* komputerów zostało sprzedanych. num:pl:nom:m3:rec

Liczby całkowite ujemne traktowane są analogicznie jak liczby naturalne. Jedyna różnica wynika z faktu, że liczba *–1* jest liczebnikiem, a nie przymiotnikiem, chyba że zostanie użyta jako forma liczebnika porządkowego. Minus jest częścią takich segmentów liczbowych.

[4] Przypadek wątpliwy, ale zaklasyfikowany jako przymiotnik na podstawie dystrybucji.

Liczby rzeczywiste niecałkowite są znakowane jako formy liczebnikowe.

### 3.5.4. Imiona, nazwiska, inicjały

Imiona, nazwiska i inicjały znakowane są jako rzeczowniki o wartości rodzaju odpowiadającej płci danej osoby, czyli m1 lub f. Segmentacja i lematyzacja inicjałów została omówiona w p. 3.5.1.

### 3.5.5. Symbole typu %, $, €, ¥ itp.

Formy hasłowe symboli typu %, $, €, ¥ to te same symbole, natomiast ich znaczniki są takie jak znaczniki odpowiadających im pełnych form, np.:

(3.34) Kosztowało to 5$.
- forma hasłowa: *$*
- znacznik: subst:pl:gen:m3

(3.35) Już tylko 5% wyborców nie popiera Leppera.
- forma hasłowa: %
- znacznik: subst:sg:nom:m3

# 4

# Przeszukiwanie korpusu



Do przeszukiwania Korpusu IPI PAN i wyświetlania konkordancji służy program Poliqarp[1], stworzony w ramach niniejszego projektu przez Zygmunta Krynickiego i Daniela Janusa, pod kierunkiem autora[2]. Poliqarp ma ambicje być przeszukiwarką uniwersalną: tagset używany w danym korpusie nie jest wbudowany w program, tylko zadany przez zewnętrzny plik konfiguracyjny, zaś wewnętrznie wykorzystywany format kodowania znaków to uniwersalny format UTF-8. Nic więc nie stoi na przeszkodzie, by używać tej przeszukiwarki do korpusów innych niż Korpus IPI PAN, w tym do korpusów innych języków.

[1] *POLyinterpretation Indexing Query and Retrieval Processor*.

[2] W początkowych etapach prac projektowych i implementacyjnych udział brał także Mateusz Przepiórkowski.

Program Poliqarp istnieje w trzech wersjach:

— wersja graficzna, opisana w p. 4.2.2, dla systemów operacyjnych: Windows 2000, Windows XP (oraz, być może, innych systemów klasy Windows, nie był jednak dla nich testowany) i GNU/Linux;
— wersja tekstowa, opisana w p. 4.2.3, dla systemu GNU/Linux;
— wersja internetowa, opisana w p. 4.2.1, pozwalająca na korzystanie z Korpusu IPI PAN poprzez Internet za pomocą dowolnej przeglądarki internetowej typu Mozilla, Internet Explorer, Opera czy Links.

Wspólna dla tych trzech wersji jest bogata składnia zapytań, opisana w p. 4.1.

## 4.1. Składnia zapytań

Składnia zapytań w programie Poliqarp jest wzorowana na składni używanej w programie Corpus Query Processor (CQP) stworzonym na Uniwersytecie w Stuttgarcie (Christ, 1994), zawiera jednak szereg rozszerzeń i udogodnień. Niniejszy rozdział omawia składnię zapytań programu Poliqarp i ilustruje ją wieloma przykładami.

### 4.1.1. Zapytania o segmenty

W najprostszym wypadku zapytanie to ciąg szukanych segmentów, np.:

(4.1) `przyszedł czas`
(4.2) `przyszedł em rano`

W zapytaniu (4.2) powyżej występują trzy segmenty odpowiadające dwóm słowom (por. p. 3.1): *przyszedłem* i *rano*. W wypadku prostych zapytań o formy Poliqarp próbuje wykryć sytuacje, w których słowo składa się z kilku segmentów i znaleźć odpowiednie ciągi segmentów, a zatem prawidłowe wyniki dadzą także zapytania:

(4.3) `przyszedłem rano`
(4.4) `długom szedł`

W tym drugim wypadku znalezione zostaną zarówno wszystkie wystąpienia trzysegmentowych ciągów |*długo*|*m*| |*szedł*|, interpretowanych jako

przysłówek, aglutynant i pseudoimiesłów, jak i dwusegmentowych ciągów *długom szedł*, interpretowanych jako celownikowa forma nominalna i pseudoimiesłów.

Domyślnie rozróżniana jest kasztowość[3] (wielkość) liter, a zatem poniższe dwa zapytania dadzą różne wyniki:

(4.5) `przyszedł`

(4.6) `Przyszedł`

Aby znaleźć wszystkie wystąpienia formy *przyszedł*, niezależnie od wielkości poszczególnych liter, należy użyć flagi `/i`. Poniższe zapytania dadzą te same wyniki, zawierające wszystkie odpowiedzi na oba powyższe zapytania.

(4.7) `przyszedł/i`

(4.8) `Przyszedł/i`

W wersji graficznej i tekstowej programu kasztowość można ustawić także globalnie — dla całego zapytania lub szeregu zapytań (por. pp. 4.2.2 i 4.2.3).

W zapytaniach o segmenty mogą wystąpić standardowe wyrażenia regularne wykorzystujące następujące znaki specjalne: `?`, `*`, `+`, `.`, `,`, `|`, `{`, `}`, `[`, `]`, `(`, `)` oraz liczby naturalne pisane cyframi arabskimi, np. `0` czy `21`, przy czym specyfikacje segmentów zawierające wyrażenia regularne muszą być ujęte w cudzysłowy `"`. Ponieważ formalny opis wyrażeń regularnych wykracza poza ramy niniejszej publikacji, ograniczymy się tutaj do kilku przykładów, które powinny pozwolić użytkownikowi na szybkie przyswojenie składni i znaczenia takich wyrażeń.

(4.9) `"Ala|Ela"`
znak `|` oznacza alternatywę dwóch wyrażeń, a zatem zapytanie to może zostać użyte do znalezienia wszystkich wystąpień segmentów o postaci *Ala* lub *Ela*,

(4.10) `"[AE]la"`
nawiasy kwadratowe oznaczają alternatywę znaków, a zatem zapytanie to może zostać użyte do znalezienia tych segmentów, których

[3] Termin ten, odpowiadający angielskiemu *case sensitivity*, został zaproponowany w pracach (Bień, 1991, 2001).

pierwszy znak to *A* lub *E*, a następne to *la*, tj. zapytanie to jest równoważne poprzedniemu,

(4.11) `"beza?"`
znak zapytania oznacza opcjonalność znaku (tutaj *a*) lub ujętego w nawiasy okrągłe wyrażenia bezpośrednio poprzedzającego znak `?`, a zatem w wyniku zadania tego zapytania znalezione zostaną segmenty *bez* i *beza*,

(4.12) `"bez."`
kropka oznacza dowolny znak, a zatem wynikiem tego zapytania będą segmenty *beza*, *bezy*, *bezą* itp., ale nie *bez* czy *bezami*,

(4.13) `"bez.?"`
*bez*, *beza*, *bezy*, *bezą* itp., ale nie *bezami*,

(4.14) `".z.z."`
segmenty pięcioznakowe, w których 2. i 4. znak to *z* (np. *czczą* i *rzezi*),

(4.15) `".z.z..?"`
segmenty składające się z pięciu lub sześciu znaków, w których 2. i 4. znak to *z*, np. *czczą*, *rzezi* i *szczyt*,

(4.16) `"a*by"`
gwiazdka oznacza dowolną liczbę wystąpień znaku lub wyrażenia bezpośrednio przed nią, a zatem zapytanie to może posłużyć do znalezienia segmentów składających się z dowolnej liczby liter *a*, po których następuje ciąg *by*, np. *by* (zero wystąpień *a*), *aby*, *aaaaby* itp.,

(4.17) `"Ala.*"`
segmenty zaczynające się na *Ala*, np. *Ala* i *Alabama*,

(4.18) `"ala.*"/i`
segmenty zaczynające się na *ala*, *Ala*, *aLa*, *ALA* itd., np. *Ala*, *alabaster* i *ALABAMA*,

(4.19) `".*al+"`
plus ma działanie podobne do gwiazdki i oznacza dowolną *większą od zera* liczbę wystąpień znaku lub wyrażenia bezpośrednio przed nim, a zatem wynikiem tego zapytania będzie znalezienie segmentów kończących się na *al*, *all*, *alll* itd., ale nie na *a*, np. *dal*, *robal* i *Gall*,

(4.20) `"a{1,3}b.*"/i`
konstrukcja typu `{n,m}` oznacza od `n` do `m` wystąpień znaku lub wyrażenia bezpośrednio przed nią, a zatem zapytanie to pomoże znaleźć segmenty zaczynające się od ciągu od 1 do 3 liter *a* lub *A*,

po którym następuje litera *b* lub *B*, a następnie dowolny ciąg znaków (por. `.*`), np. *Aby*, *aaaby*, *absolutnie*, *ABBA*,

(4.21) `".*(la){3,}.*"`
konstrukcja typu `{n,}` oznacza co najmniej `n` wystąpień znaku lub ujętego w nawiasy okrągłe wyrażenia bezpośrednio przed nią, a zatem zapytanie to może posłużyć do znalezienia segmentów, w których ciąg *la* występuje przynajmniej 3 razy pod rząd, np. *tralalala*, *sialalala*,

(4.22) `"[bcćdfghjklłmnńprsśtwzźż]{4,}[aąeęioóuy]"/i`
segmenty składające się z co najmniej 4 spółgłosek i dokładnie jednej samogłoski, np. *źdźbła* i *Chrzczę* — wyrażenie `[bcćdfghjklłmnńprsśtwzźż]{4,}` oznacza co najmniej czterokrotne powtórzenie znaku pasującego do `[bcćdfghjklłmnńprsśtwzźż]`, tj. co najmniej cztery wystąpienia spółgłoski (niekoniecznie tej samej),

(4.23) `"([bcćdfghjklłmnńprsśtwzźż]{3}[aąeęioóuy]){2,}"/i`
segmenty składające się z co najmniej dwukrotnego powtórzenia wzorca CCCV, gdzie C to spółgłoska, a V to samogłoska, np. *wszystko*, *Zdmuchnąwszy* i *Szmajdziński* — konstrukcja typu `{n}` oznacza *dokładnie* `n` wystąpień znaku lub ujętego w nawiasy okrągłe wyrażenia bezpośrednio przed nią,

(4.24) `"(pod|na|za)jecha.*"`
segmenty zaczynające się od *podjecha*, *najecha* i *zajecha*, np. *podjechał*, *zajechawszy*.

Specyfikacje segmentów podane powyżej muszą pasować do całych segmentów — stąd konieczność umieszczenia po obu stronach ciągu `(la){3,}` w zapytaniu (4.21) o segmenty zawierające ciąg *lalala* wyrażenia `.*`, pasującego do dowolnego ciągu znaków. Ten sam efekt zostanie osiągnięty przy użyciu flagi `/x`, oznaczającej, że dana specyfikacja segmentu musi pasować przynajmniej do części danego segmentu, niekoniecznie do całego segmentu:

(4.25) `"(la){3,}"/x`
segmenty, w których ciąg *la* występuje przynajmniej 3 razy pod rząd, np. *tralalala*, *sialalala*,

(4.26) `"(la){3,}"/ix`
segmenty, w których występuje ciąg *lalala*, *LaLAla* itp., np. *tralalala*, *SiaLaLALA*.

### 4.1.2. Zapytania o formy podstawowe

Aby znaleźć wszystkie formy leksemu KORPUS, można użyć następującego zapytania:

(4.27) `[base=korpus]`

Atrybut `base` jest jednym z wielu możliwych atrybutów, jakie mogą pojawić się w zapytaniu. Wartością tego atrybutu powinna być specyfikacja formy podstawowej (hasłowej) w rozumieniu p. 3.4.3, a zatem zapytanie `[base=pisać]` może być użyte do znalezienia form typu *pisać*, *piszę*, *pisała*, *piszcie*, *pisanie*, *pisano*, *pisane* itp.

Innym możliwym atrybutem jest `orth` — wartości tego atrybutu określają same segmenty, a zatem następujące pary zapytań są równoważne:

(4.28) a. `przyszedł`
b. `[orth=przyszedł]`

(4.29) a. `Przyszedł/i`
b. `[orth=Przyszedł/i]`

(4.30) a. `przyszedł czas`
b. `[orth=przyszedł][orth=czas]`

Nie są jednak równoważne zapytania:

(4.31) a. `przyszedłem rano`
b. `[orth=przyszedłem][orth=rano]`

W pierwszym wypadku program spróbuje rozbić słowo *przyszedłem* na dwa segmenty, *przyszedł* i *em*, podczas gdy wartość atrybutu `orth` jest zawsze interpretowana jako specyfikacja pojedynczego segmentu.

Wartościami atrybutów `base` i `orth` mogą być wyrażenia regularne, podobnie jak w wypadku prostych zapytań o formy opisanych w p. 4.1.1, np.:

(4.32) `[orth="bez.?"/i]`
znalezione zostaną segmenty *bez*, *Beza*, *bezy* itp., ale nie *bzem* czy *bezami*,

(4.33) `[base="bez.?"/i]`
znalezione zostaną wszystkie segmenty, których forma hasłowa ma 3 lub 4 litery i zaczyna się od *bez* (przy dowolnej kasztowości), czyli np. segmenty *bzem*, *bez*, *bezami* itp.

### 4.1.3. Zapytania wyższego rzędu

Zapytania o segmenty i o formy podstawowe segmentów można łączyć. Na przykład, aby znaleźć wszystkie wystąpienia segmentu *minę* rozumianego jako forma leksemu MINA (a nie na przykład leksemu MIJAĆ), można zadać następujące zapytanie:

(4.34) `[orth=minę & base=mina]`

Podobne znaczenie ma następujące zapytanie o te wystąpienia segmentu *minę*, które *nie* są interpretowane jako formy leksemu MIJAĆ.

(4.35) `[orth=minę & base!=mijać]`

Warunek, że forma hasłowa to nie *mijać*, można zadać także, umieszczając znak negacji (wykrzyknik) przed nazwą atrybutu, a więc poniższe zapytanie jest w pełni równoważne zapytaniu powyższemu.

(4.36) `[orth=minę & !base=mijać]`

Podobnie jak w rachunku zdań, zapytanie z podwójną negacją jest równoważna zapytaniu bez negacji, a zatem następujące zapytania o segment *nie* jako formę zaimka osobowego ON są w pełni równoważne:

(4.37) `[orth=nie & base=on]`
(4.38) `[orth=nie & !base!=on]`
(4.39) `[orth=nie & !!!base!=on]`
(4.40) `[orth=nie & !!base=on]`

W powyższych zapytaniach operator `&` spełnia rolę logicznej koniunkcji. Operatorem do niego dualnym jest operator `|`, spełniający rolę logicznej alternatywy. Oto kilka przykładów użycia tego operatora:

(4.41) `[base=on | base=ja]`
wszystkie formy zaimków ON i JA, równoważne zapytaniu `[base="on|ja"]`,

(4.42) `[base=on | orth=mnie | orth=ciebie]`
wszystkie formy zaimka ON, a także segmenty *mnie* i *ciebie*,

(4.43) `[orth=pora & !(base=por | base=pora)]`
segment *pora* nie będący ani formą leksemu POR, ani formą leksemu PORA.

Aby lepiej zrozumieć różnicę pomiędzy operatorami `&` i `|`, porównajmy następujące dwa zapytania:

(4.34) `[orth=minę & base=mina]`

(4.44) `[orth=minę | base=mina]`

W wyniku zadania pierwszego zapytania znalezione zostaną te segmenty, które są jednocześnie (koniunkcja) segmentem *minę* i formą leksemu MINA, a więc wyłącznie te wystąpienia segmentu *minę*, które są interpretowane jako formy leksemu MINA. W wyniku zadania drugiego zapytania znalezione natomiast zostaną te segmenty, które są albo dowolnie interpretowanym segmentem *minę*, albo formą leksemu MINA (alternatywa), czyli wszystkie wystąpienia zarówno segmentu *minę*, jak i segmentów *mina*, *miny*, *minami* itp. interpretowanych jako formy leksemu MINA.

Specyfikacje pozycji w korpusie, ujęte w nawiasy kwadratowe, mogą zawierać dowolną liczbę warunków typu `atrybut=wartość` (na przykład `orth=nie`) połączonych operatorami `!`, `&` i `|`, tak jak pokazują to powyższe przykłady. Możliwe jest także całkowite pominięcie jakichkolwiek warunków — poniższe zapytanie mogłoby posłużyć do znalezienia wszystkich segmentów w korpusie.[4]

(4.45) `[]`

Taka trywialna specyfikacja pozycji w korpusie, pasująca do dowolnego segmentu, może posłużyć na przykład do znalezienia dwóch form oddzielonych od siebie dowolnymi dwoma segmentami, np.:

(4.46) `[orth=się][][][base=bać]`

W wyniku tego zapytania zostaną znalezione ciągi takie jak *się nikogo nie bać*, *się Boga nie boicie* itp.

Dla wielu zastosowań ciekawsza byłaby możliwość zapytania na przykład o formy oddalone od siebie o *najwyżej* pięć pozycji. Poliqarp umożliwia zadawanie takich pytań, gdyż pozwala na formułowanie wyrażeń regularnych także na poziomie pozycji korpusu. Na przykład zapytanie o formę leksemu BAĆ występującą dwie, trzy lub cztery pozycje dalej niż forma *się* może wyglądać następująco:

[4] Mogłoby, gdyby nie fakt, że Poliqarp zawiera wewnętrzne ograniczenia na liczbę możliwych wyników zapytania.

(4.47) `[orth=się][]{2,4}[base=bać]`

W wyniku tego zapytania zostaną znalezione ciągi uzyskane w wyniku poprzedniego zapytania, a także na przykład ciąg *się każdy następny Rywin będzie bał*.

Zapewne bardziej poprawnym zapytaniem o różne wystąpienia form tzw. czasownika zwrotnego BAĆ SIĘ byłoby zapytanie o *się* w pewnej odległości przed formą leksemu BAĆ, ale bez znaku interpunkcyjnego pomiędzy tymi formami, lub bezpośrednio za taką formą, ewentualnie oddzielone od formy BAĆ zaimkiem osobowym:[5]

(4.48)
```
[orth=się][orth!="[.!?,:]"]{,5}[base=bać]
|[base=bać][base="on|ja|ty|my|wy"]?[orth=się]
```

### 4.1.4. Zapytania o znaczniki morfosyntaktyczne

Zapytanie (4.48) można uprościć poprzez zastąpienie warunku `orth!="[.!?,:]"` bezpośrednim odwołaniem do „klasy gramatycznej” interp (por. rozdz. 3):

(4.48′)
```
[orth=się][pos!=interp]{,5}[base=bać]
|[base=bać][base="on|ja|ty|my|wy"]?[orth=się]
```

Ogólniej, wartościami atrybutu `pos` (ang. *part of speech* ‘część mowy’) są skróty nazw klas gramatycznych omówionych w p. 3.4 (por. tabela na str. 34). Na przykład zapytanie o sekwencję dwóch form rzeczownikowych rozpoczynających się na *a* może być sformułowane w sposób następujący:

(4.49) `[pos=subst & orth="a.*"]{2}`

Podobnie jak to miało miejsce w wypadku specyfikacji form tekstowych i form hasłowych, także specyfikacje klas gramatycznych mogą zawierać wyrażenia regularne. Na przykład, zważywszy na to, że zaimki osobowe należą do klasy zaimków trzecioosobowych ppron3 i do klasy zaimków nietrzecioosobowych ppron12, poniższe zapytania mogą posłużyć do znalezienia dowolnych form dowolnych zaimków osobowych:

(4.50) `[pos=ppron12 | pos=ppron3]`

(4.51) `[pos="ppron12|ppron3"]`

[5] Zapytanie to zostało złamane pomiędzy dwie linie z powodów typograficznych.

(4.52) `[pos="ppron(12|3)"]`
(4.53) `[pos="ppron[123]+"]`
(4.54) `[pos="ppron.+"]`
(4.55) `[pos=ppron/x]`

A zatem zapytanie (4.48) może zostać jeszcze bardziej uproszczone do następującego zapytania:

(4.48″) `[orth=się][pos!=interp]{,5}[base=bać]`
`|[base=bać][pos=ppron/x]?[orth=się]`

W zapytaniach można określać wartości nie tylko formy wyrazowej (za pomocą atrybutu `orth`), formy hasłowej (za pomocą `base`) i klasy gramatycznej (za pomocą `pos`), ale także wartości poszczególnych kategorii gramatycznych, takich jak przypadek czy rodzaj. Służą do tego następujące atrybuty (por. p. 3.3):

| **atrybut** | **kategoria** | **możliwe wartości** |
|---|---|---|
| `number` | liczba | `sg pl` |
| `case` | przypadek | `nom gen dat acc inst loc voc` |
| `gender` | rodzaj | `m1 m2 m3 f n` |
| `person` | osoba | `pri sec ter` |
| `degree` | stopień | `pos comp sup` |
| `aspect` | aspekt | `imperf perf` |
| `negation` | zanegowanie | `aff neg` |
| `accentability` | akcentowość | `akc nakc` |
| `post-prepositionality` | poprzyimkowość | `npraep praep` |
| `accommodability` | akomodacyjność | `congr rec` |
| `agglutination` | aglutynacyjność | `agl nagl` |
| `vocalicity` | wokaliczność | `nwok wok` |

A zatem możliwe jest zadanie na przykład następujących zapytań:

(4.56) `[number=sg]`
znalezione zostaną wszystkie formy w liczbie pojedynczej,
(4.57) `[pos=subst & number=sg]`
znalezione zostaną formy rzeczownikowe w liczbie pojedynczej,
(4.58) `[pos=subst & gender!=f]`
formy rzeczownikowe rodzaju męskiego lub nijakiego,

(4.59) `[number=sg & case="nom|acc" & gender="m[123]"]`
pojedyncze mianownikowe lub biernikowe formy męskie.

Zamiast pełnych nazw atrybutów można używać ich trzyliterowych skrótów:

| atrybut | skrót |
|---|---|
| number | nmb |
| case | cas |
| gender | gnd |
| person | per |
| degree | deg |
| aspect | asp |
| negation | neg |
| accommodability | acm |
| accentability | acn |
| post-prepositionality | ppr |
| agglutination | agg |
| vocalicity | vcl |

A zatem zapytanie (4.59) można zadać także w sposób następujący:

(4.59′) `[nmb=sg & cas="nom|acc" & gnd="m[123]"]`

Wersje graficzna i tekstowa umożliwiają zdefiniowanie tzw. aliasów, czyli skrótów alternatyw wartości danego atrybutu, których następnie można używać jak zwykłych wartości atrybutów. W obecnej wersji Korpusu IPI PAN predefiniowane są cztery takie aliasy:

| alias | definicja |
|---|---|
| masc | m1 m2 m3 |
| noun | subst depr ger xxs ppron12 ppron3 |
| pron | ppron12 ppron3 siebie |
| verb | fin praet aglt bedzie inf imps impt pact ppas pcon pant ger winien |

Przy tak zdefiniowanych aliasach `noun` i `masc`, poniższe dwa zapytania są równoważne:

(4.60) `[pos=noun & gender=masc]`

(4.61) `[pos="subst|depr|ger|xxs|ppron12|ppron3" & gender="m1|m2|m3"]`

O klasy gramatyczne i kategorie gramatyczne można także pytać łącznie, używając do tego atrybutu `tag`. Na przykład, aby znaleźć wszystkie rzeczowniki nijakie w mianowniku o pojedynczej wartości liczby, można zadać następujące zapytanie:

(4.62) `[tag=subst:sg:nom:n]`

Wartości atrybutu `tag` mają postać $\texttt{kl:kat}_1\texttt{:kat}_2\texttt{:...:kat}_n$, gdzie `kl` to nazwa klasy gramatycznej, a $\texttt{kat}_i$ to wartości kategorii przysługujących tej klasie w kolejności, w jakiej zostały podane w tabeli na str. 30 (p. 3.4.2).

Tak jak w wypadku innych atrybutów, specyfikacja atrybutu `tag` może być zadana wyrażeniem regularnym, np.:

(4.59″) `[tag=".*:sg:(nom|acc):m[123].*"]`

(4.59‴) `[tag="sg:(nom|acc):m[123]"/x]`

### 4.1.5. Wieloznaczności i dezambiguacja

Jedną z cech, które wyróżniają Korpus IPI PAN i program Poliqarp, jest możliwość reprezentacji wieloznaczności morfosyntaktycznych, zgodnie z postulatem wyrażonym m.in. w pracy Oliva 2001. Jak zostało powiedziane w p. 2.2, istnieją konstrukcje, w których nie jest możliwe ujednoznacznienie interpretacji danej formy, na przykład dlatego, że kilka możliwych interpretacji prowadzi do identycznego znaczenia całego zdania. Ilustruje to przykład (2.5), z p. 2.2, powtórzony poniżej.

(2.5) Pamiętam ją pijaną.

(2.6) a. Pamiętam go pijanego.
b. Pamiętam go pijanym.

W przykładzie tym nie można stwierdzić, czy forma *pijaną* jest formą biernikową, jak *pijanego* w (2.6a), czy też formą narzędnikową, jak *pijanym* w (2.6). Dlatego też w korpusie segment *pijaną* w (2.5) powinien dostać dwie interpretacje: biernikową i narzędnikową.

W takim wypadku powstaje jednak pytanie, czy segment *pijaną* w (2.5) powinien zostać znaleziony w wypadku zadania zapytania typu `[case=acc]`, czy też nie: za odpowiedzią pozytywną przemawia fakt, że interpretacja biernikowa jest poprawną interpretacją tego segmentu w tym kontekście, zaś za odpowiedzią negatywną — fakt, że nie można z całkowitą pewnością uznać tej formy za formę biernikową.

Zamiast narzucać jedną z tych interpretacji, Poliqarp pozwala na zadanie obu rodzajów zapytań. Wynikiem zapytania `[case=acc]` będzie znalezienie tych segmentów, w wypadku których przynajmniej jedna z interpretacji poprawnych w danym kontekście jest interpretacją biernikową, a zatem, w wypadku zdania (2.5), zostanie znaleziony zarówno segment *pijaną*, jak i segment *ją*, zaś w wypadku zdania (2.6a) — zarówno *pijanego*, jak i *go*. Aby natomiast znaleźć te segmenty, które są biernikowe z całą pewnością, tj. których każda poprawna w danym kontekście interpretacja jest biernikowa, należy posłużyć się zapytaniem `[case==acc]`. Wynikiem tego ostatniego zapytania w wypadku zdania (2.5) będzie znalezienie formy *ją*, ale nie formy *pijaną*, zaś w wypadku zdania (2.6a) zostaną ponownie znalezione obie formy *go* i *pijanego* — choć są to formy synkretyczne posiadające także interpretacje dopełniaczowe, w kontekście zdania (2.6a) powinny zostać zdezambiguowane do interpretacji biernikowej.

Dzięki temu rozróżnieniu możliwe jest na przykład wyszukiwanie form, które mogą być w danym kontekście interpretowane jako biernikowe lub dopełniaczowe (por. (2.3) na str. 14), a zatem w poprawnie oznakowanym korpusie poniższe zapytanie może dać niepuste wyniki.

(4.63) `[case=acc & case=gen]`

Z drugiej strony, zapytanie przedstawione poniżej, jako zapytanie o te formy, których wszystkie poprawne w danym kontekście interpretacje są jednocześnie biernikowe i dopełniaczowe, oczywiście da wynik pusty.

(4.64) `[case==acc & case==gen]`

Powyższe zapytania dotyczą interpretacji segmentów uzyskanych w procesie dezambiguacji, tj. w wyniku ujednoznacznienia (a raczej *unietakwieloznacznienia*) interpretacji morfosyntaktycznych. Jak zostało powiedziane w p. 2.2, w Korpusie IPI PAN zachowane są jednak wszystkie interpretacje danego segmentu znane analizatorowi morfologicznemu, a nie tylko te, które zostały uznane za poprawne w danym kontekście.

W pewnych sytuacjach przydatny jest dostęp do tych odrzuconych interpretacji, na przykład wtedy, gdy chcemy znaleźć wszystkie występujące w korpusie formy synkretyczne o możliwych interpretacjach biernikowej i dopełniaczowej, niezależnie od tego, które z tych interpretacji są poprawne w danym kontekście, albo gdy chcemy zignorować potencjalnie błędne wyniki działania dezambiguatora. W pierwszym z tych wypadków posłużyć się możemy następującym zapytaniem:

(4.65) `[case~acc & case~gen]`

W wyniku tego zapytania znalezione zostaną synkretyczne formy *go* i *pijanego*, ale nie formy *ją* i *pijaną*, które takiego synkretyzmu nie wykazują.

Ostatnim rodzajem równości dostępnym w zapytaniach jest `~~`. Zapytanie poniższe może posłużyć do znalezienia tych form, o których jeszcze przed ujednoznacznianiem wiadomo, że są wyłącznie biernikowe:

(4.66) `[case~~acc]`

Poniższa tabela podsumowuje znaczenie poszczególnych operatorów równości.

| | **w wynikach analizy morfologicznej** | **w wynikach dezambiguacji** |
|---|---|---|
| **co najmniej jedna interpretacja** | `~` | `=` |
| **każda interpretacja** | `~~` | `==` |

A zatem prawdziwe są następujące implikacje:

— `[atrybut~~wartość]` → `[atrybut==wartość]`
tj. każda odpowiedź na zapytanie `[atrybut~~wartość]` jest także odpowiedzią na zapytanie `[atrybut==wartość]`
— `[atrybut==wartość]` → `[atrybut=wartość]`
tj. każda odpowiedź na zapytanie `[atrybut==wartość]` jest także odpowiedzią na zapytanie `[atrybut=wartość]`
— `[atrybut=wartość]` → `[atrybut~wartość]`
tj. każda odpowiedź na zapytanie `[atrybut=wartość]` jest także odpowiedzią na zapytanie `[atrybut~wartość]`

### 4.1.6. Ograniczenie zapytania do zdania lub akapitu

Tekst zawarty w Korpusie IPI PAN został automatycznie podzielony na zdania i na akapity. Informację tę można wykorzystać w zapytaniach, na przykład ograniczając dopasowanie do jednego zdania.

Aby ograniczyć zasięg zapytania, należy dopisać do zapytania słowo kluczowe `within`, a po nim `s` lub `p`, w zależności od tego, czy zasięg ma być ograniczony do zdania (ang. *sentence*) czy do akapitu (ang. *paragraph*). Ilustruje to następujący przykład zapytania o zdania, w których forma *się* występuje za formą leksemu BYĆ, lecz nie bezpośrednio za nią:

(4.67) `[base=bać][orth!=się]+[orth=się] within s`

### 4.1.7. Ograniczenie zapytania za pomocą metadanych

Z każdym utworem znajdującym się w Korpusie IPI PAN związane są tzw. metadane, czyli informacje o tytule i autorze utworu, jego pochodzeniu, wydawcy itp. Część z tych informacji jest dostępna za pomocą programu Poliqarp i może zostać użyta do ograniczenia zasięgu zapytania, na przykład do tekstów danego autora lub tekstów powstałych w danym przedziale czasowym.

W obecnej wersji Korpusu IPI PAN dostępne są trzy rodzaje metadanych, którym odpowiada pięć metaatrybutów:

— imię i nazwisko autora lub autorów — atrybut `author`,
— tytuł utworu — atrybut `title`,
— data powstania lub publikacji utworu — atrybuty:
  — `created` — kiedy utwór powstał,
  — `first_published` — kiedy został opublikowany po raz pierwszy,
  — `published` — kiedy została opublikowana wersja znajdująca się w korpusie.

Zwykle tylko niektóre z tych atrybutów będą miały przypisaną wartość, na przykład wtedy, gdy znana jest tylko data publikacji, a nie data pierwszej publikacji i powstania, albo w wypadku krótkich notatek prasowych, których autor — a czasami i tytuł — nie są znane.

Aby ograniczyć zasięg zapytania do określonych wartości metadanych, należy na końcu zapytania wpisać słowo kluczowe `meta`, po którym następuje specyfikacja wartości metaatrybutów. W wypadku, gdy zapyta-

nie jest ograniczone zarówno do zdania czy akapitu, jak i na podstawie metadanych, specyfikacja ograniczenia do zdania lub akapitu występuje w zapytaniu przed specyfikacją metadanych, na przykład:

(4.68) `[pos=subst]{6,} within s meta author=Kowalski`

Wartości atrybutów `author` i `title` mogą być zadane wyrażeniami regularnymi, np. poniższe zapytanie może posłużyć do znalezienia form leksemu WIRUS w utworach, których tytuł zawiera jeden z napisów *windows* lub *microsoft*.

(4.69) `[base=wirus] meta title="windows|microsoft"`

Specyfikacje atrybutów `author` i `title` są rozumiane w sposób bezkasztowy (wielkość liter nie ma znaczenia) i są interpretowane jako części pełnych wartości, a zatem wyniki poniższego zapytania będą pochodziły z utworów m.in. Pola, Polkowskiego i Rampolskiego (o ile ci autorzy są reprezentowani w korpusie):

(4.70) `[pos=subst]{5} meta author=Pol`

Flagi `/X` i `/I` działają w sposób dualny do opisanych powyżej flag `/x` i `/i`: flaga `/X` powoduje, że dana specyfikacja atrybutu jest rozumiana jako specyfikacja pełnej wartości tego atrybutu, a nie tylko jego części, zaś flaga `/I` powoduje, że kasztowość specyfikacji jest dopasowywana do kasztowości wartości atrybutu. Ilustrują to poniższe przykłady:

(4.71) `[pos=subst]{5} meta author=Pol/I`
przeszukiwanie ograniczone do utworów Pola, Polkowskiego itd., ale nie Rampolskiego

(4.72) `[pos=subst]{5} meta author="Marek Pol"/X`
przeszukiwanie ograniczone do utworów Marka Pola

(4.73) `[pos=subst]{5} meta author=Pol/X`
przeszukiwanie ograniczone do utworów autora, którego pełna specyfikacja imienia i nazwiska to *Pol* (lub *POL*, *pol* itd.)

(4.74) `[pos=subst]{5} meta author=".* Pol"/I`
przeszukiwanie ograniczone do utworów autora, którego nazwisko to *Pol*

Odmiennie traktowane są atrybuty `created`, `first_published` i `published`, którego wartościami mogą być wyłącznie daty. W wypadku atrybutów tego typu nie można użyć wyrażeń regularnych, natomiast można użyć znaków < i > w celu określenia przedziału czasowego, z jakiego dany utwór ma pochodzić, na przykład:

(4.75) `[pos=subst]{5} meta created>1950`
przeszukiwanie ograniczone do utworów powstałych po roku 1950

Warunki dotyczące metadanych można łączyć za pomocą operatorów `&`, `|` i `!`:

(4.76) `[pos=subst]{5} meta created>=1951 & created<=1960`
przeszukiwanie ograniczone do utworów powstałych w latach 1951–1960

(4.77) `[pos=subst]{5} meta published>1900 & author!=Sienkiewicz`
przeszukiwanie ograniczone do utworów wydanych po roku 1900, z wyjątkiem tekstów Sienkiewicza,

(4.78) `[pos=subst]{5} meta (author=sienkiewicz & title=potop) | (author=żeromski & title=przedwiośnie)`
przeszukiwanie ograniczone do *Potopu* Sienkiewicza i *Przedwiośnia* Żeromskiego

W obecnej wersji korpusu wiele utworów nie posiada kompletnych metadanych. Efektem powyższych zapytań, w których podane są ograniczenia na metaatrybuty, będzie znalezienie tylko tych utworów, dla których stosowne metaatrybuty mają przypisane właściwe wartości. A zatem wynikiem pierwszego z poniższych zapytań będzie znalezienie znacznie mniejszej liczby wyników niż w wypadku drugiego zapytania.

(4.79) `[pos=subst] meta published>1900 | published<=1900`
(4.80) `[pos=subst]`

W wypadku, gdy dany metaatrybut nie ma przypisanej odpowiedniej wartości, przyjmuje się, że jego wartością jest pusty ciąg, tj. `""`. A zatem poniższe zapytanie równoważne jest drugiemu z zapytań powyżej.

(4.81) `[pos=subst] meta published>1900 | published<=1900 | published=""`

### 4.1.8. Wyrównywanie wyników

W części zapytania dotyczącej form, tj. przed kwalifikatorami `within` i `meta`, można używać znacznika wyrównania `^`, na przykład:

(4.82) `[pos=adj & case=nom]+^[pos=subst & case=nom]+`

W wyniku zadania powyższego zapytania wyświetlone zostaną cztery kolumny zawierające, odpowiednio, lewy kontekst, tj. fragment tekstu bezpośrednio poprzedzający znalezione dopasowanie, lewe dopasowanie, tj. fragment tekstu pasujący do części zapytania przed znakiem `^` (tutaj: ciąg przymiotników w mianowniku), prawe dopasowanie (tutaj: ciąg rzeczowników w mianowniku) oraz prawy kontekst, jak zostało to pokazane na Rys. 4.1.

## 4.2. Poliqarp

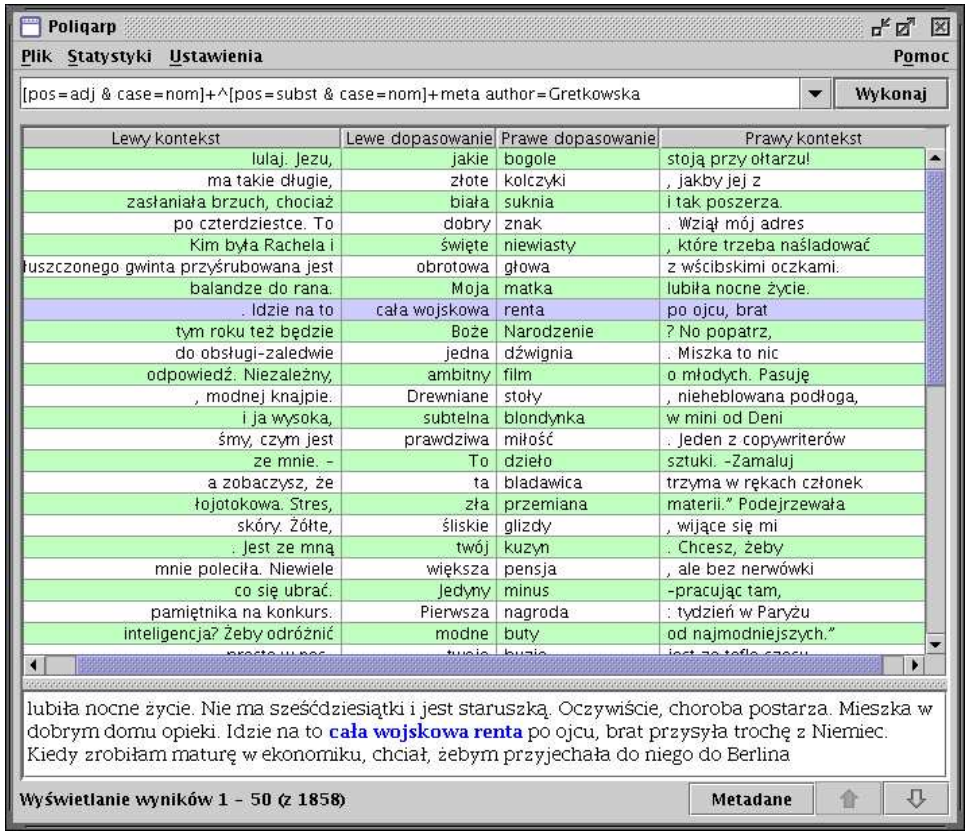
Niniejszy punkt zawiera omówienie trzech wersji interfejsu do programu Poliqarp, począwszy od wersji najuboższej, internetowej (p. 4.2.1), poprzez wersję graficzną (p. 4.2.2), przeznaczoną dla znacznej większości użytkowników, a skończywszy na linuksowej wersji tekstowej (p. 4.2.3), o największych możliwościach i najszybszej, ale posiadającej najmniej atrakcyjny interfejs użytkownika.

### 4.2.1. Wersja internetowa

W wersji sieciowej Korpus IPI PAN dostępny jest pod adresem internetowym `korpus.pl`. W chwili obecnej korpus dostępny jest bez ograniczeń, choć być może ograniczenia w korzystaniu z korpusu zostaną wprowadzone, jeżeli popularność tej usługi będzie większa niż pozwala na to moc serwera przeznaczonego do obsługi korpusu.

Choć funkcjonalność obecnej wersji internetowej jest dosyć ograniczona w stosunku do wersji graficznej i tekstowej, nie jest ograniczona w stosunku do opisu w p. 4.1 składnia zapytań. Bezpośrednio po połączeniu się z adresem `korpus.pl` i po wybraniu korpusu, wyświetlony jest pasek zapytań, jak na Rys. 4.2.

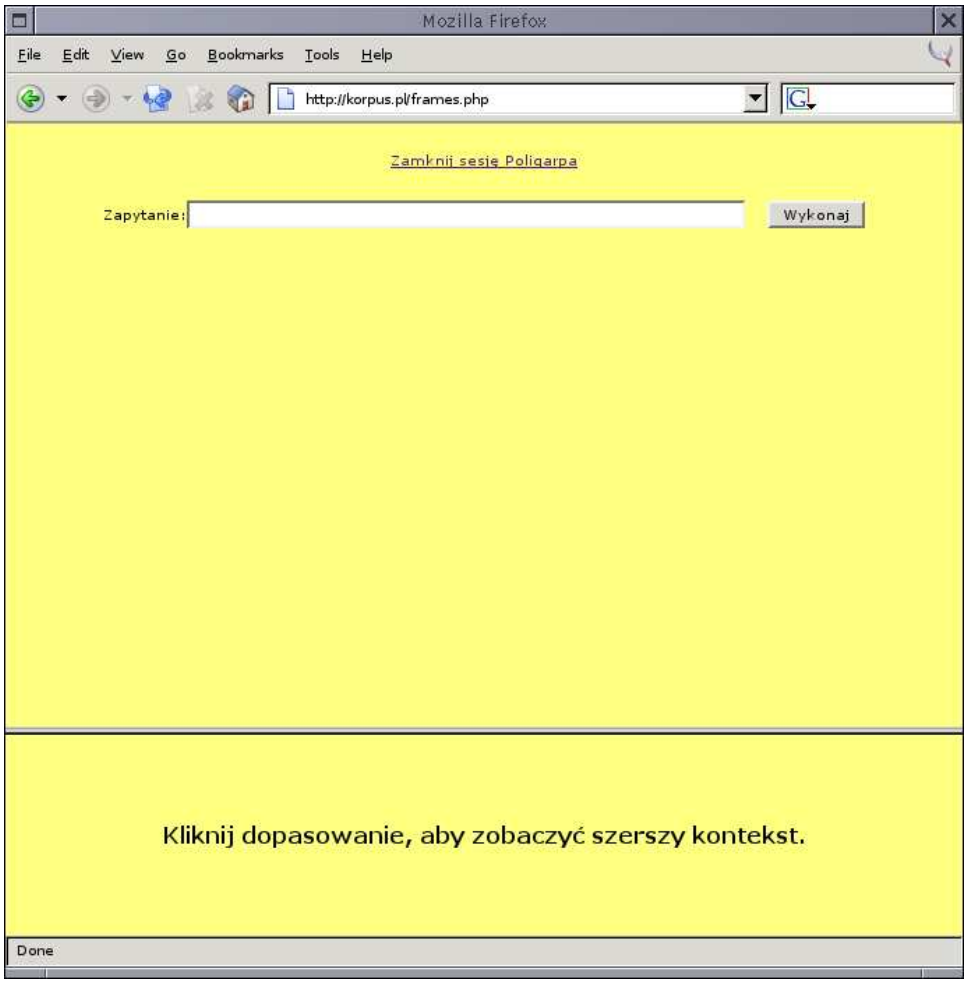
W pasku tym można wpisać dowolne zapytanie zgodne z opisem w p. 4.1, a następnie nacisnąć Enter lub kliknąć przycisk Wykonaj. Wyni-




Rysunek 4.1. Wyrównywanie wyników w wersji graficznej

ki zostaną wyświetlone w trzech lub czterech kolumnach, w zależności od tego, czy w zapytaniu był obecny znacznik wyrównania, por. Rys. 4.3. Kliknięcie jednego z wyników przeszukiwania, a ściślej centralnej części wyniku odpowiadającej dopasowaniu do zapytania, spowoduje wyświetlenie szerszego kontekstu w dolnej części strony, jak to pokazano na Rys. 4.4. Wyniki przeszukiwania pokazywane są w partiach po 25 wyników. Aby przejść do następnych 25 wyników, należy kliknąć napis Następne 25, a aby wrócić do poprzednich wyników — napis Poprzednie 25.

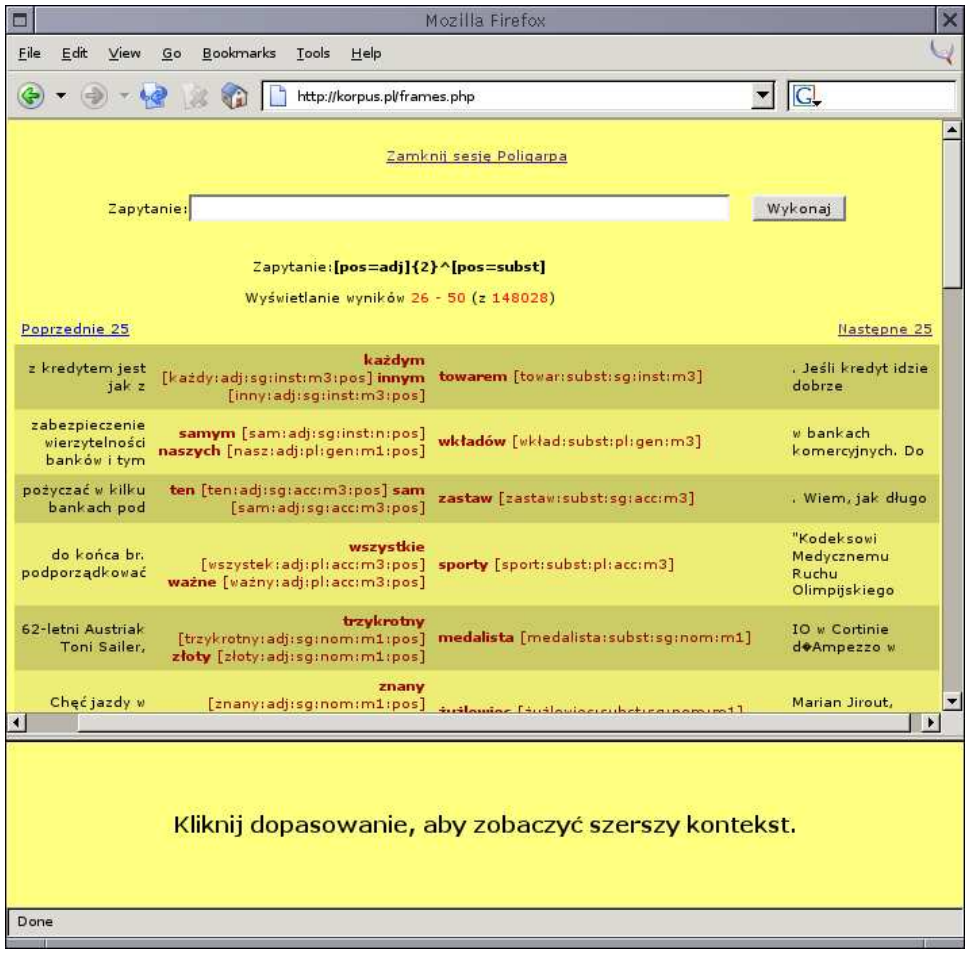
Jedyną dodatkową funkcjonalnością dostępną w obecnej wersji sieciowej jest możliwość posortowania wyników według dowolnej kolumny, al-



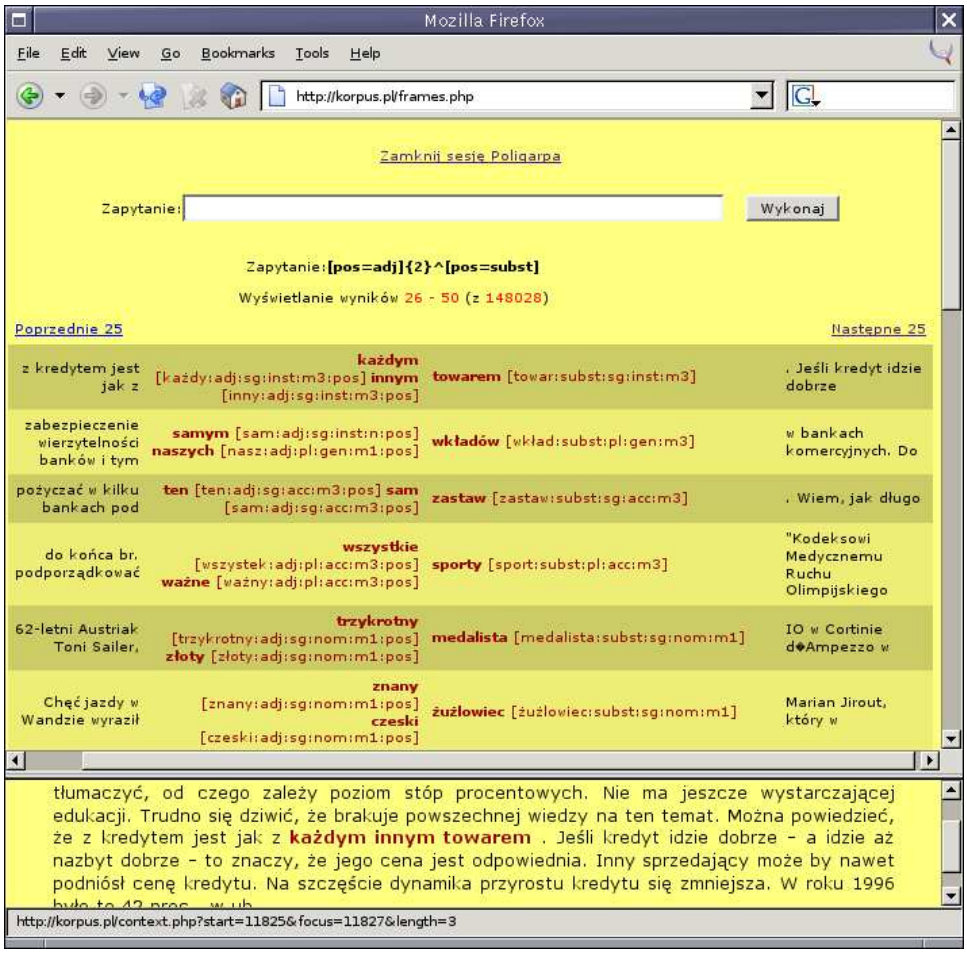

Rysunek 4.2. Wersja sieciowa bezpośrednio po wybraniu korpusu

fabetycznie *a fronte* (od początku wartości) lub *a tergo* (od końca wartości), w porządku rosnącym lub malejącym.

Wersja sieciowa przeszukiwarki Poliqarp jest wersją obecnie najszybciej się zmieniającą, możliwe jest więc, że w chwili ukazania się niniejszej publikacji będzie się ona istotnie różnić od postaci tutaj opisanej.




Rysunek 4.3. Wersja sieciowa: wyniki wyszukiwania

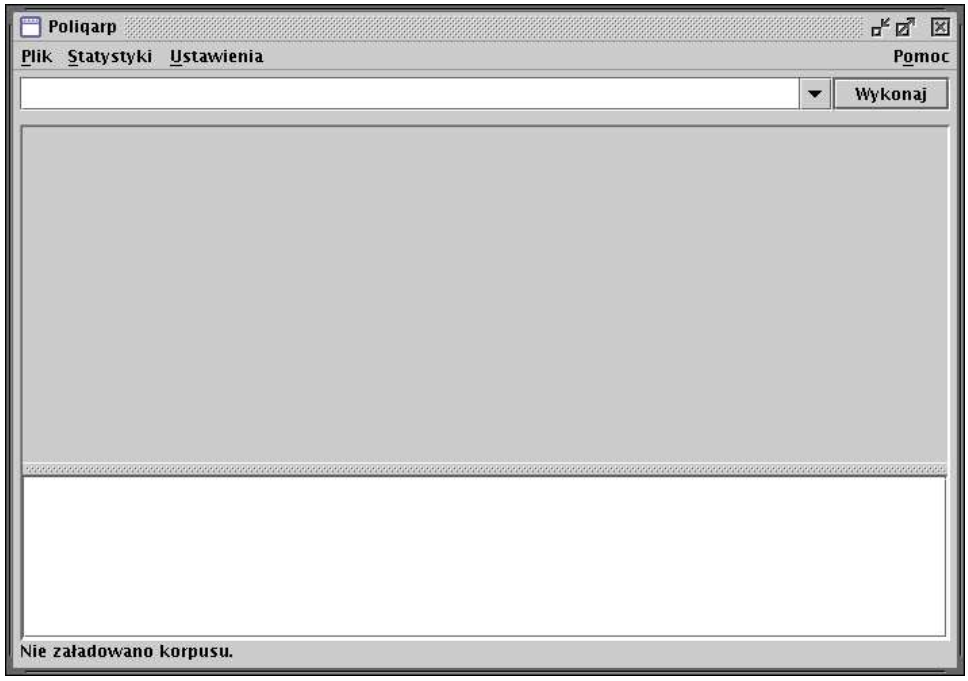



Rysunek 4.4. Wersja sieciowa: szerszy kontekst

### 4.2.2. Wersja graficzna

Podstawową wersją przeszukiwarki Poliqarp jest wersja graficzna. Oprócz menu, okienko wersji graficznej zawiera pasek zapytań, górną część służącą do wyświetlania wyników zapytania, oraz dolną część zawierającą bardziej szczegółowe informacje o zaznaczonym wyniku: szerszy kontekst, tak jak na Rys. 4.1 na str. 59, lub metadane.

Bezpośrednio po uruchomieniu przeszukiwarki, pasek zapytań i obie części okna są puste, jak na Rys. 4.5. Zanim możliwe będzie przeszukiwa-

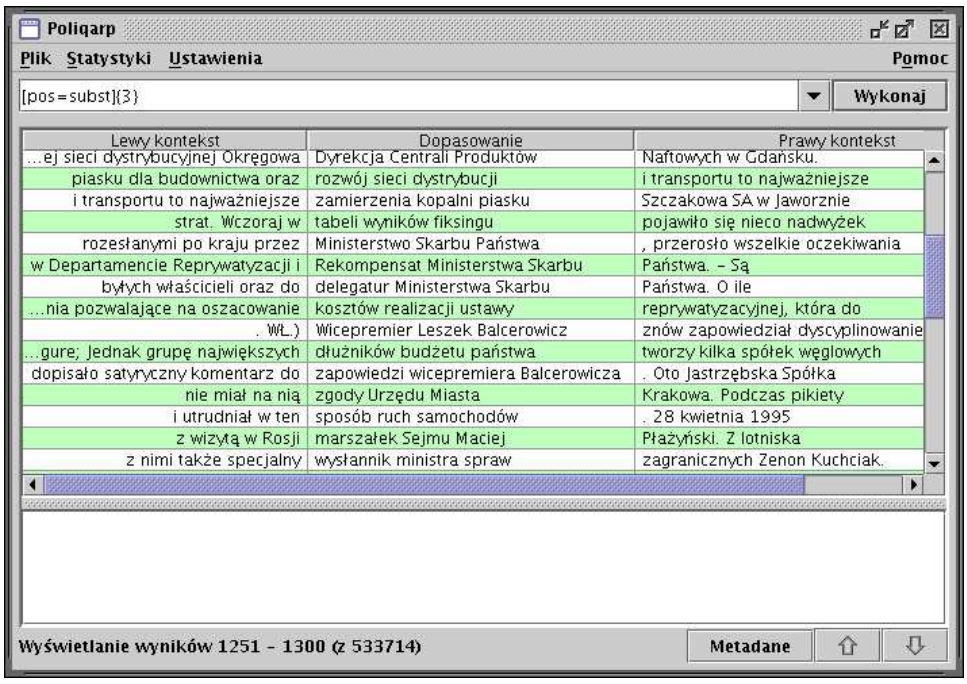



Rysunek 4.5. Wersja graficzna bezpośrednio po uruchomieniu

nie korpusu, należy najpierw korpus otworzyć ('załadować'), wybierając w menu Plik i dalej Otwórz korpus..., a następnie lokalizując katalog z Korpusem IPI PAN na płycie CD-ROM lub na dysku, w zależności od tego, gdzie korpus został zainstalowany (por. dodatek A). W ten sam sposób można zmienić bieżący korpus, bez potrzeby wychodzenia z programu. Przy wczytywaniu korpusu zapamiętywana jest informacja o jego lokali-

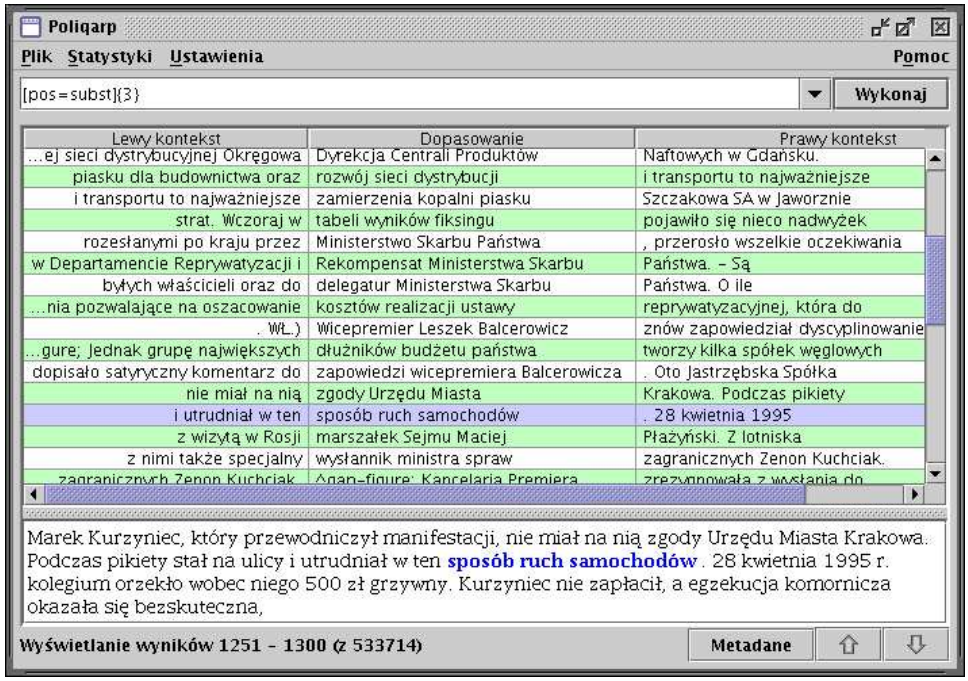
zacji — przy ponownym uruchomieniu programu Poliqarp wystarczy wybrać w menu Plik i dalej Ostatnio używane lub nacisnąć kombinację klawiszy Ctrl-Alt-1 w celu wczytania pierwszego z zapamiętanych korpusów, Ctrl-Alt-2, aby wczytać drugi z zapamiętanych korpusów itd.

Po wczytaniu korpusu można wpisać zapytanie w pasku zapytań i uruchomić wyszukiwanie poprzez naciśnięcie Enter lub kliknięcie przycisku Wykonaj. Jeżeli zapytanie nie zawierało znacznika wyrównania wyników, w górnej części okna pokazane zostaną wyniki przeszukiwania w postaci trzech kolumn zawierających, odpowiednio, lewy kontekst, dopasowanie i prawy kontekst, tak jak pokazuje to Rys. 4.6, przy czym szerokość kolumn można zmieniać, przeciągając myszką granice nagłówków kolumn. W wypadku zadania zapytania ze znacznikiem wyrównania ^ pokazane zostaną cztery kolumny, tak jak zostało to wyjaśnione w p. 4.1.8.



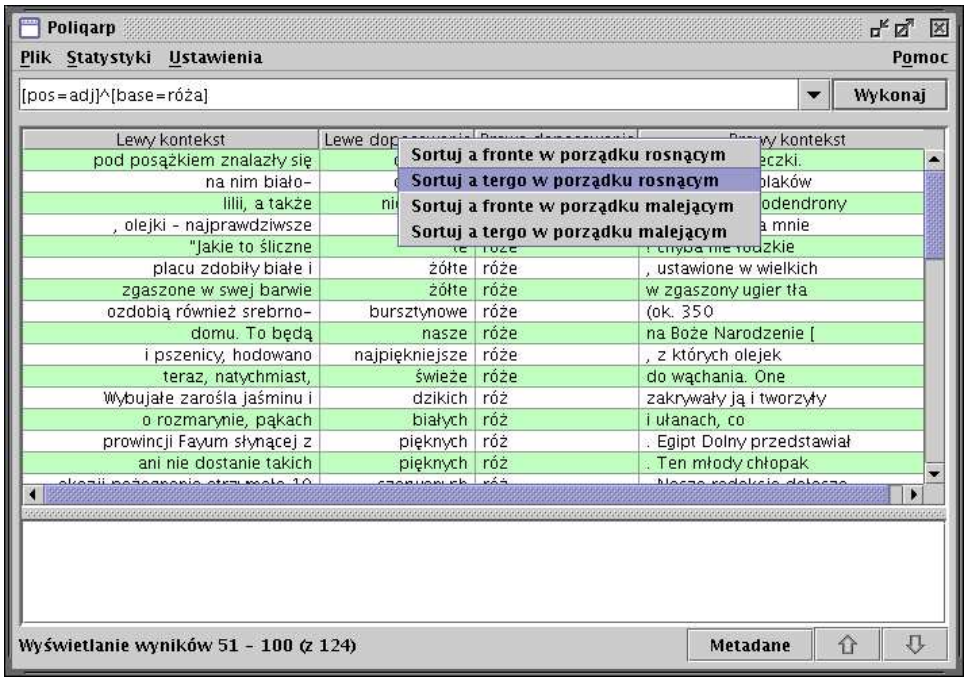

Rysunek 4.6. Wyniki zapytania bez wyrównywania

Program Poliqarp przechowuje historię zapytań zadanych w bieżącej i w poprzednich sesjach przeszukiwania korpusu. Historia zapytań dostępna jest m.in. za pomocą małego przycisku znajdującego się pomiędzy paskiem zapytań a przyciskiem Wykonaj. W wyniku zadania zapytania w górnej części okna programu Poliqarp pokazanych zostanie jedynie pierwsze pięćdziesiąt wyników. Aby obejrzeć następne wyniki, należy kliknąć strzałkę w dół znajdującą się w prawym dolnym rogu okna. Powrót do poprzednich wyników możliwy jest poprzez kliknięcie znajdującej się na lewo od niej strzałki w górę.




Rysunek 4.7. Szerszy kontekst wyniku przeszukiwania

Aby zobaczyć szerszy kontekst wybranego wyniku, należy kliknąć odpowiadający mu wiersz w górnej części okna. W wyniku tego wiersz zostanie zaznaczony innym kolorem, zaś szerszy kontekst zostanie pokazany w dolnej części okna. W dolnej części prezentowane są także informacje o utworze, z którego dany fragment pochodzi, czyli tzw. metadane, zaś do




Rysunek 4.8. Sortowanie *a tergo* w porządku rosnącym

przechodzenia pomiędzy trybem prezentacji kontekstu a trybem prezentacji metadanych służy przycisk na dole okna oznaczony Metadane w trybie prezentacji kontekstu, zaś Kontekst — w trybie prezentacji metadanych.

Wyniki przeszukiwania korpusu można sortować po wartościach każdej z kolumn. W najprostszym wypadku, aby posortować wyniki alfabetycznie w porządku rosnącym, należy kliknąć nagłówek odpowiedniej kolumny lewym przyciskiem myszy, na przykład kolumny Dopasowanie (w wypadku zapytania bez znacznika wyrównania). Ponowne kliknięcie spowoduje posortowanie wyników według wartości tej kolumny, lecz w porządku malejącym, zaś kolejne — znowu posortowanie w porządku rosnącym.

Możliwe jest także sortowanie *a tergo*, tj. od końca wartości, zarówno w porządku rosnącym, jak i malejącym. Funkcja ta jest dostępna z menu, które pojawia się po kliknięciu nagłówka odpowiedniej kolumny prawym klawiszem myszki, tak jak zostało to pokazane na Rys. 4.8 ilustrującym

sortowanie *a tergo* w porządku rosnącym po wartościach kolumny Lewe dopasowanie.

Sposób sortowania wyników można ustawić także w menu Ustawienia, wybierając Opcje... a następnie zakładkę Sortowanie. Wartości wybrane w ten sposób są stosowane przy prezentacji wyników kolejnych zapytań oraz zapamiętywane na potrzeby następnych sesji.

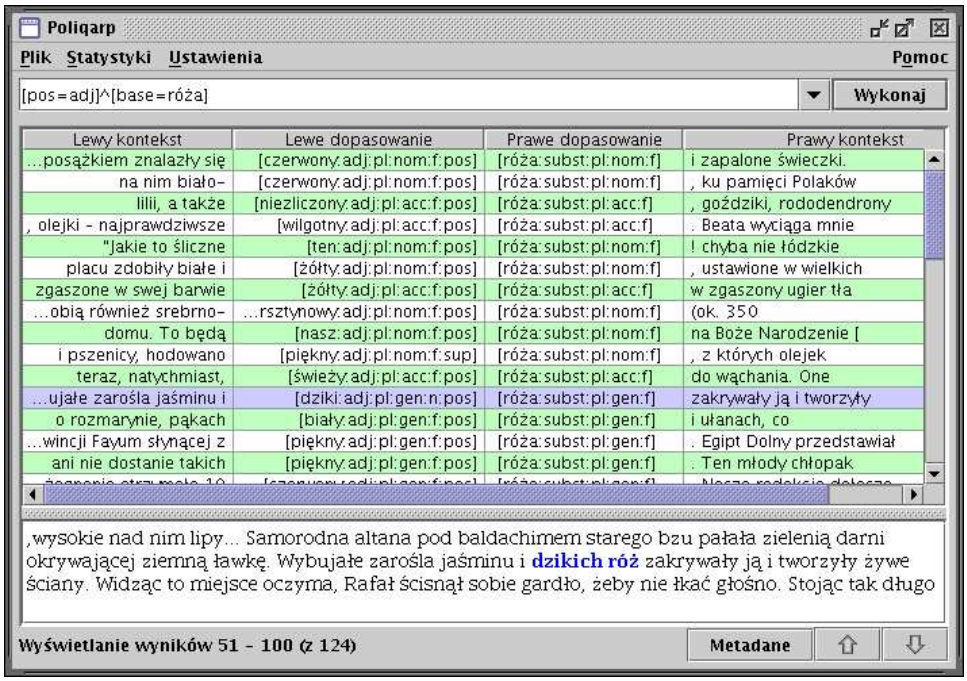
Wśród innych opcji, które można ustawić z menu Ustawienia → Opcje..., jest liczona w segmentach szerokość kontekstu, jaki będzie prezentowany dla każdego znalezionego wyniku zapytania: zarówno szerokość lewego i prawego kontekstu w górnej części okna, jak i większego kontekstu w dolnej części okna (zakładka Kontekst).

Ważną cechą programu Poliqarp jest możliwość wyświetlania nie tylko form wyrazowych występujących w korpusie, lecz także ich form podstawowych i znaczników morfosyntaktycznych. W obecnej wersji programu opcje wyświetlania form wyrazowych, form podstawowych i tagów można określić oddzielnie dla kolumn z dopasowaniem i oddzielnie dla kolumn z kontekstem. Na przykład efektem ustawień z Rys. 4.9 będzie formatowanie wyników pokazane na Rys. 4.10.


Rysunek 4.9. Ustawienia formatowania wyników

Jak zostało powiedziane powyżej (str. 51), wersja graficzna przeszukiwarki Poliqarp umożliwia zdefiniowanie tzw. aliasów, czyli skrótów dla alternatyw wartości danego atrybutu. Do edycji aliasów, w tym do ich dodawania, służy zakładka Aliasy w Ustawienia → Opcje... W obecnej wersji programu aliasy dodane w trakcie jednej sesji nie są zapamiętywane na potrzeby sesji następnych.

Kolejna zakładka, Flagi, umożliwia określenie, jak mają być interpretowane podane przez użytkownika wartości atrybutów w części zapytania




Rysunek 4.10. Wyniki formatowane według ustawień z Rys. 4.9

dotyczącej szukanych form oraz w części zapytania dotyczącej metadanych. Domyślnie zapytanie typu `[pos=ppro]` znajdzie te segmenty, których klasa gramatyczna to ppro, a zatem wynik takiego zapytania będzie pusty. Aby znaleźć różne formy zaimkowe, a zatem takie, których klasa gramatyczna to ppron12 lub ppron3, można użyć flagi `/x`, tak jak zostało to opisane powyżej (str. 45). Aby natomiast zmienić zachowanie domyślne programu w taki sposób, by interpretował wszystkie zapytania o formy tak, jakby miały dodaną flagę `/x`, należy ustawić wartość opcji Tylko całe słowa w kolumnie Zapytanie w zakładce Flagi tak, by opcja ta nie była zaznaczona.

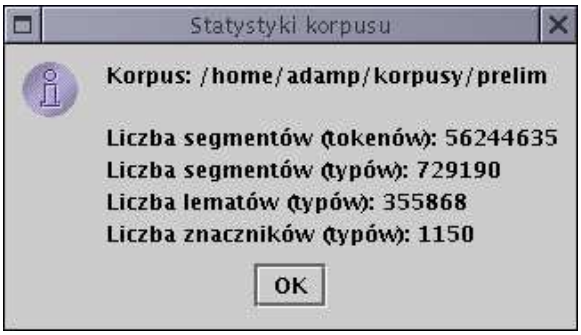
W analogiczny sposób można zmienić interpretację specyfikacji metaatrybutów. Jak zostało to opisane na str. 56, domyślnie specyfikacja metaatrybutu powinna pasować przynajmniej do części wartości metaatrybutu, zaś zachowanie to zmienić można na potrzeby danego zapytania za pomo-

cą flagi /X. Aby zachowanie to zmienić na stałe, należy zaznaczyć opcję Tylko całe słowa w kolumnie Metadane zakładki Flagi.

Podobnie, za pomocą odpowiednich ustawień opcji Ignoruj wielkość liter, mogą zostać zmienione standardowe interpretacje kasztowości w obu częściach zapytań.

Kolejną zakładką w menu Ustawienia → Opcje... jest zakładka Metadane. Za jej pomocą można ograniczyć kolejne zapytania do z góry zadanych wartości metaatrybutów, na przykład do utworów danego autora lub utworów opublikowanych w danym przedziale czasowym. Ostatnia zakładka, Wygląd, pozwala ustawić rozmiar czcionki.

Oprócz podmenu Plik i podmenu Ustawienia, w głównym menu programu Poliqarp dostępne jest także podmenu Statystyki. W obecnej wersji programu wybranie tego podmenu spowoduje wyświetlenie podstawowych informacji liczbowych o bieżącym korpusie. Na przykład podkorpus Korpusu IPI PAN, którego dotyczą informacje pokazane na Rys. 4.11, zawiera ponad 56 milionów pozycji, na które składa się ponad 729 tysięcy różnych segmentów, będących formami ponad 355 tysięcy różnych form hasłowych. Liczba różnych znaczników morfosyntaktycznych, rozumianych jako ciągi $\texttt{kl:kat}_1\texttt{:kat}_2\texttt{:...:kat}_n$ (por. str. 52) w tej wersji korpusu to 1150.




Rysunek 4.11. Informacje liczbowe o bieżącym korpusie

Do zapisywania wyników przeszukiwania służy pozycja Zapisz wyniki... w menu Plik, zaś do kończenia pracy z programem — pozycja Wyjście w tym samym menu.

### 4.2.3. Wersja tekstowa

Wersja tekstowa programu Poliqarp jest przeznaczona dla systemu operacyjnego GNU/Linux działającego na komputerach klasy PC.

Program można uruchomić za pomocą polecenia `poliqarp korpus`, gdzie `korpus` to nazwa korpusu (pierwszy człon nazw plików `wstepny.cfg` i `wstepny.poliqarp.corpus.image`), łącznie ze ścieżką do tego korpusu. Na przykład, jeżeli korpus znajduje się w katalogu `./korpus/` i składa się z plików typu `wstepny.cfg`, `wstepny.poliqarp.chunk.image` itp., program można wywołać w sposób następujący:

(4.83) `$ poliqarp korpus/wstepny`

W wyniku wykonania powyższego polecenia, linia poleceń powłoki zostanie zastąpiona poprzez linię poleceń programu Poliqarp, np.:

(4.84) `KORPUS/WSTEPNY>`

Linia poleceń programu Poliqarp służy do wpisywania zapytań i innych komend. Aby zadać zapytanie, wystarczy je wpisać w linii poleceń i nacisnąć klawisz Enter. Na przykład w wyniku zapytania pokazanego w (4.85) zostaną wyświetlone wyniki jak na Rys. 4.12.

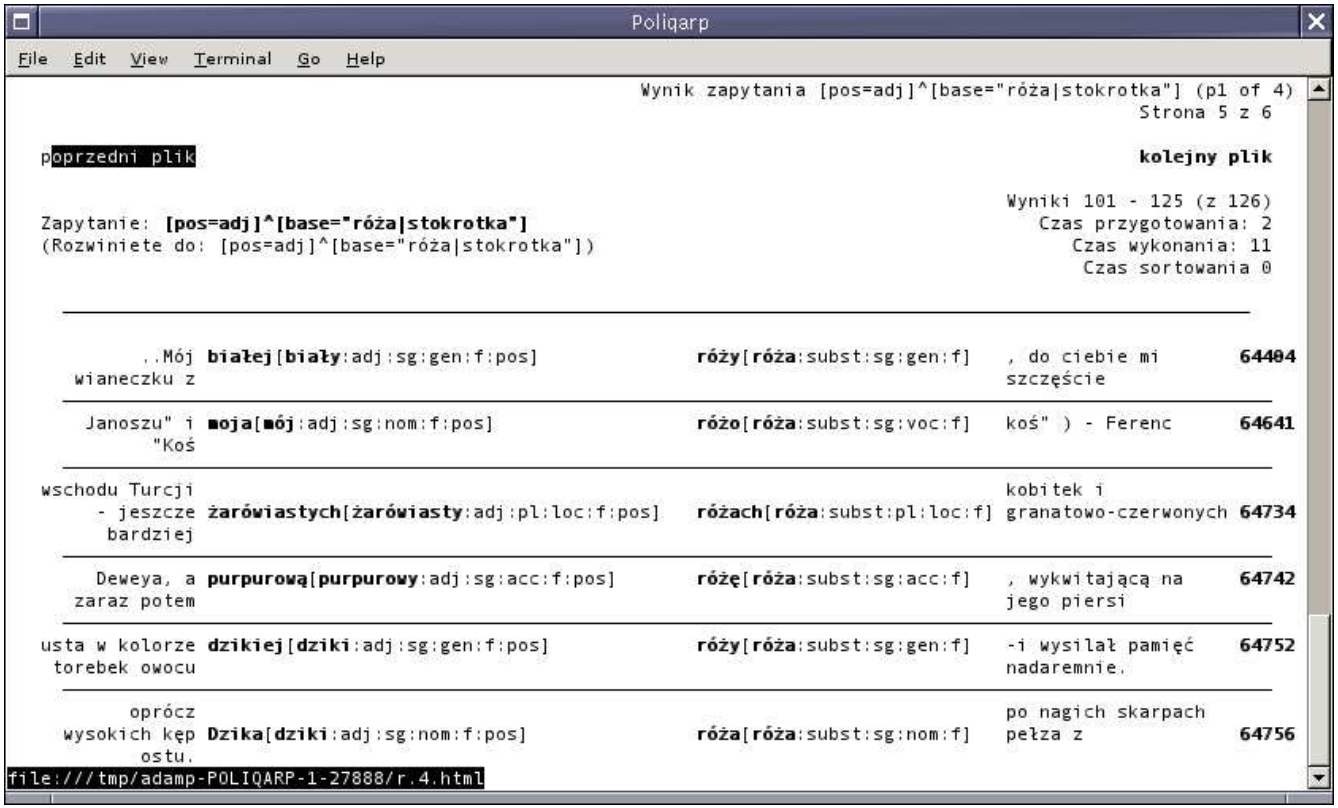
(4.85) `KORPUS/WSTEPNY> [pos=adj]^[base="róża|stokrotka"]`

Jeżeli użytkownik nie określił tego inaczej, wyniki zapytania generowane są w postaci HTML, a do ich wyświetlania służy program Links.

#### 4.2.3.1. Edytowanie zapytań w linii poleceń

Podobnie jak w wypadku wersji graficznej programu, także wersja tekstowa zapamiętuje pewną liczbę wcześniejszych zapytań (oraz komend wydawanych z linii poleceń). Dostęp do tych poleceń można uzyskać za pomocą klawiszy ze strzałką w górę i ze strzałką w dół.

W ograniczonym zakresie działa także przeszukiwanie historii zapytań: w wyniku naciśnięcia kombinacji klawiszy Ctrl-r i wpisania ciągu znaków znalezione zostanie (w historii zapytań, a nie w korpusie!) ostatnie zapytanie zawierające ten ciąg znaków.




Rysunek 4.12. Wynik zapytania w wersji tekstowej (konfiguracja domyślna)

Zapytanie można edytować za pomocą klawiszy ze strzałkami w lewo i w prawo, klawisza Backspace oraz m.in. następujących kombinacji klawiszy znanych z programu Emacs:

| klawisz | efekt |
|---|---|
| Ctrl-b | przejście kursora o jeden znak w lewo |
| Ctrl-f | przejście kursora o jeden znak w prawo |
| Ctrl-a | przejście kursora na początek wiersza |
| Ctrl-e | przejście kursora na koniec wiersza |
| Esc-b | przejście kursora o jedno słowo w lewo |
| Esc-f | przejście kursora o jedno słowo w prawo |
| Ctrl-d | skasowanie znaku na pozycji kursora |
| Esc-d | skasowanie znaków od pozycji kursora do końca słowa |
| Esc-Backspace | skasowanie znaków od początku słowa do znaku przed kursorem |
| Ctrl-k | skasowanie znaków od pozycji kursora do końca wiersza |
| Esc-4 Ctrl-f | jak czterokrotne naciśnięcie Ctrl-f, tj. przejście kursora o cztery znaki w prawo |
| Esc-2 Esc-d | jak dwukrotne naciśnięcie Esc-d |
| | itd. |

Nie działają natomiast w sposób oczekiwany klawisze Del, Home oraz End.

W momencie zadawania zapytania (poprzez naciśnięcie klawisza Enter) kursor może znajdować się w dowolnym miejscu linii poleceń.

#### 4.2.3.2. Konfiguracja programu

Działanie wersji tekstowej programu Poliqarp można znacząco zmodyfikować poprzez edycję pliku konfiguracyjnego `.poliqarp_config` znajdującego się w katalogu domowym użytkownika. Najprostszą metodą utworzenia takiego pliku konfiguracyjnego jest wydanie z linii poleceń komendy `/dump-config`. Spowoduje to zapisanie w katalogu domowym użytkownika pliku `.poliqarp_config.new`, zawierającego domyślne ustawienia wersji tekstowej programu Poliqarp. Aby plik ten stał się obowiązującym plikiem konfiguracyjnym, należy zmienić jego nazwę na `.poliqarp_config`.

Plik konfiguracyjny jest *de facto* ciągiem poleceń dla programu Poliqarp, które wykonywane są przy starcie programu. Każde z tych poleceń, poprzedzone ukośnikiem `/`, może być także wpisane w dowolnym momencie z linii poleceń programu Poliqarp.

Na przykład, aby zmienić liczoną w segmentach długość prawego kontekstu na potrzeby danej sesji, należy wydać następujące polecenie:

(4.86) `KORPUS/WSTEPNY> /set right-context 20`

Poleceniu temu odpowiada następujący wpis w pliku konfiguracyjnym:

(4.87) `set right-context 20`

W analogiczny sposób można zmienić długość lewego kontekstu — albo wydając polecenie w (4.88) (zmiana będzie dotyczyła obecnej sesji), albo dokonując w pliku konfiguracyjnym wpisu (4.89) (zmiana będzie dotyczyła kolejnych sesji).

(4.88) `KORPUS/WSTEPNY> /set left-context 20`
(4.89) `set left-context 20`

Zmienne `right-context` i `left-context` to tylko dwie spośród wielu zmiennych tekstowej wersji programu Poliqarp, które można modyfikować za pomocą polecenia `set`. Krótkie opisy wszystkich zmiennych można uzyskać za pomocą polecenia `/desc` wydanego z linii poleceń[6], zaś aktualne wartości wszystkich zmiennych — za pomocą polecenia `/show`. Jeżeli polecenia te zostaną wywołane z argumentami będącymi nazwami zmiennych, to odpowiednie informacje wyświetlone zostaną tylko dla tych zmiennych. Na przykład, aby uzyskać informację o aktualnych wartościach zmiennych `right-context` i `left-context`, należy wydać polecenie:

(4.90) `KORPUS/WSTEPNY> /show right-context left-context`

Domyślnie, jak ilustruje to Rys. 4.12, w kolumnach z lewym i prawym kontekstem wyświetlane są wyłącznie segmenty, tj. formy występujące w tekście, zaś w kolumnie (lub kolumnach) z dopasowaniem — segmenty, formy podstawowe oraz znaczniki morfosyntaktyczne. Rodzaj pokazywanych informacji można zmienić, nadając odpowiednie wartości zmiennym

[6] Podobnie jak inne polecenia wersji tekstowej, także polecenie `desc` można umieścić w pliku konfiguracyjnym — spowoduje to wyświetlenie krótkiego opisu zmiennych przy każdym uruchomieniu programu Poliqarp.

`cl-x` (aby zmienić lewy kontekst), `ml-x` (lewe dopasowanie), `mr-x` (prawe dopasowanie lub, jeżeli w zapytaniu nie było znacznika wyrównania, całe dopasowanie) oraz `cr-x` (prawy kontekst). Wartością każdej z tych zmiennych powinien być ciąg liter. Jeżeli wśród tych liter jest litera `o` (jak `orth`), wyświetlona zostanie w danej kolumnie forma wyrazowa segmentu. Jeżeli jest wśród nich litera `b` (jak `base`), wyświetlona zostanie forma hasłowa. Jeżeli natomiast jest wśród nich litera `t` (jak `tag`), wyświetlone zostaną znaczniki morfosyntaktyczne właściwe w danym kontekście (tj. po dezambiguacji), zaś w wypadku litery `T` — wszystkie znaczniki przypisane danemu segmentowi przez analizator morfologiczny. Na przykład, jeżeli w pliku konfiguracyjnym znajdą się wiersze pokazane w (4.91), wynik zapytania będzie wyglądał jak na Rys. 4.13.

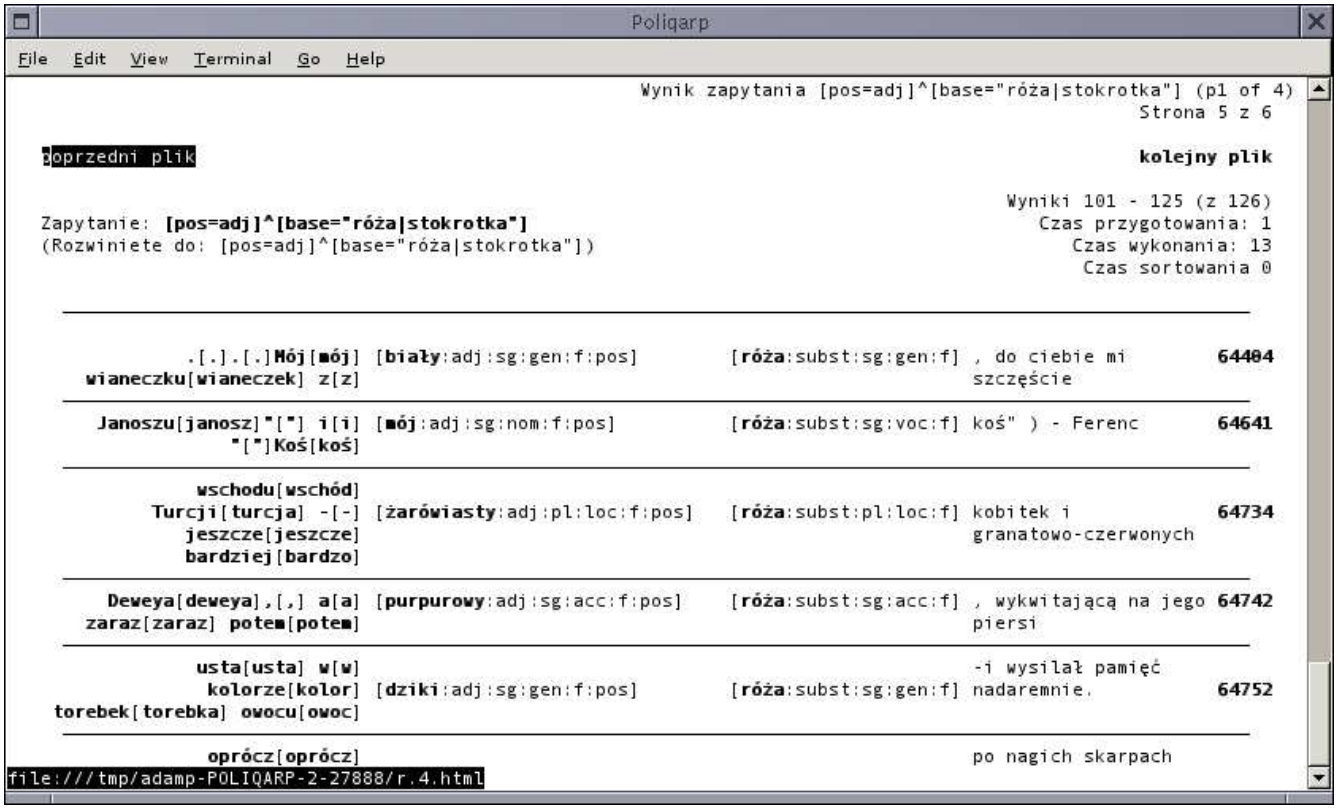
(4.91)
```
set cl-x bo
set ml-x bt
set mr-x bt
set cr-x o
```

W obecnej wersji tekstowej wyniki przeszukiwania korpusu generowane są w postaci zbioru plików HTML-owych, z których każdy, być może z wyjątkiem ostatniego, zawiera maksymalną liczbę wyników, będącą wartością zmiennej `hits-per-page`. Wyniki te są pokazywane za pomocą programu, którego nazwa jest wartością zmiennej `pager`. Na przykład wykonanie poniższych komend z poziomu linii poleceń spowoduje wyświetlanie wyników kolejnych zapytań za pomocą programu Mozilla, po 50 wyników na stronę.

(4.92) `KORPUS/WSTEPNY> /set hits-per-page 50`
(4.93) `KORPUS/WSTEPNY> /set pager mozilla`

Format wyświetlania wyników można modyfikować dosyć szczegółowo za pomocą zmiennych `result-format` i `match-format`: pierwsza z nich określa format HTML całej strony z wynikami, zaś druga — format pojedynczego wiersza w tabeli z wynikami. Wartości tych zmiennych nie powinny być modyfikowane przez użytkowników, którzy nie posiadają dogłębnej znajomości formatu HTML. Szczegółowe omówienie tych zmiennych wykracza poza ramy niniejszej publikacji.

Inne zmienne, o których należy tu wspomnieć, to `input-encoding` i `output-encoding`: ich wartości określają, odpowiednio, kodowanie




Rysunek 4.13. Formatowanie wyników przy ustawieniach w (4.91)

wejścia i wyjścia terminala, w którym uruchamiana jest wersja tekstowa. Zwykle nie należy zmieniać domyślnej wartości tych zmiennych, którą jest `latin2`.

Podobnie jak w wypadku wersji graficznej, także w wersji tekstowej możliwe jest sortowanie wyników po każdej z kolumn tabeli z wynikami, w porządku rosnącym lub malejącym, *a fronte* (alfabetycznie od przodu) lub *a tergo* (alfabetycznie od tyłu). Kolejność sortowania jest określana za pomocą zmiennej `sort-by`, której wartością jest ciąg znaków będący szeregiem specyfikacji sortowania. Na przykład komenda w (4.94) spowoduje, że wyniki kolejnych zapytań będą sortowane (*a fronte*, w porządku rosnącym) po prawym kontekście.

(4.94) `KORPUS/WSTEPNY> /set sort-by r`

Każdej kolumnie odpowiada inna litera:

| litera | kolumna sortowania |
|---|---|
| `l` | lewy kontekst |
| `r` | prawy kontekst |
| `n` | lewe dopasowanie |
| `m` | prawe dopasowanie |

Podanie w wartościach zmiennej `sort-by` małych liter spowoduje posortowanie *a fronte*, zaś podanie odpowiednich liter wielkich spowoduje posortowanie *a tergo*. Dodatkowo przed każdą literą może stać znak `+` dla sortowania rosnącego lub `-` dla sortowania malejącego. Brak znaku przed literą interpretowany jest jako znak `+`. Na przykład komenda (4.95) spowoduje sortowanie wyników po lewym kontekście, *a tergo*, w porządku malejącym.

(4.95) `KORPUS/WSTEPNY> /set sort-by -L`

Jeżeli w wartościach zmiennej `sort-by` znajdują się odnośniki do kilku kolumn, to sortowanie przebiega najpierw według wartości pierwszej podanej kolumny, następnie dla tych samych wartości w pierwszej kolumnie — według wartości drugiej podanej kolumny itd. Na przykład poniższa komenda spowoduje, że wyniki kolejnych zapytań będą sortowane w porządku rosnącym według kolumny z prawym dopasowaniem, a następnie, w obrębie wyników tego sortowania, w porządku malejącym według war-

tości kolumny z lewym dopasowaniem, tak jak to zostało przedstawione na Rys. 4.14.

(4.96) `KORPUS/WSTEPNY> /set sort-by m-n`

Wyniki ostatniego zapytania są zapamiętywane aż do wykonania kolejnego zapytania i mogą być ponownie posortowane i wyświetlone w nowym porządku. Służą do tego komendy `sort` i `view`. Na przykład, aby obejrzeć wyniki ostatniego zapytania, ale posortowane według lewego kontekstu, należy podać następujące polecenia:

(4.97) `KORPUS/WSTEPNY> /set sort-by l`
(4.98) `KORPUS/WSTEPNY> /sort`
(4.99) `KORPUS/WSTEPNY> /view`

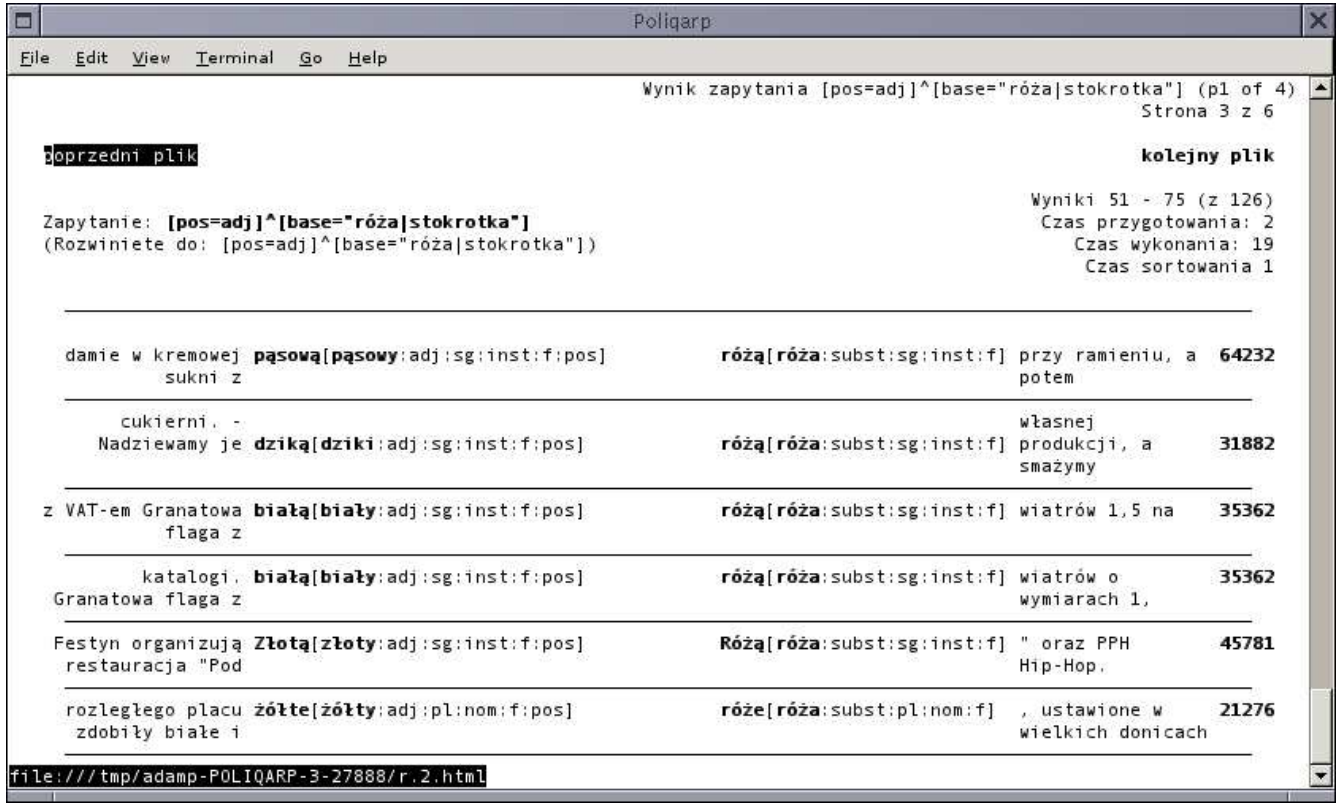
W wypadku wszystkich omówionych powyżej zmiennych, zamiast ich pełnych nazw można użyć ich skrótów, przy czym skrótem może być dowolnej długości początek nazwy zmiennej, o ile tylko pozwala on jednoznacznie zidentyfikować daną zmienną. Na przykład zamiast polecenia `set cl-x bo` w (4.91) można podać polecenie `set cl bo`, w którym nazwa zmiennej `cl-x` jest jednoznacznie skrócona do `cl`, ale nie polecenie `set c bo`, gdyż skrót `c` nie jest jednoznaczny: nie wiadomo, czy odnosi się do `cl-x` czy do `cr-x`.

Podobnie jak w wypadku wersji graficznej, możliwe jest obejrzenie danych o utworze, z którego pochodzi dany wynik przeszukiwania. Jak widać na Rys. 4.12–4.14, w ostatniej kolumnie tabeli z wynikami znajduje się pewna liczba — jest ona numerem utworu, z którego pochodzi dany cytat. Aby uzyskać informacje o tym utworze, należy w linii poleceń napisać `/meta <numer>`, na przykład `/meta 64232`.

W wersji tekstowej możliwe jest także definiowanie dodatkowych aliasów. Służy do tego polecenie `alias`. Na przykład, aby móc używać napisu `ppron` jako skrótu dla `ppron12` i `ppron3`, należy do pliku konfiguracyjnego wpisać następujący wiersz:

(4.100) `alias ppron = ppron12 ppron3`

Oczywiście, tak jak w wypadku innych poleceń, można je wydać także z linii poleceń, poprzedzając je ukośnikiem. Polecenie `alias` bez żadnych argumentów spowoduje wypisanie obecnie zdefiniowanych aliasów, zaś




Rysunek 4.14. Sortowanie wyników według specyfikacji w (4.96)

polecenie `unalias` z jednym lub większą liczbą argumentów będących nazwami aliasów spowoduje usunięcie tych aliasów.

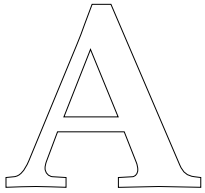
Aby zakończyć pracę z programem Poliqarp, należy użyć polecenia `exit`. Bez wychodzenia z programu można, za pomocą polecenia `open`, zamienić bieżący korpus na inny. Na przykład, aby wczytać korpus `frek` znajdujący się w katalogu `/home/adamp/korpus/`, należy wydać polecenie

(4.101) `KORPUS/WSTEPNY> /open "/home/adamp/korpus/frek"`

Zwięzłą informację o dostępnych poleceniach tekstowej wersji programu Poliqarp można uzyskać wydając polecenie `help`, zaś polecenie `stat` spowoduje wyświetlenie podstawowych informacji liczbowych o bieżącym korpusie.

#### 4.2.3.3. Zmienne systemowe

Podobnie jak wersja graficzna, także i wersja tekstowa ma dwie wersje językowe: polską i angielską. Wersja ta jest wybierana automatycznie, na podstawie wartości zmiennej systemowej `LANG`. Jeżeli wartością tej zmiennej jest `pl_PL`, jest to wersja polska; w przeciwnym wypadku — wersja angielska. Inne standardowe zmienne systemowe, których wartości mogą wpłynąć na działanie programu, to `HOME`, `TMP`, `TEMP` i `USER`. Ich rola jest dokładniej opisana w pliku `README.pl.txt` w katalogu `linux` na załączonej płycie CD-ROM.



# Zawartość płyty CD-ROM



Płyta CD-ROM będąca częścią niniejszej publikacji zawiera:
— wersję wstępną Korpusu IPI PAN (katalog `corpus`),
— trzy wersje programu Poliqarp:
  — graficzną wersję dla systemów Windows 2000 i Windows XP (katalog `windows` oraz skrypt uruchamiający proces instalacji `autorun.inf`),[1]
  — graficzną wersję dla systemu GNU/Linux dla komputerów klasy PC (katalog `linux/gui`),
  — wersję tekstową przeznaczoną dla systemu GNU/Linux dla komputerów klasy PC (katalog `linux/text`),
— niniejszą publikację w formacie PDF (katalog `pdf`).

Na płycie CD-ROM znajduje się także umowa licencyjna (plik `eula.pl.txt`) określająca zasady, na jakich udostępniane są wersje Korpusu IPI PAN i programu Poliqarp znajdujące się na tej płycie. Instalacja Korpusu IPI PAN lub programu Poliqarp na dysku twardym, uruchomienie programu lub jakakolwiek redystrybucja Korpusu IPI PAN lub programu Poliqarp oznacza akceptację warunków tej umowy licencyjnej.

[1] Wersja ta powinna także działać w systemach Windows 98 i Windows NT, nie była jednak pod tym względem ekstensywnie testowana.

## A.1. Windows

Po włożeniu płyty CD-ROM do komputera kontrolowanego przez system Microsoft Windows zostanie automatycznie uruchomiony instalator programu Poliqarp. Instalator pozwoli na wybranie katalogu, w którym zostanie zainstalowany program Poliqarp, oraz zaoferuje skopiowanie Korpusu IPI PAN z płyty CD-ROM na lokalny dysk użytkownika. Kopiowanie korpusu na dysk lokalny nie jest konieczne, można korzystać bezpośrednio z korpusu na płycie CD-ROM, jest ono jednak zalecane, gdyż znacznie przyspieszy pracę z korpusem.

Wersja graficzna programu Poliqarp, a więc także wersja przeznaczona dla systemu Windows, wymaga zainstalowanego środowiska Javy. Instalator spróbuje zlokalizować to środowisko, a w wypadku niepowodzenia zaproponuje instalację Javy dołączonej do niniejszej płyty CD-ROM.

W wyniku poprawnej instalacji programu do menu Programy dodany zostanie program Poliqarp, zaś — o ile użytkownik sobie tego zażyczył w trakcie instalacji — na pulpicie zostanie umieszczona ikonka tego programu.

## A.2. GNU/Linux

Proces instalacji wersji graficznej i wersji tekstowej programu Poliqarp opisany jest w pliku `README.pl.txt` w katalogu `linux`.

# Bibliografia


Bański, Piotr. (2001) „The proposed annotation scheme for the IPI PAN corpus". Prace IPI PAN 936, Instytut Podstaw Informatyki PAN.

———. (2003) „Anotacja zewnętrzna: wpływ architektury korpusu IPI PAN na efektywność jego tworzenia oraz wykorzystania". *Polonica* XXII–XXIII: 77–91.

Bień, Janusz S. (1991) *Koncepcja słownikowej informacji morfologicznej i jej komputerowej weryfikacji*. Warszawa, Wydawnictwa Uniwersytetu Warszawskiego.

———. (2001) „O pojęciu wyrazu morfologicznego". Gruszczyński i in. (2001), 67–78.

———. (2004) „An approach to computational morphology". Mieczysław A. Kłopotek *et al.*, red., *Intelligent Information Processing and Web Mining*. Berlin, Springer-Verlag, 191–199.

Bień, Janusz S. i Zygmunt Saloni. (1982) „Pojęcie wyrazu morfologicznego i jego zastosowanie do opisu fleksji polskiej (wersja wstępna)". *Prace Filologiczne* XXXI: 31–45.

Christ, Oli. (1994) „A modular and flexible architecture for an integrated corpus query system". *Complex'94*. Budapeszt.

Dębowski, Łukasz. (2001) „Tagowanie i dezambiguacja morfologiczna". Prace IPI PAN 934, Instytut Podstaw Informatyki PAN.

———. (2003) „A reconfigurable stochastic tagger for languages with complex tag structure". *Proceedings of* Morphological Processing of Slavic Languages, *EACL 2003*.

———. (2004) „Trigram morphosyntactic tagger for Polish". *Proceedings of IIS:IIPWM 2004*.

Erjavec, Tomaž, red. (2001) *Specifications and notation for MULTEXT-East lexicon encoding*. Ljubljana.

Gruszczyński, Włodzimierz, Urszula Andrejewicz, Mirosław Bańko i Dorota Kopcińska, red. (2001) *Nie bez znaczenia... Prace ofiarowane Profesorowi*

*Zygmuntowi Saloniemu z okazji jubileuszu 15000 dni pracy naukowej*. Białystok, Wydawnictwo Uniwersytetu Białostockiego.

Gruszczyński, Włodzimierz i Zygmunt Saloni. (1978) „Składnia grup liczebnikowych we współczesnym języku polskim". *Studia Gramatyczne* II: 17–42.

Ide, Nancy, Patrice Bonhomme i Laurent Romary. (2000) „XCES: An XML-based standard for linguistic corpora". *Proceedings of the linguistic resources and evaluation conference*. Athens, Greece.

Ide, Nancy, Greg Priest-Dorman i Jean Véronis. (1996) „Corpus encoding standard". Maszynopis, http://www.cs.vassar.edu/CES/.

Kupść, Anna. (1999) „Haplology of the Polish reflexive marker". Robert D. Borsley i Adam Przepiórkowski, red., *Slavic in Head-driven Phrase Structure Grammar*. Stanford, CA, CSLI Publications, 91–124.

Kurcz, Ida, Andrzej Lewicki, Jadwiga Sambor, Krzysztof Szafran i Jerzy Woronczak. (1990) *Słownik frekwencyjny polszczyzny współczesnej*. Kraków, Wydawnictwo Instytutu Języka Polskiego PAN.

Kurcz, Ida, Andrzej Lewicki, Jadwiga Sambor i Jerzy Woronczak. (1974) „Słownictwo współczesnego języka polskiego. Listy frekwencyjne". Maszynopis, Uniwersytet Warszawski.

Mańczak, Witold. (1956) „Ile jest rodzajów w polskim?". *Język Polski* XXXVI(2): 116–121.

Oliva, Karel. (2001) „On retaining ambiguity in disambiguated corpora". *TAL (Traitement Automatique des Langues)* 42(2).

Przepiórkowski, Adam. (2003a) „A hierarchy of Polish genders". Piotr Bański i Adam Przepiórkowski, red., *Generative linguistics in Poland: Morphosyntactic investigations*. Warszawa, Instytut Podstaw Informatyki PAN, 109–122.

———. (2003b) „Składniowe uwarunkowania znakowania morfosyntaktycznego w korpusie IPI PAN". *Polonica* XXII–XXIII: 57–76.

———. (2004) „Instrukcja konwertowania tekstów na format XML w projekcie korpusowym IPI PAN". Maszynopis, Instytut Podstaw Informatyki PAN.

Przepiórkowski, Adam, Piotr Bański, Łukasz Dębowski, Elżbieta Hajnicz i Marcin Woliński. (2003) „Konstrukcja korpusu IPI PAN". *Polonica* XXII–XXIII: 33–38.

Przepiórkowski, Adam, Elżbieta Hajnicz, Marcin Woliński i Łukasz Dębowski. (2004) „Zasady znakowania morfosyntaktycznego w Korpusie

IPI PAN". Maszynopis, Instytut Podstaw Informatyki PAN.

Przepiórkowski, Adam, Zygmunt Krynicki, Łukasz Dębowski, Marcin Woliński, Daniel Janus i Piotr Bański. (2004) „A search tool for corpora with positional tagsets and ambiguities". *Proceedings of the fourth international conference on language resources and evaluation, lrec 2004*, 1235–1238.

Przepiórkowski, Adam, Anna Kupść, Małgorzata Marciniak i Agnieszka Mykowiecka. (2002) *Formalny opis języka polskiego: Teoria i implementacja*. Warszawa, Akademicka Oficyna Wydawnicza EXIT.

Przepiórkowski, Adam i Marcin Woliński. (2003a) „A flexemic tagset for Polish". *Proceedings of* Morphological Processing of Slavic Languages, *EACL 2003*. Lizbona, 33–40.

———. (2003b) „The unbearable lightness of tagging: A case study in morphosyntactic tagging of Polish". *Proceedings of the* 4th International Workshop on Linguistically Interpreted Corpora (LINC-03), *EACL 2003*, 109–116.

Saloni, Zygmunt. (1976) „Kategoria rodzaju we współczesnym języku polskim". *Kategorie gramatyczne grup imiennych we współczesnym języku polskim*. Wrocław, Ossolineum, 41–75.

———. (1977) „Kategorie gramatyczne liczebników we współczesnym języku polskim". *Studia Gramatyczne* I: 145–173.

———. (1981) „Uwagi o opisie fleksyjnym tzw. zaimków rzeczownych". *Folia Linguistica* 2: 265–271.

———. (1988) „O tzw. formach nieosobowych [rzeczowników] męskoosobowych we współczesnej polszczyźnie". *Biuletyn Polskiego Towarzystwa Językoznawczego* XLI: 155–166.

———. (2001) *Czasownik polski. Odmiana, słownik.* Warszawa, Wiedza Powszechna.

Tokarski, Jan. (1993) *Schematyczny indeks* a tergo *polskich form wyrazowych*. Warszawa, Wydawnictwo Naukowe PWN. Opracowanie i redakcja Zygmunt Saloni

Woliński, Marcin. (2001) „Rodzajów w polszczyźnie jest osiem". Gruszczyński i in. (2001), 303–305.

———. (2003) „System znaczników morfosyntaktycznych w korpusie IPI PAN". *Polonica* XXII–XXIII: 39–55.

Woliński, Marcin i Adam Przepiórkowski. (2001) „Projekt anotacji morfosynktaktycznej korpusu języka polskiego". Prace IPI PAN 938, Instytut Podstaw Informatyki PAN.

# Skorowidz