Anotatornia — lingwistyczna baza danych

Elżbieta Hajnicz, Grzegorz Murzynowski, Marcin Woliński

Instytut Podstaw Informatyki PAN, ul. Ordona 21, 01-237 Warszawa
{hajnicz, wolinski} @ipipan.waw.pl

Streszczenie: Anotatornia jest aplikacją przeznaczoną do symultanicznej anotacji korpusów tekstów za pośrednictwem przeglądarki internetowej na różnym poziomie (morfosyntaktycznym, syntaktycznym, semantycznym). Sposób i zakres anotacji zadawany jest za pośrednictwem plików konfiguracyjnych, dzięki czemu może być łatwo zmieniany.

Summary: Anotatornia is a Ruby on Rails application for asynchronous annotation of text corpora via a web browser. The annotation could be performed on various levels (morphosyntactic, syntactic, semantic). The rules and scope of annotation are determined with configuration files and therefore may be easily adjusted.

1. Wstęp

Lingwistyka korpusowa jest obecnie jednym z ważniejszych i szybko rozwijających się działów inżynierii lingwistycznej. Korpusy — duże zbiory tekstów przechowywanych w ujednoliconym formacie i znakowane lingwistycznie — mają bowiem ogromne znaczenie dla rozwoju innych działów inżynierii lingwistycznej oraz tradycyjnej lingwistyki. Stanowią bowiem podstawę do tworzenia i weryfikacji zarówno metod, narzędzi i zasobów pochodnych (np. słowników) inżynierii lingwistycznej, jak i teorii ściśle lingwistycznych.

Przekształcenie tekstów źródłowych na format korpusu polega, poza oczywistym dopasowaniem strony kodowej, na segmentacji tekstu. Jest on dzielony na akapity, zdania oraz tzw. segmenty (w przybliżeniu słowa). Każdy segment jest ponadto znakowany morfosyntaktycznie, tzn. jest mu przypisywana klasa gramatyczna (w przybliżeniu część mowy) oraz zestaw kategorii gramatycznych właściwych dla danej klasy, takich jak rodzaj, liczba i przypadek dla rzeczowników.

Korpusy tekstów są zazwyczaj zbyt duże, aby móc je ręcznie oznakować w całości. Jednak dobra praktyka wymaga stworzenia niewielkiego, starannie dobranego, ręcznie znakowanego podkorpusu. Podkorpus taki służy następnie do trenowania i weryfikacji programów służących do automatycznej anotacji (w wypadku anotacji morfosyntaktycznej zwanych tagerami) pozostałej części korpusu. Jest on także obdarzany większym zaufaniem przez użytkowników korpusu ze względu na mniejszą liczbę błędów popełnianych przez człowieka niż maszynę.

Program Anotatornia przeznaczony został do anotacji konkretnego korpusu, czyli Korpusu IPI PAN (KIPI) stworzonego w zespole Inżynierii Lingwistycznej Instytutu Podstaw Informatyki PAN w ramach projektu KBN 7 T11C 043 20 kierowanego przez dr A. Przepiórkowskiego. Korpus ten tworzy drzewo katalogów, w którego liściach znajdują się poszczególne dokumenty. Dokumenty te przechowywane są w nieco zmodyfikowanym formacie XCES (ang. XML Corpus Encoding Standard) [3]. Dokładniejsze informacje na temat KIPI i jego struktury można znaleźć w [5, 6].

W ramach wczesnych prac nad anotacją KIPI, do ręcznej anotacji morfosyntaktycznej wykorzystywany był program DAUJC, udostępniony przez Jana Hajiča z Uniwersytetu Karola w Pradze. Za jego pomocą oznakowany został Korpus Słownika Frekwencyjnego zawierający w sumie 500 tys. segmentów. Jednak DAUJC był programem nie wystarczającym na nasze obecne potrzeby. W szczególności, weryfikacja musiała być dokonywana ręcznie (na losowo wybranej próbce danych). Zasady anotacji morfosyntaktycznej za pomocą programu DAUJC zostały opisane w [7].

Prezentowane przez nas narzędzie przezwycięża powyższe ograniczenia. Jest intuicyjne i łatwe w obsłudze. Przede wszystkim zaś wszystkie dane dotyczące zarówno zgromadzonej i przetwarzanej wiedzy lingwistycznej jak i operujących na nich osób przechowywane są na jednym serwerze, co umożliwia automatyczną synchronizację prac.

Program Anotatornia został napisany we frameworku Ruby on Rails, w ramach projektów badawczych MNiSW N N516 0165 33 pt. *Automatyczne wykrywanie zależności semantycznych*

w strukturze argumentowej czasowników w dużych korpusach tekstów anotowanych syntaktycznie kierowanego przez E. Hajnicz oraz nr 3 T11C 003 28 pt. *Automatyczna ekstrakcja wiedzy lingwistycznej z dużego korpusu języka polskiego* kierowanego przez A. Przepiórkowskiego.

Niniejsza praca stanowi pracę naukową finansowaną ze środków na naukę na lata 2007-2009 w ramach projektu badawczego MNiSW nr N N516 0165 33.

2. Zasady działania Anotatorni

Wszystkie dane, zarówno te dotyczące procesu anotacji, jak i wiedza lingwistyczna, są przechowywane w jednej bazie danych, fizycznie stanowiącej jeden plik SQLite3. Baza danych składa się z dwóch powiązanych ze sobą części, z których jedna przechowuje dane o procesie anotacji lingwistycznej, a druga właściwą wiedzę językową. Zasadą jest opracowywanie tych samych danych (anotacja zdań) przez dwóch uprawnionych językoznawców (zwanych Anotatorami). Wyniki ich pracy podlegają konfrontacji i uzgadnianiu. Dla każdego zdania zapamiętywana jest informacja o parze Anototatorów, która nad nim pracowała, a dla każdego Anotatora — lista opracowanych przezeń jednostek językowych. Proces uzgadniania przebiega iteracyjnie aż do uzyskania jednomyślności. Jest on zapamiętywany, co umożliwia oszacowanie trudności zadania.

Operacje na bazie danych uprawnieni użytkownicy wykonują przez strony programu wyświetlone w przeglądarce internetowej.

2.1. Użytkownicy

W Anotatorni wydzielonych zostało pięć typów użytkowników: Administrator Bazy, Zarządca Anotacji, Anotator, Audytor i Gość. Do zadań pierwszego należy dodawanie, usuwanie i blokowanie użytkowników w dowolnym momencie prac;[1] Jego obowiązkiem jest także wczytanie porcji danych do anotacji i wypisanie jej w formacie wyjściowym po zakończeniu procesu anotacji (istnieje możliwość zdefiniowania alternatywnych formatów wyjściowych). Z kolei Zarządca Anotacji steruje samym procesem Anotacji, widzi postępy prac poszczególnych Anotatorów, może komentować i poprawiać popełnione przez nich błędy. Jest wręcz do tego zobligowany, gdy Anotatorom zbyt długo nie udaje się osiągnąć konsensusu. Jako jedyny może też dokonywać poprawek w zdaniach, które przeszły pomyślnie proces anotacji i weryfikacji i tym samym zostały uznane za zaanotowane poprawnie.

Głównymi użytkownikami programu są rzecz jasna Anotatorzy, dokonujący właściwego znakowania lingwistycznego.

Audytor widzi pełen stan prac, lecz jego możliwość ingerencji ograniczona jest do wpisywania komentarzy. Gość natomiast ma prawo do obejrzenia wyłącznie kompletnie zaanotowanych i zweryfikowanych zdań (to prawo posiadają wszyscy użytkownicy, gdyż zdania takie uznawane są za wzorzec poprawnej weryfikacji).

2.2. Postać danych do anotacji

Szczegóły anotacji nie są wpisane w strukturę programu, lecz zadawane plikami konfiguracyjnymi podczas inicjalizacji bazy. Przede wszystkim zostają wówczas dostarczone podstawowe dane systemu, czyli fragment korpusu do znakowania, w postaci pliku XML. Struktura tego pliku, jako zgodna z międzynarodowym standardem, jest zadana na sztywno w programie. Ponadto wczytywane są do bazy pliki opisujące dopuszczalne typy fraz wraz z układem ich centrów semantycznych i syntaktycznych, lista klas gramatycznych i odpowiadających im części mowy oraz lista sensów, jakie mogą zostać przypisane w ramach anotacji semantycznej. W wersji 2.0, nad którą w chwili pisania artykułu jeszcze trwają prace, wczytywany będzie także słownik walencji, tj. wymagań czasownikowych.

Następnie należy ustalić zakres planowanej anotacji. W programie przewidziane są następujące rodzaje anotacji:

1. morfosyntaktyczna,
2. syntaktyczna (podział na frazy),

[1] Anotatora, który zaanotował choć jedno zdanie nie można usunąć, gdyż zdania zawierają informację o tym, kto je zaanotował.

3. semantyczna (znakowanie sensami pojedynczych wyrazów),
4. tworzenie słowników walencyjnych.

Wszystkie te rodzaje anotacji są od siebie niezależne i mogą być przeprowadzane łącznie bądź osobno. Dla każdej z nich wymagany jest odrębny plik konfiguracyjny.

2.2.1. Anotacja morfosyntaktyczna

W obecnej wersji programu anotacja morfosyntaktyczna ogranicza się do wyboru jednego z listy opisów (tagów), wczytanych do bazy danych z pliku źródłowego — otagowanego korpusu. Ściślej rzecz biorąc, program rozpoznaje anotowaną klasę gramatyczną segmentu (wyrazu), co jest konieczne dla zweryfikowania anotacji syntaktycznej. Pozostałą część tagu, tzn. wartości odpowiednich kategorii gramatycznych, program traktuje jako całość i nie sprawdza ich poprawności — sprawdzenie takie zostało już przeprowadzone przez program znakujący korpus (tager). Tager w szczególności wyróżnia jeden opis morfosyntaktyczny jako najbardziej prawdopodobny (niekiedy więcej niż jeden). Jeżeli Anotator zgadza się z takim wyborem, po prostu pozostawia go bez zmian.

2.2.2. Anotacja syntaktyczna

Przez anotację syntaktyczną rozumiemy oznaczenie fraz w zdaniu. Program w obecnej wersji pozwala na pozostawienie niektórych segmentów poza frazami (np. znaków interpunkcyjnych). Po oznaczeniu granic frazy Anotator wskazuje jej typ (z listy, np. czasownikowa, rzeczownikowa, przyimkowa). Wówczas program próbuje rozpoznać centrum syntaktyczne i semantyczne frazy, zgodnie z opisem fraz wczytanym z pliku konfiguracyjnego. Zazwyczaj centra syntaktyczne i semantyczne fraz są domyślnie równe, jednak np. w wypadku frazy przyimkowej jej centrum syntaktycznym jest przyimek, a semantycznym — rzeczownik. W wypadku zainstnienia więcej niż jednej możliwości program czeka na dokonanie wyboru przez Anotatora.

Aby możliwe było poprawne wyznaczanie centrów fraz, korpus musi być jednoznacznie oznakowany morfosyntaktycznie. Nie jest rzecz jasna istotne, czy anotacja taka dokonywana była w ramach tych samych prac czy też dostarczona była na wejściu.

Typowa struktura syntaktyczna zdania jest wielopoziomowa (stąd tzw. drzewa rozbioru), jednak obecna wersja Anotatorni udostępnia jedynie możliwość wyznaczania granic fraz na najwyższym poziomie składniowym.

2.2.3. Anotacja semantyczna

Gdy do zadań anotatorskich należy przypisywanie wyrazom sensów, w ramach inicjalizacji bazy niezbędne jest wczytanie do niej listy sensów opartej na zgrubnej kategoryzacji Słowosieci oraz listy sensów dopuszczalnych dla konkretnych leksemów, opracowanych przez M. Derwojedową i M. Zawisławską w ramach projektu MNiSW nr 3 T11C 018 29. W trakcie anotacji semantycznej program proponuje te właśnie sensy dla odpowiednich segmentów, gdy już został ustalony (wybrany) ich opis morfologiczny. Anotator może pozostawić sens zaproponowany przez program, wybrać inny z listy sensów dotychczas przypisanych danemu leksemowi, bądź przypisać nowy sens dopuszczalny dla danej klasy gramatycznej.

W prowadzonym obecnie procesie anotacji sensami opatrywane są rzeczowniki i czasowniki.

2.2.4. Słownik walencyjny

W czasie przekazania niniejszego artykułu na konferencję trwają prace nad wzbogaceniem Anotatorni o słownik walencyjny. Jest to słownik, który gromadzi informację na temat zależności pomiędzy pewnymi nadrzędnymi (predykatywnymi) składnikami struktury zdania (przede wszystkim czasownikami) a ich podrzędnikami (argumentami). Każdemu predykatowi przypisywana jest lista argumentów, które dzielą się na obligatoryjne (wymagane) oraz fakultatywne. Słownik walencyjny może mieć charakter ściśle syntaktyczny. Jako argumenty wskazywane są wówczas typy fraz wraz z charakterystyką ich centrów syntaktycznych (np. dla fraz rzeczownikowych i przyimkowych jest to przypadek). Do tworzenia słownika walencyjnego niezbędny jest korpus oznakowany syntaktycznie. Hasła słownika zawierają argumenty odpowiadające frazom w zdaniach. Zadaniem Anotatora jest tu określenie, które z nich są obligatoryjne. Dobierane hasło musi być dopasowane do wszystkich fraz wyznaczonych w zdaniu, jednak lista argumentów może być dłuższa.

W zdaniu może brakować nie tylko dowolnych argumentów fakultatywnych, dopuszczalna jest także nieobecność pewnych argumentów obligatoryjnych; zjawisko takie określane jest w lingwistyce mianem elipsy. Anotatornia może wymagać zaznaczenia w zdaniu wszystkich elips przed dopasowaniem hasła słownika walencyjnego, jednak jest to wymaganie opcjonalne.

Słownik walencyjny może zawierać także pewną informację semantyczną. Często ma ona postać roli semantycznej (agensa, przedmiotu, narzędzia akcji) pełnionej przez argument w zdaniu. Ten rodzaj opisu nie będzie jednak na razie uwzględniany w programie. Zamiast tego predykaty i ich argumenty uzupełniane będą o listę sensów (sens predykatu musi być jednoznaczny), przypisanych centrom semantycznym właściwych fraz, które wystąpiły w zdaniu powiązanym z danym hasłem słownikowym. Proces ten może wymagać rozbicia jednego hasła słownika walencyjnego syntaktycznego na kilka odrębnych haseł słownika semantycznego.

Słownik semantyczny tworzony będzie na podstawie korpusu znakowanego tak syntaktycznie, jak semantycznie. Dodatkowym plikiem wejściowym może być w tym wypadku istniejący słownik walencyjny syntaktyczny. Jako że słowniki takie często zawierają jedynie argumenty obligatoryjne, Anotatorzy będą mieli nadal swobodę dodawania argumentów fakultatywnych.

Szczegółowe zasady, według których przebiegać ma proces anotacji przedstawione zostały w [2].

3. Program i baza danych

Program został napisany we frameworku Ruby on Rails, który wyraźnie skłania do tworzenia aplikacji w paradygmacie Model-View-Controller. Program składa się z kilkudziesięciu niewielkich plików zawierających kod języka Ruby. Fakt, że Ruby nie wymaga kompilacji, stanowi duże ułatwienie dla dokonywania na bieżąco potrzebnych zmian i poprawek, które zostają wdrożone natychmiast.

Framework Rails dostarcza m.in. eleganckiego mechanizmu Object-Relational Mapping (ORM) automatycznie tworzącego klasy stanowiące interfejs do odpowiednich tabel bazy danych, a także mechanizmy obsługi technologii AJAX, dzięki której możliwe jest odświeżanie jedynie odpowiednich fragmentów strony w przeglądarce — bez konieczności przeładowywania całej strony.

Sercem programu Anotatornia jest baza danych SQLite3, składająca się z 26 tabel zawierających łącznie 140 atrybutów. Powiązania między nimi są na tyle złożone, że nie zamieszczamy schematu ze względu na brak miejsca — schemat uproszczony byłby całkowicie nieczytelny. Do bazy tej wczytywany jest korpus oraz wszystkie pliki konfiguracyjne. Aktualnie baza zawiera 7680 zdań, z czego ok. 4500 zaanotowanych i ma rozmiar ok. 27 MB. Mechanizm ORM Rails pozwala na proste deklarowanie powiązań między tabelami na poziomie języka Ruby i odwoływanie się do tych powiązań przez metody definiowane automatycznie przez sam fakt deklaracji powiązania, np. kod

```
class Zdanie < ActiveRecord::Base
has_many :token, :order => ”nr_wyrazu_zd”
...
```

deklaruje klasę `Zdanie` jako odpowiadającą tabeli `zdanie` oraz powiązanie jeden do wielu tej tabeli z tabelą `token`, przy czym klauzula `:order...` sprawia, że rekordy podrzędne będą dostępne w programie w porządku `nr_wyrazu_zd`.

Główną stronę anotacji zdania pokazuje rysunek 1. Warto podkreślić, że udało się opracować wygodny i intuicyjny interfejs użytkownika. Większość decyzji podejmowanych jest poprzez klikanie na odpowiednie pola i wybór elementu z rozwijanych w ten sposób list. Natomiast granice fraz oznacza się przez chwycenie myszą elementu „chwyć” przy tokenie stanowiącym początek/koniec frazy i upuszczenie go na element „upuść” w miejscu końca/początku frazy, odpowiednio, co także jest realizowane w technologii AJAX.

Co istotne, interpreter języka Ruby, framework Ruby on Rails oraz silnik bazy danych SQLite3 są dostępne na licencjach swobodnych.

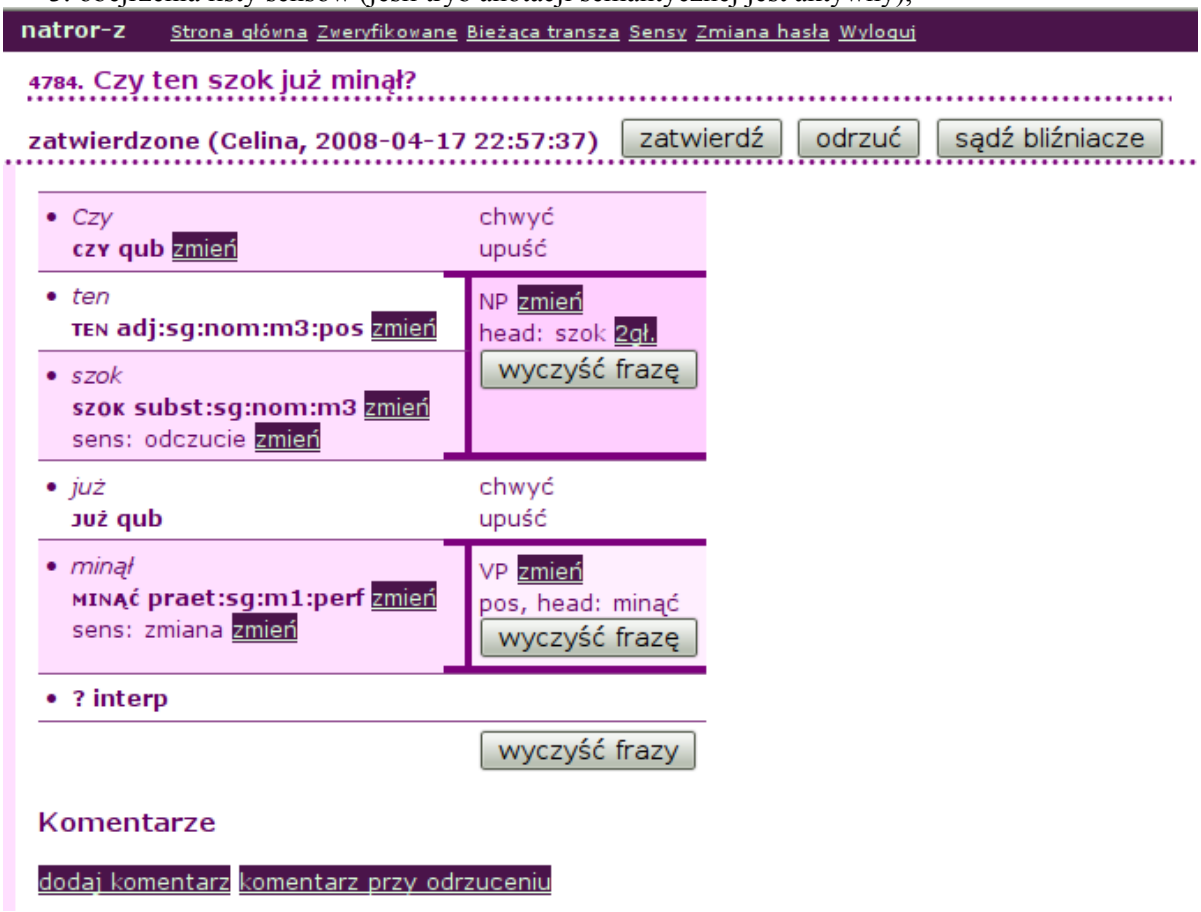
4. Proces anotacji

Anotatorzy dostają do anotacji fragmenty tekstów łączone w porcje zwane transzami. Obecnie transza stanowi zbiór niezależnych zdań, nic nie stoi jednak na przeszkodzie, by pojedynczą jednostką był np. akapit. Baza, na której Anotatorzy pracują obecnie, zawiera transze 120-zdaniowe. Podział korpusu na transze, wykonywany automatycznie w ramach inicjalizacji bazy, przebiega

w taki sposób, aby każde zdanie znalazło się dokładnie w dwóch transzach, które zostaną przydzielone różnym Anotatorom. Co więcej, transze zostają skonstruowane w taki sposób, że dowolne dwie mają nie więcej niż połowę zdań wspólną. Ma to na celu ograniczenie możliwości „dogrywania się" Anotatorów w parach.

Po zalogowaniu się do systemu Anotator ma możliwość wykonania następujących akcji.

1. zmiany hasła,
2. obejrzenia poprawnie zaanotowanych zdań (o ile takie już są obecne w systemie),
3. obejrzenia listy sensów (jeśli tryb anotacji semantycznej jest aktywny),



Rysunek 1 Strona anotacji zdania

4. pobrania nowej transzy do anotacji (jeśli liczba otwartych transz nie przekroczyła z góry ustalonego limitu),
5. wybrania transzy do anotacji (spośród otwartych).

Anotator ma możliwość pracy na więcej niż jednej transzy ze względu na różne tempo prac. Liczba otwartych transz Anotatora jest ograniczona, aby uniknąć nadmiernej desynchronizacji prac. Optymalną maksymalną liczbą otwartych transz okazało się w praktyce 5. Transzę uznaje się za zamkniętą, gdy wszystkie wchodzące w jej skład zdania są już zaanotowane i zweryfikowane (uzgodnione).

Po wybraniu transzy Anotator przechodzi do listy wchodzących w jej skład zdań. Każde zdanie opatrzone jest statusem: do anotacji, zaanotowane, do konfrontacji, zweryfikowane. Zdania, które nie przeszły jeszcze procesu weryfikacji, mogą być anotowane. Działania Anotatora są niejednokrotnie ograniczone do wyboru elementu listy, co ogranicza możliwość popełnienia pomyłki. Wstępna weryfikacja następuje w momencie zatwierdzenia zdania przez Anotatora. Program sprawdza wówczas zgodność anotacji z ograniczeniami narzuconymi w plikach konfiguracyjnych. Jeśli zdanie nie daje się poprawnie (zgodnie z zaleceniami z [2]) zaanotować, Anotator ma prawo je odrzucić.

W każdym momencie Anotator ma także prawo opatrzyć zdanie komentarzem. Komentarze służą zarówno do komunikacji między użytkownikami, jak i do opisu szczególnych przypadków anotacji

(np. wystąpienia leksykalizacji). Lista takich przypadków została wyszczególniona w [2]. W wypadku odrzucenia zdania komentarz zawierający przyczynę takiej decyzji jest obowiązkowy.

Właściwa weryfikacja następuje, gdy zdanie zostanie zaanotowane przez dwie osoby. Jeśli znakowanie jest identyczne, weryfikacja przebiega pomyślnie. W przeciwnym razie zdanie przechodzi w tryb konfrontacji bądź osądzenia w zależności od trybu pracy programu (ustalonego w fazie konfiguracji).

Osoba dokonująca konfrontacji czy osądzenia widzi obie wersje anotacji wraz z zaznaczonymi różnicami (chyba że jeden z Anotatorów zdanie odrzucił). W obecnej wersji programu, jeśli Anotatorzy nie są w stanie uzgodnić swoich wersji po dwóch wymianach opinii i korekt, zdanie zostaje oznaczone jako „do osądzenia" (przez Zarządcę) i pojawia się na odpowiedniej stronie Zarządcy Anotacji.

Zasady pracy z programem Anotatornia opisane zostały w [1].

5. Podsumowanie

W powyższym omówieniu przedstawione zostało narzędzie służące do symultanicznej anotacji korpusów tekstów za pośrednictwem przeglądarki internetowej. Cała wiedza, zarówno lingwistyczna jak i dotycząca procesu anotacji, zgromadzona jest w bazie danych (SQLite3).

Po zakończeniu obecnych prac planowana jest rozbudowa Anotatorni o pełną anotację morfosyntaktyczną (edycja tagów z kontrolą poprawności) i syntaktyczną (wyznaczanie drzew rozbioru).

Anotatornia od samego początku projektowana jest w sposób tak elastyczny, że wraz z jej dalszym rozwojem powinna stanowić uniwersalne narzędzie niezwykle przydatne do wszelkich ręcznych prac z zakresu lingwistyki korpusowej.

Bibliografia

[1] Hajnicz E., *Instrukcja obsługi programu Anotatornia* , 2008, maszynopis.

[2] Hajnicz E., *Zasady anotacji syntaktyczno-semantycznej w programie Anotatornia* , 2008, maszynopis.

[3] Ide N., Bonhomme P. i Romary L., *XCES: An XML-based standard for linguistic corpora* , Proceedings of Linguistic Resources and Evaluation Conference, Athens, 2000, Greece, s. 825–830.

[4] Murzynowski G., *Dokumentacja techniczna programu Anotatornia* , 2008, maszynopis.

[5] Przepiórkowski A., *Korpus IPI PAN. Wersja wstępna* , 2004, Instytut Podstaw Informatyki PAN, Warszawa.

[6] Przepiórkowski A., Bański P., Dębowski Ł., Hajnicz E. i Woliński M. *Konstrukcja Korpusu IPI PAN* , 2003, Polonica XXII–XXIII, s. 33–38.

[7] Przepiórkowski A., Hajnicz E., Woliński M., Dębowski Ł., *Zasady znakowania morfosyntaktycznego w Korpusie IPI PAN* , 2004, maszynopis.

[8] Thomas D., Heinemeier Hansson D., *Agile Web Development with Rails* , 2007, The Pragmatic Bookshelf, Raleigh North Carolina, Dallas, Texas.