Katarzyna Krasnowska-Kieraś

Automatyczne tworzenie składnikowo-zależnościowych rozbiorów wypowiedzeń z wykorzystaniem głębokich sieci neuronowych

rozprawa doktorska pod kierunkiem
dr. hab. Marcina Wolińskiego, prof. IPI PAN

Instytut Podstaw Informatyki
Polskiej Akademii Nauk

Warszawa 2025

Prace, których wyniki są prezentowane w rozprawie, prowadzone były w ramach projektów Dariah.lab oraz Dariah.hub.

Spis treści

Streszczenie

Przedmiotem niniejszej rozprawy jest analiza składniowa języka naturalnego, nazywana również parsowaniem składniowym. Problem ten polega na przypisaniu zadanemu fragmentowi tekstu formalnej reprezentacji opisującej jego budowę gramatyczną. Jednostkami tekstu poddawanymi analizie najczęściej są zdania lub — ogólniej — wypowiedzenia, a struktura stanowiąca jej wynik to rozbiór składniowy. Analiza składniowa stanowi jedno z podstawowych zagadnień w dziedzinie przetwarzania języka naturalnego, a jej przeprowadzenie jest niezbędne lub bardzo przydatne w wielu zastosowaniach.

Dwoma powszechnie wykorzystywanymi formalizmami reprezentowania rozbioru są drzewa składnikowe oraz zależnościowe. Metoda automatycznej analizy składniowej stanowiąca przedmiot niniejszej rozprawy w systematycznie spójny sposób łączy dwa wspomniane sposoby przedstawiania rozbioru w jedno, hybrydowe drzewo składnikowo-zależnościowe. Podejście takie pozwala na elastyczne wykorzystanie zalet obu reprezentacji, a jednocześnie nie jest szeroko zbadane i stosowane, szczególnie w przetwarzaniu tekstów w języku polskim.

Dla opracowanej przeze mnie nowej techniki parsowania do postaci hybrydowej kluczowe są elementy struktury drzewiastej, które nazywam kręgosłupami składniowymi. Kręgosłupy to swego rodzaju lokalne osie, wokół których zbudowane są fragmenty wynikowego drzewa. Stosując podejście typu „parsowanie przez tagowanie" (*parsing as tagging*), traktuję je jako etykiety (znaczniki), które należy przypisać poszczególnym słowom (segmentom) analizowanego zdania. Aby wyznaczyć sposób, w jaki na bazie kręgosłupów należy zbudować kompletną strukturę hybrydową, wykorzystuję z kolei grafową metodę parsowania zależnościowego.

W celu przeprowadzenia eksperymentalnej weryfikacji skuteczności proponowanej przeze mnie metody zaprojektowałam oraz zaimplementowałam model statystyczny w postaci głębokiej sieci neuronowej, w której skład wchodzi pretrenowany model językowy typu BERT. Model ten wykonuje wszystkie obliczenia konieczne do skonstruowania drzewa hybrydowego zgodnie z moim algorytmem. W ramach dodatkowych badań rozszerzam architekturę parsera o komponenty wykonujące

inne zadania z zakresu przetwarzania tekstu: lematyzację, analizę fleksyjną oraz rozpoznawanie jednostek nazewniczych.

Wyniki wykonanej przeze mnie ewaluacji wykazują, że parser działający zgodnie z przedstawioną metodą ma skuteczność na poziomie najlepszych osiąganych aktualnie wyników (*state-of-the-art*). W moich eksperymentach skupiam się przede wszystkim na tekstach polskich (zarówno współczesnych, jak i dawnych), ale uwzględniam również dane niemieckojęzyczne. Uzyskana jakość automatycznej analizy składniowej współczesnego języka polskiego, wyrażona powszechnie wykorzystywanymi miarami, to 96% UAS i 90,7% LAS dla wykrywania struktury zależnościowej oraz 97,6% F_1 dla wykrywania składników. Wysokie są również wyniki dla lematyzacji oraz analizy fleksyjnej – F_1 odpowiednio na poziomie około 98,5% oraz 97,5% – a także dla rozpoznawania jednostek nazewniczych – F_1 wynosi w tym przypadku 88%.

Moja implementacja zaprezentowanej metody – parser Hydra – w momencie kończenia tej rozprawy znalazła już praktyczne zastosowania, między innymi w automatycznej anotacji dużego korpusu języka polskiego, wsparciu rozbudowy ręcznie znakowanego banku drzew oraz wykrywaniu neologizmów w tekstach pochodzących z Internetu.

Abstract

The subject of this dissertation is the syntactic analysis of natural language, also known as syntactic parsing. This problem involves assigning a formal representation to a given textual passage, describing its grammatical structure. The text units subjected to analysis are most often sentences or — more generally — utterances, and the resulting structure is called a syntactic parse. Syntactic parsing is one of the fundamental issues in the field of natural language processing, and its results are essential or very useful in many applications.

Two formalisms commonly used for representing a parse are constituency trees and dependency trees. The method of automatic parsing that is the subject of this dissertation systematically and consistently combines the two aforementioned structure types into a single, hybrid constituency-dependency tree. This approach allows for flexible use of the advantages of both representations, but at the same time, it has not been widely researched and applied, particularly in processing of Polish texts.

Central to my new hybrid parsing technique are tree structure elements which I call syntactic spines. These components are a kind of local axes around which fragments of the resulting tree are constructed. Using a parsing-as-tagging approach, I treat them as labels (tags) that should be assigned to individual words (segments) of the analysed sentence. To determine the way a complete hybrid structure should be constructed starting from the spines, I use a graph-based dependency parsing method.

In order to experimentally verify the effectiveness of my proposed method, I designed and implemented a statistical model in the form of a deep neural network, which includes a pre-trained BERT-type language model. This model performs all the calculations necessary to construct the hybrid tree according to my algorithm. As part of additional research, I augmented the parser architecture with components that perform other text processing tasks: lemmatisation, inflectional analysis, and named entity recognition.

The results of my evaluation show that the parser operating according to the presented method achieves state-of-the-art performance. In my experiments, I focus primarily on Polish texts (both contemporary and historical), but I also include German-language data. The achieved quality of automatic syntactic analysis of contemporary Polish, measured by commonly used metrics, is 96% UAS and 90.7% LAS for dependency structure detection, and 97.6% F_1 score for constituent detection. The results for lemmatisation and inflectional analysis are also high – the results being approximately 98.5% and 97.5% F_1 score respectively – as well as for named entity recognition – the F_1 score value in this case is 88%.

My implementation of the presented method – the Hydra parser – had already found practical applications by the time this dissertation was completed, including automatic annotation of a large Polish corpus, support for the expansion of a manually tagged treebank, and detection of neologisms in texts sourced from the Internet.

Wstęp

Tematyka i cel pracy

Tematyka niniejszej pracy związana jest z zagadnieniem parsowania składniowego (analizy składniowej) języka naturalnego, czyli przypisywania fragmentom tekstu — najczęściej zdaniom — rozbiorów składniowych. Rozbiór zdania to formalna reprezentacja jego wewnętrznej struktury, odzwierciedlająca sposób, w jaki mniejsze jednostki (słowa, frazy) w systematyczny, hierarchiczny sposób łączą się w jednostki bardziej złożone.

Analiza składniowa jest jednym z kluczowych zadań przetwarzania języka naturalnego i znajduje wiele zastosowań w tej oraz pokrewnych dziedzinach, takich jak humanistyka cyfrowa. Struktury składniowe wypowiedzeń służyć mogą między innymi do precyzyjnego identyfikowania w dużych zbiorach tekstów (korpusach) konkretnych konstrukcji gramatycznych — czy to w pracy lingwisty bądź leksykografa, czy na przykład jako podstawa automatycznej ekstrakcji terminologii czy fraz kluczowych. Analiza składniowa stanowi również konieczny krok w niektórych metodach głębokiego przetwarzania semantycznego.

W badaniach prowadzonych zarówno w Polsce jak za granicą powszechnie znane są w szczególności dwa nurty parsowania, różniące się typem stosowanej reprezentacji rozbioru zdania: składnikowej lub zależnościowej. Oba nurty są szeroko wykorzystywane, rozwijane i stanowią narzędzie lub przedmiot licznych badań. Bardzo mało uwagi poświęca się natomiast ich łącznemu wykorzystaniu w sposób spójny i systematyczny.

W rozprawie proponuję nowy algorytm automatycznego parsowania dla wypowiedzeń w języku naturalnym, operujący na hybrydowych strukturach składniowych zawierających w sobie kompletne, w pełni ze sobą spójne reprezentacje składnikową i zależnościową. Algorytm działa w oparciu o fragmenty struktury drzewiastej, które nazywam *kręgosłupami składniowymi*, traktując je jako swego rodzaju znaczniki przypisywane poszczególnym słowom w podejściu wpisującym się w nurt *parsing as tagging*. Drugim kluczowym elementem opracowanego przeze mnie sposobu parso-

wania jest zastosowanie grafowej techniki analizy zależnościowej. Hybrydowa postać struktur produkowanych przez parser, łącząca dwa powszechne nurty reprezentacji budowy gramatycznej zdania — składnikowy oraz zależnościowy — jest nowością na gruncie automatycznego przetwarzania polszczyzny. Ważną cechą mojej metody jest również fakt, że pozwala ona na uzyskiwanie rozbiorów uwzględniających składniki nieciągłe.

W ramach opisywanych badań weryfikuję, czy z wykorzystaniem zaprojektowanej metody można skutecznie przeprowadzić analizę składniową wypowiedzeń. Szczególną uwagę poświęcam przetwarzaniu tekstów w języku polskim, ale nie ograniczam się do nich — eksperymenty przeprowadzam również na danych niemieckojęzycznych. Badam także możliwość rozszerzenia algorytmu o nanoszenie dodatkowych warstw anotacji. W przeprowadzonych eksperymentach uzyskuję pozytywne wyniki. Praktycznym efektem moich prac jest zaimplementowany przeze parser, który znalazł już zastosowanie w dalszych badaniach i projektach.

Koncepcje oraz wyniki zaprezentowane w tej pracy zostały częściowo opublikowane w postaci artykułów konferencyjnych (Krasnowska-Kieraś i Woliński, 2023, 2024). Wersja demonstracyjna parsera jest dostępna w sieci pod adresem `https://constituency.nlp.ipipan.waw.pl`.

Zawartość rozprawy

W pierwszym rozdziale pracy przedstawiam tło dla prezentowanych badań i wyników, przybliżając w punkcie 1.1 wybrane sposoby reprezentacji budowy składniowej wypowiedzeń — składnikowy i zależnościowy — oraz łączące te dwa podejścia drzewa hybrydowe. W punkcie 1.2 tego samego rozdziału omawiam natomiast korpusy anotowane składniowo, które stanowiły źródło danych wykorzystanych w części eksperymentalnej moich badań.

W rozdziale 2 objaśniam bardziej szczegółowo pojęcie analizy składniowej języka naturalnego. Następnie opisuję, w oparciu o literaturę przedmiotu, różne koncepcje i techniki wykorzystywane w parsowaniu składnikowym oraz zależnościowym. W osobnym punkcie skupiam się szczególnie na dotychczasowych pracach związanych z formalnym opisem gramatycznym oraz automatycznym przetwarzaniem składniowym języka polskiego.

Opracowanej przeze mnie metodzie parsowania oraz jej komputerowej implementacji poświęcam rozdział 3. W pierwszej kolejności prezentuję sam algorytm parsowania. Następnie opisuję wybraną przeze mnie architekturę sieci neuronowej służącej jako model statystyczny dla parsowania hybrydowego oraz implementację tego modelu. Przedstawiam również rozszerzenia zaprojektowanego przeze mnie

parsera o przetwarzanie tekstu w warstwach innych niż składniowa. Rozdział kończę dyskusją wybranych szczegółowych rozwiązań zastosowanych w opracowanym narzędziu oraz ich alternatyw.

W ostatnim rozdziale rozprawy opisuję empiryczną weryfikację skuteczności omawianej metody analizy składniowej. Przedstawiam wykorzystane miary jakości parsowania hybrydowego, przeprowadzone eksperymenty oraz uzyskane w ich efekcie wyniki liczbowe.

Pracę zamyka krótkie podsumowanie.

Podziękowania

Niniejsza praca nie powstałaby bez pomocy i wsparcia merytorycznego mojego promotora, prof. Marcina Wolińskiego, któremu jestem winna serdeczne podziękowania. Dziękuję również prof. Małgorzacie Marciniak, która była pierwszą czytelniczką końcowej wersji rozprawy i autorką wielu pomocnych uwag, a także moim Koleżankom i Kolegom z Zespołu Inżynierii Lingwistycznej IPI PAN za wiele ciekawych i inspirujących rozmów nie tylko na tematy związane z moją pracą.

Dziękuję wreszcie mojemu Mężowi, który był niezmiennie najlepszym i najskuteczniejszym wsparciem na każdym etapie pracy nad tą rozprawą.

Rozdział 1

Składnikowo-zależnościowa reprezentacja składniowa

W tym rozdziale zostaną przedstawione dane wykorzystane do trenowania i ewaluacji parsera będącego przedmiotem rozprawy. Punkt 1.1 opisuje dwie powszechnie stosowane w lingwistyce reprezentacje struktury składniowej tekstu — drzewa składnikowe oraz zależnościowe — a także ich połączenie w postaci drzewa hybrydowego. Ten ostatni rodzaj struktury będzie stanowił przedmiot mojego szczególnego zainteresowania w kolejnych rozdziałach. Punkt 1.2 przybliża zbiory danych, które posłużyły do eksperymentów opisanych w dalszej części tej rozprawy.

1.1. Drzewo jako reprezentacja składniowa

Składnia to dział gramatyki zajmujący się opisywaniem struktury wewnętrznej wypowiedzeń, to jest sposobu, w jaki są one zbudowane z jednostek niższego poziomu (por. Saloni i Świdziński 2001). W ramach badań nad składnią istnieją różne podejścia i teorie opisujące i wyjaśniające prawidłowości, według których słowa mogą łączyć się w określony sposób, tworząc większe jednostki i całe zdania lub wypowiedzenia.[1] Do reprezentacji budowy wypowiedzenia powszechnie wykorzystywane są drzewa składniowe. Przypisanie danemu wypowiedzeniu pewnego drzewa, na przykład w jeden ze sposobów opisanych dalej (1.1.1 — 1.1.4), nazywane jest *anotacją składniową* lub *znakowaniem składniowym*.

Ponieważ przedstawiona w dalszych rozdziałach metoda automatycznego otrzymywania drzew składniowych jest niezależna od szczegółów teorii lingwistycznej, niniejszy punkt skupia się na ogólnych własnościach przywoływanych typów drzew, a nie na motywowanych językoznawczo konkretnych rozstrzygnięciach dotyczących ich budowy. Przytaczane tu przykłady będą dla ustalenia uwagi utrzymane w konwencjach przyjętych w danych wykorzystanych w eksperymentach.

[1] Pojęcie *wypowiedzenia* jest szersze, niż *zdanie* i obejmuje również jednostki pozbawione centrum finitywnego (w składni szkolnej: orzeczenia). Wypowiedzeniami są więc także równoważnik zdania: *Jutro poniedziałek.*, *Wstęp surowo wzbroniony!* czy zawiadomienie: *wyjście ewakuacyjne* na ścianie budynku, *Ogniem i mieczem* na okładce książki, por. Saloni i Świdziński (2001).

W teorii grafów *drzewo* to nieskierowany, spójny graf acykliczny. Jedną z właściwości drzewa (stanowiącą też jedno z alternatywnych sformułowań jego definicji) jest istnienie dokładnie jednej ścieżki pomiędzy dowolnymi dwoma jego wierzchołkami. Graf, w którym każdy spójny podgraf jest drzewem, nazywany jest *lasem*.

Jeśli jeden element spośród zbioru V wierzchołków drzewa zostanie wyróżniony jako korzeń, mamy do czynienia z *drzewem ukorzenionym*. W naturalny sposób wyłania się wówczas hierarchia (częściowy porządek) na jego wierzchołkach: dla $v, w \in V$, $v \leqslant w$ wtedy i tylko wtedy, gdy ścieżka łącząca korzeń z w przechodzi przez v.[2] Gdy $v < w$ (czyli $v \leqslant w$ i $v \neq w$), mówimy, że v jest *przodkiem* w, natomiast w — *potomkiem* v. W szczególnym przypadku, gdy $v < w$ i istnieje krawędź łącząca v i w, v nazywamy *rodzicem* w, a w — *dzieckiem* v. Wierzchołki o wspólnym rodzicu to *rodzeństwo* lub węzły siostrzane. Jeśli w jest jedynym dzieckiem v, to łączącą je krawędź nazwiemy *krawędzią unarną*. Wierzchołek nieposiadający dzieci nosi nazwę *liścia* (terminala, wierzchołka terminalnego). Pozostałe wierzchołki to *wierzchołki wewnętrzne* (wierzchołki nieterminalne, nieterminale). W niniejszej pracy wierzchołki nazywane będą zamiennie *węzłami*.

Poddrzewo danego wierzchołka to drzewo ukorzenione w tym wierzchołku, składające się z niego samego, jego potomków i krawędzi pomiędzy nimi.

Drzewo ukorzenione jest *uporządkowane*, jeśli ustalony jest porządek liniowy na dzieciach każdego jego węzła. Porządek ten oznaczany będzie symbolem $\prec$. Symbol $\prec_d$ oznaczać będzie bezpośrednie poprzedzanie w tym porządku. Można również zdefiniować rozszerzenie $\prec$ na wszystkie węzły, które oznaczymy $\prec^*$. Jeśli $v \prec w$, to $v \prec^* w$ oraz $v' \prec^* w'$ dla każdego v' należącego do poddrzewa ukorzenionego w v oraz w' należącego do poddrzewa ukorzenionego w w. Dodatkowo jeśli $v < w$, to $v \prec^* w$.[3]

Plon wierzchołka v drzewa uporządkowanego to sekwencja liści w kolejności ich odwiedzenia przy przejściu przez poddrzewo v zgodnym z porządkiem $\prec^*$. Oznaczać go będziemy $yield(v)$.

Wszystkim bądź niektórym wierzchołkom drzewa mogą być przypisane *etykiety* z dowolnie ustalonego zbioru. W przypadku drzew składniowych etykietami najczęściej są napisy. Etykieta wierzchołka v oznaczana będzie jako $label(v)$.

---

[2] W szczególności przyjmujemy, że ścieżka prowadząca do w przechodzi przez w, zatem definicja ta obejmuje również przypadek, gdy $v = w$

[3] Pierwszy warunek, definiujący porządek na węzłach niebędących w relacji przodek-potomek, będzie istotniejszy dla kolejnych definicji. Drugi warunek, w myśl którego $\prec^*$ jest porządkiem prefiksowym, nie jest kluczowy i równie adekwatny byłby np. jego odpowiednik sufiksowy: jeśli $v < w$, to $w \prec^* v$.

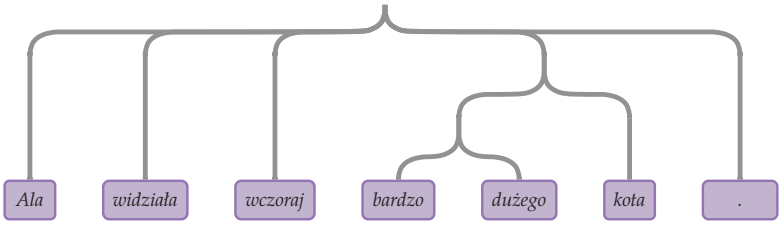
1.1.1. Drzewa składnikowe

Niech S będzie napisem w języku naturalnym (najczęściej będzie to zdanie lub, ogólniej, wypowiedzenie), a $W = s_1 \prec_s ... \prec_s s_n$ — uporządkowaną sekwencją jednostek wyodrębnionych z S, które określać będziemy *segmentami*, a podział tekstu na takie jednostki — *segmentacją*. W najprostszym przypadku będą to słowa tekstowe („od spacji do spacji") oraz znaki interpunkcyjne, występujące kolejno w tym napisie (na przykład $S = Ala\ ma\ kota.$, $s_1 = Ala$, $s_2 = ma$, $s_3 = kota$, $s_4 = .$) Relacja $\prec_s$ oznacza tu bezpośrednie poprzedzanie, jej domknięcie przechodnie oznaczać będziemy $\prec_s^+$.

Drzewo składnikowe to drzewo uporządkowane, którego liście etykietowane są segmentami danego napisu, a uporządkowanie wierzchołków wynika z porządku tekstowego segmentów przypisanych liściom. Segmenty przypisane liściom w plonie każdego nieterminala stanowią spójny podciąg W, a relacja $\prec^*$ pomiędzy liśćmi odzwierciedla relację $\prec_s$ na segmentach: jeśli $yield(v) = v_i \prec^* ... \prec^* v_{i+k}$, to musi zachodzić $label(v_i) \prec_s ... \prec_s label(v_{i+k})$. Dla każdych dwóch siostrzanych węzłów nieterminalnych v, w, gdzie $yield(v) = v_i \prec^* ... \prec^* v_{i+k}$ oraz $yield(w) = w_j \prec^* ... \prec^* w_{j+l}$, zachodzi $v \prec_d w$ wtedy i tylko wtedy, gdy $label(v_{i+k}) \prec_s label(w_j)$ (segment w ostatnim liściu plonu v bezpośrednio poprzedza segment w pierwszym liściu plonu w w porządku tekstowym, innymi słowy: sekwencje segmentów odpowiadające v i w występują w W bezpośrednio jedna po drugiej). W dalszej części pracy, dla uproszczenia sformułowań, liście drzewa i segmenty stanowiące ich etykiety będą często utożsamiane.

Węzły nieterminalne drzewa składnikowego tworzą strukturę niejako nadbudowaną nad wypowiedzeniem, przedstawiającą sposób, w jaki liście łączą się w coraz dłuższe i bardziej skomplikowane jednostki. Liczba i konkretna struktura tych nieterminali może być różna, w zależności od przyjętej teorii lingwistycznej. Rysunek 1.1 przedstawia przykładowe drzewo składnikowe dla zdania *Ala widziała wczoraj bardzo dużego kota.* Struktura zdania reprezentowana przez to drzewo jest następująca. Każdy z segmentów (słów) w liściach drzewa stanowi składnik (frazę[4]): *Ala*, *widziała*, *wczoraj*, *bardzo*, *dużego*, *kota* oraz kończąca zdanie kropka. Składniki *bardzo* oraz *dużego* połączone są w większy składnik *bardzo dużego*, który z kolei razem z *kota* tworzy składnik *bardzo dużego kota*. Składniki *Ala*, *widziała*, *wczoraj*, *bardzo dużego kota* oraz kończąca zdanie kropka składają się na całość zdania, której odpowiada korzeń drzewa.

[4] Kryteria pozwalające uznać dany fragment tekstu za frazę nie są w językoznawstwie uniwersalne i różnią się pomiędzy teoriami lingwistycznymi. Pojęcie frazy używane w ramach tej pracy jest tożsame ze składnikiem: frazą jest to, co zostało wyodrębnione z wypowiedzenia jako plon pewnego nieterminala w przypisanym mu drzewie. Lingwistyczne lub pragmatyczne umotywowanie takiego wyodrębnienia, jakkolwiek niezwykle istotne, pozostaje poza obszarem badawczym tej rozprawy.



Rysunek 1.1: Drzewo składnikowe.

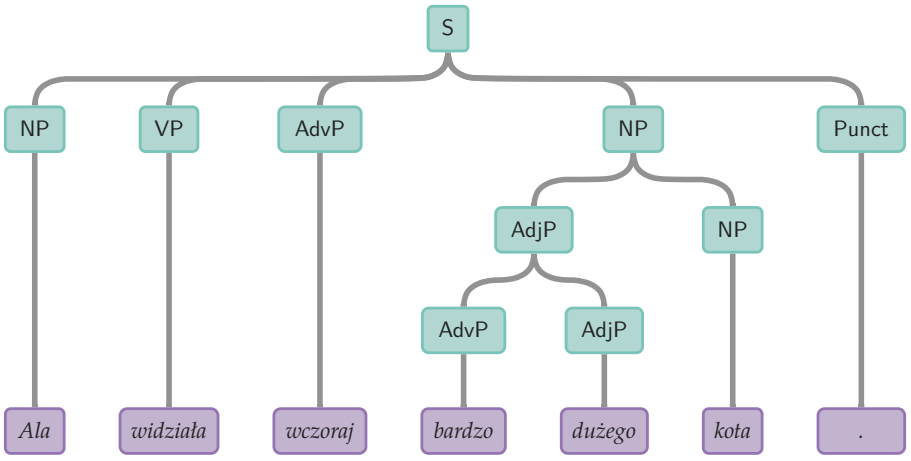
Poszczególne składniki w zdaniu są często nie tylko wyróżniane jak w przykładzie powyżej, ale również klasyfikowane ze względu na swoją wewnętrzną budowę i konstrukcje, w jakie mogą wchodzić z innymi składnikami. Typologia składników ponownie podyktowana jest przyjętą konwencją opisu składniowego. W drzewie składnikowym klasyfikacja składników odzwierciedlona jest poprzez przypisanie etykiet reprezentującym je węzłom nieterminalnym. Drzewo z rysunku 1.1 można rozszerzyć o etykiety wierzchołków na przykład tak, jak zilustrowano na rysunku 1.2. Węzeł obejmujący frazę *bardzo dużego* został tu opatrzony etykietą AdjP (od *adjectival phrase*, fraza przymiotnikowa), a jego rodzic odpowiadający frazie *bardzo dużego kota* — etykietą NP (*nominal phrase*, fraza rzeczownikowa). Korzeń drzewa, reprezentujący całe zdanie, otrzymał etykietę S (*sentence*). Podobnie jak w przypadku liści drzewa, nieterminale będą czasem dla uproszczenia utożsamiane z odpowiadającymi im składnikami.

Oprócz tego zostały wprowadzone dodatkowe węzły, obejmujące pojedyncze liście. W ten sposób również frazom składającym się z pojedynczych słów zostały przypisane typy (*kategorie składniowe*): *Ala* i *kota* to frazy rzeczownikowe, *widziała* to fraza czasownikowa (VP, *verbal phrase*), *wczoraj* oraz *bardzo* — frazy przysłówkowe (AdvP, *adverbial phrase*), zaś *dużego* — fraza przymiotnikowa. Kończący zdanie znak interpunkcyjny opatrzony został etykietą Punct (*punctuation*). Wierzchołek, którego jedynym dzieckiem jest pewien liść drzewa, nazywany jest *wierzchołkiem preterminalnym* lub preterminalem.

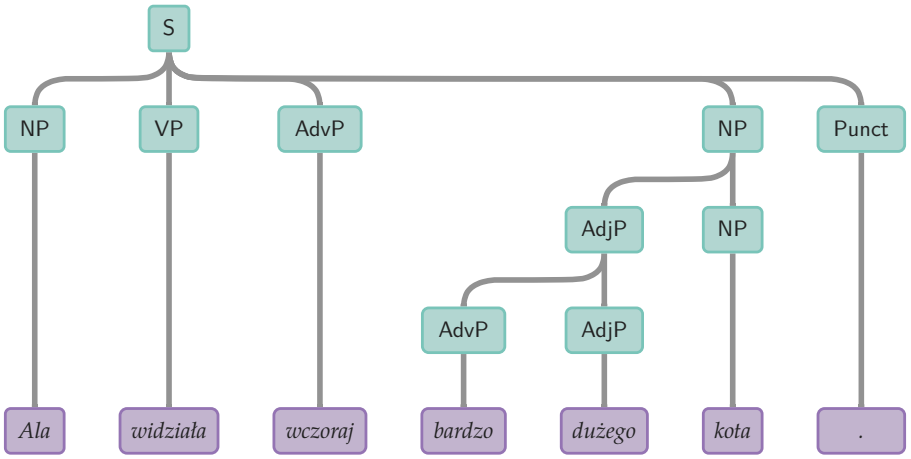
Drzewa z rysunków 1.1 oraz 1.2 można alternatywnie przedstawić w postaci nawiasowania ciągu segmentów, odpowiednio (*Ala widziała wczoraj* ((*bardzo dużego*) *kota*) .) oraz $((Ala)_{\mathsf{NP}}\ (widziała)_{\mathsf{VP}}\ (wczoraj)_{\mathsf{AdvP}}\ (((bardzo)_{\mathsf{AdvP}}\ (dużego)_{\mathsf{AdjP}})_{\mathsf{AdjP}}\ (kota)_{\mathsf{NP}})_{\mathsf{NP}}\ (.)_{\mathsf{Punct}})_{\mathsf{S}}$. W takiej notacji każdy składnik zostaje objęty parą nawiasów. W przypadku drzewa etykietowanego nawiasy mogą być opatrzone kategorią gramatyczną swojego składnika.

Centra składniowe

Kolejną informacją, o jaką może być rozszerzone drzewo składnikowe, jest oznaczenie *centrów składniowych* poszczególnych składników. W tym celu dokładnie jedno



Rysunek 1.2: Drzewo składnikowe z etykietami nieterminali.



Rysunek 1.3: Drzewo składnikowe z centrami składniowymi.

spośród dzieci każdego nieterminala zostaje wyróżnione jako *centralne*. Krawędź łączącą to dziecko z rodzicem również określać będziemy *centralną*. Ścieżka od dowolnego węzła wewnętrznego v w dół drzewa, prowadząca tylko przez węzły centralne, prowadzi wówczas do liścia $center(v)$ stanowiącego centrum składnika reprezentowanego przez v. Konkretny wybór centrów jest zależny od ujęcia lingwistycznego (o ile w ogóle zakłada ono takie rozróżnienie). Ogólnie rzecz ujmując, są nimi jednostki determinujące własności składniowe lub semantyczne całej frazy.

Przykładowe oznaczenie centrów składniowych w drzewie zilustrowane zostało na rysunku 1.3. Wierzchołki drzewa zostały wyrównane tak, aby krawędzie prowadzące do centralnych dzieci poprowadzone zostały pionowo. Z tak przedstawionego drzewa można na przykład odczytać, że centrum składnika *bardzo dużego kota* stanowi rzeczownik *kota*, a centrum całego zdania — czasownik *widziała*.

Drzewa nieciągłe

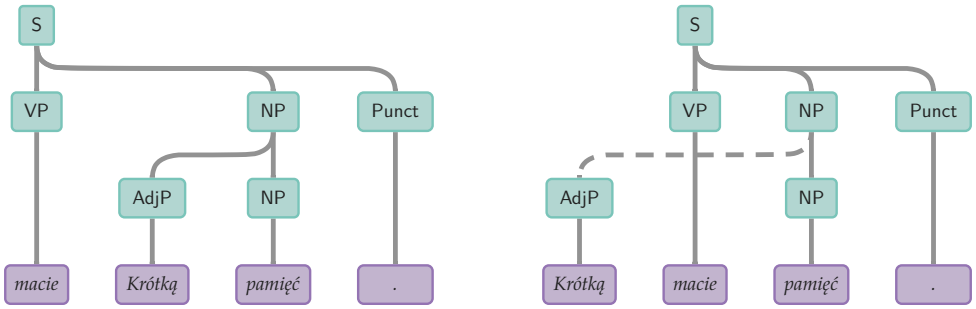
W ramach definicji drzewa składnikowego zostało wyżej sformułowane wymaganie, aby etykiety plonu każdego nieterminala, odczytane w porządku $\prec_d$, stanowiły spójny podciąg sekwencji segmentów składających się na wypowiedzenie. Istnieją jednak niekontrowersyjnie poprawne zdania, w przypadku których zachowanie tego warunku istotnie utrudnia stworzenie sensownej reprezentacji składnikowej. Na przykład dla zdania *Krótką macie pamięć*.[5] narzuca się analiza, w której słowa *Krótką* oraz *pamięć* tworzą jeden składnik, wypełniający pozycję biernikową otwieraną przez czasownik (co macie?). Taki składnik jest jednak *nieciągły*: zawiera „lukę" względem porządku tekstowego segmentów pozostawioną przez słowo *macie*.

Aby dopuścić występowanie w drzewie składników *nieciągłych* (nieciągłości), konieczna jest modyfikacja warunków dotyczących porządku na węzłach drzewa. Wobec porządku na liściach drzewa wymagamy teraz jedynie, aby zachowywał kolejność tekstową segmentów: jeśli $yield(v) = v_i \prec^* ... \prec^* v_{i+k}$, to $label(v_i) \prec_s^+ ... \prec_s^+ label(v_{i+k})$ (relację bezpośredniego poprzedzania w tekście zastępujemy jej domknięciem przechodnim). Wierzchołek v reprezentuje składnik nieciągły, jeśli $yield(v)$ zawiera liście $v_i \prec^* v_j$ takie, że $label(v_i) \not\prec_s label(v_j)$, ale dla każdego innego wierzchołka w wchodzącego w skład $yield(v)$, zachodzi $w \prec^* v_i$ lub $v_j \prec^* w$. Drzewo zawierające co najmniej jeden taki składnik określane jest drzewem nieciągłym.[6]

Przyjęcie możliwości wystąpienia składników nieciągłych wymaga zmiany definicji porządku $\prec_d$ na siostrzanych węzłach wewnętrznych. Chcąc naśladować definicję dla drzew ciągłych (zob. 1.1.1) z zamianą relacji $\prec_s$ na $\prec_s^+$, natrafimy na sprzeczności. Jako przykład niech posłuży zdanie $w_1w_2w_3w_4$ rozdzielone na dwa składniki: a obejmujący frazę w_1w_4 oraz b obejmujący w_2w_3. Próbując porównać odpowiedni segment końcowy z początkowym, otrzymamy $w_4 \not\prec_s^+ w_2$ oraz $w_3 \not\prec_s^+ w_1$ – czyli nie zachodzi ani $a \prec_d b$, ani $b \prec_d a$. Problemu takiego pozwala uniknąć na przykład przyjęcie, że dla $yield(v) = v_i \prec^* ... \prec^* v_{i+k}$ oraz $yield(w) = w_j \prec *\cdot.. \prec^* w_{j+l}$, $v \prec_d w$ wtw. $label(v_i) \prec_s^+ label(w_j)$ (porządek wyznaczony przez kolejność tekstową początkowych segmentów poszczególnych składników). W przypadku drzew składnikowych z centrami składniowymi przyjmujemy jednak jeszcze inną definicję: $v \prec w$ wtedy i tylko wtedy, gdy $label(center(v)) \prec_s^+ label(center(w))$. Innymi słowy, porządek węzłów jest zgodny z kolejnością tekstową ich centrów.

[5] Źródło zdania: Narodowy Korpus Języka Polskiego (Przepiórkowski *et al.*, 2012), https://nkjp.pl/.

[6] Terminy „nieciągłość", „nieciągły" w odniesieniu do drzew składnikowych odzwierciedlają przyjęte w anglojęzycznej literaturze przedmiotu *discontinuity/discontinuous*, które również tam nie są związane z pojęciem funkcji nieciągłej. Doskonale współgrają jednocześnie z potocznym rozumieniem tych słów – składnik nie stanowi ciągłego fragmentu zdania, występuje w nim przerwa.



Rysunek 1.4: Nieciągłe drzewo składnikowe.

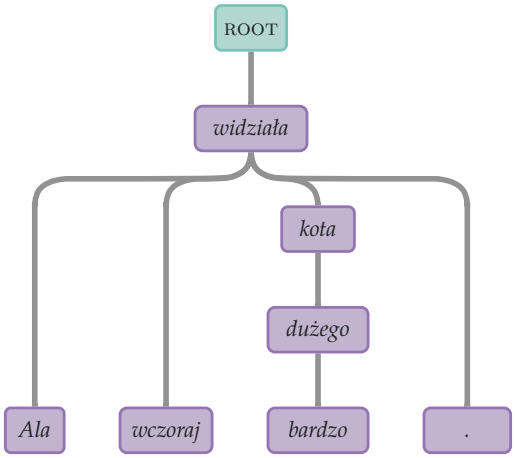
Rysunek 1.4 przedstawia drzewo z centrami składniowymi dla zdania zawierające nieciągłość: słowa *Krótką* i *pamięć*, składające się na frazę nominalną *Krótką pamięć*, są w porządku tekstowym rozdzielone czasownikiem *macie*. Drzewo z krawędziami centralnymi kreślonymi pionowo zostało narysowane na dwa sposoby. O przedstawieniu 1.4a można myśleć jako powstałym poprzez rekurencyjne rozmieszczenie węzłów wewnętrznych „od góry do dołu". Zaczynając od korzenia, dzieci każdego węzła uwzględniane są zgodnie z porządkiem na nieterminalach, a ich poddrzewa rysowane kolejno, od lewej do prawej. Po uzupełnieniu rysunku o terminale słowa w liściach, odczytane od lewej do prawej, mają kolejność inną niż tekstowa. W przedstawieniu 1.4b o graficznym układzie węzłów decyduje porządek tekstowy terminali – nieterminale zostały do nich dodane „od dołu do góry". W efekcie krzyżują się dwie krawędzie drzewa. Wyróżnienie linią przerywaną krawędzi między węzłami NP a VP jest konwencją wizualną. W niniejszej pracy do wizualizacji drzew nieciągłych używany będzie sposób 1.4b.[7]

1.1.2. Drzewa zależnościowe

Drzewo zależnościowe to drzewo ukorzenione, którego wierzchołki etykietowane są segmentami danego napisu. W odróżnieniu od drzewa składnikowego, nie zostają wprowadzone żadne dodatkowe węzły za wyjątkiem jednego, konwencjonalnie oznaczanego ROOT, stanowiącego korzeń drzewa i posiadającego dokładnie jedno dziecko (w ten sposób każdy węzeł odpowiadający segmentowi zdania ma w drzewie zależnościowym rodzica). Rodzic i dziecko w drzewie zależnościowym nazywane są też odpowiednio *nadrzędnikiem* i *podrzędnikiem*. O nadrzędniku i podrzędniku mówi się, że połączone są *relacją zależnościową*.

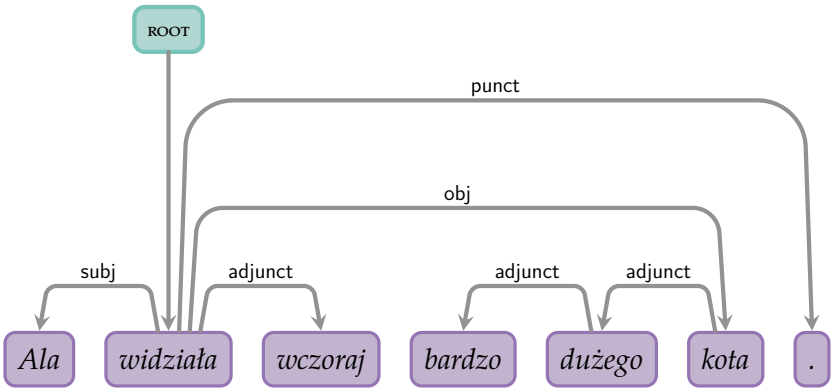
[7] Zastosowanie notacji nawiasowej dla takiego drzewa wymagałoby zmiany kolejności segmentów względem porządku tekstowego: $((\textit{macie})_{\mathsf{VP}}\ ((\textit{Krótką})_{\mathsf{AdjP}}\ (\textit{pamięć})_{\mathsf{NP}})_{\mathsf{NP}}\ (.)_{\mathsf{Punct}})_{\mathsf{S}}$.

Krawędzie drzewa zależnościowego odpowiadają powiązaniom składniowym pomiędzy poszczególnymi segmentami zdania. Przykładowe drzewo zależnościowe dla zdania *Ala widziała wczoraj bardzo dużego kota* przedstawiono na rysunku 1.5. Jedynym dzieckiem węzła ROOT jest czasownik *widziała*. Intuicyjnie ujmując, krawędzie drzewa odpowiadają sposobowi, w jaki kolejne słowa uzupełniają/rozszerzają zdanie. Dziećmi węzła *widziała* są: *Ala* (kto widział?), *wczoraj* (kiedy widziała?), *kota* (kogo widziała?) oraz, konwencjonalnie, końcowy znak interpunkcyjny. Kolejne dwie krawędzie łączą wierzchołek *kota* z *dużego* (jakiego kota?), a ten z kolei — z *bardzo* (jak dużego?). Z każdym z węzłów v drzewa zależnościowego można powiązać frazę $phrase(v)$ składającą się z etykiet wierzchołków poddrzewa węzła v, uporządkowanych w kolejności tekstowej. Na przykład węzłowi *kota* w przykładowym drzewie odpowiada w ten sposób fraza *bardzo dużego kota*.



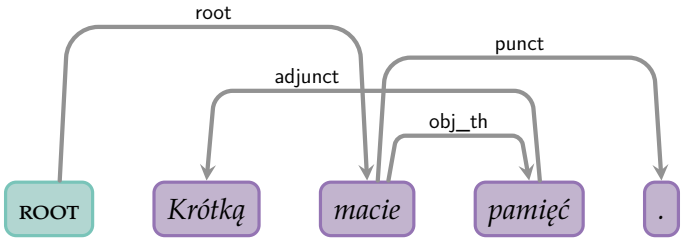
Rysunek 1.5: Drzewo zależnościowe.

Krawędziom w drzewie zależnościowym często przypisane są etykiety określające, jaki rodzaj relacji zależnościowej zachodzi pomiędzy połączonymi nimi węzłami, to jest jaką *funkcję gramatyczną* (zależnościową) pełni podrzędnik względem nadrzędnika. Rysunek 1.6 przedstawia opisywane powyżej drzewo z dodanymi typami relacji. Na przykład podrzędnikom głównego czasownika *widziała* zostały przypisane funkcje podmiotu (*Ala*, SUBJ), okolicznika (*wczoraj*, ADJUNCT) oraz dopełnienia bliższego (*kota*, OBJ). Drzewo zostało przedstawione graficznie w inny sposób niż na poprzednim rysunku — wierzchołki składające się na zdanie ułożone są w układzie liniowym zgodnie z porządkiem tekstowym, a krawędzie mają postać łuków narysowanych nad nimi, ze strzałkami skierowanymi od nadrzędnika do podrzędnika. Jest to jeden z typowych sposobów przedstawiania drzewa zależnościowego.



Rysunek 1.6: Drzewo zależnościowe z etykietami krawędzi, w układzie liniowym.

„Zależnościowym" odpowiednikiem nieciągłości jest *nieprojekcyjność*. W intuicyjnym sformułowaniu, drzewa nieprojekcyjnego nie da się narysować w układzie liniowym, z łukami krawędzi nad węzłami, bez przecinających się łuków. Bardziej formalnie oznacza to, że istnieją w nim krawędzie pomiędzy v_1 a v_2 i pomiędzy w_1 a w_2 (w dowolnym kierunku) takie, że $v_1 \prec w_1 \prec v_2 \prec w_2$ w porządku tekstowym[8] (na potrzeby definicji nieprojekcyjności przyjmujemy, że węzeł ROOT jest w tym porządku pierwszy). Przykładowe drzewo, którego nieprojekcyjność powodują przecięcia krawędzi adjunct z punct oraz adjunct z root, przedstawia rysunek 1.7. Dla większej klarowności węzeł ROOT został wyjątkowo umieszczony obok węzłów odpowiadających segmentom.



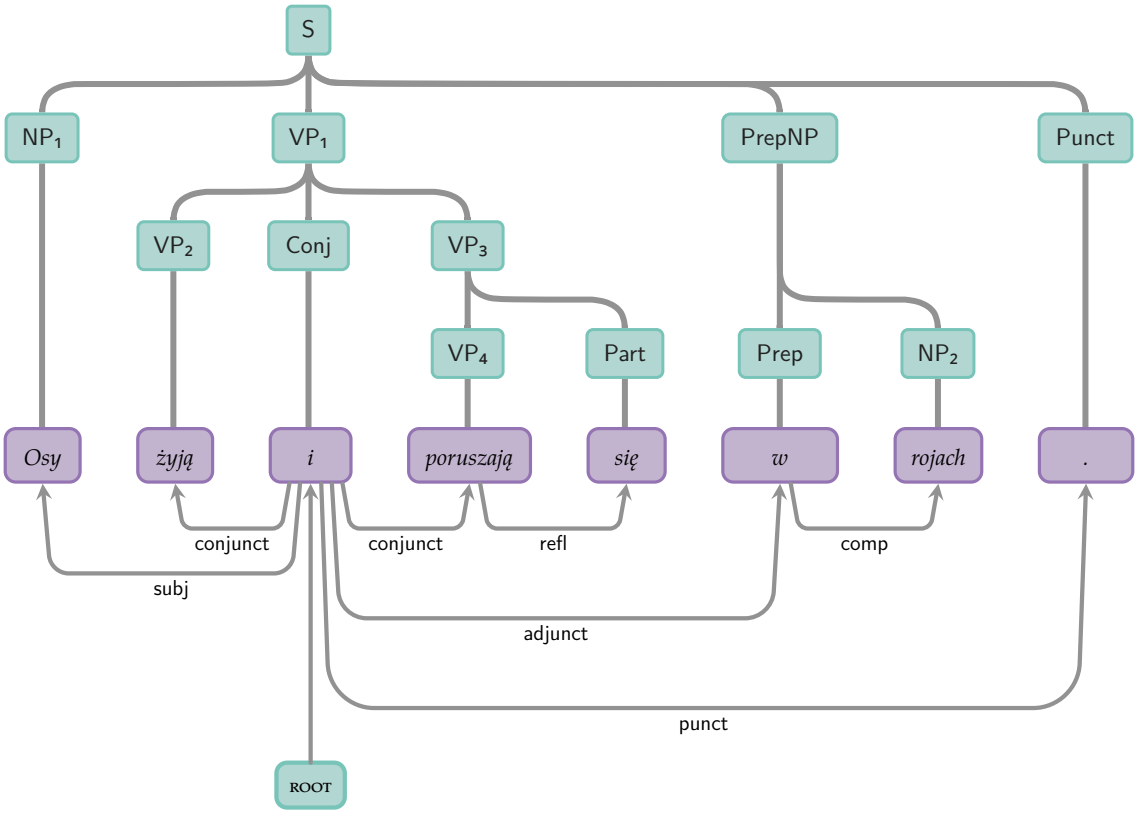
Rysunek 1.7: Nieprojekcyjne drzewo zależnościowe.

[8] Alternatywnie, jak podają m.in. Kübler *et al.* (2009), nieprojekcyjność zdefiniowana jest jako niespełnienie warunku projekcyjności sformułowanego następująco: Krawędź pomiędzy v_i a v_j jest projekcyjna, jeśli dla każdego $v_i \prec w \prec v_j$ istnieje ścieżka prowadząca do w (a) z v_i, jeżeli v_i jest nadrzędnikiem v_j (b) z v_j w przeciwnym przypadku. Drzewo zależnościowe jest projekcyjne, jeśli wszystkie jego krawędzie są projekcyjne.

1.1.3. Zależności a składniki

W ramach niniejszej pracy istotne będzie pojęcie *zgodności* drzewa składnikowego i zależnościowego dla tego samego ciągu segmentów. Najogólniej rzecz ujmując, składniki i frazy wyszczególnione w obu drzewach powinny sobie w pewien sposób odpowiadać, a przede wszystkim nie mogą nieść sprzecznych ze sobą informacji o budowie składniowej tekstu.

Zgodność drzew można bardziej formalnie zdefiniować następująco. Niech v będzie dowolnym nieterminalem drzewa składnikowego, a $children(v)$ — zbiorem jego dzieci. Jeśli $|children(v)| > 1$, to wymagamy, aby istniał dokładnie jeden węzeł $w_c \in children(v)$ taki, że dla każdego $w' \in children(w)$, $w' \neq w_c$, zachodzi $yield(w') = phrase(w_d)$ dla pewnego węzła w_d w drzewie zależnościowym (plon w' jest identyczny z pewną frazą w drzewie zależnościowym). Dla węzła w_c natomiast nie istnieje fraza drzewa zależnościowego odpowiadająca plonowi w_c. W przypadku, gdy drzewo składnikowe ma oznaczone centra składniowe, warunek zgodności jest rozszerzony następująco: wyróżniony jak powyżej wierzchołek $w_c \in children(v)$ jest centralnym dzieckiem v. Przykładowe zgodne ze sobą drzewa przedstawia ry-



Rysunek 1.8: Zgodne drzewa składnikowe i zależnościowe.

v		$w \in children(v)$	$yield(w)$	w_d: $yield(w) = phrase(w_d)$
S		NP_1	*Osy*	*Osy*
	$\star$	VP_1	*żyją i poruszają się*	×
		PrepNP	*w rojach*	*w*
		Punct	.	.
VP_1		VP_2	*żyją*	*żyją*
	$\star$	Conj	*i*	×
		VP_3	*poruszają się*	*poruszają*
VP_3	$\star$	VP_4	*poruszają*	×
		Part	*się*	*się*
PrepNP	$\star$	Prep	*w*	×
		NP_2	*rojach*	*rojach*

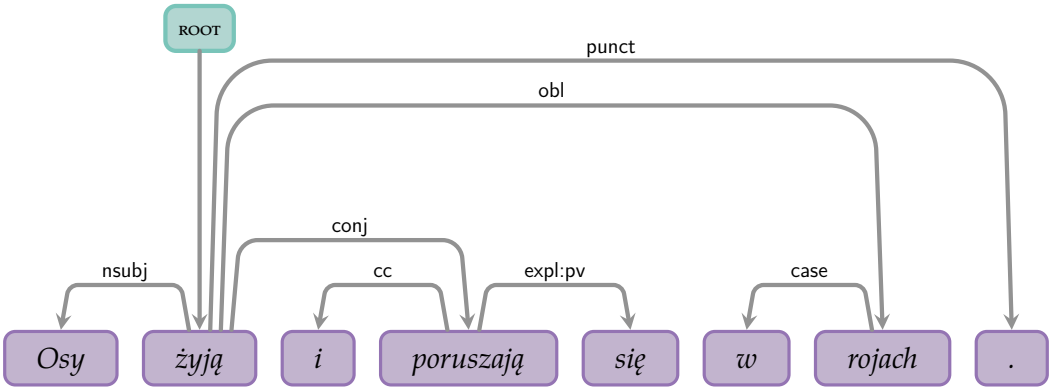
Tabela 1.1: Porównanie zgodnych drzew składnikowego i zależnościowego.

sunek 1.8.[9] Tabela 1.1 zbiera odpowiednie wierzchołki z obu drzew, pokazując, że faktycznie są one zgodne według przyjętej definicji. Centralne dziecko każdego nieterminala oznaczone jest w tabeli symbolem $\star$, natomiast × oznacza brak odpowiedniej frazy w drzewie zależnościowym.

Przykład drzewa zależnościowego niezgodnego z drzewem składnikowym z rysunku 1.8 przedstawia rysunek 1.9. Jest to drzewo skonstruowane zgodnie ze schematem Universal Dependencies (UD, de Marneffe *et al.* 2021). Węzły powodujące niezgodność podsumowane są w tabeli 1.2. Węzeł VP_1 ma aż dwoje dzieci bez odpowiadającej frazy w drzewie zależnościowym. Węzeł PrepNP ma co prawda dokładnie jedno takie dziecko, ale nie jest ono centralne. Rozbieżności te wynikają z innego sposobu anotacji koordynacji oraz fraz przyimkowych w schemacie UD.

Należy zwrócić w tym miejscu uwagę, że o ile liczba węzłów w drzewie zależnościowym dla danego zdania podzielonego na n segmentów jest ustalona i wynosi $n + 1$ (uwzględniając ROOT), o tyle drzewo składnikowe o tych samych segmentach jako liściach typowo ma ich więcej ($n+1$ węzłów oznaczałoby trywialne drzewo o jed-

[9] Liczby w indeksach dolnych przy niektórych etykietach nieterminali służą do ich rozróżnienia. Źródło zdania: Korpus Współczesnego Języka Polskiego, 2011–2020, (Marciniak *et al.*, 2023; Kieraś *et al.*, 2025), https://kwjp.pl.



Rysunek 1.9: Drzewo zależnościowe w schemacie UD.

v		$w \in children(v)$	$yield(w)$	w_d: $yield(w) = phrase(w_d)$
VP_1		VP_2	*żyją*	×
	⋆	Conj	*i*	*i*
		VP_3	*poruszają się*	×
PrepNP	⋆	Prep	*w*	*w*
		NP_2	*rojach*	×

Tabela 1.2: Porównanie niezgodnych drzew składnikowego i zależnościowego.

nym nieterminalu obejmującym wszystkie liście).[10] Ta większa liczba wierzchołków pozwala przypisać zdaniu strukturę składniową bogatszą niż ta wyrażalna w sposób zależnościowy.

W przytoczonym jako przykład drzewie składnikowym (rysunek 1.8) występuje w szczególności rodzaj konstrukcji składnikowej, która nie posiada odpowiednika zależnościowego odwzorowującego ją w pełni. Chodzi o frazy powstałe w wyniku „rozbudowywania" drzewa wokół centrum na więcej niż jednym poziomie. W przykładowym drzewie dzieje się tak w przypadku spójnika *i*. Na poziomie węzła VP_1 łączy on czasowniki *żyją* oraz *poruszają się* (taka konstrukcja składniowa nazywana jest koordynacją), a dopiero wyżej (w nieterminalu S) dołączone są podmiot *Osy* oraz okolicznik *w rojach*, współdzielone przez oba czasowniki (obie czynności wykonywane są przez osy i odbywają się w rojach). W drzewie zależnościowym nie ma

[10] W szczególności, jeśli pozwolimy, aby dowolny nieterminal posiadał tylko jedno dziecko, drzewo składnikowe może mieć dowolnie wiele węzłów.

sposobu, aby wyrazić taką hierarchiczną konstrukcję za pomocą krawędzi między segmentami.[11]

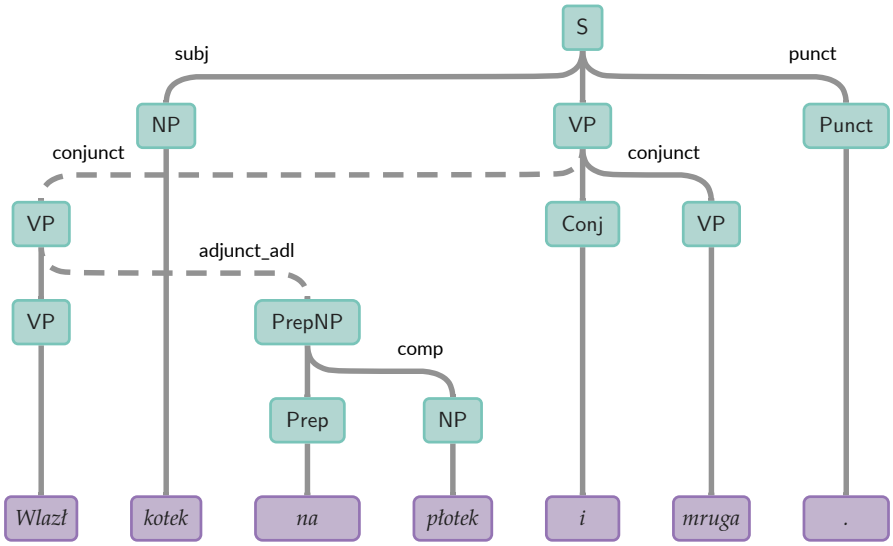
1.1.4. Drzewa hybrydowe

Informację zawartą w zgodnych ze sobą drzewie składnikowym T_c oraz etykietowanym drzewie zależnościowym T_d można przedstawić za pomocą jednej struktury. Jeśli w drzewie składnikowym nie są zaznaczone centra składniowe, możliwe jest jednoznaczne ich wyznaczenie na podstawie drzewa zależnościowego. Jako centralne zostają wówczas oznaczone dzieci nieposiadające odpowiadającej frazy w T_d dla węzłów o więcej niż jednym dziecku oraz jedyne dzieci dla pozostałych węzłów.

Jedyną informacją z T_d nieobecną w zgodnym z nim drzewie składnikowym z centrami są etykiety krawędzi (relacje zależnościowe). Etykiety te mogą zostać przeniesione na odpowiednie krawędzie T_c. Niech w będzie niecentralnym dzieckiem v w T_c, a w_d — węzłem T_d takim, że $yield(w) = phrase(w_d)$. Krawędź między v a w otrzymuje wówczas etykietę odpowiadającą funkcji zależnościowej w_d. Krawędzie centralne nie otrzymują etykiet.

Przykład otrzymanej w opisany powyżej sposób struktury, którą określać będziemy terminem *drzewo hybrydowe*, przedstawia rysunek 1.10. Wygodną cechą przyjętej tutaj konwencji graficznej jest fakt, że krawędzie prowadzące do niecentralnych dzieci, rysowane poziomo, odpowiadają wizualnie krawędziom drzewa zależnościowego w układzie liniowym (zob. rysunek 1.8). Można dzięki temu łatwo prześledzić

[11] Można ją natomiast do pewnego stopnia przekazać pośrednio poprzez etykiety krawędzi. W omawianym przykładzie wskazują na nią relacje zależnościowe conjunct opisujące koordynację.



Rysunek 1.10: Drzewo hybrydowe.

strukturę zależnościową, łącznie z relacjami zależnościowymi, zakodowaną w drzewie hybrydowym.

Eksperymenty przedstawione w niniejszej rozprawie będą skupione na automatycznym generowaniu struktur hybrydowych o opisanej powyżej budowie i własnościach. Narzędzie realizujące to zadanie pozwoli *de facto* na uzyskiwanie jednocześnie rozbiorów składnikowych i zależnościowych z zachowaniem spójności pomiędzy nimi.

1.2. Korpusy anotowane składniowo

Korpusy anotowane składniowo, nazywane również bankami drzew przez analogię do angielskiego terminu *treebank*, to zbiory zdań wraz z przypisanymi im drzewami składniowymi. Korpus taki może powstać w wyniku całkowicie ręcznego procesu, w którym drzewa są konstruowane przez anotatorów. Praca anotatorów może również być wspomagana użyciem parsera produkującego jedną lub wiele możliwych struktur składniowych. Anotacja polega wówczas na korekcie lub wyborze spośród wyników parsera. Tworzenie korpusu anotowanego składniowo może być wreszcie w pełni automatyczne, wyłącznie z wykorzystaniem parsera — korpus taki to w nazewnictwie anglojęzycznym *parsebank*.

Tworzenie banków drzew poprzez anotację wykonaną przez specjalistów jest kosztowne — zajmuje dużo czasu oraz wymaga nakładów finansowych. Koszt ten jest tym większy, im wyższa ma być jakość tworzonego zasobu. Przykładowym modelem pracy nad korpusem jest system dwóch anotatorów i jednego superanotatora, który rozstrzyga rozbieżności między ich decyzjami oraz ma szansę zauważyć ich niedopatrzenia — taki system wymaga jednak potrojenia wykonanej pracy w porównaniu z powierzeniem anotacji danego zdania jednej osobie. Korpusy ręcznie anotowane stanowią cenne i pożądane zasoby, które mogą posłużyć między innymi do trenowania i ewaluacji coraz lepszych narzędzi do przetwarzania tekstu.

W przypadku parsebanków jakość powstającego zasobu trudniej kontrolować, ale jest ona w oczywisty sposób zależna od jakości użytego narzędzia. Niewątpliwą zaletą korpusów tworzonych automatycznie jest łatwość ich pozyskiwania oraz możliwość wzbogacenia tekstów o anotacje w skali nieosiągalnej w przypadku procedur ręcznych. Informacje składniowe mogą być przydatne na przykład w badaniach lingwistycznych, ułatwiając przeszukiwanie tekstów pod kątem interesujących badacza zjawisk językowych.

Poniżej zostaną krótko opisane istniejące składniowe zasoby w językach polskim oraz niemieckim, które zostały wykorzystane w eksperymentach będących przedmiotem tej rozprawy.

1.2.1. Składnica

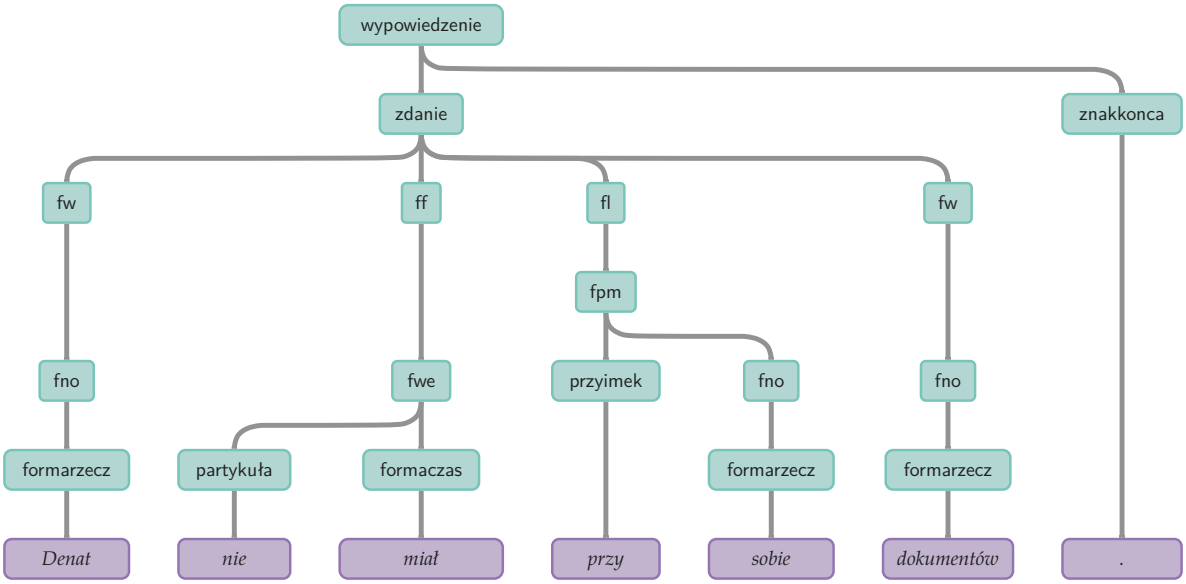
Bank drzew składnikowych Składnica[12] (Świdziński i Woliński, 2010; Woliński *et al.*, 2018; Woliński, 2019; Woliński i Hajnicz, 2021) powstał w wyniku ręcznego ujednoznacznienia przez anotatorów wyników automatycznej analizy składniowej wykonanej za pomocą regułowego parsera Świgra 2 (zob. 2.4). Wyjściowy materiał w procesie tworzenia Składnicy stanowiło 20 tysięcy zdań wybranych losowo z ręcznie anotowanego między innymi morfoskładniowo[13] podkorpusu Narodowego Korpusu Języka Polskiego (NKJP, Przepiórkowski *et al.* 2012). Spośród lasu wszystkich zgodnych z gramatyką Świgry 2 drzew dla danego zdania dwoje anotatorów wybierało jedno właściwe, a ich decyzje weryfikował superanotator. Warunkiem włączenia zdania do korpusu była jego zgodność z gramatyką. Rozszerzanie gramatyki o kolejne konstrukcje umożliwiało stopniowe powiększanie treebanku, który w chwili obecnej liczy około 14 tysięcy zdań.

Drzewa Składnicy posiadają oznaczenia centrów składniowych. Charakterystyczną cechą tych drzew, podyktowaną regułami gramatyki zaimplementowanej w parserze Świgra 2, jest struktura zbudowana wokół nieterminala zdanie, którego centralnym dzieckiem jest ff (fraza finitywna, w najprostszym ujęciu — główny czasownik zdania), a pozostałe dzieci, o etykietach fw (fraza wymagana) lub fl (fraza luźna) to odpowiednio argumenty frazy finitywnej oraz okoliczniki. Taka struktura koresponduje ze stosunkowo swobodnym szykiem zdania w języku polskim, w którym składniki mogą występować w różnej kolejności. Etykiety nieterminali są utrzymane w innej konwencji niż w przykładowych drzewach w punkcie 1.1 i nawiązują do polskich nazw poszczególnych kategorii składniowych.

Przykładowe drzewo ze Składnicy zostało przedstawione na rysunku 1.11. Etykieta wypowiedzenie zarezerwowana jest dla korzenia drzewa, który, oprócz centralnego składnika zdanie, obejmuje końcową kropkę (znakkonca). Pierwszym dzieckiem nieterminala *zdanie* jest fraza wymagana realizowana przez frazę nominalną (fno) składającą się z jednej formy rzeczownikowej *Denat* (formarzecz). Kolejna jest centralna fraza finitywna, czyli fraza werbalna (fwe) złożona z formy czasownikowej *miał* oraz partykuły przeczącej *nie*. Po niej następuje fraza luźna w postaci frazy przyimkowej (fpm) *przy sobie* oraz fraza wymagana *dokumentów*. Uwagę zwraca

[12] https://zil.ipipan.waw.pl/Składnica

[13] Przez *znakowanie morfoskładniowe* rozumiemy podział tekstu na segmenty oraz przypisanie każdemu segmentowi interpretacji w postaci formy hasłowej, klasy fleksyjnej (w dużym uproszczeniu: części mowy) oraz wartości adekwatnych kategorii gramatycznych takich jak liczba, rodzaj czy przypadek. Zamiennie używany będzie termin *znakowanie fleksyjne*. Określenia te nie są w pełni tożsame, ale różnice pomiędzy nimi można uznać za zaniedbywalne w kontekście niniejszej rozprawy.

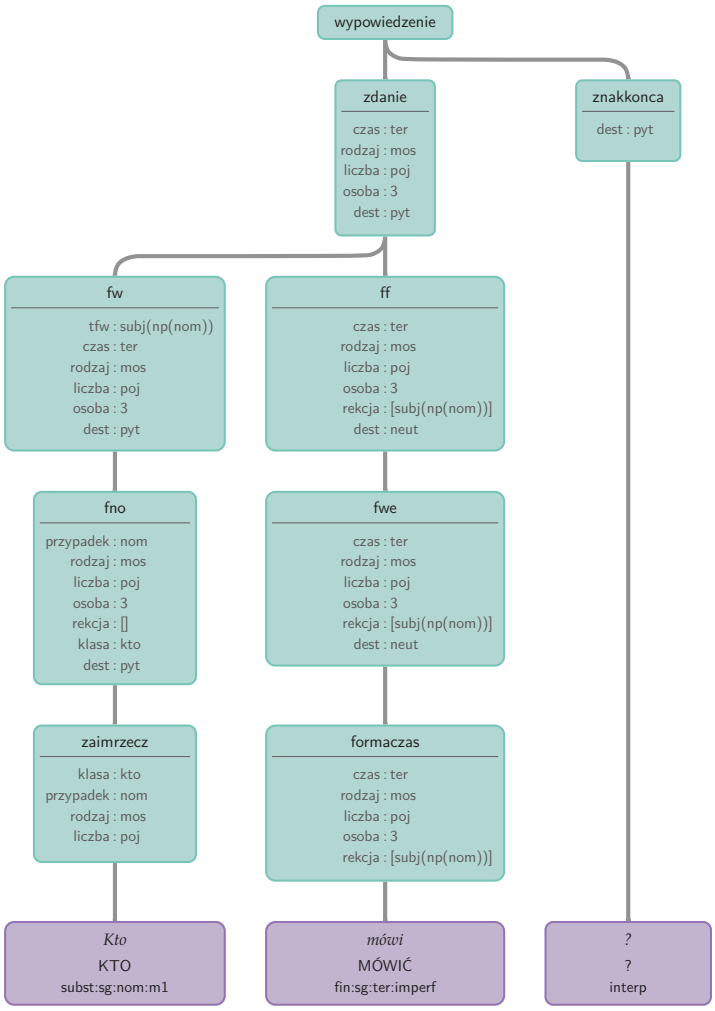


Rysunek 1.11: Drzewo ze Składnicy.

fakt, że nawet dla prostych, jednowyrazowych składników jak *Denat* czy *dokumentów* występuje pełen łańcuch krawędzi unarnych wynikających z reguł gramatyki.

W węzłach wewnętrznych drzew Składnicy przechowywane są, oprócz typów reprezentowanych przez nie fraz, zestawy przypisanych im atrybutów. Atrybuty te pochodzą z reguł gramatyki parsera Świgra i pozwalają na zwięzły zapis oddziaływań składniowych warunkujących poprawność wypowiedzenia (zob. 2.1). Ich obecność w zbudowanych przez parser drzewach daje z kolei między innymi możliwość szczegółowego przeszukiwania korpusu (na przykład zapytania o frazy rzeczownikowe w konkretnym przypadku). Terminale natomiast zawierają charakterystykę fleksyjną odpowiednich segmentów zaczerpniętą z NKJP — oprócz formy tekstowej odnotowana jest forma hasłowa oraz znacznik morfoskładniowy. Przykład drzewa wraz z (uproszczonym na potrzeby ilustracji) zestawem atrybutów przedstawia rysunek 1.12. Widać na nim na przykład uzgodnienie wartości osoby i liczby między podmiotem a orzeczeniem zdania — niedopuszczalne przez gramatykę jako wypowiedzenia byłyby napisy *Kto mówią?* czy *Ty mówi?*.

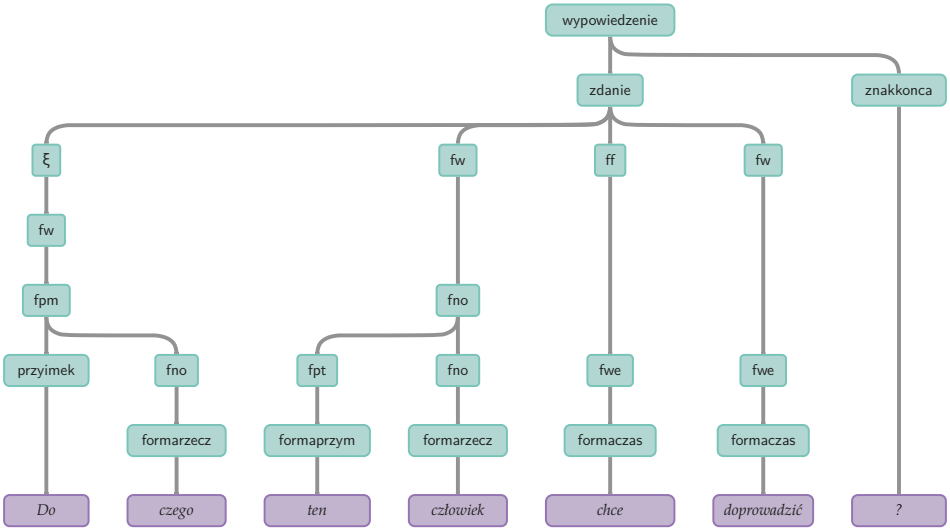
Składnica zawiera wyłącznie drzewa ciągłe, jednak część z nich reprezentuje nieciągłe struktury składniowe za pomocą specjalnego nieterminala ξ. Rysunek 1.13a pokazuje przykład takiego drzewa. Węzeł ξ reprezentuje frazę *Do czego*, która nie jest tak naprawdę argumentem czasownika *chce*, ale jego argumentu *doprowadzić* (*Do czego doprowadzić*, nie: *Do czego chce*). Miejsce, z którego został „wyjęty" wierzchołek ξ oznaczone jest odpowiednim atrybutem węzła — w przykładowym drzewie atrybut taki nosi węzeł fw odpowiadający frazie *doprowadzić*. Drzewo z wierzchołkiem ξ



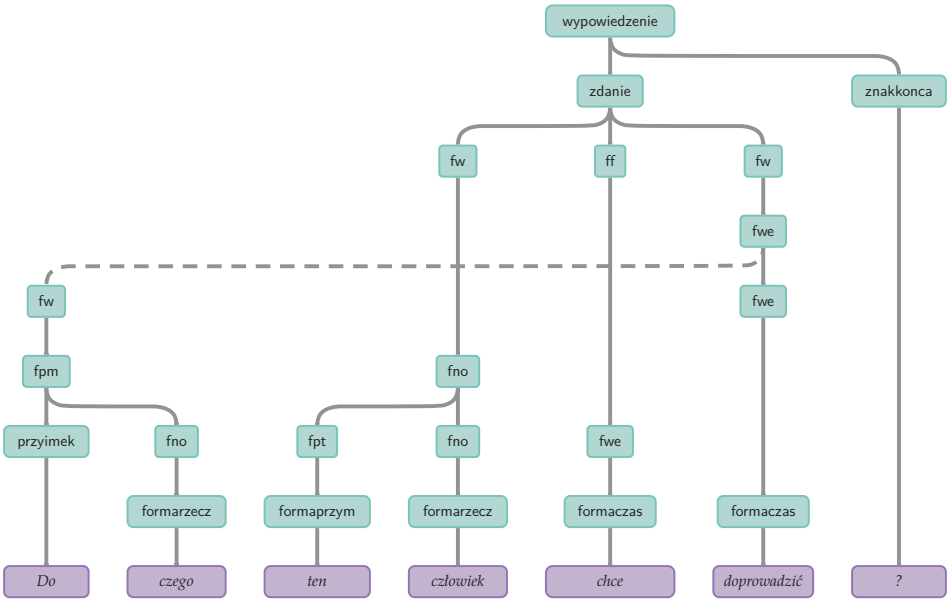
Rysunek 1.12: Drzewo ze Składnicy wraz z atrybutami węzłów.

można zatem w jednoznaczny sposób przekształcić w drzewo z nieciągłością. Rysunek 1.13b przedstawia drzewo nieciągłe dla przytoczonego wyżej przykładu.

W Składnicy zastosowany został sposób segmentacji tekstu obecny również w innych zasobach korpusowych dla języka polskiego (w tym w NKJP oraz PDB, który zostanie przedstawiony w punkcie 1.2.2) oraz zaimplementowany w analizatorze morfologicznym Morfeusz 2 (Kieraś i Woliński, 2017). W większości przypadków jest to podział na słowa tekstowe „od spacji do spacji" oraz oddzielenie znaków interpunkcyjnych, jednak w pewnych sytuacjach segmentacja jest drobniejsza. Charakterystyczne jest przede wszystkim traktowanie form czasownikowych w czasie przeszłym oraz w trybie przypuszczającym, polegające na wydzieleniu partykuły przypuszczającej *-by* oraz cząstek posiłkowych *-am*, *-eś*, *-śmy* itd. Jest to motywowane faktem, iż cząstki te wykazują pewną mobilność w obrębie wypowiedzenia: *Co zrobili·**ście**?* vs. *Co·**ście** zrobili?*; *Piotr śpiewał·**by**.* vs. *Piotr **by** śpiewał.*; *Chciała·**by**·**m***



(a)



(b)

Rysunek 1.13: Drzewo ze Składnicy z węzłem ξ oraz odpowiadające mu drzewo nieciągłe.

gdzieś wyjechać. vs. *Gdzieś* ***by·m*** *chciała wyjechać.* Innym przykładem segmentacji drobniejszej niż granice spacji, jest oddzielenie nieakcentowanej formy zaimka *-ń* od poprzedzającego przyimka: *do·**ń*** (= *do niego*), *przeze·**ń*** itp.

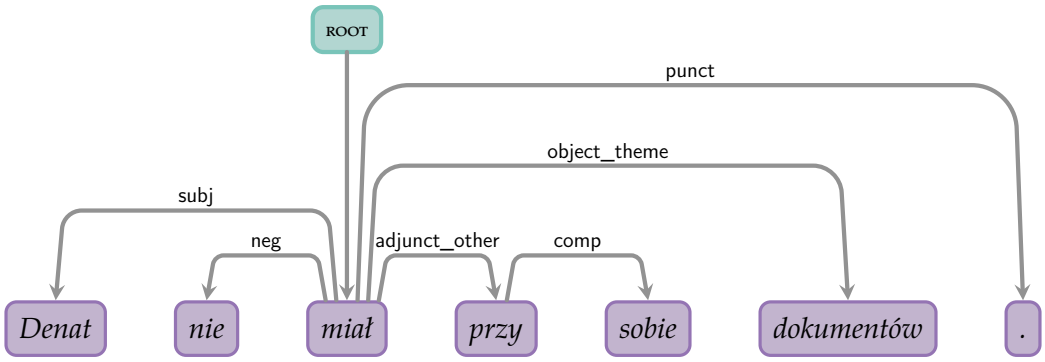
Z kolei przykładami segmentacji, w której znaki interpunkcyjne nie są traktowane jako oddzielne jednostki, są apostrofy oraz dywizy w formach fleksyjnych (*Lagrange'a*, *NFZ-u*), notacja dziesiętna z kropką lub przecinkiem (*3,14*, *2.5*) czy wyrazy takie jak *25-letni* czy *5-krotnie*. Wszystkie przytoczone napisy traktowane są jako jeden segment.

1.2.2. Polski Bank Drzew Zależnościowych

Polski Bank Drzew Zależnościowych[14] (Polish Dependency Bank, PDB, Wróblewska 2014, 2018) to pierwszy i do tej pory największy zasób tego typu dla polszczyzny. W swojej najstarszej wersji powstał na drodze konwersji liczącej wówczas około 8 tysięcy drzew Składnicy (Wróblewska i Woliński, 2012). Od tamtej pory ten anotowany zależnościowo korpus był systematycznie rozwijany, korygowany i rozszerzany o kolejne zdania (anotowane ręcznie ze wsparciem automatycznych narzędzi do parsowania) pochodzące ze zróżnicowanych źródeł, aby w momencie prowadzenia prac opisanych w dalszych rozdziałach niniejszej rozprawy osiągnąć wielkość około 22 tysięcy drzew. Przykładowe drzewo z treebanku PDB zamieszczone zostało na rysunku 1.14.

Istotną cechą korpusu PDB jest opracowany na potrzeby jego anotacji rozbudowany system relacji zależnościowych. W wersji wykorzystanej do eksperymentów, które zostaną przedstawione w niniejszej rozprawie, liczył on 72 różne etykiety krawędzi. Część nazw relacji, w szczególności tych opisujących okoliczniki czasownika, zawiera informację semantyczną. Przykładami takich relacji są adjunct_locat (okolicznik miejsca), adjunct_purp (okolicznik celu) czy adjunct_caus (okolicznik

[14] https://zil.ipipan.waw.pl/PDB



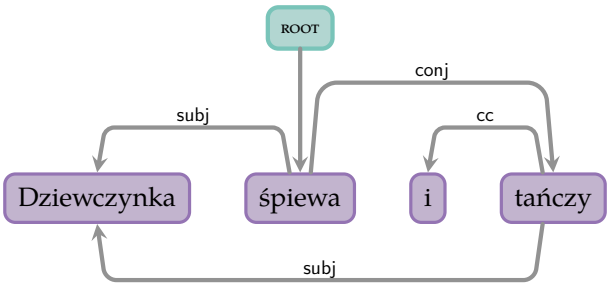
Rysunek 1.14: Drzewo z PDB.

przyczyny). PDB posiada również wersję zgodną ze wspomnianym wcześniej schematem opisu zależnościowego UD, zawierającą w szczególności dodatkowe krawędzie (enhanced dependencies,[15], Nivre *et al.* 2020), opisaną w artykule Wróblewskiej (2018).

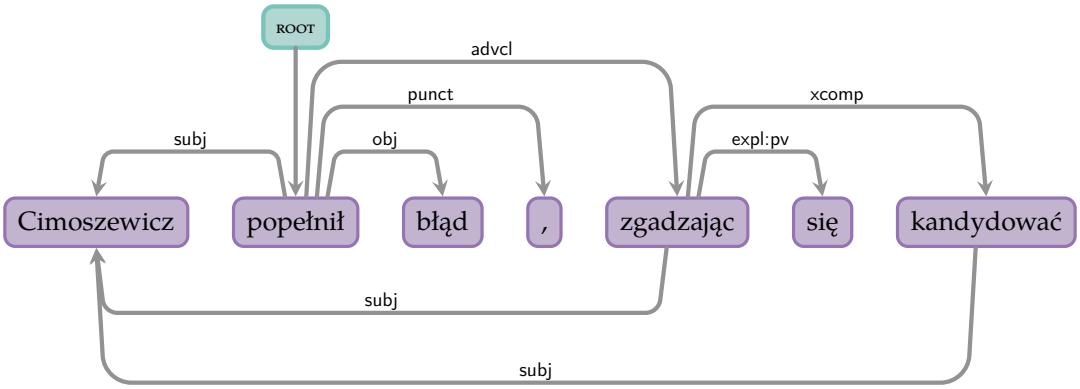
1.2.3. Zależnica

Bank drzew hybrydowych Zależnica został stworzony przez Marcina Wolińskiego przez połączenie informacji składniowej oraz konwencji znakowania Składnicy oraz PDB. Oba korpusy posiadają wspólną początkową historię oraz, co za tym idzie, duży zakres odpowiadających sobie (z dokładnością do zależnościowego lub składnikowego charakteru zasobów) rozstrzygnięć dotyczących opisu składnio-

[15] Dodatkowe krawędzie (*enhanced edges*) pozwalają na uogólnienie drzew zależnościowych do grafów kodujących dodatkowe informacje składniowe. Przykładem może być następujący graf (przytoczony za Wróblewską, 2018, krawędź dodatkowa nakreślona jest pod segmentami):



Krawędź subj prowadząca do segmentu *Dziewczynka* oznacza, że podmiot jest w tym zdaniu współdzielony przez oba skoordynowane czasowniki. Ze względu na specyfikę opisu koordynacji w UD, bez użycia dodatkowej krawędzi możliwe jest również odczytanie, w którym dziewczynka śpiewa, ale tańczy (nieznany) ktoś inny. W innym przykładzie (za Patejuk i Przepiórkowskim, 2018):



krawędzie dodatkowe posłużyły do zareprezentowania *explicite* zjawiska kontroli: rzeczownik *Cimoszewicz* uznawany jest nie tylko za podmiot głównego czasownika *popełnić*, ale również jego bezpośredniego podrzędnika *zgadzać* (*się*) oraz jeszcze głębiej zagnieżdżonego *popełnić*.

Parsowanie do takich reprezentacji grafowych znajduje się poza obszarem badawczym tej pracy.

wego. Proces łączenia był zatem w dużej mierze naturalny i wymagał tylko nielicznych zmian czy uspójnień.

W szczególności rozstrzygnięcia wymagała systematyczna rozbieżność w opisie konstrukcji biernych typu *był zrobiony*, *zostały napisane*. Wybór padł na konwencję stosowaną w Składnicy, według której forma finitywna czasownika (*był*, *zostały*) stanowi nadrzędnik imiesłowu biernego (*zrobiony*, *napisane*). W schemacie anotacji PDB relacja zależnościowa prowadzi w odwrotnym kierunku — od imiesłowu do czasownika posiłkowego.

Istotną (aczkolwiek niedotyczącą samej struktury drzew) modyfikacją wobec Składnicy była zmiana pochodzących z języka polskiego etykiet nieterminali na etykiety kojarzące się z nazwami angielskimi. Na przykład nazwa fno (fraza nominalna) została zastąpiona przez NP, fps (fraza przysłówkowa) — przez AdvP, formaczas (forma czasownikowa) — przez V, zdanie — przez S, itd.

Do powstałego treebanku hybrydowego została również włączona anotacja fleksyjna, która obecna jest w obu wyjściowych korpusach składniowych. Ta warstwa znakowania jest jednak drugorzędna dla prac relacjonowanych w niniejszej rozprawie.

1.2.4. TIGER

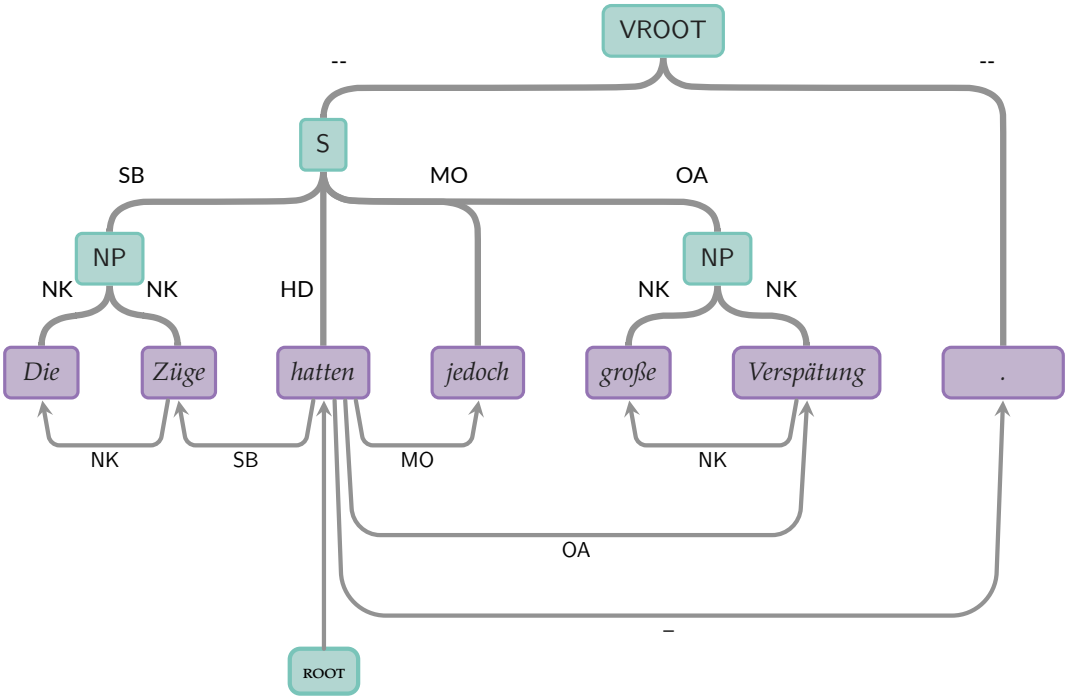
Aby rozszerzyć przeprowadzane eksperymenty i ewaluację poza zasoby polskojęzyczne, wykorzystany został niemiecki treebank TIGER (Brants *et al.*, 2004). Wybór ten podyktowany był możliwością stosunkowo łatwej konwersji tego korpusu do postaci drzew hybrydowych. Według mojego stanu wiedzy nie istnieją korpusy *explicite* anotowane składniowo strukturami hybrydowymi rozumianymi tak, jak w tej pracy. Dość częste są przypadki równoległego znakowania składnikowego i zależnościowego, jednak niekoniecznie jest ono spójne według definicji z punktu 1.1.3. Sensowne rozstrzyganie ewentualnych niespójności wymagałoby natomiast dobrej znajomości nie tylko przyjętych schematów anotacji, ale również ogólnej wiedzy o składni danego języka.

Znakowany składnikowo korpus tekstów prasowych TIGER zawiera struktury pokrewne drzewom hybrydowym: krawędzie są częściowo etykietowane funkcjami gramatycznymi. Schemat anotacji dopuszcza również, interesujące z mojego punktu widzenia, drzewa nieciągłe. Obecne są również oznaczenia centrów składniowych, nie dotyczy to jednak wszystkich nieterminali. Lista kategorii składniowych występujących w treebanku została umieszczona w tabeli A.1.

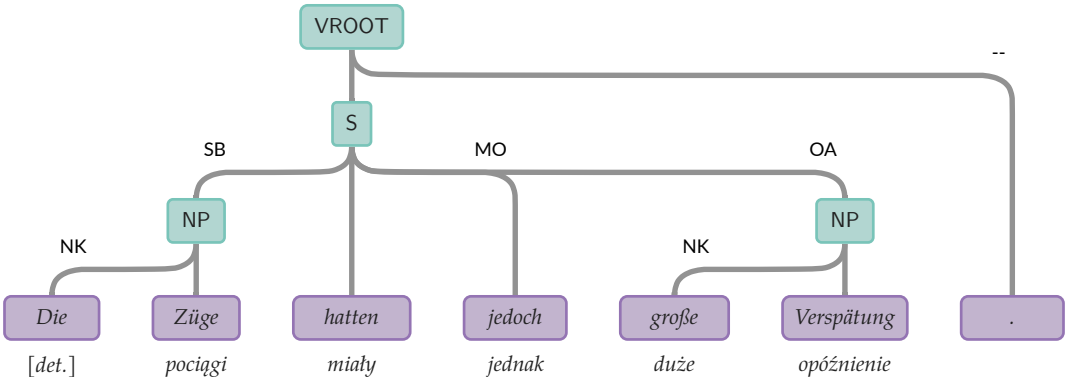
Zasobem umożliwiającym stosunkowo łatwe uzyskanie struktur hybrydowych odpowiadających założeniom tej pracy jest wersja korpusu TIGER przekształcona do postaci spójnej ze schematem UD (Falenska *et al.*, 2020). Zawiera ona drzewa

zależnościowe spójne (lub w systematyczny sposób przekształcalne do spójnych) ze strukturami składnikowymi banku drzew TIGER. Połączenie informacji składniowych zawartych w tych dwóch znakowaniach treebanku pozwala na uzyskanie zbioru drzew hybrydowych dla wypowiedzeń w języku niemieckim.

Rysunek 1.15 przedstawia przykładowe drzewa z omawianych zasobów. W strukturze składnikowej (rys. 1.15a) wszystkie krawędzie posiadają etykiety funkcji gramatycznych, jednak tylko główny czasownik *hatten 'miały'* został oznaczony explicite jako centralne dziecko swojego rodzica S za pomocą etykiety HD. Struktura zależnościowa (również rys. 1.15a, przedstawiona poniżej składnikowej) uzupełnia tę informację, pozwalając na wyznaczenie centrów składniowych pozostałych nieterminali. Struktura hybrydowa łącząca te dwa drzewa, pochodzące odpowiednio z wersji składnikowej oraz zależnościowej treebanku TIGER, została przedstawiona na rysunku 1.15b.



(a) Drzewa składnikowe oraz zależnościowe z korpusu TIGER.



(b) Drzewo hybrydowe.

Rysunek 1.15: Drzewa z korpusów niemieckojęzycznych.

Rozdział 2

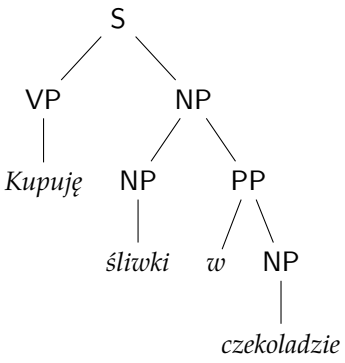
Parsowanie

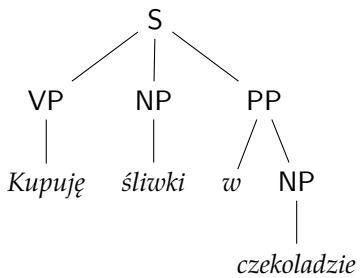
Analiza składniowa lub *parsowanie* polega na przypisaniu danym wejściowym struktury składniowej (rozbioru, analizy) reprezentującej wewnętrzną budowę tych danych. W tej rozprawie obiektem zainteresowania będzie sytuacja, w której danymi wejściowymi są wypowiedzenia, a proces parsowania wykonywany jest przez program komputerowy (*parser*, *analizator składniowy*). Wynikiem jego działania są drzewa opisujące budowę gramatyczną zadanych wypowiedzeń.

W regułowym podejściu do parsowania języka naturalnego podstawą jest gramatyka formalna definiująca zbiór poprawnych zdań. Zadaniem parsera jest wówczas przypisanie każdemu poprawnemu zdaniu jednego lub więcej drzewa odpowiadającego wyprowadzeniu zdania przy użyciu reguł gramatyki. Uzyskanie w wyniku więcej niż jednego drzewa możliwe jest, kiedy istnieje więcej niż jeden sposób wyprowadzenia zdania zgodny z gramatyką. Mamy wówczas do czynienia z *niejednoznacznością* składniową.[1]

[1] Przykładem może być szeroko występująca w różnych językach niejednoznaczność podłączenia frazy przyimkowej (*PP-attachment*), którą ilustruje skrajnie uproszczona gramatyka oraz dwa zgodne z nią drzewa dla zdania *Kupuję śliwki w czekoladzie*:

S → VP NP PP
S → VP NP
NP → NP PP
VP → *Kupuję*
NP → *śliwki*
NP → *sklepie*
NP → *czekoladzie*
PP → *w* NP





Pierwsze drzewo odpowiada narzucającemu się rozumieniu rozpatrywanego zdania (*w czekoladzie* są śliwki, a nie odbywa się czynność kupowania). Gdyby natomiast zdanie brzmiało *Kupuję śliwki w sklepie*, z drzew o analogicznym kształcie właściwsze wydawałoby się to drugie (*w sklepie* to lokalizacja zdarzenia, a nie cecha śliwek). Bez dodatkowego mechanizmu uwzględniającego znaczenia słów i wyrażeń, gramatyka powierzchniowoskładniowa akceptująca oba przytoczone zdania musi być obarczona podobną niejednoznacznością.

Dla zdań spoza języka gramatyki (niepoprawnych lub zawierających konstrukcje składniowe nieuwzględniane przez gramatykę) parser nie produkuje żadnego drzewa. Zadanie parsowania regułowego można zatem rozumieć jako udzielenie odpowiedzi na dwa pytania: (1) czy zadane zdanie należy do języka użytej gramatyki? oraz (2) jeśli tak, to jakie są możliwe rozbiory tego zdania według tej gramatyki?

Niewątpliwą siłą i zaletą podejścia regułowego jest oparcie na tworzonych przez ekspertów (lingwistów), często głęboko podbudowanych teoretycznie gramatykach. W idealnej sytuacji daje to pewność, że każde zaakceptowane przez parser wypowiedzenie jest składniowo poprawne, a przypisane mu drzewa potwierdzają to właściwymi wyprowadzeniami. Brak wyników parsowania dla wypowiedzeń niezgodnych z gramatyką jest istotną przeszkodą w praktycznym zastosowaniu takiego parsera na przykład jako etapu przetwarzania tekstu w bardziej złożonych zadaniach.[2] W miarę rozbudowywania zestawów reguł w celu uwzględnienia jak największej liczby zjawisk składniowych mogą pojawić się problemy wydajnościowe: zbyt wysokie wymagania pamięciowe lub nieakceptowalnie długi czas działania dla skomplikowanych wypowiedzeń. Liczba możliwych drzew potrafi też bardzo szybko rosnąć wraz z długością zdania, ponieważ gramatyki nie biorące pod uwagę znaczenia tekstu dopuszczają wszystkie gramatycznie poprawne analizy, również te mało prawdopodobne lub wręcz absurdalne z semantycznego i pragmatycznego punktu widzenia.

Możliwym rozwiązaniem tego ostatniego problemu jest wykorzystanie heurystyk lub modelu statystycznego (probabilistycznego), aby ograniczyć liczbę rozbiorów lub uszeregować je od najbardziej do najmniej pożądanych. Heurystyki mogą na przykład nadawać większy lub mniejszy priorytet regułom gramatyki lub wskazywać preferowane cechy otrzymywanych drzew. Do wytrenowania modelu statystycznego, przypisującego poszczególnym produkowanym przez parser drzewom prawdopodobieństwa, niezbędny jest bank drzew.

Podejściu regułowemu można przeciwstawić metody w pełni statystyczne, oparte na danych (ang. *data-driven*), w których jedynym źródłem informacji dla parsera są korpusy znakowane składniowo. Rozwiązania takie mogą w pewnym sensie naśladować metody regułowe — na przykład poprzez automatyczne stworzenie gramatyki w ustalonym formalizmie na podstawie drzew w korpusie, a następnie dodanie do niej modułu statystycznego. Taki parser posiada komponent regułowy, jakkolwiek nie stworzony przez eksperta (wiedza lingwistyczna jest jednak potrzebna do przeprowadzenia anotacji treebanku oraz opracowania procedury pozyskiwania

[2] Możliwym, częściowym rozwiązaniem jest celowe uwzględnienie w gramatyce wybranych konstrukcji substandardowych lub wręcz błędnych. Zwiększa to jednak liczbę i poziom skomplikowania reguł, a także stawia przed twórcami gramatyki pytanie o granicę takiego „rozluźniania" opisu.

gramatyki z danych). Parser oparty na danych może również nie zawierać *explicite* żadnego zestawu reguł, a jedynie ogólny algorytm parsowania oraz wytrenowany na danych korpusowych model służący do podejmowania decyzji, jaka konfiguracja działań doprowadzi do uzyskania najlepszego drzewa.

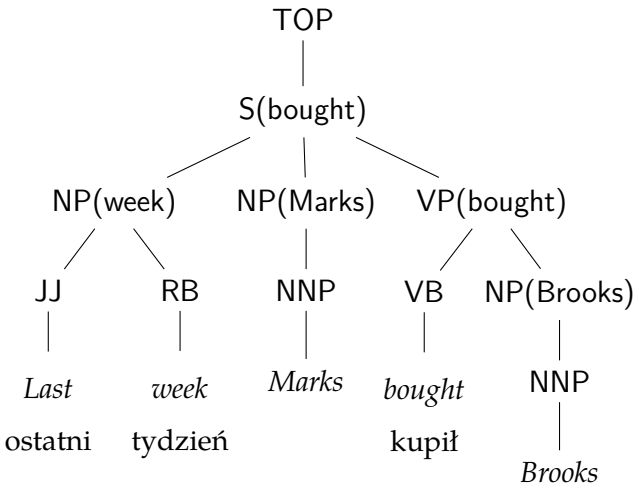
Zaletą rozwiązań opartych na danych jest potencjalnie większe niż dla ręcznie tworzonych gramatyk pokrycie przetwarzanych tekstów (a dla narzędzi działających bez reguł, wręcz stuprocentowe). Oznacza to w szczególności, że zadanie stawiane w tym przypadku parserowi jest inne niż w sytuacji użycia ręcznie stworzonej gramatyki. Nie ma tu już pytania o poprawność (względem gramatyki), istotne jest natomiast przypisanie jak najlepszej struktury każdemu wypowiedzeniu. Praktycznie niemożliwe do wyeliminowania jest w tym wypadku pojawianie się pewnych nieprawidłowości w wynikach — czy to w postaci przypisania wypowiedzeniu nieprawidłowej struktury przez statystyczny model, czy też wynikające z niegramatyczności samego zdania, dla którego mimo wszystko parser stara się wygenerować przynajmniej lokalnie poprawny rozbiór. Wyzwaniem jest minimalizacja skali tych błędów. Skuteczność parsera w podejściu *data-driven* zależy od jakości danych treningowych oraz samego algorytmu uczenia i parsowania. Niniejsza praca opisuje eksperymenty, których celem jest stworzenie skutecznie działającego, opartego na danych parsera produkującego drzewa hybrydowe takie, jak opisano to w punkcie 4.2.2.

W kolejnych częściach rozdziału zostaną krótko przedstawione opisane w literaturze różne podejścia do parsowania składnikowego (2.1), zależnościowego (2.2) oraz łączącego te rodzaje opisów (2.3).[3] W ostatnim punkcie (2.4) przywołane będą najważniejsze prace związane z formalnymi opisami oraz parserami dla polszczyzny.

2.1. Parsowanie składnikowe

Wczesne podejścia do parsowania składnikowego bazowały na efektywnych algorytmach uzyskiwania analiz zgodnych z regułami gramatyki bezkontekstowej (*context-free grammar*, CFG, Chomsky 1956; Backus 1959; Naur *et al.* 1960). Wykorzystywały one techniki programowania dynamicznego do przyrostowego budowania struktury danych przechowującej elementy informacji o możliwych rozbiorach (często więcej niż jednym). Proces ten przypomina stopniowe wypełnianie tabeli, dlatego rodzina tego typu algorytmów nazywana jest parsowaniem tablicowym (*chart parsing*, Kaplan 1973; Kay 1982). Na przykład w algorytmie

[3] Temat jest bardzo szeroki, więc omówienie w tej pracy nie ma szans go wyczerpać. Obszerne opracowanie znaleźć można na przykład w odpowiednich rozdziałach podręcznika Jurafsky'ego i Martina (2009; 2024).



„W zeszłym tygodniu Marks kupił Brooks"

Rysunek 2.1: Drzewo zleksykalizowane (za Collinsem, 1997).

Cocke'a-Youngera-Kasamiego (CYK lub CKY, Kasami 1965; Younger 1967) drzewa budowane są od liści do korzenia, czyli wstępująco (*bottom-up*). Z kolei parser operujący w przeciwnej kolejności (analiza zstępująca, *top-down*) zaproponował Earley (1968, 1970). Czas działania obu algorytmów jest w ogólnym przypadku sześcienny względem liczby segmentów w analizowanym wypowiedzeniu.

Możliwym rozwiązaniem problemu niejednoznaczności uzyskiwanych w taki sposób rozbiorów jest przypisanie regułom gramatyki prawdopodobieństw — mamy wówczas do czynienia z probabilistyczną gramatyką bezkontekstową (*probabilistic context-free grammar*, PCFG,[4] Booth 1969; Salomaa 1969), dzięki której dla każdego zgodnego z nią drzewa można obliczyć jego prawdopodobieństwo. Prace związane z trenowaniem gramatyk PCFG oraz parsowaniem z ich użyciem prowadzili m.in. Jelinek i Lafferty (1991) czy Stolcke (1995).

Innym istotnym rozszerzeniem gramatyk bezkontekstowych jest wzbogacenie nieterminali (oraz, co za tym idzie, reguł) o informację leksykalną (*lexicalised* PCFG, m.in. Collins 1997, 2003; Charniak 1997, 2000). Każdy węzeł wewnętrzny struktury składnikowej zostaje wówczas opatrzony, oprócz etykiety reprezentującej jego typ składnika, słowem stanowiącym jego centrum składniowe. Koncepcja ta jest bardzo zbliżona do oznaczania centralnego dziecka (zob. 1.1.1). Przykład drzewa zleksykalizowanego przedstawia rysunek 2.1.

Formalizm TAG (*tree adjoining grammar*, gramatyka dołączania drzew, Joshi 1985) jest przykładem gramatyki umiarkowanie kontekstowej (*mildly context-sensitive grammar*). Gramatyki takie, o sile wyrazu mieszczącej się pomiędzy CFG a gramatykami

[4] Używane jest też określenie *stochastic context-free grammar*.

kontekstowymi w hierarchii Chomsky'ego, pozwalają na uchwycenie pewnych zjawisk językowych niewyrażalnych poprzez gramatyki bezkontekstowe. Gramatyki TAG zostały, podobnie jak CFG, rozszerzone o leksykalizację (LTAG, Schabes *et al.* 1988) oraz warianty probabilistyczne (Resnik, 1992; Schabes, 1992).

Reguły gramatyk unifikacyjnych (*unification grammars*) operują, oprócz prostych kategorii gramatycznych, na strukturach atrybutów (*attribute-value matrices*, *feature structures*) powiązanych z tymi kategoriami. Struktura atrybutów to zestaw par atrybut–wartość opisujących cechy danej konstrukcji składniowej. Mogą to być na przykład cechy morfoskładniowe (liczba, rodzaj itd.) lub funkcje gramatyczne składników tworzących konstrukcję. Bardzo prostym przykładem może być następująca reguła:

$$\begin{aligned} \mathsf{NP}_1 \quad \rightarrow \quad & \mathsf{AdjP}\ \mathsf{NP}_2 \\ & \mathsf{NP}_2\ \textsc{case} = \mathsf{AdjP}\ \textsc{case} \\ & \mathsf{NP}_1\ \textsc{case} = \mathsf{NP}_2\ \textsc{case} \end{aligned}$$

— opisuje ona konstrukcję frazy nominalnej poprzez modyfikację frazą przymiotnikową, z wymaganiem, aby atrybuty CASE struktur odpowiadających składnikom NP_1, AdjP oraz NP_2 miały tę samą wartość.[5] Innymi słowy (jeśli przyjmiemy, że w całej hipotetycznej gramatyce atrybut CASE reprezentuje przypadek), wymusza zgodność wartości przypadka pomiędzy frazą nominalną, modyfikującą ją frazą przymiotnikową oraz powstałą z ich połączenia większą frazą nominalną.[6] Kluczowy jest tu mechanizm unifikacji struktur atrybutowych, który zapewnia spełnienie warunków narzuconych przez reguły na poszczególne wartości (oraz sam w sobie stanowi algorytm konstruowania odpowiednich struktur) w przypadku prawidłowych wyprowadzeń oraz odrzucenie wyprowadzeń niespełniających tych warunków.

Wśród gramatyk unifikacyjnych wymienić można gramatyki metamorficzne (Colmerauer, 1978) oraz ich późniejsze wcielenie Definite Clause Grammars (DCG, Pereira i Warren 1980), których reguły bezpośrednio przypisują wartości atrybutów poszczególnym węzłom drzewa. W formalizmie Lexical-Functional Grammmar (LFG, Bresnan 1982; Dalrymple 2023) struktury składnikowa (*c-structure*) oraz atrybutowa (*f-structure*) są ściśle powiązane, ale budowane równolegle, jako osobne warstwy reprezentacji zdania. Z kolei formalizm Head-Driven Phrase Structure

[5] Wyrażenie takiej zależności za pomocą gramatyki bezkontekstowej jest możliwe, ale wymagałoby wprowadzenia osobnych nieterminali reprezentujących frazy nominalne i przymiotnikowe w poszczególnych przypadkach oraz zdefiniowania osobnej reguły dla każdego przypadka. Próba opisania w ten sposób większej liczby zjawisk składniowych szybko spowodowałaby eksplozję rozmiaru gramatyki.

[6] Oczywiście w sensownej gramatyce reguła taka powinna wyrażać dodatkowo co najmniej analogiczne uzgodnienie liczby i rodzaju.

Grammar (HPSG, Pollard i Sag 1994; Müller *et al.* 2024) w ogóle nie przewiduje reguł przepisywania, a jedynie zbiór ograniczeń opisujących poprawne struktury atrybutowe. Z dwoma ostatnimi formalizmami ściśle związane są odpowiadające im rozbudowane teorie lingwistyczne.

Technika parsowania nazywana supertagowaniem (*supertagging*) stanowi uogólnienie *tagowania*, czyli zadania przypisania każdemu elementowi (słowu) w sekwencji wejściowej odpowiedniej etykiety (znacznika) — najczęściej chodzi o część mowy lub bardziej szczegółową charakterystykę morfoskładniową. W przypadku supertagowania znaczniki stają się dużo bardziej rozbudowane i reprezentują fragmenty (zleksykalizowanego) drzewa, z których można zbudować pełną strukturę składnikową. Na przykład Joshi i Bangalore (1994) oraz Bangalore i Joshi (1999) zaproponowali statystyczne modele do znakowania supertagami opartymi na formalizmie LTAG.

W przypadku supertagowania (super)znaczniki przypisywane segmentom są jednostkami występującymi w pewnym formalizmie (jak wspomniana powyżej LTAG) lub nim motywowanymi. Bardziej ogólną techniką jest *parsing-as-tagging* (parsowanie jako tagowanie), w którym poszukiwane etykiety, z których sekwencji zostaje odczytane wynikowe drzewo, mogą mieć bardzo różną postać i w szczególności pochodzić wyłącznie z danych uczących, bez bezpośredniego odwołania do żadnej gramatyki. Na przykład Collobert (2011); Collobert *et al.* (2011) stosują znakowanie typu IOB (zob. 3.4.3) do iteracyjnego wyznaczania składników na coraz wyższych poziomach drzewa. Gómez-Rodríguez i Vilares (2018) proponują procedurę odwracalnej dekompozycji drzewa na fragmenty służące jako znaczniki przypisywane segmentom przez model.

Ważnym przełomem w dziedzinie przetwarzania języka naturalnego było upowszechnienie się gęstych[7] dystrybucyjnych[8] reprezentacji wektorowych jednostek tekstu, w szczególności słów — tak zwanych *word embeddings*.[9] Jednym ze sposobów otrzymania takich wektorów jest obliczenie na dużym korpusie macierzy statystyk

[7] Reprezentacje gęste (*dense*), w odróżnieniu od rzadkich (*sparse*), kodują informację przy użyciu stosunkowo niewielkiej — względem wielkości zbioru reprezentowanych jednostek V — liczby parametrów. Ekstremalnym przykładem reprezentacji rzadkiej jest kodowanie *one-hot*, w którym i-tej jednostce odpowiada wektor długości $|V|$ składający się z samych zer poza liczbą 1 na pozycji i. Używanymi w praktyce wektorami rzadkimi były na przykład częstości wystąpień słów w poszczególnych tekstach korpusu referencyjnego ważone metodą tf-idf (*term frequency-inverse document frequency*) lub współwystąpienia słów ważone metodą PPMI (*Positive Pointwise Mutual Information*). Sama idea reprezentacji słów za pomocą wektorów sięga lat 50. ubiegłego wieku (Osgood *et al.*, 1957).

[8] Hipoteza dystrybucyjna w językoznawstwie związana jest z koncepcjami spopularyzowanymi przez Firtha ((1957)) i zakłada, że słowa o podobnych znaczeniach występują w podobnych kontekstach. Reprezentacje dystrybucyjne słów tworzone są zatem na podstawie ich otoczeń tekstowych.

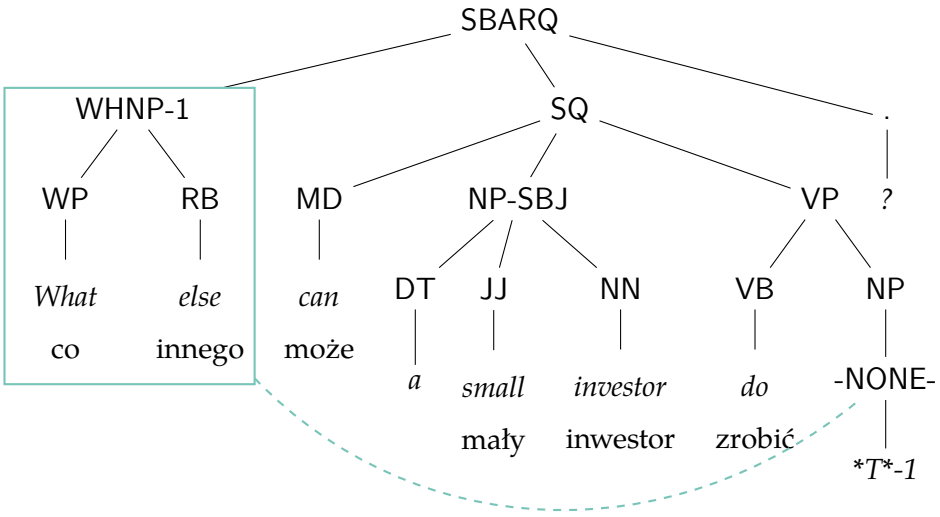
[9] Określenie tłumaczone czasem jako „zanurzenia słów" (w przestrzeni wektorowej).

dla słów (na przykład liczby współwystąpień w kontekście ustalonej długości) a następnie poddanie jej odpowiedniej faktoryzacji lub transformacji. W taki sposób powstały na przykład wektory GloVe (Pennington *et al.*, 2014). Innym podejściem jest uzyskanie zanurzeń słów jako części modelu trenowanego na zadaniu pomocniczym (często jest to przewidywanie kolejnego lub brakującego słowa w zadanym kontekście lub odwrotnie — kontekstu towarzyszącego słowu). Tego typu modele neuronowe opisali Bengio *et al.* (2003) czy Collobert *et al.* (2011). Dalsze prace w tym nurcie obejmowały między innymi wektory Word2vec (Mikolov *et al.*, 2013) oraz fastText (Bojanowski *et al.*, 2017) — te ostatnie uwzględniają, oprócz całych słów, ich krótsze fragmenty, co pozwala na obliczenie przybliżonych reprezentacji nieznanych słów.

Opisane do tej pory reprezentacje były statyczne — raz obliczone, stanowiły ustalone przypisanie wektorów słowom z określonego zbioru. Nowsze podejścia, możliwe dzięki rozwojowi efektywnych metod trenowania głębokich sieci neuronowych, polegają na wyuczaniu modeli zdolnych do produkowania „na bieżąco" reprezentacji słów (lub mniejszych jednostek, zob. 3.3) w kontekście fragmentu tekstu, na przykład zdania. Wśród modeli takich można wymienić między innymi ELMo (Peters *et al.*, 2018) wykorzystujący dwukierunkową architekturę LSTM (Hochreiter i Schmidhuber, 1997) czy BERT (Devlin *et al.*, 2019) oparty na architekturze transformer (Vaswani *et al.*, 2017). Również w tym przypadku modele trenowane są do zadania przewidywania kolejnego słowa w tekście (*causal language modelling*) lub uzupełniania luk w tekście (*masked language modelling*).

Sieci neuronowe znalazły oczywiście zastosowanie nie tylko jako źródło zanurzeń słów, ale także jako modele stanowiące podstawę działania parserów. Wspomniana wcześniej technika parsowania zastosowana przez Colloberta (2011) została zaimplementowana z użyciem sieci konwolucyjnej. Socher *et al.* (2013) zaproponowali rekurencyjną sieć neuronową aplikowaną wstępująco do każdego możliwego rozgałęzienia binarnego w celu wyznaczenia najlepszego drzewa. Wadą tego podejścia była niemożność zastosowania programowania dynamicznego, a co za tym idzie — zaporowa złożoność pełnego przeszukiwania przestrzeni rozwiązań, którą z tego powodu ograniczono heurystycznie.

W drugiej połowie lat 2010. w parsowaniu składnikowym były powszechnie wykorzystywane sieci LSTM, które zostały zastosowane w różnych zaproponowanych wówczas technikach. Cross i Huang (2016) opisują wstępujący parser stosowy operujący na częściowo zanalizowanych (niekoniecznie do pełnego poddrzewa) zakresach segmentów. W parserze stosowym Dyera *et al.* (2016) została użyta zstępująca strategia analizy. Stern *et al.* (2017) połączyli z kolei model neuronowy ze zstępującym parsowaniem tablicowym. Vinyals *et al.* (2015) oraz Choe i Charniak



„Co innego może zrobić drobny inwestor?"

Rysunek 2.2: Reprezentacja nieciągłej struktury składniowej w korpusie Penn Treebank.

(2016) ujęli problem parsowania w ramy odpowiednio generowania sekwencji (*sequence-to-sequence*) i modelowania języka zastosowanego do nawiasowego zapisu drzew (zob. 1.1.1).

W narzędziu Berkeley Neural Parser (Kitaev i Klein, 2018; Kitaev *et al.*, 2019) została zaimplementowana wersja algorytmu CKY, której nie towarzyszy gramatyka, ustalony jest jedynie zbiór terminali. Parser uwzględnia wszystkie możliwe produkcje, a ich prawdopodobieństw dostarcza model neuronowy operujący na reprezentacjach jednostek przetwarzanego tekstu pochodzących z modelu BERT.

Badania poświęcone parsowaniu składnikowemu koncentrują się tradycyjnie na strukturach ciągłych, wpisując to ograniczenie w proponowane rozwiązania. Automatycznej analizie uwzględniającej nieciągłości w drzewach poświęca się w literaturze przedmiotu wyraźnie mniej uwagi. Po części wynika to również z faktu, że obecność takich struktur w dostępnych korpusach nie jest oczywista. Niektóre schematy anotacji w ogóle nie przewidują reprezentowania takich konstrukcji, inne sygnalizują je w sposób niebezpośredni. Przykładem tego drugiego podejścia, obok wspomnianych w poprzednim rozdziale węzłów ξ w polskim banku drzew Składnica (por. 1.2.1), może być sposób przedstawiania struktur nieciągłych w treebanku Penn (Marcus *et al.*, 1993), zilustrowany za pomocą przykładowego drzewa z tego korpusu umieszczonego na rysunku 2.2. Specjalne, „puste" poddrzewo -NONE- sygnalizuje systematyczną dla języka angielskiego zmianę szyku w zdaniu pytajnym: przeniesiona na początek zdania fraza WHNP-1 stanowi dopełnienie kończącego je czasownika *do*.

Uzyskiwaniu struktur faktycznie nieciągłych (tak, jak zdefiniowano to w punkcie 1.1.1), poświęcona jest na przykład praca, w której Fernández-González i Martins (2015) opisali redukcję parsowania składnikowego do zależnościowego, kodując nieterminale jako etykiety krawędzi i sprowadzając nieciągłe składniki do krawędzi nieprojekcyjnych. Z kolei Coavoux i Cohen (2019) uogólnili parser stosowy, zastępując stos zbiorem umożliwiającym dostęp do każdego elementu. Neuronowy parser tablicowy zaproponowany przez Corro (2020) operuje na sekwencjach segmentów, w których dopuszczalna jest jedna nieciągłość. Fernández-González i Gómez-Rodríguez (2021) wprowadzają dodatkowy model *pointer network* permutujący segmenty wejściowe tak, aby możliwe było zastosowanie parsera składnikowego nieuwzględniającego nieciągłości — ostateczne drzewo powstaje przez odwrócenie permutacji z zachowaniem struktury nieterminali i krawędzi.

2.2. Parsowanie zależnościowe

Najstarszym znanym opisem języka naturalnego jest *Aṣṭādhyāyī* — datowane na połowę IV wieku p.n.e. dzieło starożytnego indyjskiego uczonego Pāṇiniego zawierające gramatykę zależnościową sanskrytu. Nowożytne teorie zależnościowe zostały natomiast zapoczątkowane przez prace Tesnière'a (1959), stanowiące podstawę dla rozwijanych później koncepcji takich jak Meaning-Text Theory (Mel'čuk, 1988) czy wywodząca się ze szkoły praskiej Functional Generative Description (Sgall *et al.*, 1986).

Jako przykłady regułowej analizy zależnościowej przytoczyć można bezkontekstowy formalizm Link Grammar (Sleator i Temperley, 1993) wraz z jego wersją probabilistyczną (Lafferty *et al.*, 1992), a także parser oparty na regułach Constraint Grammar (Karlsson *et al.*, 1995).

2.2.1. Parsery stosowe

Dwie główne w pełni statystyczne metody parsowania zależnościowego to podejście stosowe (*transition-based*) oraz grafowe (*graph-based*). Stosowy parser zależnościowy to rodzaj automatu, w którym sekwencja przejść (działań, czynności) prowadzi do zbudowania drzewa zależnościowego. Pojedynczy stan (konfiguracja) parsera składa się z bufora segmentów wejściowych (początkowo zawierającego całe zdanie do przetworzenia), stosu częściowo przetworzonych segmentów (początkowo zawierającego jedynie element ROOT) oraz zbioru utworzonych już krawędzi (początkowo pustego). Przejścia w najbardziej podstawowej wersji obejmują utworzenie krawędzi pomiędzy górnym elementem stosu a pierwszym elementem bufora (towarzyszy temu odpowiednia modyfikacja stosu i/lub bufora — zob. przykład

dalej) — LEFT-ARC oraz RIGHT-ARC w zależności od kierunku krawędzi — oraz przeniesienie pierwszego elementu bufora na wierzch stosu — SHIFT. Przebieg automatu kończy się wraz z opróżnieniem bufora. Tak zdefiniowany parser produkuje wyłącznie drzewa projekcyjne, a liczba jego przejść jest liniowa względem liczby segmentów w wejściowym zdaniu.[10]

Przykładowy przebieg działań parsera stosowego ilustruje tabela 2.1. Pierwszą czynnością jest w tym przypadku SHIFT — pierwszy segment *Kot* trafia z bufora na stos. Kolejne przejście — LEFT-ARC — polega na utworzeniu krawędzi skierowanej w lewo *Kot* ← *śpi*. Segment na szczycie stosu (tutaj: *Kot*) zostaje przy tym usunięty i nie będzie dalej przetwarzany. Po dwóch kolejnych przejściach SHIFT, przenoszących na stos segmenty *śpi* oraz *w*, następuje działanie RIGHT-ARC (*w* → *koszyku*). Modyfikuje ono stos i bufor w następujący sposób: pierwszy segment bufora (tutaj: *koszyku*) zostaje usunięty, a segment ze szczytu stosu (*w*) wraca na początek bufora. Eliminuje go dopiero kolejne działanie RIGHT-ARC, po którym przetwarzanie tego segmentu zostaje zakończone: ukorzenione w nim poddrzewo (*w koszyku*) jest kompletne

[10] Taka koncepcja parsera pokrewna jest parserom typu *shift-reduce* dla gramatyk bezkontekstowych. Na przejścia LEFT-ARC oraz RIGHT-ARC można spojrzeć jak na analogi przejścia SHIFT, z dwiema kluczowymi różnicami. (1) Przejście SHIFT może zbudować większą strukturę z jednego lub więcej dotychczas skonstruowanych poddrzew, przejścia -ARC zawsze łączą dokładnie dwa elementy. (2) Przejścia SHIFT dodają do wynikowej struktury węzły nieterminalne, stosowy parser zależnościowy buduje drzewo wyłącznie z segmentów wejściowych.

stos	**bufor**	**działanie**	**krawędź**
ROOT	*Kot*, *śpi*, *w*, *koszyku*, .	SHIFT	
ROOT, *Kot*	*śpi*, *w*, *koszyku*, .	LEFT-ARC	*Kot* ← *śpi*
ROOT	*śpi*, *w*, *koszyku*, .	SHIFT	
ROOT, *śpi*	*w*, *koszyku*, .	SHIFT	
ROOT, *śpi*, *w*	*koszyku*, .	RIGHT-ARC	*w* → *koszyku*
ROOT, *śpi*	*w*, .	RIGHT-ARC	*śpi* → *w*
ROOT	*śpi*, .	SHIFT	
ROOT, *śpi*	.	RIGHT-ARC	*śpi* → .
ROOT	*śpi*	RIGHT-ARC	ROOT → *śpi*
	ROOT	SHIFT	
ROOT		parsowanie zakończone	

Tabela 2.1: Przykładowy przebieg stosowego parsowania zależnościowego.

oraz został wyznaczony jego nadrzędnik *śpi*. Kolejne przejścia uzupełniają drzewo o ostatnie dwie krawędzie, wszystkie segmenty zostają w ten sposób przetworzone, a bufor opróżniony.

Uzyskanie poprawnego drzewa wymaga zastosowania właściwej sekwencji akcji. Teoretyczny, idealny parser korzysta z tzw. wyroczni (*oracle*), czyli funkcji, która każdemu zdaniu wejściowemu przypisuje odpowiedni ciąg działań. W praktyce o wyborze czynności parsera w danej konfiguracji decyduje model statystyczny wytrenowany na banku drzew. Aby uzyskać taki model, konieczne jest zakodowanie struktur zależnościowych z korpusu jako sekwencji przejść (par stan–działanie). Wejściem dla modelu jest wówczas reprezentacja pojedynczej konfiguracji parsera, wyjściem natomiast — właściwe w takiej konfiguracji działanie.

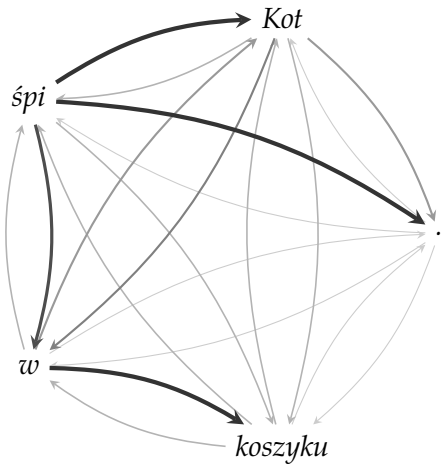
Algorytm stanowiący punkt wyjścia dla metod stosowych opisał Covington (2001). Yamada i Matsumoto (2003) zaproponowali jeden z pierwszych parserów stosowych, produkujący projekcyjne drzewa nieetykietowane oraz wykorzystujący klasyfikator Support Vector Machines do wyboru wykonywanych przejść. Nivre i Nilsson (2005) zastosowali odwracalne przekształcenie struktur nieprojekcyjnych do drzew bez krzyżujących się krawędzi, sprowadzając problem do parsowania projekcyjnego. W systemie MaltParser (Nivre *et al.*, 2006) zostały zaimplementowane dwa algorytmy: działający w czasie liniowym, ograniczony do struktur projekcyjnych (Nivre, 2003) oraz kwadratowy, uwzględniający nieprojekcyjność (Covington, 2001). Attardi (2006) rozszerzył algorytm Yamady i Matsumoto, wprowadzając dodatkowe typy przejść automatu pozwalające na uzyskanie drzew nieprojekcyjnych.

We wczesnych systemach parsowania stosowego reprezentacje stanów automatu były projektowane przez twórców tych narzędzi, którzy wybierali zestaw cech i elementów konfiguracji[11] składających się na taką reprezentację. Ważnym krokiem naprzód był Stanford Parser (Chen i Manning, 2014), w którym reprezentacje stanów wykorzystywały zanurzenia wektorowe wybranych segmentów, znaczników części mowy oraz relacji zależnościowych. Model wybierający działania automatu również był neuronowy. Oparte na sieciach neuronowych systemy parsowania stosowego rozwijali następnie między innymi Dyer *et al.* (2015), Andor *et al.* (2016), Kiperwasser i Goldberg (2016) oraz Kulmizev *et al.* (2019).

2.2.2. Parsery grafowe

W podejściu grafowym najlepsze drzewo dla danej sekwencji segmentów wybierane jest spośród wszystkich możliwych na podstawie wartości funkcji oceniającej. Funkcja ta agreguje (najczęściej sumuje) oceny poszczególnych składowych (kompo-

[11] Takich, jak części mowy kilku pierwszych segmentów na stosie i w buforze, ich pozycja w zdaniu, elementy wcześniej dodanych łuków, itp.



Rysunek 2.3: Przykładowe wagi przypisane krawędziom w parserze typu *arc-factored*.

nentów, *factors*) drzewa. Na przykład w parserach z faktoryzacją pierwszego rzędu podlegającym ocenie komponentem jest pojedyncza krawędź (stąd nazwa *arc-factored models*). Możliwa jest również faktoryzacja na bardziej złożone komponenty: w faktoryzacji n-tego rzędu przy ocenie danej krawędzi uwzględnianych jest również wybranych $n - 1$ krawędzi sąsiednich — taki komponent składa się łącznie z n krawędzi. Ocen poszczególnych składowych drzewa dostarcza model statystyczny wytrenowany na korpusie znakowanym zależnościowo.

Jeden z etapów działania parsera grafowego pierwszego rzędu został przedstawiony schematycznie na rysunku 2.3. Ocenie przez model została już poddana każda możliwa krawędź — im grubszy i ciemniejszy łuk na rysunku, tym wyższa ocena. Do ukończenia działania parsera pozostaje jeden krok — wybór drzewa obejmującego wszystkie wierzchołki i maksymalizującego sumę wag (ocen) krawędzi.

Jednym z pierwszych modeli grafowych był probabilistyczny algorytm projekcyjnego parsowania regułowego opracowany przez Eisnera (1996; 2000). Najbardziej prawdopodobne drzewo wyznaczane było przez dynamiczny parser wstępujący, dla którego autor zaproponował trzy różne techniki faktoryzacji i modelowania odpowiednich prawdopodobieństw. Na algorytmie Eisnera bazowały między innymi metody zaproponowane przez McDonalda *et al.* (2005b) czy Carrerasa (2007).

Parser MST (McDonald *et al.*, 2005a) zawdzięcza swoją nazwę algorytmowi maksymalnego drzewa rozpinającego (*Maximum Spanning Tree*), którego wersja dla grafów skierowanych (Chu i Liu, 1965; Edmonds, 1967) służy do wyboru drzewa optymalnego pod względem sumy wag krawędzi (być może nieprojekcyjnego). McDonald i Pereira (2006) oraz McDonald *et al.* (2006) rozszerzyli tę metodę m.in.

o faktoryzację drugiego rzędu,[12] możliwość przewidywania dodatkowych krawędzi (grafy zależnościowe) oraz etykietowanie krawędzi. System Mate (Bohnet, 2010) wykorzystuje algorytm wyszukiwania najlepszego drzewa z faktoryzacją drugiego rzędu (Carreras, 2007), przybliżoną technikę parsowania nieprojekcyjnego (McDonald *et al.*, 2006) oraz zoptymalizowaną pod kątem szybkości procedurę uczenia modelu.

Głębokie sieci neuronowe i wektorowe reprezentacje słów, które upowszechniły się w drugiej dekadzie XXI wieku, znalazły swoje zastosowanie również tutaj. Kiperwasser i Goldberg (2016) wykorzystali dwukierunkową sieć LSTM zarówno w parserze stosowym, jak i grafowym. Zbliżoną koncepcję parsowania grafowego opisali Zhang *et al.* (2017). Hashimoto *et al.* (2017) oraz Dozat i Manning (2017) zmodyfikowali architekturę grafową Kiperwassera i Goldberga, w szczególności wprowadzając do funkcji oceny krawędzi odpowiednio przekształcenie dwuliniowe i biafiniczne, uwzględniające iloczynową interakcję wektorów każdej rozpatrywanej pary nadrzędnik-podrzędnik. Warianty architektury Dozata i Manninga (*Deep Biaffine Attention*) zostały zaimplementowane na przykład jako część pakietu narzędzi Stanza (Qi *et al.*, 2020), a także w czeskim systemie UDPipe (Straka, 2018; Straka *et al.*, 2019). Wykorzystują ją również narzędzia zaprojektowane z myślą o zastosowaniach wielojęzycznych — trenowany łącznie na danych w 75 językach UDify (Kondratyuk i Straka, 2019) czy Trankit (Nguyen *et al.*, 2021), w którym współdzielony model połączony jest ze specyficznymi dla poszczególnych języków przekształceniami jego parametrów (adapterami). Wśród nowoczesnych, neuronowych parserów grafowych wymienić należy również polski system wstępnego przetwarzania tekstu COMBO (Rybak i Wróblewska, 2018a,b; Klimaszewski i Wróblewska, 2021), oferujący między innymi możliwość generowania grafów (enhanced dependencies). Ciekawą analizę i porównanie skuteczności różnych technik stosowanych w parsowaniu grafowym z użyciem sieci neuronowych przeprowadzili Grünewald *et al.* (2021).

2.3. Podejścia łączone

W literaturze przedmiotu znaleźć można prace, w których analiza składnikowa oraz zależnościowa zostały w mniej lub bardziej ścisły sposób powiązane. Jednym z takich nurtów jest tworzenie modeli, które równolegle produkują drzewa obu typów. Podejścia takie nie wymagają ani nie zapewniają systematycznej spójności pomiędzy anotacją zależnościową a składnikową danych. Ponieważ jednak moduły

[12] McDonald i Pereira (2006) dowodzą, że nieprojekcyjne parsowanie MST z faktoryzacją drugiego rzędu jest NP-trudne i dlatego stosują algorytm przybliżony.

parsera odpowiedzialne za każdą z nich trenowane są łącznie i współdzielą część parametrów, każdy z nich ma szansę skorzystać podczas uczenia z informacji składniowej wyrażonej na oba sposoby. Takie równoległe systemy parsowania zaproponowali na przykład Strzyz *et al.* (2019), a także Fernández-González i Gómez-Rodríguez (2022). Ściślejsze powiązania pomiędzy strukturami składnikowymi a zależnościowymi przyjmują Zhou i Zhao (2019) oraz Zhou *et al.* (2020), faktycznie operując na połączeniu tych reprezentacji. W tym przypadku jednak również nie jest zakładana pełna, systematyczna zgodność pomiędzy nimi.

2.4. Gramatyki i parsery dla języka polskiego

2.4.1. Podejścia składnikowe

Pierwszym obszernym formalnym opisem składnikowym języka polskiego była gramatyka metamorficzna Szpakowicza (1978; 1983). Podejścia do jej implementacji komputerowej oraz towarzyszące im, szczególnie w latach 80., trudności techniczne wynikające z ograniczeń sprzętowych, opisuje Bień (2009). Rozwinięcie zawartych w opisie Szpakowicza koncepcji stanowiła pozostająca w tym samym formalizmie gramatyka Świdzińskiego (GFJP, 1992). GFJP doczekała się szeregu implementacji w postaci parserów AMOS (Bień, 1996, 1997), AS (Bień *et al.*, 2001) oraz Świgra (Woliński, 2004, 2005). Wszystkie one posługiwały się pokrewnym gramatykom metamorficznym formalizmem DCG. Ogrodniczuk (2005, 2006) zmodyfikował gramatykę Świgry, między innymi rozbudowując opis fraz nominalnych oraz wprowadzając reguły dla konstrukcji liczebnikowych.

GFJP stanowiła również punkt wyjścia dla gramatyki parsera Świgra 2 (Woliński, 2015, 2019), będącej *de facto* nową gramatyką DCG dla języka polskiego, obejmującą w szczególności elastyczny mechanizm dopasowywania schematów składniowych słownika walencyjnego Walenty (Hajnicz *et al.*, 2016; Przepiórkowski *et al.*, 2017). Bank drzew, w którego tworzeniu wykorzystany był ten parser, opisany już został w punkcie 1.2.1. Z kolei Woliński i Rogozińska (2016) oraz Woliński (2019) opisują eksperymenty rozszerzające Świgrę 2 o wytrenowany na tym korpusie komponent probabilistyczny wybierający najbardziej prawdopodobny z lasu rozbiorów parsera.

Wśród innych powstałych narzędzi do parsowania języka polskiego wymienić można parser DCG włączony do systemu POLINT (Vetulani, 2004; Vetulani *et al.*, 2010), a także parser Gralińskiego (2002; 2007) wchodzący w skład systemu tłumaczenia maszynowego POLENG (Jassem, 2006). Prace związane z parsowaniem polszczyzny prowadził również M. Rudolf (zob. Derwojedowa *et al.* 2003).

2.4.2. Podejścia zależnościowe

Jednym z pierwszych opisów polszczyzny w duchu zależnościowym była gramatyka Klemensiewicza (1969), do dziś stanowiąca w Polsce podstawę szkolnego nauczania o składni. Późniejsze prace nad zależnościowym ujęciem składni polskiej prowadzili Świdziński (1989) oraz Derwojedowa (2011) — to ostatnie opracowanie skupione było na opisie fraz liczebnikowych. Autorem pierwszego polskiego parsera zależnościowego (regułowego) oraz opartej na teorii Meaning-Text gramatyki stanowiącej podstawę jego działania jest Obrębski (2002, 2003).

Wczesne prace związane z w pełni statystycznym parsowaniem zależnościowym na gruncie polskim obejmowały stworzenie pierwszej wersji Polskiego Banku Drzew Zależnościowych (Wróblewska, 2012) oraz wytrenowanie na nim modeli (Wróblewska i Woliński, 2012). Rozprawa doktorska Wróblewskiej (2014) obejmuje dalsze eksperymenty w tym zakresie, w szczególności eksplorację możliwości poprawy skuteczności trenowanych parserów poprzez wzbogacenie danych uczących o drzewa rzutowane z obcojęzycznych korpusów znakowanych zależnościowo. Dalszy rozwój PDB (Wróblewska, 2018) dostarczał coraz bardziej rozbudowanego materiału treningowego dla kolejnych systemów, w tym parsera COMBO (Klimaszewski i Wróblewska, 2021).

2.4.3. Inne ujęcia

Szczegółowe opracowanie opisu języka polskiego w ramach teorii HPSG przedstawili w cyklu prac Mykowiecka (1999), Przepiórkowski (1999), Kupść (2001), Marciniak (2001) oraz Przepiórkowski *et al.* (2002). Pracom teoretycznym towarzyszyła częściowa implementacja, mająca jednak charakter środka weryfikacji poprawności i spójności powstałej formalizacji, a nie narzędzia o zastosowaniu praktycznym.

Z kolei gramatyka POLFIE (Patejuk, 2015; Patejuk i Przepiórkowski, 2017) jest zaimplementowanym w środowisku XLE (Xerox Linguistic Environment) opisem polszczyzny w formalizmie LFG. Wyjściowym materiałem dla POLFIE były reguły Świgry 2 oraz elementy wspomnianego wyżej opisu HPSG (zob. Patejuk i Przepiórkowski 2012). Następnie gramatyka ta została między innymi zintegrowana ze słownikiem walencyjnym Walenty (Patejuk, 2016) oraz rozszerzona o rozbudowany opis konstrukcji skoordynowanych (Patejuk, 2015). Wraz z gramatyką powstał korpus wyprodukowanych z jej użyciem a następnie ręcznie ujednoznacznionych analiz (Patejuk i Przepiórkowski, 2014). Zasób ten został dodatkowo skonwertowany do struktur zależnościowych (Patejuk i Przepiórkowski, 2018; Przepiórkowski i Patejuk, 2020) w podstawowym (drzewiastym) oraz rozszerzonym (grafowym) schemacie UD.

Na koniec przytoczyć można systemy parsowania płytkiego (*shallow parsing*). Podejście to nie zostało przywołane we wcześniejszych częściach tego rozdziału, ponieważ nie prowadzi do powstania kompletnych rozbiorów, przez co znajduje się niejako poza obszarem tematycznym tej pracy. Zamiast tego wydzielane są niektóre jednostki składniowe, na przykład grupy rzeczownikowe czy czasownikowe. Do parsowania płytkiego można zastosować formalizmy nierekurencyjne, o mniejszej sile wyrazu, na przykład kaskady reguł typu bezkontekstowego. Dziś w zasadzie uznawane za przestarzałe, podejścia takie były swego czasu popularne jako praktyczna, mniej wymagająca obliczeniowo alternatywa dla pełnego parsowania. Jako przykłady płytkich parserów dla języka polskiego przywołać można narzędzia Spejd (Przepiórkowski, 2008) oraz Puddle (Graliński *et al.*, 2013). Pierwsze z nich zostało wykorzystane do wzbogacenia znakowania NKJP o warstwę grup składniowych.

Rozdział 3

Parser składnikowo-zależnościowy Hydra

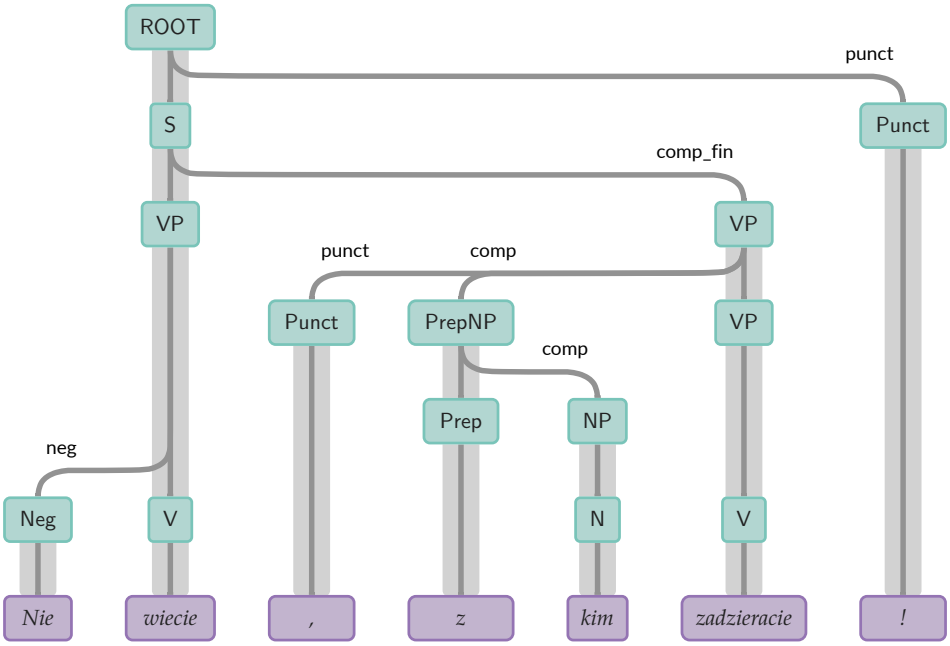
W tym rozdziale przedstawiamy parser hybrydowy Hydra,[1] którego implementacja stanowiła jeden z celów postawionych we wstępie do rozprawy. W pierwszej kolejności, w punkcie 3.1, prezentowana jest opracowana technika parsowania składniowego zbudowana wokół koncepcji kręgosłupów składniowych. Punkt 3.2 stanowi opis architektury modelu neuronowego służącego do wykonania zaprojektowanej procedury. Następnie podane zostaną istotne szczegóły implementacji tego modelu (3.3). W punkcie 3.4 pokazane są dodatkowe zadania z zakresu przetwarzania tekstu, które Hydra może realizować obok analizy składniowej. Omówione zostaną również wybrane decyzje podjęte przy opracowywaniu narzędzia oraz ich niewykorzystane ostatecznie alternatywy (punkt 3.5).

3.1. Algorytm parsowania

Koncepcja parsowania opracowana i zaimplementowana jako narzędzie Hydra opiera się na wyodrębnieniu ze struktury drzewa hybrydowego (lub składnikowego z centrami) jednostek nazywanych dalej *kręgosłupami składniowymi*.[2] Kręgosłup to maksymalna ścieżka w drzewie kończąca się w liściu oraz prowadząca wyłącznie wzdłuż krawędzi centralnych. O kręgosłupie prowadzącym do liścia v będziemy mówić, że dominuje on v, odpowiada v lub znajduje się nad v. Dla uproszczenia będziemy czasem utożsamiać liść z segmentem stanowiącym jego etykietę i analogicznie mówić o kręgosłupie odpowiadającym segmentowi.

[1] Nazwa parsera to akronim zdania „HYdra Daje Rozbiór Automatycznie".

[2] Koncepcja kręgosłupa jako jednostki w opisie lingwistycznym jest częściowo pokrewna pojęciu *projection path* występującemu w teorii X-bar (Jackendoff, 1977), ale sama w sobie nie niesie żadnych założeń co do teorii składniowej. Termin *spine* (kręgosłup) można znaleźć w literaturze dotyczącej analizy składniowej. Na przykład Ballesteros i Carreras (2015, 2017) opisują parsery stosowe, w których kręgosłupy nie stanowią z technicznego punktu widzenia niepodzielnej jednostki, ale konstruowane są w kolejnych przejściach węzeł po węźle. Koncepcję najbardziej zbliżoną do prezentowanej w tym rozdziale zaproponowali Carreras *et al.* (2008). Te ostatnie badania ograniczają się przy tym do drzew ciągłych.



Rysunek 3.1: Drzewo z wyróżnionymi kręgosłupami.

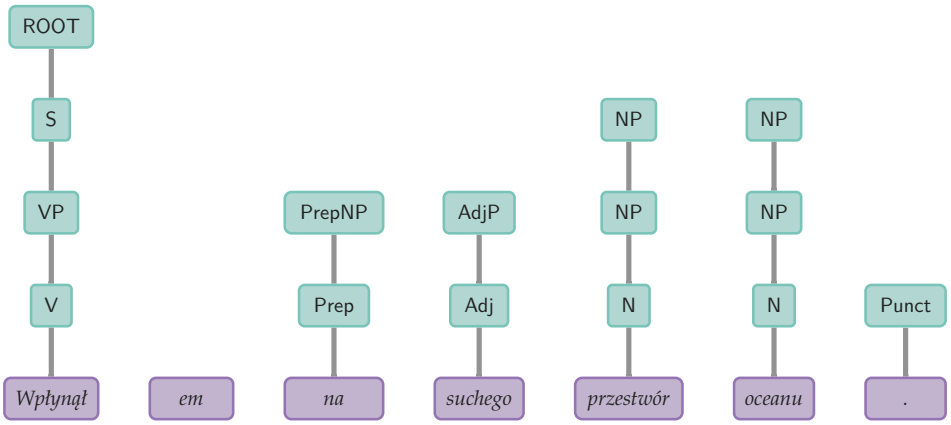
W przyjętej konwencji przedstawiania drzew (zob. 1.1.1) kręgosłupy składniowe są dobrze widoczne jako łańcuchy krawędzi kreślonych pionowo. Dla pełnej jasności rysunek 3.1[3] pokazuje drzewo hybrydowe z kręgosłupami dodatkowo wyróżnionymi graficznie za pomocą jasnoszarego pogrubienia krawędzi. I tak na przykład kręgosłup odpowiadający segmentowi *Nie* składa się tylko z jednej krawędzi i obejmuje jeden nieterminal Neg. Z kolei kręgosłup dominujący formę czasownikową stanowiącą centrum składniowe całego wypowiedzenia to ścieżka do terminala *wiecie* od korzenia drzewa ROOT, przechodząca kolejno przez terminale S, VP oraz V. Ten ostatni kręgosłup oddać można zapisem ROOT → S → VP → V.

Na szereg kręgosłupów wchodzących w skład drzewa hybrydowego można spojrzeć jak na swego rodzaju szkielet, na którym rozpięta jest cała struktura. Podążając za tą obserwacją, proponuję algorytm parsowania składający się z następujących kroków:

1. Każdemu segmentowi przypisać prowadzący do niego kręgosłup.
2. Dodać krawędzie niecentralne.
3. Opatrzyć krawędzie niecentralne etykietami zależnościowymi.

Jako ilustracja szczegółowego opisu powyższej procedury posłuży rysunek 3.2, przedstawiający jej przykładowy przebieg dla zdania *Wpłynąłem na suchego przestwór oceanu*. Pierwszy krok polega na przypisaniu poszczególnym segmentom pewnych

[3] Źródło zdania: NKJP.



(a) Segmenty z przypisanymi kręgosłupami.

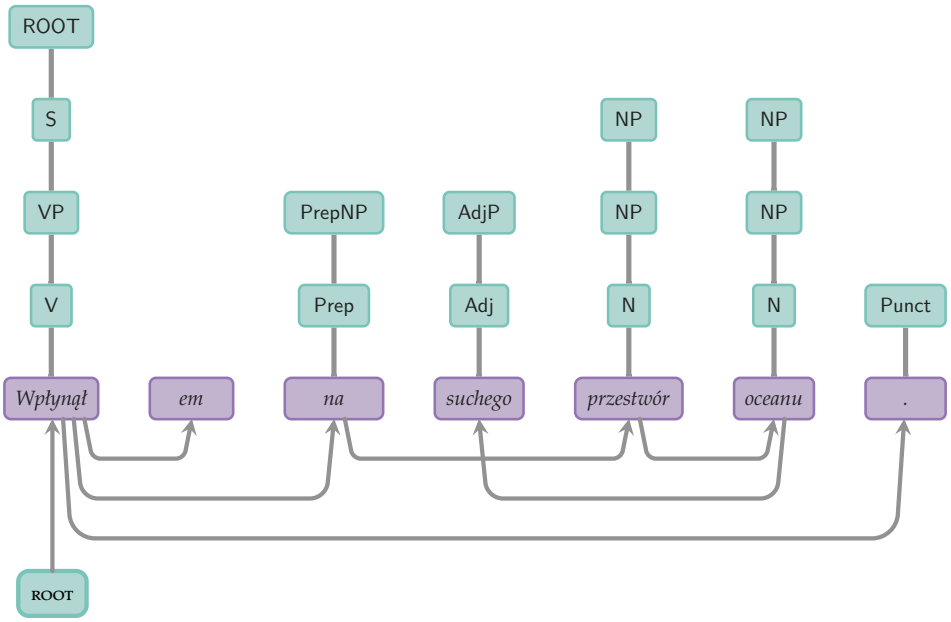
Rysunek 3.2: Procedura parsowania do drzewa hybrydowego.

fragmentów docelowej struktury składniowej, przypomina zatem (super)tagowanie. Opisywane podejście można również zaliczyć do nurtu *parsing-as-tagging* (zob. 2.1). Na rysunku 3.2a widać segmenty składające się na zdanie przytoczone jako przykład. Nad każdym segmentem został narysowany odpowiadający mu kręgosłup. Kręgosłup dla segmentu *em* jest tutaj pusty.

W punkcie 1.1.3 została opisana odpowiedniość między krawędziami zależnościowymi a niecentralnymi krawędziami składnikowymi. Dzięki niej, aby wykonać krok drugi, można skorzystać z technik parsowania zależnościowego. Krawędzie zależnościowe pomiędzy segmentami wskazują, które kręgosłupy należy „połączyć" krawędziami niecentralnymi. Krawędzie te będą zawsze prowadzić do najwyższego węzła kręgosłupa przypisanego podrzędnikowi. Na rysunku 3.2b do informacji o kręgosłupach dodane zostały powiązania zależnościowe pomiędzy segmentami.

Na przykład krawędź *na* → *przestwór* wskazuje, że z jednego z wierzchołków kręgosłupa dla segmentu *na* musi zostać poprowadzona krawędź do najwyższego nieterminala NP kręgosłupa dla segmentu *przestwór*. Krawędź ROOT → *Wpłynął* oznacza, że do kręgosłupa *Wpłynął*, jako jedynego, ma nie prowadzić krawędź niecentralna — jego szczytowy węzeł ROOT będzie stanowił korzeń powstającego drzewa. W przypadku pustego kręgosłupa (segment *em*) krawędź (tutaj: wychodząca z kręgosłupa *Wpłynął*) będzie wiodła bezpośrednio do liścia.

Do wyznaczenia krawędzi niecentralnych potrzebna jest jeszcze informacja, z którego węzła w kręgosłupie nadrzędnika należy wyprowadzić daną krawędź. Do każdego segmentu prowadzi dokładnie jedna krawędź zależnościowa (i analogicznie: do każdego kręgosłupa poza odpowiadającym korzeniowi drzewa prowadzi dokładnie jedna krawędź niecentralna). Miejsca, z których wychodzą krawędzie nie-



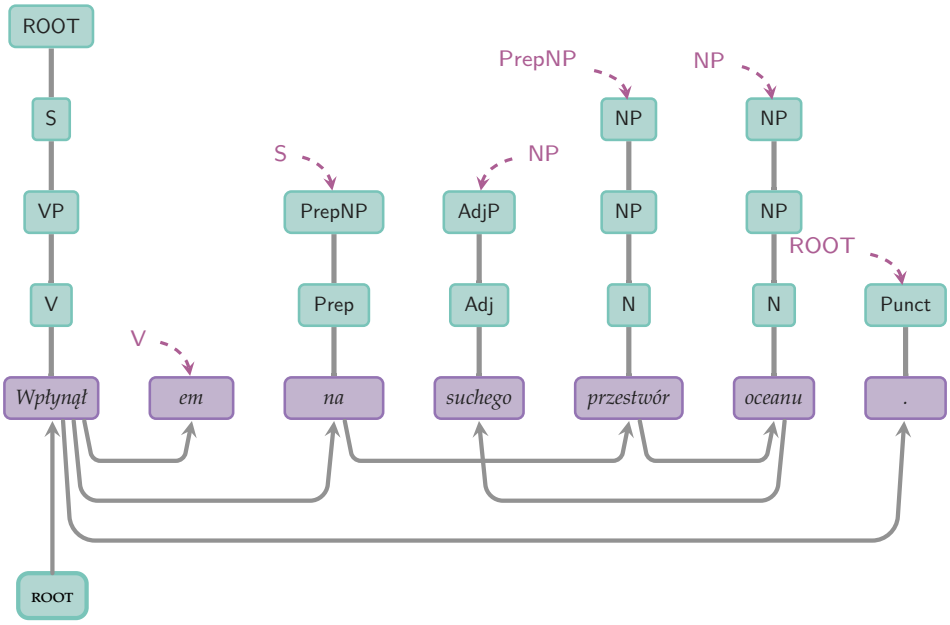
(b) Po dodaniu powiązań zależnościowych.

Rysunek 3.2: Procedura parsowania do drzewa hybrydowego.

centralne, można więc przypisać sekwencyjnie kolejnym segmentom, podobnie jak kręgosłupy. Na rysunku 3.2c przy każdym segmencie poza centrum całego wypowiedzenia *Wpłynął* zanotowano kategorię składniową nieterminala, do którego należy poprowadzić krawędź niecentralną. Na przykład od segmentu *na* (a dokładnie: od szczytowego węzła jego kręgosłupa PrepNP) należy poprowadzić krawędź do węzła S położonego na kręgosłupie przypisanym jego nadrzędnikowi zależnościowemu *Wpłynął*.

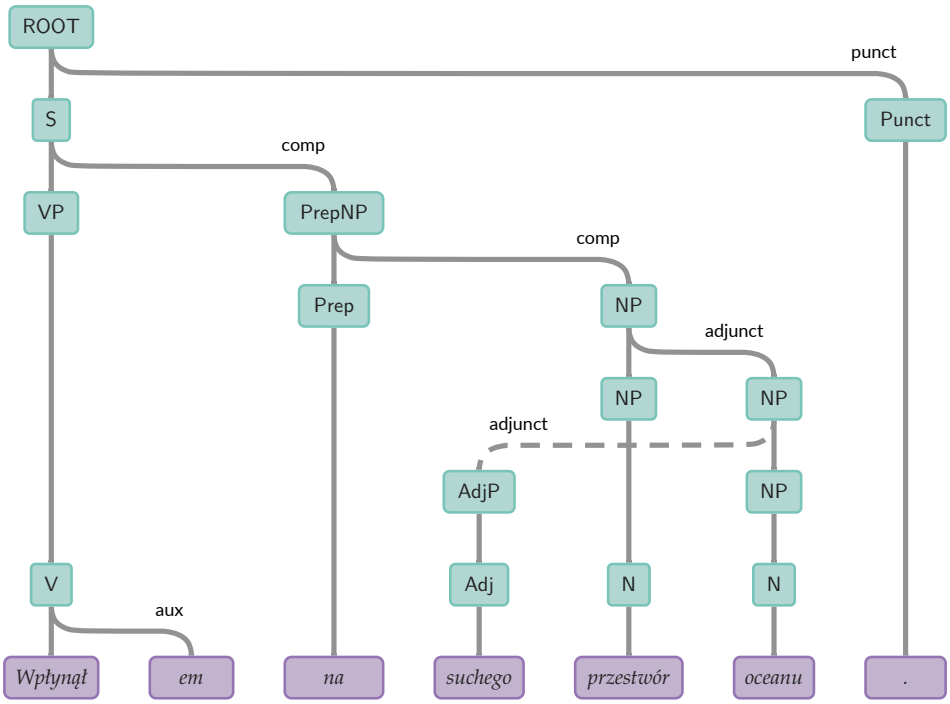
Wreszcie etykiety zależnościowe również można przyporządkować segmentom, do których prowadzą odpowiednie krawędzie. Drzewo hybrydowe powstałe z informacji zgromadzonych w poprzednich krokach, wraz z naniesionymi etykietami krawędzi niecentralnych, umieszczone zostało na rysunku 3.2d.

Należy w tym miejscu zaznaczyć, że – o ile tylko wykorzystana metoda parsowania zależnościowego umożliwia uzyskanie drzewa nieprojekcyjnego – cała opisana powyżej procedura pozwala stworzyć nieciągłe drzewo hybrydowe. Widać to w szczególności w omawianym przykładzie: nieciągły składnik *suchego oceanu* powstał dzięki wyznaczeniu na wcześniejszym etapie nieprojekcyjnego drzewa zależnościowego, w którym krzyżują się krawędzie *oceanu* → suchego oraz *na* → przestwór.



(c) Po ustaleniu węzłów wyjściowych krawędzi niecentralnych.

Rysunek 3.2: Procedura parsowania do drzewa hybrydowego.



(d) Kompletne drzewo.

Rysunek 3.2: Procedura parsowania do drzewa hybrydowego.

3.2. Architektura parsera

3.2.1. Dane wejściowe i wyjściowe modelu

Model parsera, który zostanie opisany w punkcie 3.2.2, operuje na sekwencji segmentów, przypisując każdemu z nich szereg informacji służących następnie do odtworzenia drzewa według ogólnej procedury z punktu 3.1. Materiał służący do wytrenowania modelu musi zatem zostać odpowiednio zakodowany w postaci danych wejściowych i wyjściowych modelu. Dane wejściowe to przykłady informacji, jaka będzie przekazywana do przetworzenia gotowemu modelowi. Dane wyjściowe to przykłady odpowiedzi, jakich model powinien udzielić dla danego wejścia. Nie jest to jeszcze gotowe drzewo (stanowiące efekt końcowy działania parsera), ale pośrednia postać danych — komplet informacji wystarczających do skonstruowania takiego drzewa.

Postać danych wejściowych jest prosta — każde zdanie reprezentowane jest jako lista składających się na nie segmentów. Dane wyjściowe zawierają informację pozwalającą w pełni odtworzyć drzewo hybrydowe zgodnie z procedurą opisaną wcześniej. Tabela 3.1 przedstawia dane wejściowe oraz wyjściowe odpowiadające drzewu z rysunku 3.2d. Pierwsza kolumna zawiera numery kolejne segmentów i dodana jest dla czytelności. Druga kolumna — segmenty — stanowi dane wejściowe. Kolejne kolumny tabeli to dane wyjściowe w postaci pięciu sekwencji o długości równej liczbie segmentów, zawierających odpowiadające poszczególnym segmentom:

— Nadrzędnik zależnościowy w postaci jego numeru kolejnego lub zera, jeśli nadrzędnikiem jest ROOT.
— Relacja zależnościowa łącząca segment z jego nadrzędnikiem (root, jeśli nadrzędnikiem jest ROOT).

#	segment	nadrzędnik	relacja	kręgosłup	rodzic	
1	*Wpłynął*	0	root	ROOT → S → VP → V	—	—
2	*em*	1	aux	—	V	1
3	*na*	1	comp	PrepNP → Prep	S	1
4	*suchego*	6	adjunct	AdjP → Adj	NP	2
5	*przestwór*	3	comp	NP → NP → N	PrepNP	1
6	*oceanu*	5	adjunct	NP → NP → N	NP	2
7	.	1	punct	Punct	ROOT	1

Tabela 3.1: Drzewo z rysunku 3.2d zakodowane jako dane wejściowe i wyjściowe.

— Kręgosłup nad segmentem w postaci ciągu nieterminali. Kręgosłupy traktowane są w tym ujęciu jako wartości atomowe (zob. również 3.5.1). Możliwy jest pusty kręgosłup, jak dla segmentu *em* w przykładzie.
— Etykieta węzła będącego rodzicem najwyższego nieterminala kręgosłupa (miejsce, w którym kręgosłup podrzędnika należy podłączyć do kręgosłupa nadrzędnika krawędzią niecentralną).
— Wysokość rodzica w kręgosłupie nadrzędnika, w obrębie węzłów o tym samym typie składniowym, liczona od dołu. Pojedynczy kręgosłup może zawierać więcej niż jeden węzeł o tej samej etykiecie, konieczna jest zatem możliwość ich rozróżnienia. Na przykład węzeł AdjP kończący kręgosłup nad segmentem *suchego* jest podłączony do drugiego (licząc od liścia w górę) węzła NP kręgosłupa nad segmentem *oceanu*, stąd wartość 2 w ostatniej kolumnie w wierszu dla *suchego*.

3.2.2. Sieć neuronowa

Podstawę działania parsera stanowi sieć neuronowa składająca się z dwóch głównych modułów: pretrenowanego modelu językowego dostarczającego wektorowych reprezentacji segmentów wejściowych oraz komponentu przewidującego poszczególne elementy danych wyjściowych.

Reprezentacje wektorowe modelu BERT

W pierwszej kolejności dane wejściowe przetwarzane są przez model językowy typu BERT (Devlin *et al.*, 2019). Modele z rodziny BERT składają się z wielu (najczęściej kilkunastu) warstw o architekturze Transformer (Vaswani *et al.*, 2017), wykorzystujących mechanizm uwagi (*attention*), aby obliczać reprezentacje wektorowe poszczególnych segmentów z uwzględnieniem pozostałych segmentów w sekwencji wejściowej jako towarzyszącego im kontekstu. Architektura kodująca Transformer, wykorzystywana w modelach BERT, została opisana w dodatku C. Podczas trenowania modelu BERT jego zadaniem jest uzupełnianie losowo usuniętych z sekwencji wejściowej segmentów (*masked language modeling*). Takie trenowanie, nazywane *pretrenowaniem*, nie wymaga nanoszenia na dane żadnej anotacji lingwistycznej (jedyną potrzebną informacją jest sam tekst). Można zatem w tym celu wykorzystać bardzo duże ilości danych tekstowych. Pretrenowanie modeli językowych jest natomiast zadaniem kosztownym czasowo i obliczeniowo.

Pretrenowany model BERT można pozbawić ostatnich warstw, odpowiedzialnych za przewidywanie brakujących słów w zadaniu, którego został nauczony, a następnie wykorzystać reprezentacje wektorowe[4] obliczane przez jedną z kolejnych warstw

[4] Mogą to być reprezentacje poszczególnych segmentów lub reprezentacja całego tekstu wejściowego obliczana dla specjalnego dodatkowego segmentu CLS wprowadzanego w architekturze BERT.

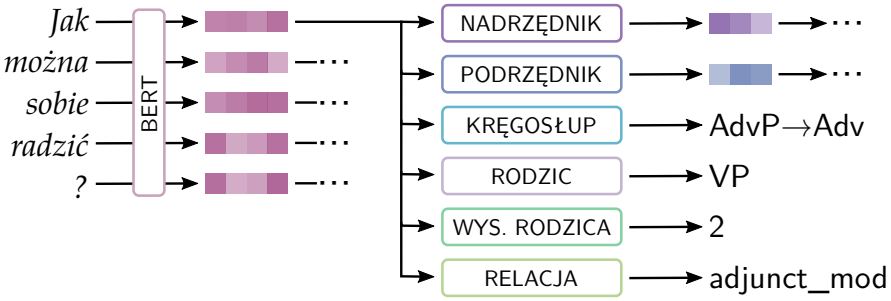
Transformer, czyniąc je wejściami dla innego klasyfikatora. Jeśli klasyfikatorem tym jest sieć neuronowa, a w ramach jej trenowania modyfikowane są również parametry modelu językowego, mamy do czynienia z *dostrajaniem* (*fine-tuning*) modelu językowego do danego zadania. Jest to obecnie częsty scenariusz w różnych zastosowaniach z zakresu przetwarzania języka naturalnego, wykorzystany również w parserze Hydra. Dostrajanie modelu językowego, w odróżnieniu od pretrenowania, przeważnie odbywa się z użyciem danych opatrzonych jakiegoś rodzaju anotacją. Są to dane trudniej dostępne, a zatem najczęściej dużo mniej liczne.

Należy tutaj zaznaczyć, że używając określenia *modele językowe*, mamy na myśli sieci neuronowe o architekturach typu *encoder*, których rozmiary liczone są w dziesiątkach lub setkach milionów parametrów. Odróżniamy je tym samym od dużych, generatywnych modeli językowych (LLM, *large language models*) z rodzin takich jak GPT (OpenAI, 2024), Llama (Grattafiori *et al.*, 2024) czy DeepSeek (DeepSeek-AI, 2025), liczących nawet setki miliardów parametrów. Typowe scenariusze wykorzystania takich modeli w zadaniach NLP różnią się od mojego podejścia i polegają na odczytaniu napisu w języku naturalnym wygenerowanego przez model w odpowiedzi na tekstowe zapytanie (instrukcję, *prompt*) zadane na wejściu. Model najczęściej nie jest przy tym w żaden sposób dostrajany, jedyną stosowaną formą nadzoru jest zawarcie w prompcie jednego lub kilku przykładów wskazujących schemat, według którego ma być sformułowana odpowiedź (tzw. *one-* lub *few-shot*, w odróżnieniu od scenariusza *zero-shot* nie uwzględniającego nawet przykładów).

Skala i specyfika dużych modeli językowych czynią je mało adekwatnymi w przypadku podejść takich jak moje, zakładających dostrojenie modelu na niewielkim (w porównaniu z danymi wykorzystanymi do pretreningu) zbiorze specjalistycznie anotowanych danych, jakim jest treebank, oraz operowanie na wektorowych reprezentacjach poszczególnych segmentów. Zagadnienie wykorzystania LLM-ów do uzyskiwania rozbiorów składniowych poprzez odpowiednie konstruowanie promptów badali na przykład Tian *et al.* (2024), jednak ich wyniki wskazują, że lepiej sprawdzają się uczone z nadzorem parsery wykorzystujące (małe) modele językowe. Jednocześnie, mimo wyraźnego skupienia uwagi badaczy na modelach typu LLM, prowadzone są również prace nad udoskonalaniem małych modeli – przykładem może być opracowany niedawno ModernBERT (Warner *et al.*, 2025), stanowiący istotne unowocześnienie i optymalizację architektury BERT.

Klasyfikatory

Wszystkie elementy danych wyjściowych wymienione w punkcie 3.2.1 poza powiązaniami zależnościowymi przewidywane są przez klasyfikatory emitujące roz-



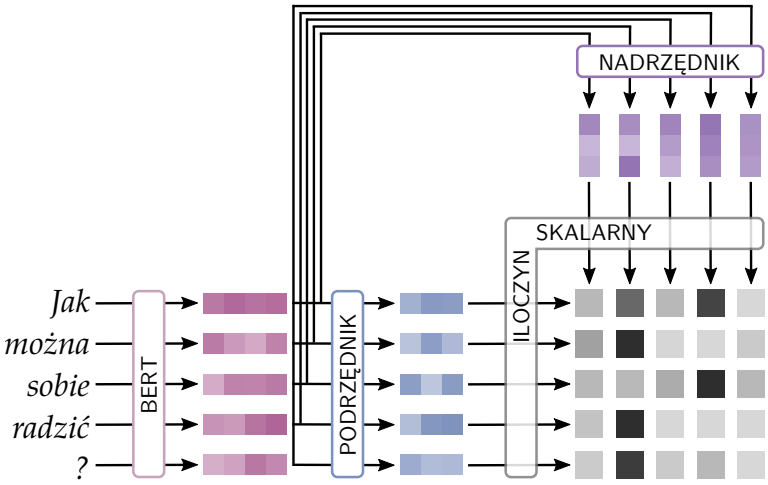
Rysunek 3.3: Schemat architektury parsera.

kład prawdopodobieństwa[5] możliwych wartości przyjmowanych przez ten element. Dla każdej z tych czterech składowych przeznaczony jest jeden klasyfikator w postaci pojedynczej warstwy gęstej — RELACJA zależnościowa, KRĘGOSŁUP, RODZIC oraz WYS. RODZICA — uruchamiany równolegle na reprezentacji wektorowej każdego segmentu. Na wektorach pochodzących z modelu BERT operują, równolegle z czterema klasyfikatorami wymienionymi powyżej, dwie warstwy gęste NADRZĘDNIKI oraz PODRZĘDNIKI, produkujące pośrednie reprezentacje wektorowe. Stanowią one wejście dla posiadającego nieco bardziej skomplikowaną budowę klasyfikatora wyznaczającego nadrzędniki zależnościowe. Opisywaną strukturę modelu schematycznie przedstawia rysunek 3.3.

Do wyznaczania krawędzi zależnościowych (czyli nadrzędnika dla każdego segmentu) zastosowałam bardziej złożoną konstrukcję, zainspirowaną parserem COMBO (Klimaszewski i Wróblewska, 2021),[6] osiągającego w momencie pracy nad Hydrą najlepszą jakość parsowania zależnościowego dla języka polskiego. Pośrednie reprezentacje wektorowe odpowiadające nadrzędnikom oraz podrzędnikom są parami mnożone skalarnie, co daje w wyniku macierz M wag krawędzi zależnościowych. Schemat klasyfikatora przedstawia rysunek 3.4. Iloczyn skalarny i-tego wektora podrzędnika oraz j-tego wektora nadrzędnika stanowi element M_{ij} tej macierzy i odpowiada logitowi prawdopodobieństwa, że j-ty segment jest nadrzędnikiem zależnościowym segmentu i-tego. Innymi słowy, wiersze macierzy wag to przewidywane przez model rozkłady prawdopodobieństwa wyboru nadrzędnika

[5] Z technicznego punktu widzenia na tym etapie nie mamy jeszcze do czynienia z prawdopodobieństwami, ale z tzw. logitami przyjmującymi wartości rzeczywiste. Wektor logitów $(x_1, ..., x_K)$ przekształcany jest za pomocą funkcji $softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$ do postaci rozkładu prawdopodobieństwa, czyli liczb z przedziału $(0, 1)$ (takie wartości przyjmuje funkcja *softmax*) sumujących się do jedności. *Softmax* stanowi wielowymiarowe uogólnienie funkcji logistycznej $\frac{1}{1+e^{-x}}$, $x \in \mathbf{R}$, której funkcją odwrotną jest funkcja logitowa $ln(\frac{x}{1-x})$, $x \in (0, 1)$. Stąd pochodzi konwencja określania w kontekście uczenia maszynowego wartości, które mają zostać przekazane do funkcji *softmax*, logitami.

[6] Podobne podejście stosowali również m.in. Dozat i Manning (2017).



Rysunek 3.4: Schemat klasyfikatora dla krawędzi zależnościowych.

dla poszczególnych segmentów. Wartość elementu M_{ii} reprezentuje wybór węzła ROOT jako nadrzędnika i-tego segmentu.[7]

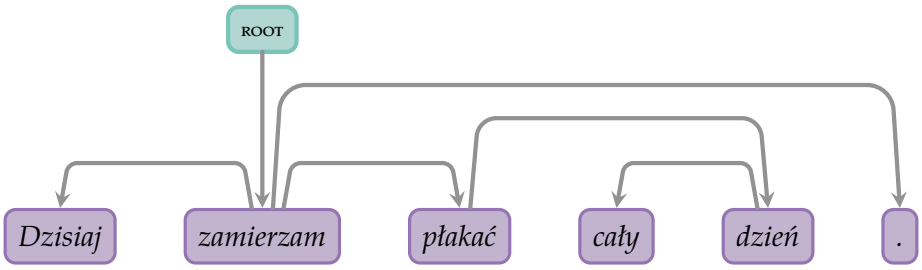
Na rysunku 3.5 zostały przedstawione przykładowe macierze odpowiadające zdaniu *Dzisiaj zamierzam płakać cały dzień*.[8] Wartości 1 w macierzy na rysunku 3.5b odpowiadają istniejącym w drzewie krawędziom. Po lewej stronie oraz nad macierzą wypisano segmenty odpowiadające wierszom i kolumnom. Jedyna niezerowa wartość w każdym wierszu wskazuje na nadrzędnik segmentu (wypisany po prawej stronie wiersza). Na przykład jedynka w czwartym wierszu i piątej kolumnie odpowiada relacji zależnościowej *dzień* → *cały*. Jedyna niezerowa wartość na diagonali macierzy — w wierszu i kolumnie dla *zamierzam* — wskazuje ten segment jako nadrzędnik całego zdania. Wynik zastosowania funkcji *softmax* do poszczególnych wierszy macierzy logitów (rys. 3.5c), stanowiącej wynik działania klasyfikatora dla krawędzi zależnościowych, bardzo niewiele różni się od macierzy wzorcowej.

Ostateczny kształt drzewa zależnościowego jest wyznaczany z macierzy wag za pomocą algorytmu Tarjana (1977) znajdującego maksymalne drzewo rozpinające (MST) w grafie skierowanym – moduł zależnościowy Hydry jest zatem parserem grafowym typu *arc-factored*. Parser ten nie jest w żaden sposób ograniczony do drzew projekcyjnych, a co za tym idzie – pozwala na uzyskanie w ostatecznym wyniku drzew hybrydowych z nieciągłościami.

Wspomniany algorytm MST został połączony z procedurą *Root Reweighting Algorithm* (Stanojević i Cohen, 2021). Meta-algorytm *Root Reweighting Algorithm* polega na modyfikacji macierzy wag krawędzi w sposób gwarantujący, że każda użyta następ-

[7] Inaczej niż w parserze COMBO, gdzie węzłowi ROOT odpowiadają dodatkowy wiersz i kolumna macierzy.

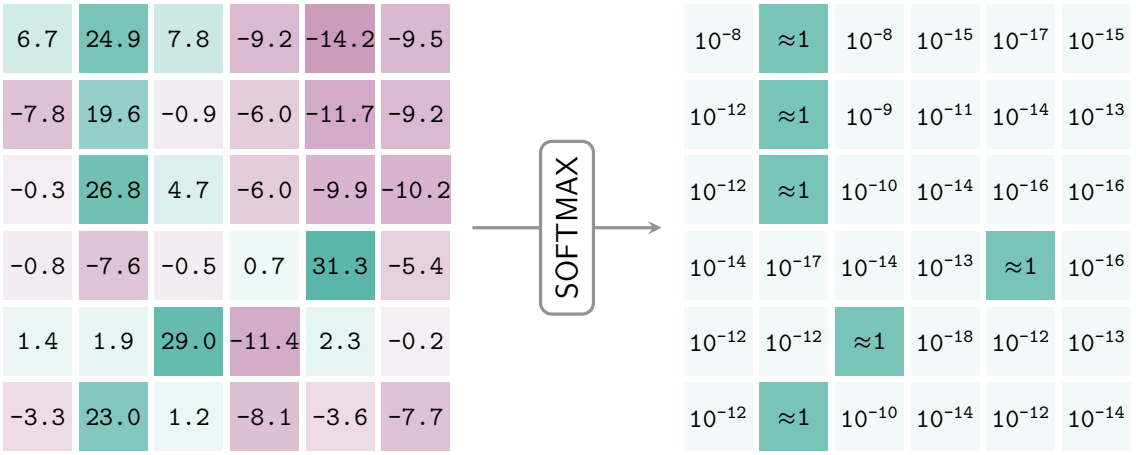
[8] Źródło: KWJP.



(a) Drzewo zależnościowe.

	Dzisiaj	*zamierzam*	*płakać*	*cały*	*dzień*	.	
Dzisiaj	0	1	0	0	0	0	*zamierzam*
zamierzam	0	1	0	0	0	0	ROOT
płakać	0	1	0	0	0	0	*zamierzam*
cały	0	0	0	0	1	0	*dzień*
dzień	0	0	1	0	0	0	*płakać*
.	0	1	0	0	0	0	*zamierzam*

(b) Macierz wzorcowa w danych uczących.



(c) Macierz logitów oraz wynik zastosowania funkcji *softmax*.

Rysunek 3.5: Porównanie wzorcowej oraz obliczonej przez model macierzy wag.

nie procedura znajdowania maksymalnego drzewa rozpinającego przypisze węzeł ROOT jako nadrzędnik dokładnie jednemu segmentowi (z węzła ROOT wychodzić będzie dokładnie jedna krawędź). Wymaganie to nie jest w szczególności spełnione

przez algorytmy MST typu Chu-Liu-Edmonds w ich ogólnej postaci i wymaga dodatkowego uwzględnienia w implementacjach parserów zależnościowych (zob. Zmigrod *et al.* 2020).

3.2.3. Konstruowanie drzewa

Działania opisane w poprzednich punktach dostarczają wszystkich informacji potrzebnych do skonstruowania drzewa hybrydowego zgodnie z procedurą zaproponowaną w punkcie 3.1. Aby wyeliminować niespójności, które mogły się pojawić w wyniku niezależnego przewidywania poszczególnych komponentów drzewa, podczas jego budowania stosowane są następujące korekty:

— Jeśli węzeł ROOT występuje w kręgosłupie segmentu innego niż korzeń drzewa zależnościowego, zostaje usunięty.
— Jeśli kręgosłup przypisany korzeniowi drzewa zależnościowego zaczyna się od węzła innego niż ROOT, węzeł taki jest do niego dodawany. Ten oraz poprzedni krok zapewniają, że drzewo zawiera dokładnie jeden nieterminal o etykiecie ROOT i jest on korzeniem całej struktury hybrydowej.[9]
— Jeśli w kręgosłupie nadrzędnika występuje mniej powtórzeń węzła pewnego typu, niż wymaga tego przewidziane przez model podłączenie kręgosłupa podrzędnika, są one dodawane. Na przykład jeśli wymagane jest podłączenie VP-2 (jako dziecko drugiego od dołu wierzchołka VP) do kręgosłupa S → VP → V, kręgosłup ten wydłużany jest do postaci S → VP → VP → V.
— Jeśli w kręgosłupie nadrzędnika nie występuje węzeł typu wymaganego do podłączenia podrzędnika, zostaje on podłączony jako dziecko najwyższego węzła. Na przykład podłączenie NP-1 do kręgosłupa VP → V rozstrzygane jest przez podłączenie do węzła VP.
— Jeśli model wskazuje na podłączenie do pustego kręgosłupa, podłączenie jest zamiast tego wykonywane do najbliższego zależnościowego przodka o niepustym kręgosłupie. Jest to jedyna operacja korekcyjna zmieniająca strukturę zależnościową przewidzianą przez model.
— Z drzewa usuwane są wszystkie ewentualne krawędzie unarne postaci X → X (nieterminal posiadający jedno dziecko tego samego typu, co on sam, zostaje zastąpiony tym dzieckiem).

[9] Implementacja Hydry pozwala na zdefiniowanie dowolnej etykiety, która ma być traktowana w ten specjalny sposób, lub na pominięcie tego kroku, jeśli schemat anotacji danych nie wyróżnia takiego typu nieterminala.

3.3. Implementacja parsera

3.3.1. Wybór narzędzi programistycznych

Włączanie modeli językowych w systemy rozwiązujące różne zadania z dziedziny przetwarzania języka naturalnego stało się w ostatnich latach powszechne. Powstały również narzędzia i pakiety programistyczne ułatwiające korzystanie z tych modeli, takie jak użyte do implementacji parsera Hydra biblioteki dla języka Python 3: Huggingface Transformers (nazwa ta zapisywana jest przez twórców jako 🤗Transformers)[10] oraz 🤗Datasets.[11] Pierwsza z wymienionych bibliotek udostępnia API programistyczne do wielu modeli językowych z rodziny BERT i nie tylko. Oferuje również szereg schematów nadbudowanych nad tymi modelami klasyfikatorów, sprawdzających się w wielu typowych zastosowaniach NLP. Na przykład klasyfikator emitujący jedną etykietę dla całego tekstu wejściowego można wytrenować i wykorzystać do analizy wydźwięku, wykrywania spamu czy treści szkodliwych, itp. 🤗Transformers umożliwia wybór biblioteki realizującej operacje matematyczne na tensorach, odpowiedzialnej za obliczenia konieczne do uczenia i uruchamiania sieci neuronowych. W przypadku implementacji Hydry została wybrana biblioteka TensorFlow.[12] Z kolei biblioteka 🤗Datasets dostarcza sposobów reprezentacji oraz funkcji użytecznych przy operowaniu na danych wejściowych dla modeli.

3.3.2. Implementacja sieci neuronowej

Spośród typów klasyfikatora oferowanych przez 🤗Transformers, najbliższy architekturze zaprojektowanej dla parsera Hydra jest scenariusz klasyfikacji poszczególnych segmentów. Pozwala on na przypisanie każdemu segmentowi wejściowemu jednej etykiety z ustalonego zbioru (znaczników morfoskładniowych, znaczników typu IOB w wykrywaniu nazw własnych itp.). Nie ma natomiast możliwości równoczesnego wyboru z kilku zbiorów etykiet, jak to zostało opisane w punkcie 3.2. Architektura przedstawiona na rysunkach 3.3 oraz 3.4 została zatem zaimplementowana w oparciu o dostępną w 🤗Transformers klasę `TFBertForTokenClassification`.

Wyjściem z modelu BERT jest macierz reprezentacji wektorowych dla poszczególnych elementów tekstu wejściowego (wiersze macierzy odpowiadają kolejnym elementom). Macierz ta jest najpierw przekazywana do warstwy typu *dropout*.[13]

[10] https://huggingface.co/docs/transformers

[11] https://huggingface.co/docs/datasets

[12] https://www.tensorflow.org

[13] Warstwa *dropout* jest mechanizmem regularyzacji modelu, czyli zapobiegania nadmiernemu dopasowaniu do danych uczących (*overfitting*) kosztem możliwości uogólniania (*generalisation*) wytrenowanego modelu. Podczas uczenia warstwa ta zmienia losowo wybrane wartości w macierzy

Każdy klasyfikator przypisujący segmentom etykiety ma postać pojedynczej warstwy gęstej. Warstwa gęsta to jeden z podstawowych komponentów stosowanych w sieciach neuronowych. Matematycznie odpowiada ona przekształceniu afinicznemu:

$$Dense(X) = X \cdot \mathbf{W} + \mathbf{b} \tag{3.1}$$

Wartości macierzy i wektora **W** i **b**, nazywane parametrami lub wagami, optymalizowane są podczas trenowania sieci. Wymiar $n \times m$ macierzy X oraz oczekiwany wymiar wyniku $n \times k$ determinują wymiary **W** i **b** jako odpowiednio $m \times k$ oraz $n \times k$.[14] Tensor **b** nazywany jest wyrazem wolnym (*bias*), a przypadku jego braku warstwa gęsta odpowiada przekształceniu liniowemu. W opisywanym modelu m będzie długością pojedynczej reprezentacji wektorowej pochodzącej z BERT-a, n — liczbą segmentów (upraszczając, zob. 3.3.3) w zadanym tekście, a k — liczbą możliwych etykiet. Ta ostatnia wartość zależna jest od konkretnego klasyfikatora.

Powstaje w ten sposób macierz n wektorów długości k — każdemu segmentowi wejściowemu odpowiada wektor logitów prawdopodobieństw. Poszczególne współrzędne wektora powiązane są jednoznacznie z możliwymi etykietami, więc ostateczny wybór etykiety sprowadza się do znalezienia największego elementu wektora.

Jako funkcja straty podczas trenowania modelu używana jest entropia krzyżowa, będąca miarą rozbieżności rozkładów prawdopodobieństwa. Jeśli X jest dyskretną zmienną losową przyjmującą wartości ze zbioru $x_1, ..., x_n$, a p oraz q — określonymi dla nich rozkładami prawdopodobieństwa, to entropia krzyżowa między p a q zdefiniowana jest jako

$$H(p, q, X) = -\sum_{i=1}^{n} p(x_i) \log q(x_i)$$

— przy czym p rozumiany jest jako rozkład prawdziwy, a q — jako rozkład, którego nietrafność względem p wyraża entropia krzyżowa. W przypadku opisywanego parsera rozkładami ocenianymi miarą entropii krzyżowej są wektory zwra-

na zera, „skłaniając" model do większej elastyczności i odporności na zaburzenia w danych wejściowych. Podczas klasyfikacji natomiast warstwa *dropout* niczego nie zmienia — model powinien móc wykorzystać całą przekazaną mu informację.

[14] W ogólności **W** i **b** są tensorami dowolnego rzędu. Źródłami dodatkowych wymiarów mogą być charakter danych (na przykład piksele w klasyfikacji obrazów ułożone w dwóch wymiarach) lub *batching*. W faktycznych implementacjach sieci neuronowych zarówno podczas uczenia, jak podczas przetwarzania nowych danych przykłady wejściowe są grupowane w partie (*batches*). Obliczenia modelu na przykładach w obrębie jednej partii są wykonywane równolegle, aby przyspieszyć cały proces oraz wykorzystać możliwości zrównoleglania obliczeń oferowane przez graficzne jednostki obliczeniowe.

cane przez poszczególne klasyfikatory dla kolejnych segmentów (znormalizowane funkcją *softmax*), natomiast jako rozkłady prawdziwe (wzorcowe) przyjmowane są rozkłady punktowe przypisujące prawdopodobieństwo 1 wartościom występującym w danych uczących. Łączna funkcja straty, względem której odbywa się propagacja wsteczna,[15] to suma entropii krzyżowych dla wszystkich klasyfikatorów i dla wszystkich segmentów.

3.3.3. Spójność tokenizacji z segmentacją

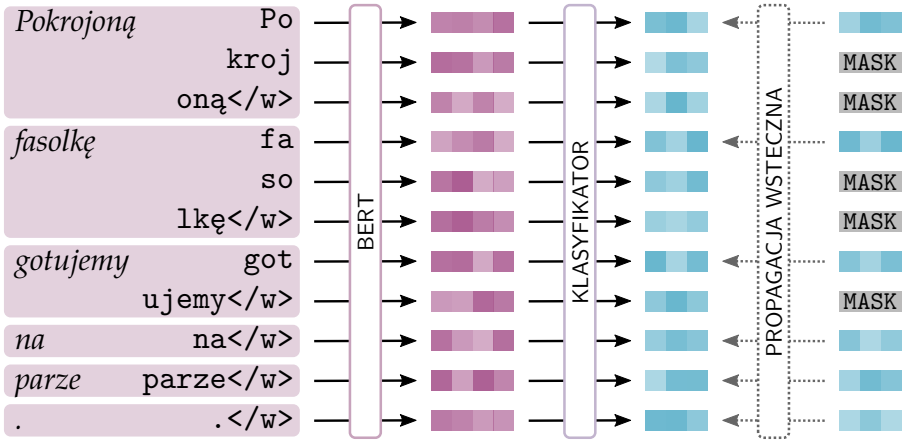
Właściwością modeli BERT wymagającą uwzględnienia w implementacji klasyfikatora operującego na poszczególnych segmentach jest mechanizm tokenizacji (algorytm WordPiece, Wu *et al.* 2016). Aby utrzymać rozsądny z obliczeniowego punktu widzenia rozmiar słownika oraz wyeliminować problem nieznanych słów (*out of vocabulary*, OOV), modele te dzielą tekst wejściowy na jednostki (tokeny) często mniejsze niż słowa. Powoduje to, że podział wypowiedzenia na tokeny przetwarzane przez model BERT może być drobniejszy niż podział na segmenty według anotacji danych. W efekcie również długość sekwencyj etykiet emitowanych przez model może być większa niż liczba segmentów, dla których ma zostać skonstruowane drzewo (oraz sekwencja wzorcowych etykiet, względem których ma być obliczona wartość funkcji straty).

Opisana rozbieżność wymaga zignorowania niektórych wektorów produkowanych przed model. Prostym i powszechnie stosowanym rozwiązaniem jest uwzględnienie tylko pierwszego tokenu pochodzącego z każdego segmentu wejściowego, jak zilustrowano na rysunku 3.6. Podczas przetwarzania przykładowego zdania,[16] składającego się z sześciu segmentów, tokenizator BERT-a podzielił słowa *Pokrojoną*, *fasolkę* oraz *gotujemy* na więcej części, produkując w rezultacie łącznie jedenaście tokenów.[17] Dane wyjściowe kodujące prawidłowe drzewo dla tego zdania zawierają

[15] Propagacja wsteczna (*backpropagation*) jest fundamentalną procedurą w trenowaniu współczesnych sieci neuronowych. Stanowi ona praktyczną implementację reguły łańcuchowej dla obliczania pochodnej funkcji złożonej i pozwala na obliczenie gradientu funkcji straty względem parametrów sieci neuronowej. Gradient ten pozwala wyznaczyć korektę parametrów modelu stosowaną w danym kroku uczenia tak, aby stopniowo przesuwać wartość funkcji straty w kierunku jej minimum. Metody będące prekursorami propagacji wstecznej sięgają lat 50. ubiegłego stulecia, a rozwój współczesnych algorytmów przypadał na lata 70. i 80. Druga dekada XXI wieku przyniosła natomiast praktyczne możliwości ich zastosowania na szeroką skalę dzięki powszechnej dostępności graficznych jednostek obliczeniowych (GPU).

[16] Źródło: KWJP.

[17] Wykorzystywana implementacja tokenizatora dla modelu BERT umożliwia podział ciągłego tekstu lub określenia wstępnej tokenizacji, która może zostać poddana dalszym podziałom. W parserze Hydra wykorzystywana jest ta druga możliwość, dlatego tokenizacja BERT-a zachowuje wszystkie występujące w danych wejściowych granice segmentów i potencjalnie wprowadza kolejne.



Rysunek 3.6: Maskowanie tokenizacji BERT-a.

po sześć etykiet dla każdego klasyfikatora. Aby zrównoleglić dane z faktycznym wyjściem modelu, w którym każdy klasyfikator obliczy jedenaście wyników — po jednym dla każdego tokenu BERT-a — należy dodać pięć wektorów. Przypisanie tym wektorom specjalnej wartości maskującej, rozpoznawanej przez używane biblioteki, spowoduje ich pominięcie podczas obliczania funkcji straty.[18]

3.4. Pozostałe moduły narzędzia Hydra

Narzędzie Hydra może nie tylko pełnić rolę parsera, ale również dostarczać innych rodzajów automatycznego znakowania tekstu. W tym punkcie zostaną przedstawione moduły zaimplementowane oprócz opisanego wyżej mechanizmu analizy składniowej. Podczas trenowania modelu Hydry można wybrać dowolny podzbiór dostępnych modułów. Wszystkie wybrane moduły korzystają ze wspólnego modelu BERT, trenowane są łącznie i działają równolegle jako jeden wielowyjściowy model — odpowiednie klasyfikatory dodawane są analogicznie, jak zostało to przedstawione w przypadku parsowania w punkcie 3.2.2. Moduły znakowania automatycznego Hydry to:

— hybrydowe znakowanie składniowe,
— zależnościowe znakowanie składniowe,
— segmentacja (zob. 3.4.1),
— tagowanie (zob. 3.4.2),
— lematyzacja (zob. 3.4.2),
— wykrywanie jednostek nazewniczych (zob. 3.4.3).

[18] A zatem wyzerowanie odpowiednich pochodnych częściowych (gradientów) i zablokowanie propagacji wstecznej dla zamaskowanych wyjść z modelu.

Moduł parsowania hybrydowego został szczegółowo opisany w punktach 3.1 oraz 3.2. Powiązania i relacje zależnościowe są wyznaczane w ramach działania tego modułu, można jednak użyć Hydry jako parsera *stricte* zależnościowego — moduł parsowania ograniczony zostaje w takim wypadku do klasyfikatorów NADRZĘDNIKI, PODRZĘDNIKI oraz RELACJA. Pozostałe cztery moduły zostaną przedstawione kolejno w niniejszym punkcie.

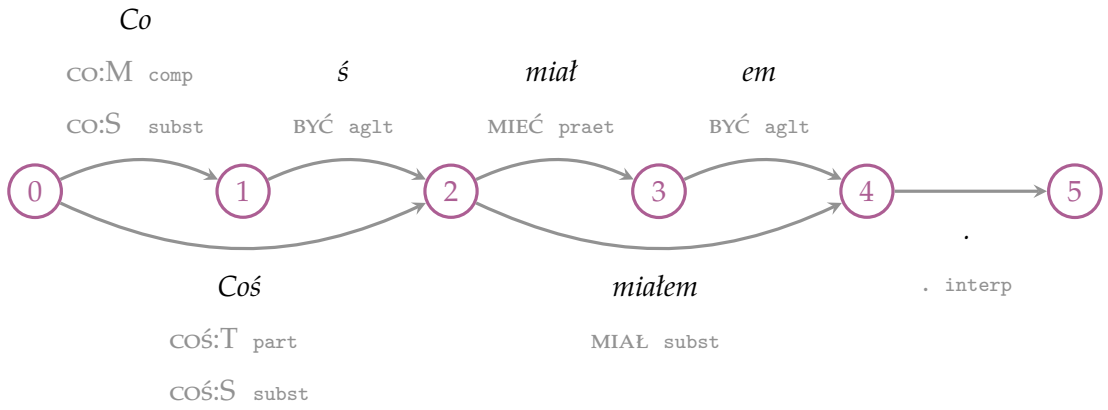
3.4.1. Segmentacja

W powyższych opisach działania Hydry uwaga skupiona była na procesie uzyskiwania analizy składniowej. Dlatego zostało przyjęte założenie, że przetwarzany tekst jest już podzielony na segmenty. Jest to prawdą w przypadku wykorzystywanych danych uczących. Jeśli jednak parser ma posłużyć do analizy składniowej dowolnego tekstu, powinien być w stanie sam wykonać segmentację, czyli prawidłowo wyznaczyć jednostki, które znajdą się w liściach drzewa (zob. 1.1.1).

Istotną cechą podejścia do tego zagadnienia opracowanego w ramach narzędzia Hydra jest wykorzystanie informacji o możliwych wariantach segmentacji dostarczanej przez analizator morfologiczny Morfeusz 2, którego słownik bazuje na danych Słownika gramatycznego języka polskiego (SGJP, Saloni *et al.* 2015).

Wynikiem działania Morfeusza jest acykliczny graf skierowany reprezentujący dopuszczane przez reguły analizatora podziały wejściowego tekstu na segmenty. Węzły grafu są uporządkowane i odpowiadają pozycjom w tekście, które mogą stanowić granicę segmentu, natomiast pojedyncza krawędź obejmuje możliwy segment, łącząc jego początek z końcem. Każda ścieżka od pierwszego do ostatniego węzła odpowiada jednej z możliwych segmentacji. Segmentom przypisane są zbiory interpretacji (forma hasłowa oraz znacznik morfoskładniowy, tutaj przedstawione w uproszczonej postaci) zgodne ze słownikiem analizatora. Rysunek 3.7 przedstawia graf analizy dla zdania *Coś miałem.* Węzły 0 oraz 5 odpowiadają początkowi i końcowi tekstu. Węzły 2 oraz 4 stanowią granice segmentów wyznaczone przez spację oraz znak interpunkcyjny. Obecność węzłów 1 oraz 3 wynika z niejednoznacznej segmentacji słów *Coś* (pojedynczy rzeczownik coś lub rzeczownik co z cząstką posiłkową *-ś*) i *miałem* (czasownik MIEĆ z cząstką posiłkową *-em* lub forma rzeczownika MIAŁ).

Punktem wyjścia dla procedury segmentacji zaimplementowanej w parserze Hydra jest maksymalny (najdrobniejszy) podział zdania odczytany z grafu analizy Morfeusza. Oznacza to, że początkowo każdy węzeł grafu jest uznawany za potencjalną granicę segmentu. Dla przykładowego tekstu *Coś miałem.* i grafu z rysunku 3.7 takim podziałem będzie *Co·ś·miał·em·.* Każdemu segmentowi przypisywana jest binarna etykieta sygnalizująca, czy w danych uczących stanowi on faktycznie początek nowego segmentu (`B`, *beginning*) czy kontynuację poprzedniego (`I`, *inside*). Tabela 3.2



Rysunek 3.7: Graf analizy Morfeusza.

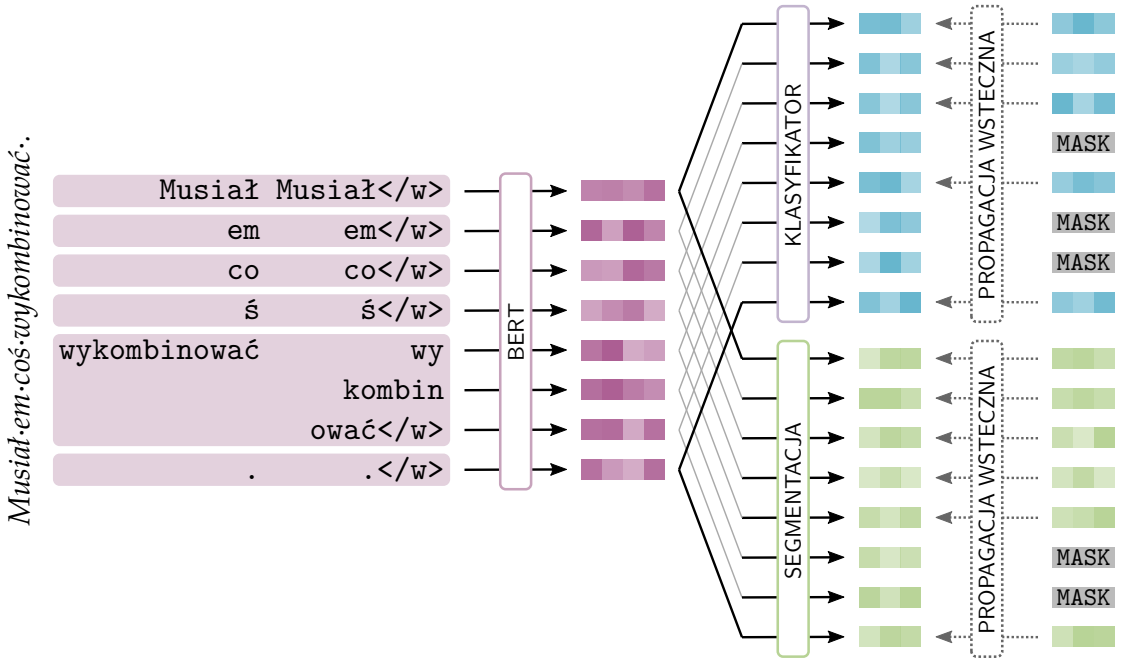
dane wejściowe	*Czyżby*		*chciał*	*mi*	*coś*		*powiedzieć*	*?*
Morfeusz	*Czyż*	*by*	*chciał*	*mi*	*co*	*ś*	*powiedzieć*	*?*
etykiety	B	I	B	B	B	I	B	B

Tabela 3.2: Etykiety dla segmentacji.

przedstawia etykiety B/I dla zdania *Czyżby chciał mi coś powiedzieć?* z prawidłową segmentacją podaną w pierwszym wierszu i maksymalną segmentacją Morfeusza w drugim wierszu.

Analogicznie, jak opisano to w punkcie 3.3, segmenty pochodzące z Morfeusza poddawane są tokenizacji BERT-a, a więc konieczne jest zastosowanie maskowania. Maskowanie dla klasyfikatora przewidującego segmentację jest inne niż dla pozostałych klasyfikatorów — uwzględniane są pierwsze tokeny segmentów Morfeusza, a maskowane pozostałe. Różnicę w maskowaniu ilustruje rysunek 3.8. Dla zdania wejściowego *Musiałem coś wykombinować.* token ś</w> (pochodzący ze słowa *coś*) zostaje dla klasyfikatorów nieprzeznaczonych do segmentacji zamaskowany, ponieważ nie jest początkiem faktycznego segmentu w tym zdaniu — nie jest mu przypisany znacznik morfoskładniowy ani żadna z etykiet kodujących drzewo. Stanowi on natomiast początek (i zarazem całość) segmentu w grafie Morfeusza, dlatego posiada etykietę segmentacji (w tym przypadku I).[19]

[19] Zbliżona koncepcja stoi za metodą segmentacji zaimplementowaną w narzędziu Trankit (Nguyen *et al.*, 2021), gdzie klasyfikacji jako początek lub kontynuacja segmentu poddawane są bezpośrednio tokeny pochodzące z podziału dla modelu językowego.



Rysunek 3.8: Maskowanie tokenizacji BERT-a vs maskowanie segmentacji.

3.4.2. Tagowanie i lematyzacja

Możliwość rozszerzenia działania parsera Hydra o tagowanie morfoskładniowe, czyli przypisanie każdemu segmentowi wypowiedzenia jego charakterystyki fleksyjnej, jest oczywista. Wystarczy dodać kolejny klasyfikator (w postaci warstwy gęstej), przewidujący odpowiednie znaczniki jako etykiety wyjściowe. Zestaw możliwych napisów (tagów), które kodują cechy morfoskładniowe segmentów, nazywany jest *tagsetem*.

Zadanie lematyzacji (hasłowania), czyli przypisania segmentom odpowiednich form hasłowych, jest natomiast realizowane za pomocą dwóch klasyfikatorów, przewidujących regułę lematyzacji oraz kasztowość pierwszej litery lematu (pisownię wielką lub małą literą). Reguły lematyzacji składają się z trzech sygnałów, odpowiadających trzem operacjom na napisach prowadzącym do przekształcenia formy tekstowej segmentu we właściwą formę hasłową: usunięcie pewnej liczby liter na początku, usunięcie pewnej liczby liter na końcu oraz dopisanie sufiksu. Na przykład, aby z formy imiesłowu *niedojedzonymi* uzyskać lemat *dojeść*, należy usunąć 3 pierwsze litery (uzyskując napis *dojedzonymi*), usunąć 7 ostatnich liter (uzyskując *doje*), i wreszcie dopisać sufiks *-ść*. Reguła ta reprezentowana jest jako `3_7_ść`. Tabela 3.3 przedstawia przykładowe zdanie[20] wraz z etykietami reguł lematyzacji i kasztowości odpowiadającymi jego segmentom. Reguła `0_0_`, właściwa dla segmentów *Może*,

[20] Źródło: KWJP.

segment	lemat	reguła	kasztowość
Może	może	`0_0_`	`x`
medale	medal	`0_1_`	`x`
nie	nie	`0_0_`	`x`
są	być	`0_2_być`	`x`
najważniejsze	ważny	`3_6_y`	`x`
dla	dla	`0_0_`	`x`
Polaków	Polak	`0_2_`	`X`
?	?	`0_0_`	`x`

Tabela 3.3: Etykiety dla lematyzacji.

nie, *dla* oraz *?*, oznacza brak jakichkolwiek zmian — lematy tych segmentów są im równokształtne z dokładnością do kasztowości. Reguła `0_2_być`, zastosowana do segmentu *są* składającego się z dwóch liter, skutkuje zastąpieniem całego napisu przez *być*. Segment *Polaków* ma etykietę kasztowości `X`, zatem przypisany mu lemat Polak pisany jest z wielkiej litery. Pozostałe segmenty posiadają etykietę `x`, oznaczającą małą literę — w szczególności dla rozpoczynającego zdanie *Może* właściwym lematem jest może.

Dyskusja dotycząca wyboru opisywanej metody lematyzacji została umieszczona w punkcie 3.5.3.

Korekta znakowania fleksyjnego

Mimo starań o jak najlepszą jakość danych uczących oraz konstrukcję modeli, błędne wyniki automatycznego przetwarzania są w zasadzie nieuniknione, a celem jest minimalizacja ich liczby. Zgodność lematyzacji wykonanej narzędziem Hydra z danymi wzorcowymi jest bardzo wysoka — stwierdzenie to zostanie podparte wynikami empirycznymi w punkcie 4.3. Ponieważ jednak każda możliwość poprawy jakości automatycznego znakowania jest ważna, została zaimplementowana prosta procedura korekty hasłowania przy użyciu obszernych danych słownikowych dostarczanych przez przywołany już w punkcie 3.4.1 analizator Morfeusz 2.

Warunkiem użycia procedury korekty jest obecność w modelu Hydry zarówno modułu lematyzacji, jak i tagowania. Dla każdego segmentu s zinterpretowanego automatycznie przeglądana jest wówczas lista jego interpretacji pochodząca z Morfeusza. Niech l oraz t będą odpowiednio formą hasłową oraz znacznikiem fleksyjnym przypisanymi przez model:

1. Jeśli segment nie jest notowany w słowniku analizatora, l i t pozostają bez zmian.
2. W przeciwnym wypadku spośród interpretacji podanych przez Morfeusza wybierany jest zbiór kandydatów. Dla każdej interpretacji $\langle l', t', k\rangle$, gdzie l' to lemat, t' to znacznik, a k — lista kwalifikatorów stylistycznych,[21] para $\langle l', t'\rangle$ zostaje włączony do zbioru, jeśli zachodzą oba następujące warunki:
   — Zgadza się kasztowość pierwszej litery lematów l i l' (oba zaczynają się małą literą lub oba wielką) lub s zawiera > 1 wielką literę.
   — Znaczniki t i t' są (niemal[22]) identyczne.

   Każda dopasowana w ten sposób interpretacja otrzymuje rangę w postaci pary liczb $\langle r_1, r_2\rangle$, gdzie:
   — r_1 wynosi 0, jeśli $t = t'$, a w przeciwnym przypadku (zob. przypis powyżej) — 1.
   — r_2 wynosi 0, jeśli lista k jest pusta; 1, jeśli jest niepusta, ale nie zawiera wartości `daw.`, `przest.`, `rzad.`, `środ.`, `gwar.` ani `indyw.`; 2 w przeciwnym przypadku.
3. Jeśli zbiór kandydatów jest niepusty, wybierane są z niego elementy $\langle l_1, t_1\rangle$, ... $\langle l_n, t_n\rangle$ (gdzie $n \geq 1$) o najniższej randze $\langle r'_1, r'_2\rangle$:
   — Jeśli $r'_1 = 0$ (udało się dopasować interpretację Morfeusza ze znacznikiem t), lemat l zastępowany jest przez l_1.
   — Jeśli $r'_1 = 1$ oraz $n = 1$ (dopasowanie znacznika nie jest dokładne, ale wybór najlepszego kandydata jest jednoznaczny), l i t zastępowane są odpowiednio przez l_1 i t_1.
4. Jeśli w poprzednim kroku nie wybrano nowej interpretacji, l i t pozostają bez zmian.

W tabeli 3.4 zostały zebrane przykłady działania opisanej procedury w odniesieniu do hipotetycznych decyzji Hydry. W pierwszym przykładzie model poprawnie uznał słowo *uprzęży* za rzeczownik rodzaju żeńskiego w dopełniaczu liczby pojedynczej (`subst:sg:gen:f`), ale przypisał mu lemat `uprzęża`.[23] Brak interpretacji `uprzęża subst:sg:gen:f` oraz dopasowanie znacznika w interpretacji `uprząż subst:sg:gen:f` powoduje zmianę formy hasłowej na prawidłową `uprząż`. Drugi przykład jest podobny, ale dodatkowo ilustruje zastosowanie reguły szeregującej kandydatów na nową interpretację na podstawie kwalifikatorów. Dawny lemat `podróża` (równoznaczny współczesnemu `podróż`) otrzymuje dalszą rangę, zatem spośród

[21] Oprócz tych trzech informacji, analizy Morfeusza zawierają również klasyfikację nazw własnych, jednak ta ostatnia nie jest brana pod uwagę w opisywanej procedurze.

[22] Dopuszczalna jest pojedyncza różnica w wartościach na mniej istotnych pozycjach znacznika.

[23] Decyzja taka jest oczywiście błędna, ale nie całkowicie bezsensowna z punktu widzenia modelu statystycznego. Końcówka *-a* i jej wymiana na -y przy tworzeniu formy dopełniacza lp. jest w języku polskim najpowszechniejsza wśród rzeczowników rodzaju żeńskiego, z kolei rzeczowniki żeńskie kończące się na *-ąż* nie są częste.

model	Morfeusz	wybrana interpretacja
uprzęża subst:sg:gen:f	uprząż subst:sg:gen:f [] uprząż subst:sg:dat:f [] (...) kontekst:	uprząż subst:sg:gen:f *Wspinał się bez uprzęży.*
podróża subst:pl:loc:f	podróża subst:pl:loc:f [daw.] podróż subst:pl:loc:f [] kontekst:	podróż subst:pl:loc:f *Marzę o dalekich podróżach.*
Eklerka subst:sg:nom:f	eklerka subst:sg:nom:f [] eklerek subst:sg:gen.m2 [] eklerek subst:sg:acc:m2 [] kontekst:	Eklerka subst:sg:nom:f *Cukiernia „Eklerka”*
Łapka subst:sg:nom:m2	łapka subst:sg:nom:f [] kontekst:	Łapka subst:sg:nom:m2 *Kocur nazywa się Łapka.*

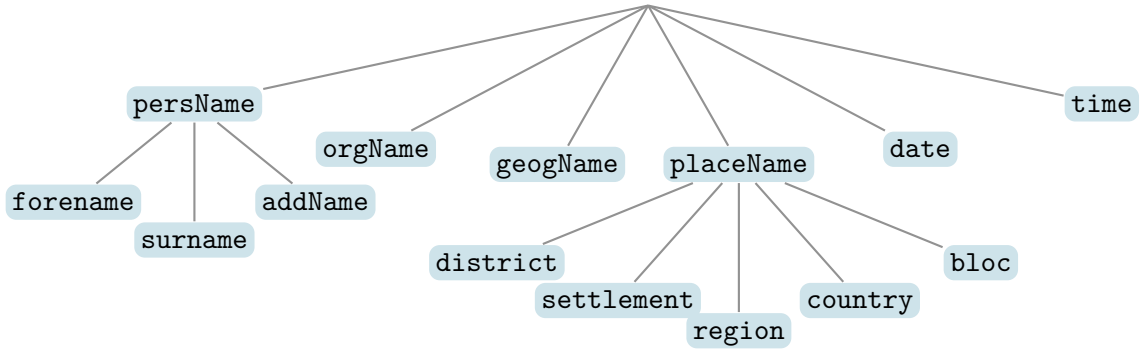
Tabela 3.4: Przykłady działania korekty znakowania fleksyjnego.

dwóch lematów powiązanych z tym samym znacznikiem preferowana jest bardziej adekwatna wersja podróż. Kolejny przykład pokazuje działanie reguły dotyczącej kasztowości liter w lematach: bez niej lemat słowa *Eklerka*, występującego w podanym kontekście nietypowo — jako nazwa własna — zostałby niepotrzebnie zmieniony na odpowiadający rzeczownikowi pospolitemu. W ostatnim przykładzie brakuje dopasowania na poziomie znaczników, zatem ponownie zostaje zachowana (prawidłowa) decyzja modelu: chodzi o imię zwierzęcia, o którym mówimy w rodzaju męskim żywotnym.

3.4.3. Rozpoznawanie jednostek nazewniczych

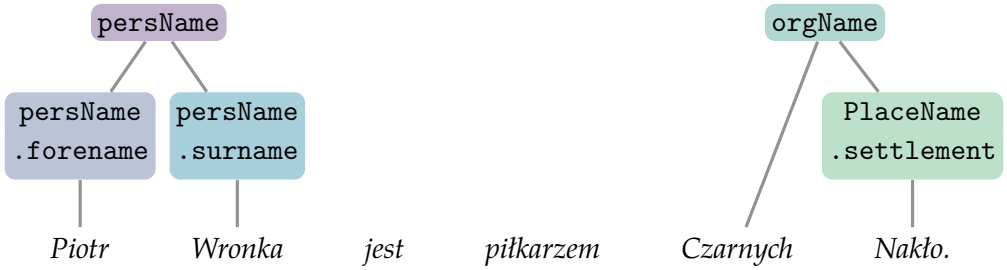
Kolejnym zadaniem z dziedziny przetwarzania języka naturalnego, o którego implementację rozszerzony został parser Hydra, jest rozpoznawanie jednostek nazewniczych, czyli wykrywanie w tekście fragmentów stanowiących nazwy własne (określenia osób, organizacji, nazwy geograficzne itp.). W przeprowadzonych eksperymentach została wykorzystana warstwa anotacji jednostkami nazewniczymi obecna w milionowym, ręcznie znakowanym podkorpusie NKJP. Przyjęta w tym korpusie hierarchia typów jednostek została przytoczona za Savary *et al.* (2012)

na rysunku 3.9. Schemat anotacji przewiduje 14 różnych etykiet zorganizowanych w dwupoziomową hierarchię — dwa typy z pierwszego poziomu (`persName`, nazwa osoby, oraz `placeName`, nazwa miejsca) posiadają odpowiednio trzy i pięć podtypów. Typy jednostek z drugiego poziomu hierarchii mogą odpowiadać częściom składowym jednostek swojego typu nadrzędnego lub stanowić samodzielne jednostki.



Rysunek 3.9: Typy jednostek nazewniczych w znakowaniu NKJP.

Rysunek 3.10 pokazuje anotację jednostkami nazewniczymi dla przykładowego zdania z NKJP *Piotr Wronka jest piłkarzem Czarnych Nakło.* Widać tu, że anotacja przewiduje zagnieżdżanie jednostek. Imię (`personName.forename`) i nazwisko (`personName.surname`) składają się na nazwę osoby *Piotr Wronka*. Nazwa organizacji (`orgName`) *Czarnych Nakło* natomiast zawiera w sobie nazwę miejscowości (`PlaceName.settlement`) *Nakło*.



Rysunek 3.10: Przykład znakowania jednostkami nazewniczymi w NKJP.

Na potrzeby działania modelu jednostki nazewnicze zostały zakodowane w notacji IOB (z ang. *inside, outside, beginning* — wewnątrz, na zewnątrz, początek): pierwszy segment jednostki typu `x` otrzymuje etykietę `Bx`, kolejne zaś — `Ix`. Segmenty niewchodzące w skład żadnej jednostki oznaczane są jako `O`. Tabela 3.5 przedstawia segmenty zdania *Piotr Wronka jest piłkarzem Czarnych Nakło.* opatrzone etykietami nazw własnych zgodnie z anotacją z rysunku 3.10. W przypadku przypisania danemu segmentowi więcej niż jednej etykiety (tutaj: segmenty *Piotr*, *Wronka* oraz

segment	etykiety nazw własnych	
Piotr	B:persName	B:persName.forename
Wronka	I:persName	B:persName.surname
jest	O	
piłkarzem	O	
Czarnych	B:orgName	
Nakło	I:orgName	B:placeName.settlement
.	O	

Tabela 3.5: Etykiety jednostek nazewniczych.

Nakło) w wyniku zagnieżdżenia jednostek, zbiór wszystkich przypisanych etykiet traktowany jest jako wartość atomowa.

3.5. Inne rozważane warianty parsera

W tym punkcie rozprawy przedyskutowane zostaną powody przyjęcia określonych rozwiązań w architekturze parsera. Wspomniane będą również rozstrzygnięcia alternatywne, które były brane pod uwagę podczas tworzenia Hydry, ale nie zostały wykorzystane w ostatecznej wersji parsera.

3.5.1. Reprezentacja kręgosłupów składniowych

Najważniejszym rozstrzygnięciem dotyczącym postaci kręgosłupów, na jakiej operuje parser, był wybór pomiędzy reprezentacją atomową a sekwencyjną. Ostatecznie wybrane przedstawienie atomowe oznacza, że każdy występujący w danych treningowych łańcuch nieterminali stanowiący kręgosłup pewnego segmentu traktowany jest jako niepodzielna całość. Z punktu widzenia modelu staje się on jedną ze zbioru etykiet, które można przydzielić segmentowi. Reprezentacja atomowa *a priori* „ukrywa" przed modelem wewnętrzną strukturę kręgosłupa oraz stopień jego podobieństwa do innych. Na przykład kręgosłupy VP → V, VP → VP → V oraz AdjP → Adj są na początku uczenia uznawane za nawzajem do siebie (nie)podobne w losowym stopniu. Wszelkich podobieństw i wzajemnych zależności pomiędzy typami kręgosłupów model musi od podstaw nauczyć się sam.

Alternatywnym branym pod uwagę na etapie projektowania architektury Hydry wariantem była reprezentacja kręgosłupów jako sekwencji nieterminali i dekodowanie ich na wyjściu z modelu węzeł po węźle. Takie podejście jest koncepcyjnie

atrakcyjne, ponieważ dawałoby parserowi możliwość nauczenia się zarówno znaczenia poszczególnych symboli nieterminalnych, jak i zasad łączenia ich w kręgosłupy. Dopuszczalność tworzenia kręgosłupów niespotykanych w danych uczących zwiększyłaby przy tym elastyczność modelu. Ta ostatnia cecha, obok większego skomplikowania modelu, może jednak również potencjalnie stanowić wadę, umożliwiając halucynowanie dowolnie długich, nieuzasadnionych lingwistycznie ścieżek w drzewach.

Ze względu na niewielkie liczby różnych kręgosłupów występujących w danych składniowych, z jakimi mamy do czynienia (zob. tab. 4.3), oraz dużo większą prostotę reprezentacji atomowej została ona wybrana jako pierwsza do zaimplementowania i wstępnego przetestowania. Jej skuteczność okazała się satysfakcjonująca, co w połączeniu z mniejszą złożonością zadecydowało o przyjęciu jej w kolejnych etapach rozwoju narzędzia Hydra.

Wśród atomowych sposobów przedstawiania kręgosłupów składniowych również zostały wzięte pod uwagę dwa podejścia. Obok faktycznie zastosowanego podejścia bezpośredniego, zakładającego zaczerpnięcie zestawu możliwych typów z danych bez żadnej modyfikacji, wypróbowana została również prosta metoda ich uogólniania. Polegała ona na stratnej „kompresji" grup nieterminali o tej samej kategorii składniowej. W ten sposób np. kręgosłupy NP → N, NP → NP → N, NP → NP → NP → N itd. zostały zebrane pod wspólną etykietą NP → N. Podczas konstruowania drzewa dodatkowe węzły NP byłyby nadbudowywane w miarę potrzeby zgodnie z przewidzianą przez model wysokością podłączenia do kręgosłupa krawędzi niecentralnych (zob. 3.1). Zaletami takiego podejścia są mniejsza liczba etykiet oraz bardziej „elegancki" i (pozornie) elastyczny sposób decydowania o wysokości kręgosłupa.

Wstępne eksperymenty nie wykazały istotnych różnic w jakości drzew produkowanych przez Hydrę przy zastosowaniu takiej skompresowanej reprezentacji kręgosłupów. Należy również zwrócić uwagę, że reprezentacja bezpośrednia w połączeniu z procedurami opisanymi w punkcie 3.2.3 również zostawia miejsce na dostosowanie liczby powtórzeń nieterminali w obrębie kręgosłupa. Struktura ostatecznie powstałego drzewa pozostaje zatem w podobny sposób elastyczna względem decyzji klasyfikatora dla kręgosłupów składniowych. Wobec powyższych spostrzeżeń wybrana została reprezentacja bezpośrednia.

3.5.2. Architektura klasyfikatora dla etykiet zależnościowych

We współczesnych neuronowych parserach grafowych (Dozat i Manning 2017 oraz pochodne architektury, zob. 2.2.2) do przewidzenia typów relacji zależnościowych wykorzystywana była — mniej lub bardziej bezpośrednio — obliczona przez

model struktura powiązań zależnościowych pomiędzy segmentami. Dla ustalenia uwagi w przytaczanych dalej wzorach przyjmujemy, że liczba możliwych etykiet wynosi l, natomiast liczba segmentów w przetwarzanym wypowiedzeniu — s. Na przykład Dozat i Manning (2017) podają następującą formułę[24] dla obliczenia wektora $labels_i$ reprezentującego rozkład prawdopodobieństwa etykiet dla relacji łączącej segment i z jego nadrzędnikiem j wyznaczonym uprzednio przez moduł przewidujący nieetykietowane drzewo ($|labels_i| = L$):

$$labels_i := h_j^{\mathrm{T}} \cdot \mathbf{U} \cdot d_i \tag{3.2}$$

W powyższym wzorze h_j oraz d_i to pośrednie reprezentacje wektorowe nadrzędnika i podrzędnika (por. podobny zabieg w pkt. 3.2.2) odpowiednio dla segmentu i oraz j, a $\mathbf{U}$ jest wytrenowaną macierzą parametrów.

Z kolei parser COMBO (Klimaszewski i Wróblewska, 2021) przewiduje dystrybucję etykiet poprzez zastosowanie warstwy gęstej modelu neuronowego (czyli przekształcenia wyrażonego przez wytrenowaną macierz $\mathbf{W}$) do konkatenacji dwóch wektorów:

$$labels_i := (a_i^{\mathrm{T}} \cdot H \oplus d_i) \cdot \mathbf{W} \tag{3.3}$$

Pierwszy z nich stanowi sumę wszystkich reprezentacji nadrzędnika (zebranych w macierzy H o wymiarach $s \times e$, gdzie e jest długością reprezentacji wektorowych h) ważoną i-tym wierszem obliczonej uprzednio macierzy wag krawędzi drzewa (wektor a_i o długości s).[25] Drugi wektor to reprezentacja podrzędnika dla i.

Sposób przewidywania typów relacji zależnościowych zastosowany w Hydrze jest w porównaniu z przytoczonymi powyżej bardzo prosty — sprowadza się do zastosowania warstwy gęstej do reprezentacji b_i segmentu i dostarczonej przez model BERT: $labels_i := b_i^{\mathrm{T}} \cdot \mathbf{W}$. Reprezentacja ta sama w sobie osadzona jest w kontekście całego zdania, a łączne dostrajanie modelu językowego pozwala przypuszczać, że produkowane przezeń reprezentacje zawierają informację składniową. Mimo to nasuwa się oczywiste pytanie, czy takie podejście nie jest jednak zbyt naiwne i nie powoduje obniżenia jakości analizy składniowej.

Ponieważ w momencie podjęcia prac nad przedstawianym w niniejszej rozprawie parserem najlepszym dostępnym narzędziem automatycznej analizy zależnościowej

[24] We wszystkich przytaczanych tutaj wzorach dla uproszczenia pomijamy wyrazy wolne występujące w stosowanych przekształceniach oraz następujące po nich nieliniowe aktywacje warstw modeli neuronowych i skupiamy się na interakcji pomiędzy reprezentacjami wektorowymi poszczególnych segmentów.

[25] Jest to zatem reprezentacja nadrzędnika segmentu i uwzględniająca stopień pewności modelu co do jego wyboru — im bardziej dana wartość a_{ij} zbliża się do jedności, z tym większym „przekonaniem"model wskazuje j jako nadrzędnik i, a udział h_j w średniej ważonej z H rośnie.

dla polszczyzny było COMBO, właśnie jego architektura została przyjęta za punkt odniesienia. Zaimplementowano oraz przetestowano trzy warianty klasyfikatora dla etykiet zależnościowych o architekturze analogicznej jak w COMBO:

— z osobnymi pośrednimi reprezentacjami ($h_i \neq d_i$, architektura analogiczna do COMBO),
— ze wspólnymi pośrednimi reprezentacjami ($h_i = d_i$),
— z reprezentacjami uzyskanymi bezpośrednio z modelu BERT ($h_i = d_i = b_i$).

Wyniki tych wstępnych eksperymentów, wraz z wynikami uzyskanymi za pomocą COMBO, a także prostego klasyfikatora ostatecznie zastosowanego w Hydrze, zebrane zostały w tabeli 3.6. Trenowanie oraz testy przeprowadzono na danych Polskiego Banku Drzew Zależnościowych w wersji UD (zob. 1.2.2). Uzyskane wyniki miary LAS (zob. 4.2.1), uwzględniającej zarówno poprawność krawędzi, jak ich etykiet zależnościowych, są o około punkt procentowy wyższe niż w przypadku COMBO, natomiast pomiędzy sobą różnią się jedynie minimalnie. Uzyskane wartości sugerują, że zanurzenia wektorowe generowane przez model BERT zawierają informację wystarczająco bogatą, aby wprowadzenie do klasyfikatora informacji o strukturze drzewa nie wpływało znacząco na jego skuteczność.

parser	**wariant**	**UAS**	**LAS**
COMBO		95,60%	93,93%
Hydra	$h_i \neq d_i$	96,61%	94,97%
	$h_i = d_i$	96,51%	94,91%
	$h_i = d_i = b_i$	96,49%	94,86%
	prosty klasyfikator	96,53%	94,94%

Tabela 3.6: Jakość parsowania zależnościowego z użyciem różnych architektur.

Na zakończenie tego punktu warto nadmienić, że w przypadku klasyfikatora dla krawędzi zależnościowych próby podobnego uproszczenia nie przyniosły wyników wartych dalszego badania. Sprowadzenie parsowania zależnościowego wymaga w szczególności decyzji, jaką informację przekazuje pojedyncza etykieta. Może to być po prostu pozycja nadrzędnika w wypowiedzeniu lub, na przykład, jego odległość od segmentu podrzędnego. Nieoczywiste jest również rozstrzygnięcie, jak reprezentować taką etykiety na potrzeby modelu. W przypadku liczby rzeczywistej trudności nastręczają wartości niecałkowite oraz wykraczające poza długość zdania. Z kolei wybór etykiet kategorialnych z góry ogranicza dopuszczalną długość wypo-

wiedzenia. Efekty wstępnej eksploracji różnych możliwości okazały się jednocześnie na tyle słabe, aby porzucić ten kierunek badań.

3.5.3. Lematyzacja

Zadanie automatycznej lematyzacji tekstów w językach fleksyjnych, takich jak polski, jest niełatwe ze względu na wielość schematów, według których formy odmienione mogą być tworzone z podstawowych (przypisanie formie tekstowej lematu wymaga mniej lub bardziej jawnego odwrócenia tej operacji). Na przykład SGJP (Saloni *et al.*, 2015) notuje około 1000 wzorów odmiany — swego rodzaju klas abstrakcji paradygmatów odmiany[26] polskich słów sklasyfikowanych ze względu na końcówki poszczególnych form. Uproszczony przykład wzoru odmiany z SGJP przedstawia tabela 3.7. W taki sposób odmieniają się na przykład rzeczowniki SINGIEL, BAJGIEL CZY CYRKIEL, ale nie WŁAŚCICIEL, GABRIEL CZY TOPIEL.

	lp.	**lm.**
mianownik	*sing* ***·iel***	*sing* ***·le***
dopełniacz	*sing* ***·la***	*sing* ***·lów ·li***
celownik	*sing* ***·lowi***	*sing* ***·lom***
biernik	*sing* ***·la***	*sing* ***·li ·le***
narzędnik	*sing* ***·lem***	*sing* ***·lami***
miejscownik	*sing* ***·lu***	*sing* ***·lach***
wołacz	*sing* ***·lu***	*sing* ***·le***

Tabela 3.7: Przykładowy wzór odmiany według SGJP.

Za ilustrację różnicy pomiędzy zróżnicowaniem fleksyjnym — a co za tym idzie: poziomem trudności zadania lematyzacji — języków polskiego oraz angielskiego może posłużyć porównanie dwóch zasobów: słownika analizatora Morfeusz 2 oraz danych fleksyjnych dla języka angielskiego opracowanych w projekcie UniMorph[27] (Batsuren *et al.*, 2022). Oba zasoby notują podobną liczbę form hasłowych — odpowiednio 348 098 oraz 399 757, natomiast pomiędzy liczbą form tekstowych występuje różnica jednego rzędu wielkości: 5 023 691 w danych polskich oraz 594 776 w angielskich. Oznacza to, że dla każdej angielskiej formy hasłowej notowanych jest średnio

[26] Paradygmat odmiany leksemu w rozumieniu Saloniego to komplet jego form fleksyjnych ustrukturyzowany według kategorii fleksyjnych. Na przykład paradygmaty dla rzeczowników przewidują miejsca (klatki) dla form w poszczególnych przypadkach i liczbach.

[27] https://github.com/unimorph/eng

1,53 różnych tekstowo form fleksyjnych, natomiast w przypadku słownika polskiego liczba ta wynosi 15,08.

Lematyzacja poprzez przypisanie formy hasłowej wprost, jako etykiety wybieranej z zamkniętego zbioru przez klasyfikator, jest metodą w oczywisty sposób niemożliwą do sensownego zaimplementowania, niezależnie od języka — zbiór słownictwa dowolnego z nich, nawet ograniczony do leksemów poświadczonych korpusowo, jest ogromny. Dla języków o ograniczonej fleksji, jak angielski, można pokusić się o ręczne opracowanie odpowiednich reguł czy procedur. W przypadku języka polskiego zadanie lematyzacji było przez długi czas powiązane z dezambiguacją morfoskładniową (wyborem właściwych interpretacji z zamkniętego zbioru pochodzącego z analizatora fleksyjnego), stanowiąc w pewnym sensie jej skutek uboczny.

Wczesne tagery dla polszczyzny były technicznie rzecz biorąc dezambiguatorami (zob. Radziszewski i Acedański 2012). Stosowane w nich modele statystyczne decydowały o wyborze interpretacji przede wszystkim na podstawie znacznika, co determinowało przypisanie segmentowi powiązanej z tym znacznikiem formy hasłowej. Takie podejście pociągało za sobą w szczególności dwa problemy: niejednoznaczności nierozstrzygalne na poziomie znacznika fleksyjnego (*mają* to forma finitywna liczby mnogiej trzeciej osoby, ale którego czasownika: MIEĆ czy MAIĆ?) oraz słowa w ogóle nieznane analizatorowi. Przypisanie interpretacji w takich przypadkach wymagało zastosowania mniej lub bardziej wyrafinowanych heurystyk, dodatkowego modułu statystycznego — tzw. *guessera* — do lematyzacji słów nieznanych, itp. Pierwszym dezambiguatorem zaimplementowanym dla polszczyzny był tager trigramowy Dębowskiego (2004). W nurt ten wpisały się następnie TaKIPI (Piasecki, 2007), wykorzystana w automatycznej anotacji pełnego korpusu NKJP Pantera (Acedański, 2010), WMBT (Radziszewski i Śniatowski, 2011), WCRFT (Radziszewski, 2013) czy Concraft (Waszczuk, 2012). O analizie morfoskładniowej dla języka polskiego zob. też Kobyliński i Kieraś (2016); Krasnowska-Kieraś i Kobyliński (2019).

Wektorowe zanurzenia słów wykorzystywane przez współczesne sieci neuronowe wydają się na tyle bogate w informację, że można pokusić się o pominięcie wsparcia w postaci wstępnej analizy morfoskładniowej i działać wyłącznie na danych tekstowych. Oznacza to jednak, że zadanie przypisania segmentom form hasłowych spoczywa wyłącznie na modelu neuronowym: nie ma już niewielkiej listy interpretacji fleksyjnych do wyboru,[28] ale pełen repertuar znaczników oraz ogromna

[28] Pośrednie rozwiązanie zostało zaimplementowane na przykład w opartym o architekturę neuronową LSTM tagerze Toygger (Krasnowska-Kieraś, 2017): wybór znacznika nie był w żaden sposób ograniczony, ale informacja o wyniku analizy była dostępna dla modelu obok reprezentacji

przestrzeń napisów stanowiących potencjalne lematy (teoretycznie nieskończona, w praktyce ograniczona do tych o sensownej długości). Istnieją różne możliwe podejścia do tak sformułowanego problemu hasłowania. Na przykład we wspominanym już narzędziu COMBO (Klimaszewski i Wróblewska, 2021) moduł lematyzujący oparty jest na konwolucyjnej architekturze *encoder-decoder* operującej na pojedynczych znakach. Innymi słowy, lemat odtwarzany jest „znak po znaku". Metoda taka w żaden sposób nie ogranicza produkowanych przez model napisów. Z jednej strony może on dzięki temu w elastyczny sposób reagować na wszelkie nietypowe słowa w analizowanym tekście. Z drugiej — jak zawsze w przypadku głębokich sieci neuronowych — istnieje ryzyko halucynacji napisów odbiegających bardzo daleko od właściwej odpowiedzi.

Reguły lematyzacji stanowiące podstawę zaimplementowanego w parserze Hydra mechanizmu lematyzacji (opisanego w punkcie 3.4.2) są inspirowane opisem fleksji w postaci wzorów odmiany SGJP. Stanowią one niejako operację odwrotną do zakodowanego we wzorach sposobu tworzenia form fleksyjnych z formy hasłowej. Należy przy tym zaznaczyć, że reguły nie są w żaden sposób pogrupowane w zestawy odpowiadające całym paradygmatom, tworzone są wyłącznie na podstawie pojedynczych par forma-lemat. Zastosowanie wzoru odmiany polega na dołączeniu określonych w nim sufiksów (*zakończeń*) do tego samego prefiksu formy hasłowej (nazywanego *trzonem*[29]). Reguły lematyzacji są natomiast wyznaczane tak, aby dokonywać jak najmniejszej modyfikacji — niepodlegający zmianie „trzon" może różnić się pomiędzy formami tego samego lematu. Przykładowe różnice pomiędzy operacjami na napisach zachodzącymi przy aplikowaniu wzorów odmiany oraz reguł lematyzacji przedstawia tabela 3.8.

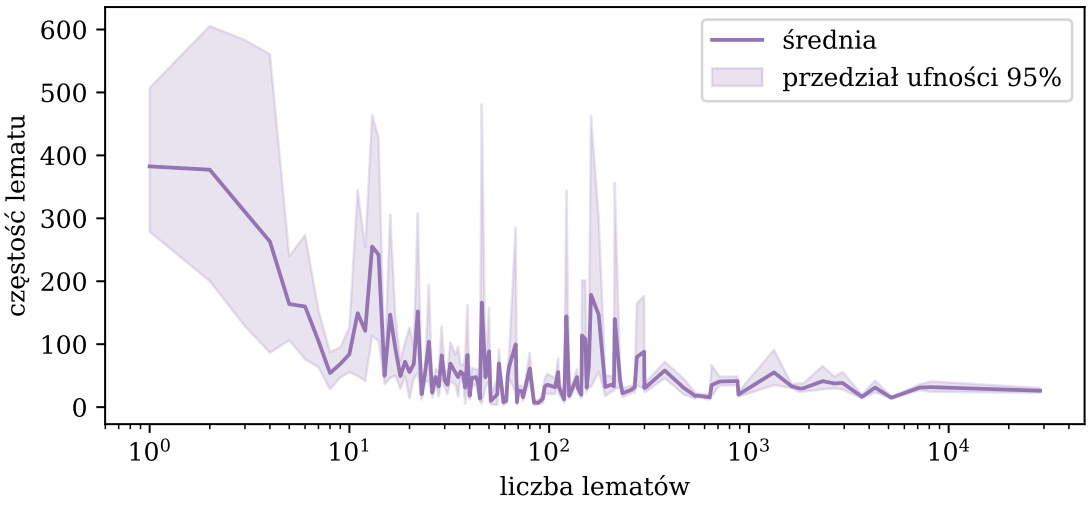
	odmiana wg. wzoru	**lematyzacja wg. reguły**	
mianownik lp.	*ko* ·***t***	`kot +`ϵ	`0_0_`
dopełniacz lp.	*ko* ·***ta***	`kot ~~a~~ +`ϵ	`0_1_`
wołacz lp.	*ko* ·***cie***	`ko ~~cie~~ +t`	`0_3_t`

Tabela 3.8: Porównanie odmiany według wzoru SGJP i lematyzacji za pomocą reguł.

wektorowej segmentu. Narzędzie to przypisywało jednak wyłącznie znaczniki morfoskładniowe, nie wykonując lematyzacji.

[29] Trzon oraz zakończenie w rozumieniu SGJP nie są tożsame z tradycyjnymi, wprowadzanymi już w składni szkolnej pojęciami tematu i końcówki. W szczególności trzon stanowi (niekoniecznie maksymalny) wspólny prefiks wszystkich form paradygmatu, temat natomiast może abstrahować od oboczności (np. odmiana *kot — kocie* opisywana jest tradycyjnie jako połączenie tematu *kot* z końcówką *-e* z obocznością w postaci zmiękczenia ostatniej głoski tematu z *t* na *ci*).

Zestaw reguł indukowany jest z danych, zatem sposoby, na jakie model może potencjalnie przypisać segmentowi formę hasłową, są ograniczone do przekształceń poświadczonych w danych. Może to budzić obawę o możliwość poprawnej lematyzacji w przypadku pojawienia się słów nieobecnych w korpusie uczącym — czy w repertuarze operacji na napisach dostępnym parserowi nie zabraknie tych właściwych dla słów bardzo rzadkich, neologizmów itd. W językach naturalnych istnieje jednak ogólna tendencja, w myśl której w sposób idiosynkratyczny odmieniają się słowa częste i podstawowe, natomiast te występujące rzadziej przyjmują paradygmaty bardziej produktywne. Za przykład posłużyć mogą polskie słowa IŚĆ, CZŁOWIEK czy ZŁY, które wymagają osobnych wzorów ze względu na supletywizmy: *iść–szła*, *człowiek–ludzie*, *zły–gorszy*, a jednocześnie należą do grupy podstawowego słownictwa i występują często w tekstach mówionych i pisanych. Natomiast empiryczną ilustrację tego faktu stanowić może proste obliczenie wykonane na tekstach NKJP 1M. Dla zebranych z korpusu reguł lematyzacji wyznaczono z jednej strony liczbę rożnych lematów, do których uzyskania posłużyła ta reguła, a z drugiej — średnią frekwencję form tych lematów. Zależność ta przedstawiona została w postaci wykresu na rysunku 3.11. Widać, że jeśli reguła jest „wyspecjalizowana” do niewielu leksemów (jak na przykład reguła `0_6_zły`, stosowalna wyłącznie do form *gorszy* i *niegorszy*), to leksemy te wykazują dużą częstość. Z kolei reguły powiązane z wieloma różnymi lematami mają tendencję do dopasowywania się do umiarkowanie częstych i rzadszych słów. Biorąc pod uwagę powyższe, można spodziewać się, że parser będzie „znał” schemat odmiany, do którego należy dopasować nowo napotkane słowa.



Rysunek 3.11: Empiryczna zależność między nietypowością a produktywnością reguł lematyzacji.

Aby dokładniej przetestować tak postawioną hipotezę, skonfrontowano reguły lematyzacji pozyskane z NKJP 1M ze zbiorem danych PolEval (zob. 4.3). Dla każdego segmentu w tym drugim korpusie sprawdzono, czy w zbiorze zebranym z NKJP 1M obecna jest reguła pozwalająca przekształcić formę tekstową tego segmentu do przypisanej mu formy hasłowej. Istnienie takiej reguły jest warunkiem koniecznym, aby hipotetyczny model wytrenowany na pierwszym korpusie poprawnie zlematyzował dany segment. Wszystkie przedstawiane dalej wyliczenia przeprowadzono na dwa sposoby: zliczając każde wystąpienie pary forma-lemat osobno oraz zliczając unikatowe typy takich par w korpusie, a wyniki zebrane zostały w tabeli 3.9. Brak odpowiedniej reguły stwierdzono dla 0,11% spośród 55033 segmentów występujących w zbiorze danych PolEval. Oznacza to, że najlepszy możliwy model (wybierający właściwą regułę, o ile tylko ma do niej dostęp) wytrenowany na NKJP 1M miałby szansę osiągnąć dokładność lematyzacji 99,89%.

		wystąpienia		**typy**	
		liczba	**brak reguły**	**liczba**	**brak reguły**
łącznie		55 033	61 (0,11%)	17 927	44 (0,25%)
NKJP 1M	✓	41 061	—	13 375	—
	×	13 972	61 (0,44%)	4 552	44 (0,97%)
Morfeusz	✓	53 991	15 (0,03%)	17 166	15 (0,09%)
	×	1 042	46 (4,41%)	761	29 (3,81%)

Tabela 3.9: Pokrycie danych PolEval regułami lematyzacji pozyskanymi z NKJP 1M.

Obliczenia jak powyżej przeprowadzono także, dzieląc pary forma-lemat z danych PolEval według dwóch kryteriów. Pierwszym z nich była obecność danej pary w NKJP 1M — odpowiada to rozróżnieniu na przypadki poznane przez model podczas uczenia oraz nowo napotkane w danych testowych. W tym przypadku reguły zabrakło dla 61 spośród 13 972 wystąpień. Drugi podział uwzględniał obecność pary w słowniku analizatora Morfeusz 2. Dla segmentów znanych i nieznanych analizatorowi udało się znaleźć regułę odpowiednio w 99,97% oraz 95,59% przypadków. Z tą ostatnią liczbą związanych jest 46 przypadków, w tym wyrazy zapisane w sposób błędny.

Otrzymane wartości pokazują, że dla ponad 99% słów sobie „nieznanych" model posiada w swoim repertuarze regułę pozwalającą na właściwe przypisanie formy hasłowej. Skalę problemu niepełnego pokrycia danych w podejściu regułowym można zatem uznać za marginalną.

Rozdział 4

Eksperymenty i ewaluacja

4.1. Kwestie techniczne

Jako pretrenowane modele językowe zostały wykorzystane HerBERT w wersji LARGE[1] (Mroczkowski *et al.*, 2021) oraz German BERT[2] odpowiednio dla języków polskiego i niemieckiego. W przypadku języka polskiego został również przetestowany szereg innych modeli językowych. Omówieniu uzyskanych z ich użyciem wyników poświęcony jest punkt 4.6 tego rozdziału.

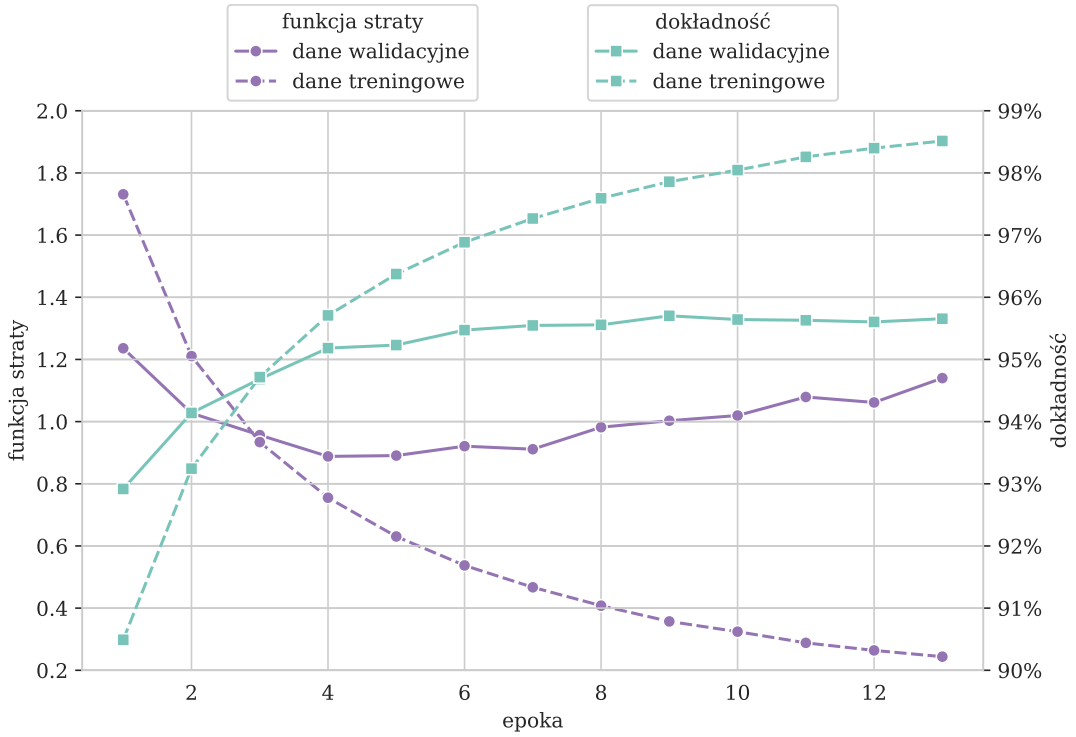
Do trenowania modeli został użyty algorytm optymalizacji RMSprop (Hinton *et al.*, 2012) z początkowym współczynnikiem uczenia (*learning rate*) 10^{-5}. Każdy z opisywanych w tym rozdziale modeli trenowany był przez maksymalnie 50 epok.[3] Jeżeli przez 4 kolejne epoki nie rosła wartość miary dokładności (*accuracy*) obliczanej łącznie (w postaci średniej) dla wszystkich wyjść z modelu przy użyciu danych walidacyjnych, proces uczenia był zatrzymywany oraz przywracane były wartości parametrów z epoki, po której dokładność była najwyższa.

Wybór miary dokładności zamiast funkcji straty (która oczywiście nadal stanowiła kryterium optymalizacji wag modelu) jako warunku wcześniejszego wstrzymania uczenia podyktowany był obserwacją empiryczną. Podczas trenowania modelu następuje moment, w którym wartość funkcji straty na danych treningowych nadal spada, ale dla danych walidacyjnych zaczyna powoli wzrastać. Sygnalizuje to ryzyko przeuczenia modelu: dopuszczenie do zbyt silnego dopasowania parametrów modelu do przykładów uczących odbywa się kosztem jego zdolności do generalizacji. W przypadku Hydry widać jednak, że po osiągnięciu (lokalnego) minimum funkcji straty przez kilka epok wciąż jeszcze rośnie dokładność. Dlatego to właśnie moment uzyskania lokalnego maksimum dokładności wyznacza koniec procesu trenowania.

[1] https://huggingface.co/allegro/herbert-large-cased

[2] https://huggingface.co/google-bert/bert-base-german-cased

[3] Epoka (*epoch*) w uczeniu maszynowym oznacza część przebiegu procedury uczenia, w której każdy element zbioru treningowego został wykorzystany dokładnie raz.



Rysunek 4.1: Przykładowy przebieg uczenia modelu Hydry.

Wykres na rysunku 4.1 przedstawia wartości funkcji straty oraz średniej dokładności w przykładowym przebiegu uczenia Hydry.

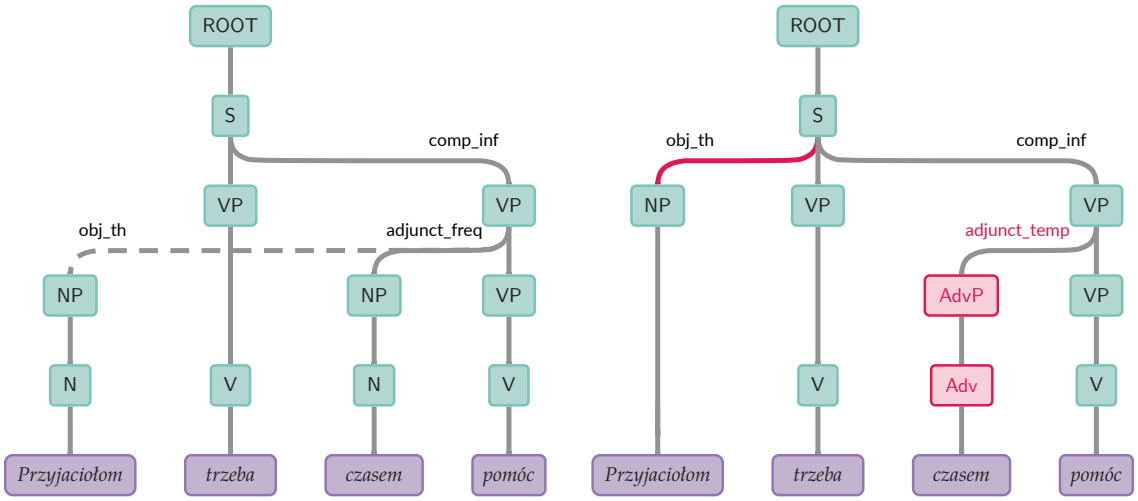
4.2. Analiza składniowa

4.2.1. Miary

Miary użyte do oceny skuteczności Hydry jako parsera składnikowo-zależnościowego zostaną wymienione poniżej oraz zilustrowane na przykładzie pary drzew przedstawionych na rysunku 4.2.[4] Przyjmujemy, że drzewo po lewej stronie należy do zbioru danych testowych, a to po prawej zostało wygenerowane z użyciem ewaluowanego modelu dla ciągu segmentów *Przyjaciołom*, *trzeba*, *czasem*, *pomóc*. W tabeli 4.1 na stronie 89 wypisane zostały elementy obu drzew służące do obliczenia poszczególnych miar oraz wynikające z nich wartości.

UAS (*Unlabeled Attachment Score*) to miara standardowo używana w ewaluacji parserów zależnościowych. Jest to odsetek segmentów w danych testowych, którym zostały przypisane właściwe nadrzędniki. W przytoczonym przykładzie jeden seg-

[4] Źródło zdania: KWJP.



Rysunek 4.2: Przykładowe drzewo wzorcowe oraz podlegające ocenie.

ment (*Przyjaciołom*) z czterech otrzymał niewłaściwy nadrzędnik: *trzeba* zamiast *pomóc*.

LAS (*Labeled Attachment Score*) jest kolejną miarą typową dla zadania parsowania zależnościowego. Jest ona bardziej restrykcyjna od UAS, ponieważ wyraża odsetek segmentów, którym zostały przypisane nie tylko właściwe nadrzędniki, ale również prowadzące do nich krawędzie opatrzone są prawidłowymi etykietami. W przykładzie jako błąd należy uwzględnić, oprócz segmentu *Przyjaciołom* o niewłaściwym nadrzędniku, również *czasem*, który został połączony z poprawnym nadrzędnikiem, ale za pomocą niepoprawnego typu relacji. Wartość miary LAS jest zawsze równa co najwyżej mierze UAS, a w tym przypadku jest od niej niższa.

Nawiasy — miara F_1[5] dla wyznaczonych przez parser (być może nieciągłych) sekwencji segmentów stanowiących plon pewnego wierzchołka nieterminalnego drzewa. Nazwa miary nawiązuje do notacji nawiasowej dla drzew składnikowych (zob. 1.1.1). Jak widać w tabeli 4.1, miara ta nie rozróżnia identycznych sekwencji segmentów stanowiących plon różnych wierzchołków — na przykład element (*Przyjaciołom trzeba czasem pomóc*) odpowiada w obu drzewach dwóm nieterminalom ROOT oraz S, ale uwzględniony jest tylko raz.

[5] Miara F_1 jest średnią harmoniczną miar precyzji (*precision*) oraz czułości (*recall*): $F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$. Precyzja klasyfikatora zdefiniowana jest jako odsetek liczby prawidłowych wskazań wśród wszystkich wykrytych przezeń elementów (tutaj: składników). Czułość natomiast to odsetek liczby faktycznie wskazanych przez klasyfikator elementów do wszystkich elementów występujących w danych testowych.

Składniki — miara F_1 obliczona dla wskazanych przez parser składników, czyli nieterminali wraz z etykietą oraz plonem. Miara ta jest w oczywisty sposób bardziej restrykcyjna od poprzedniej, ponieważ wymaga nie tylko wykrycia składnika, ale również poprawnego określenia wszystkich odpowiadających mu typów składniowych. W ten sposób w omawianym przykładzie zostają wzięte pod uwagę kolejne różnice między porównywanymi drzewami: niewłaściwe sklasyfikowanie *czasem* jako frazy przysłówkowej[6] oraz pominięcie preterminala N dla segmentu *Przyjaciołom*.

Składniki+centra — najbardziej restrykcyjna miara F_1 dla składników, wymagająca dodatkowo trafnego określenia, czy dany składnik stanowi centrum składniowe swojego rodzica. W omawianym przykładzie wartość tej miary jest identyczna jak poprzedniej, nie uwzględniono jej zatem w tabeli. Błędem wykrytym przez tę miarę, a niewidocznym dla miary *składniki* mogłoby zostać spowodowane na przykład uznaniem przez parser węzła AdvP za centralne dziecko $\mathsf{VP}_{\textit{czasem pomóc}}$.

Krawędzie — miara F_1 dla krawędzi drzewa hybrydowego. Dla każdej krawędzi brane są pod uwagę (czyli muszą zostać poprawnie określone): etykiety wierzchołków połączonych krawędzią, centralne segmenty tych dwóch wierzchołków oraz etykieta zależnościowa, którą jest opatrzona krawędź (lub brak etykiety dla krawędzi centralnych). W odróżnieniu od wcześniej wymienionych, miara ta została zaprojektowana specyficznie z myślą o drzewach hybrydowych, ponieważ uwzględnia zarówno składnikowy, jak zależnościowy aspekt struktury.[7] Według mojej wiedzy tak zdefiniowana miara nie występuje w literaturze przedmiotu. Wśród wykrytych przez nią w drzewach z rys. 4.2 nieprawidłowości są między innymi uznanie frazy *czasem* za przysłówkową (dotyczy wyłącznie struktury składnikowej) oraz przypisanie jej roli zależnościowej adjunct_temp zamiast adjunct_freq (dotyczy wyłącznie struktury zależnościowej).

[6] Klasyfikacja słowa *czasem* jako rzeczownika może nie być intuicyjna, ale wynika z decyzji o niewprowadzeniu do SGJP jednostek przysłówkowych *czasem*/*czasami*, która została utrzymana w analizatorze Morfeusz 2. Taka właśnie rzeczownikowa interpretacja obowiązuje zatem również w treebankach Składnica oraz Zależnica.

[7] Na miarę *krawędzie* można spojrzeć również jak na rodzaj rozszerzenia miary *LAS*. Dla krawędzi niecentralnych (odpowiadających zależnościowym) wymaga ona dodatkowo właściwych etykiet wejściowego i wyjściowego wierzchołka zależnościowego – pod tym względem jest więc bardziej restrykcyjna. Nie oznacza to jednak, że jej wartość nie może przekroczyć wartości *LAS*, ponieważ dodatkowo brane są pod uwagę krawędzie centralne, które nie mają odpowiedników zależnościowych.

	elementy			
	Przyjaciołom	*trzeba*	*czasem*	*pomóc*
$\mathcal{W}$	*pomóc*	ROOT	*pomóc*	*trzeba*
	obj_th	_	adjunct_freq	comp_inf
$\mathcal{P}$	*trzeba*	ROOT	*pomóc*	*trzeba*
	obj_th	_	adjunct_temp	comp_inf
	$UAS = 75\%$, $LAS = 50\%$			
$\mathcal{W}$	(*Przyjaciołom trzeba czasem pomóc*) (*Przyjaciołom czasem pomóc*)			
	(*Przyjaciołom*) (*trzeba*) (*czasem*) (*pomóc*)			
$\mathcal{P}$	(*Przyjaciołom trzeba czasem pomóc*) (*czasem pomóc*)			
	(*Przyjaciołom*) (*trzeba*) (*czasem*) (*pomóc*)			
	precision $= 5/6$, *recall* $= 5/6$, *nawiasy* $\approx 83\%$			
$\mathcal{W}$	$(\textit{Przyjaciołom trzeba czasem pomóc})_{\mathsf{ROOT}}$ $(\textit{Przyjaciołom trzeba czasem pomóc})_{\mathsf{S}}$			
	$(\textit{Przyjaciołom czasem pomóc})_{\mathsf{VP}}$ $(\textit{Przyjaciołom})_{\mathsf{NP}}$ $(\textit{Przyjaciołom})_{\mathsf{N}}$			
	$(\textit{trzeba})_{\mathsf{VP}}$ $(\textit{trzeba})_{\mathsf{V}}$ $(\textit{czasem})_{\mathsf{NP}}$ $(\textit{czasem})_{\mathsf{N}}$ $(\textit{pomóc})_{\mathsf{VP}}$ $(\textit{pomóc})_{\mathsf{V}}$			
$\mathcal{P}$	$(\textit{Przyjaciołom trzeba czasem pomóc})_{\mathsf{ROOT}}$ $(\textit{Przyjaciołom trzeba czasem pomóc})_{\mathsf{S}}$			
	$(\textit{czasem pomóc})_{\mathsf{VP}}$ $(\textit{Przyjaciołom})_{\mathsf{NP}}$			
	$(\textit{trzeba})_{\mathsf{VP}}$ $(\textit{trzeba})_{\mathsf{V}}$ $(\textit{czasem})_{\mathsf{AdvP}}$ $(\textit{czasem})_{\mathsf{Adv}}$ $(\textit{pomóc})_{\mathsf{VP}}$ $(\textit{pomóc})_{\mathsf{V}}$			
	precision $= 7/10$, *recall* $= 7/11$, *składniki* $\approx 67\%$			
$\mathcal{W}$	$(\mathsf{ROOT}_{\textit{trzeba}},\_,\mathsf{S}_{\textit{trzeba}})$ $(\mathsf{S}_{\textit{trzeba}},\_,\mathsf{VP}_{\textit{trzeba}})$ $(\mathsf{VP}_{\textit{trzeba}},\_,\mathsf{V}_{\textit{trzeba}})$			
	$(\mathsf{S}_{\textit{trzeba}},\mathsf{comp\_inf},_{\textit{pomóc}})$ $(\mathsf{VP}_{\textit{pomóc}},\_,\mathsf{VP}_{\textit{pomóc}})$ $(\mathsf{VP}_{\textit{pomóc}},\_,\mathsf{V}_{\textit{pomóc}})$			
	$(\mathsf{VP}_{\textit{pomóc}},\mathsf{obj\_th},\mathsf{NP}_{\textit{Przyjaciołom}})$ $(\mathsf{NP}_{\textit{Przyjaciołom}},\_,\mathsf{N}_{\textit{Przyjaciołom}})$			
	$(\mathsf{VP}_{\textit{pomóc}},\mathsf{adjunct\_freq},\mathsf{NP}_{\textit{czasem}})$ $(\mathsf{NP}_{\textit{czasem}},\_,\mathsf{N}_{\textit{czasem}})$			
$\mathcal{P}$	$(\mathsf{ROOT}_{\textit{trzeba}},\_,\mathsf{S}_{\textit{trzeba}})$ $(\mathsf{S}_{\textit{trzeba}},\_,\mathsf{VP}_{\textit{trzeba}})$ $(\mathsf{VP}_{\textit{trzeba}},\_,\mathsf{V}_{\textit{trzeba}})$			
	$(\mathsf{S}_{\textit{trzeba}},\mathsf{comp\_inf},_{\textit{pomóc}})$ $(\mathsf{VP}_{\textit{pomóc}},\_,\mathsf{VP}_{\textit{pomóc}})$ $(\mathsf{VP}_{\textit{pomóc}},\_,\mathsf{V}_{\textit{pomóc}})$			
	$(\mathsf{VP}_{\textit{trzeba}},\mathsf{obj\_th},\mathsf{NP}_{\textit{Przyjaciołom}})$			
	$(\mathsf{VP}_{\textit{pomóc}},\mathsf{adjunct\_temp},\mathsf{AdvP}_{\textit{czasem}})$ $(\mathsf{AdvP}_{\textit{czasem}},\_,\mathsf{Adv}_{\textit{czasem}})$			
	precision $= 6/9$, *recall* $= 6/10$, *krawędzie* $\approx 63\%$			

Tabela 4.1: Obliczanie wartości miar dla dla pary drzew z rysunku 4.2. Na niebiesko wyróżniono elementy drzewa wzorcowego $\mathcal{W}$ nieobecne w ocenianym $\mathcal{P}$, na czerwono — zbędne elementy drzewa wyznaczonego przez parser.

4.2.2. Parsowanie hybrydowe

Najistotniejszymi spośród opisywanych w tym rozdziale eksperymentów były te dotyczące automatycznej analizy zdań do postaci drzew hybrydowych. Do tego celu posłużyły dwa zasoby przedstawione już w rozdziale 1: Zależnica (zob. 1.2.3) oraz treebank hybrydowy stworzony na podstawie korpusu TIGER (zob. 1.2.4). Wymienione zbiory danych będą dalej oznaczane odpowiednio ZALEŻNICA oraz TIGER.

Tabela 4.2 zawiera informacje o liczbie drzew w obu treebankach z podziałem na ich części treningową, walidacyjną i testową, ze wskazaniem, jaka część zawartych w niej struktur hybrydowych zawiera nieciągłość składniową. Dla każdego podzbioru drzew podana jest również łączna liczba składających się na nie segmentów.

Tabela 4.3 zbiera podstawowe informacje o zróżnicowaniu danych pod względem rozmiarów zbiorów etykiet, spośród których muszą wybierać poszczególne klasyfikatory modelu (zob. 3.2.2). Odpowiednie liczby dla obu treebanków należą do tego samego rzędu wielkości, przy czym liczby różnych występujących w nich kręgosłupów oraz typów nadrzędnika są wyższe dla danych pochodzących z korpusu TIGER. Z kolei przejęty z PDB zestaw relacji zależnościowych jest liczniejszy od tego występującego w danych niemieckojęzycznych. Bardziej szczegółowe dane liczbowe dotyczące najczęściej występujących w obu zbiorach danych kręgosłupów składniowych (tab. A.2), typów nadrzędnika (tab. A.3) oraz etykiet zależnościowych (tab. A.4) zamieszczone zostały w załączniku A.

Podstawowym sposobem sprawdzenia skuteczności Hydry jako narzędzia do automatycznej analizy składnikowo-zależnościowej było wykonanie testu na danych hybrydowych ZALEŻNICA oraz TIGER. Tabela 4.4 przedstawia wartości miar wprowadzonych w punkcie 4.2.1 uzyskane dla danych polsko- i niemieckojęzycznych. Wartości miar *nawiasy*, *składniki* oraz *składniki+centra* zostały dodatkowo obliczone wyłącznie dla składników nieciągłych.

Wartości zamieszczone w tabeli są dość wysokie, szczególną uwagę zwracają przy tym wyniki dla języka polskiego, przekraczające 97% w przypadku miar biorących pod uwagę wyodrębnione przez parser składniki oraz niewiele niższe dla miar *UAS* oraz *krawędzie*. Obserwacja ta wskazuje, że udało się osiągnąć praktyczny cel prac nad Hydrą, jakim było stworzenie wysokiej jakości narzędzia do automatycznej analizy wypowiedzeń w języku polskim do postaci hybrydowych drzew składniowych.

Tabela B.1 przedstawia bardziej szczegółową ewaluację składnikowego aspektu działania modelu. Dla danych polskojęzycznych zostały obliczone miary precyzji, czułości oraz ich średnia harmoniczna F_1 osobno dla poszczególnych kategorii składniowych. W tabeli podana została również liczba faktycznych wystąpień składników każdego typu w danych testowych. Z kolei tabela B.2 zawiera te same miary

dane	podzbiór		wszystkie	ciągłe		nieciągłe	
ZALEŻNICA	treningowy	drzewa	17675	15918	(90,06%)	1757	(9,94%)
		segmenty	280466	239911	(85,54%)	40555	(14,46%)
		seg./drzewo	15,87	15,07		23,08	
	walidacyjny	drzewa	2212	1981	(89,56%)	231	(10,44%)
		segmenty	34580	29546	(85,44%)	5034	(14,56%)
		seg./drzewo	15,63	14,91		21,79	
	testowy	drzewa	2206	1991	(90,25%)	215	(9,75%)
		segmenty	33379	28564	(85,57%)	4815	(14,43%)
		seg./drzewo	15,13	14,35		22,40	
	łącznie	drzewa	22093	19890		2203	
		segmenty	348425	298021		50404	
		seg./drzewo	15,77	14,98		22,88	
TIGER	treningowy	drzewa	37725	25524	(67,66%)	12201	(32,34%)
		segmenty	709074	413878	(58,37%)	295196	(41,63%)
		seg./drzewo	18,80	16,22		24,19	
	walidacyjny	drzewa	4312	2986	(69,25%)	1326	(30,75%)
		segmenty	79588	47605	(59,81%)	31983	(40,19%)
		seg./drzewo	18,46	15,94		24,12	
	testowy	drzewa	4228	2961	(70,03%)	1267	(29,97%)
		segmenty	76315	46333	(60,71%)	29982	(39,29%)
		seg./drzewo	18,05	15,65		23,66	
	łącznie	drzewa	46265	31471		14794	
		segmenty	864977	507816		357161	
		seg./drzewo	18,70	16,14		24,14	

Tabela 4.2: Rozmiar i podział zbiorów danych ZALEŻNICA oraz TIGER.

kolumna	ZALEŻNICA	TIGER
kręgosłup	111	299
nadrzędnik	20	27
wys. nadrzędnika	3	4
relacja zależnościowa	72	44

Tabela 4.3: Zróżnicowanie etykiet w danych hybrydowych.

dane	UAS	LAS	nawiasy	składniki	składniki +centra	krawędzie
ZALEŻNICA	95,99%	90,70%	97,62%	97,58%	97,05%	95,73%
		nieciągłe:	48,92%	48,00%	47,85%	
TIGER	96,21%	94,96%	93,77%	92,58%	92,16%	94,80%
		nieciągłe:	74,84%	73,88%	73,86%	

Tabela 4.4: Wyniki ewaluacji parsowania hybrydowego.

obliczone dla relacji zależnościowych wyznaczonych przez parser. Macierz błędów dla relacji zależnościowych przedstawiona została na rysunku B.1.

Sensowna dyskusja przedstawionych powyżej wyników wymaga oczywiście porównania ze skutecznością innych narzędzi i metod parsowania. W przypadku Hydry jest to utrudnione przez bardzo specyficzny charakter konstruowanych przez nią reprezentacji składniowych. Z tego powodu w dwóch kolejnych punktach koncentrujemy się osobno na zależnościowym oraz składnikowym aspekcie działania opisywanego parsera.

4.2.3. Parsowanie zależnościowe

Motywacją do stworzenia narzędzia Hydra była możliwość automatycznego uzyskiwania hybrydowych struktur składniowych. Częścią procedury ich tworzenia jest jednak dokonanie rozbioru zależnościowego, dlatego Hydrę w oczywisty sposób można też wykorzystać jako parser *stricte* zależnościowy. Otwiera to możliwość wytrenowania i ewaluacji modeli na istniejących treebankach zależnościowych, a także porównania z innymi systemami do automatycznej analizy zależnościowej.

	parser	dane treningowe	dane testowe	UAS	LAS
	Hydra	ZALEŻNICA	ZALEŻNICA	96,29%	91,04%
	COMBO	ZALEŻNICA	ZALEŻNICA	94,63%	89,33%
	Hydra	UD 2.9	UD 2.9	96,53%	94,94%
⋆	COMBO	UD 2.9	UD 2.9	95,23%	93,66%
⋄	Trankit	UD	UD	95,89%	94,34%
⋄	Stanza	UD	UD	91,62%	89,34%
⋄	spaCy	UD	UD	89,41%	81,54%
⋄	UDPipe	UD	UD	86,82%	83,14%

Tabela 4.5: Parsowanie zależnościowe dla języka polskiego: wyniki ewaluacji i porównanie z innymi systemami.

W tabeli 4.5 wyniki ewaluacji Hydry na danych zależnościowych zostały zestawione z wartościami uzyskanymi dla innych narzędzi. Wiersze oznaczone symbolem ⋄ zostały zaczerpnięte z platformy ewaluacyjnej NLPre-PL[8] (Wiącek *et al.*, 2024). Symbol ⋆ oznacza wynik przytoczony na podstawie publikacji. Brak symbolu oznacza wartość uzyskaną eksperymentalnie w ramach prac opisywanych w niniejszej rozprawie. Wyniki otrzymane przy użyciu mojego modelu okazały się wyższe niż dotychczas raportowany poziom *state-of-the-art* dla języka polskiego osiągany przez systemy COMBO oraz Trankit. Dotyczy to zarówno danych ZALEŻNICA (w tym eksperymencie: ograniczonych do drzew zależnościowych), jak i UD 2.9 (treebank PDB-UD wchodzący w skład wydania 2.9 Universal Dependencies[9]). Wyraźnie niższe wartości *LAS* uzyskane dla tego samego systemu trenowanego i testowanego na treebanku ZALEŻNICA wynikają prawdopodobnie z bardziej szczegółowego repertuaru ról zależnościowych.[10]

Hydra w roli parsera zależnościowego została również wykorzystana w eksperymencie związanym z przetwarzaniem tekstów historycznych. Jako dane posłużyły w tym przypadku PDB oraz treebank zależnościowy tekstów z epoki średniopolskiej

[8] https://nlpre-pl.clarin-pl.eu/

[9] https://lindat.mff.cuni.cz/repository/items/32ee1b06-e9e7-4942-ac6d-3de64af47c6d

[10] System ról zależnościowych UD przewiduje podstawowy zestaw etykiet, wspólny dla wszystkich banków drzew ujętych w zasobie, oraz możliwość wydzielania podtypów każdej etykiety specyficzny dla danego języka. Przyjęta w ramach projektu UD procedura ewaluacji uwzględnia wyłącznie zestaw podstawowy. Jednocześnie przeniesienie schematu anotacji PDB na schemat UD wymagało zakodowania części informacji właśnie przy użyciu podtypów relacji.

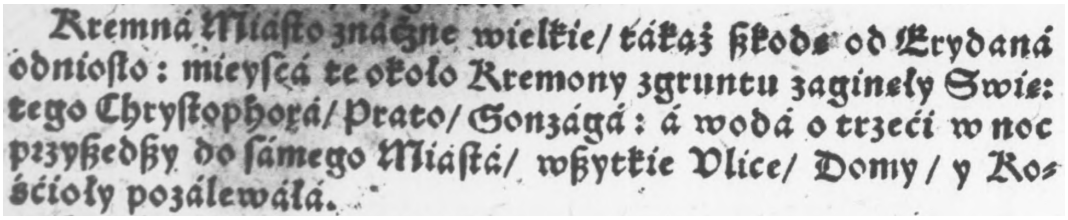
(Wieczorek, 2022). Ten ostatni zasób zawiera zdania wybrane z korpusu KorBa (zob. 4.3.2), które zostały następnie ręcznie zaanotowane składniowo w sposób spójny ze schematem PDB. Automatyczna analiza składniowa tekstów z XVII i XVIII w. stanowi ciekawe wyzwanie. Dostępne dane treningowe są mało liczne, a średnia długość występujących w nich wypowiedzeń jest większa niż w przypadku danych współczesnych. Ilustruje to tabela 4.6: zbiór danych KORBA$_{\text{DEP}}$ jest około 10 razy mniejszy niż PDB, jednocześnie średnia długość zdania jest niemal półtora raza większa. Co więcej, teksty dawne wykazują dużo większe zróżnicowanie wewnętrzne, w tym niejednolitą odmianę i grafię typowe dla okresu, w którym zasady gramatyki oraz pisowni dopiero się kształtowały.[11] Na rysunku 4.3 przedstawione zostało przykładowe zdanie włączone do treebanku średniopolskiego w postaci transliterowanej (oddającej oryginalną pisownię) oraz transkrybowanej (uwspółcześnionej, używanej w opisywanych eksperymentach).

dane	**podzbiór**	**zdania**	**segmenty**	**średnio segm./zdanie**
PDB	treningowy	17722	281687	15,89
	walidacyjny	2215	34677	15,66
	testowy	2215	33616	15,18
	łącznie	22152	349980	15,80
KORBA$_{\text{DEP}}$	treningowy	1612	37762	23,43
	walidacyjny	210	4964	23,64
	testowy	196	4547	23,20
	łącznie	2018	47273	23,43
łącznie		24170	397253	16,44

Tabela 4.6: Rozmiar i podział zbiorów danych zależnościowych dla polszczyzny współczesnej i średniopolskiej.

Opisywany eksperyment miał dwa główne cele. Pierwszym było sprawdzenie, jak skuteczny w przetwarzaniu danych historycznych okaże się model współczesny oraz zbadanie efektu połączenia obu zasobów jako materiału treningowego. Drugi cel

[11] Prace dążące do kodyfikacji ortografii polskiej sięgają XV w., jednak przez długi czas miały one raczej charakter (niekiedy konkurencyjnych) postulatów. Do pierwszej ogólnopolskiej reformy ortografii posiadającej moc urzędową doszło dopiero pod koniec dwudziestolecia międzywojennego, w 1936 r.


Kremná Miásto znáczne wielkie/ tákąż ſzkodę od Erydaná odnioſło: mieyſcá te około Kremony zgruntu zaginęły Swię: tego Chryſtophorá/ Prato/ Gonzágá: á wodá o trzeći w noc przyſzedſzy do ſámego Miáſtá/ wſzytkie Vlice/ Domy/ y Koſćioły pozálewálá.

Transliteracja: *Kremná Miásto znáczne wielkie/ tákąż szkodę od Erydaná odniosło: mieyscá te około Kremony zgruntu zaginęły Swiętego Chrystophorá/ Prato/ Gonzágá: á wodá o trzeći w noc przyszedszy do sámego Miástá/ wszytkie Vlice/ Domy/ y Kościoły pozálewáłá.*

Transkrypcja: *Kremona Miasto znaczne wielkie/ takąż szkodę od Erydana odniosło: miejsca te około Kremony zgruntu zaginęły Świętego Chrystofora/ Prato/ Gonzaga: a woda o trzeci w noc przyszedszy do samego Miasta/ wszytkie Ulice/ Domy/ i Kościoły pozalewała.*

Rysunek 4.3: Przykładowe zdanie z korpusu średniopolskiego.[12]

stanowiło natomiast przetestowanie jakości modelu Hydry wyuczonego na niewielkim zbiorze danych. Wytrenowane zostały modele Hydry wyuczone na podzbiorze treningowym PDB oraz połączonych podzbiorach treningowych PDB i $\text{KORBA}_{\text{DEP}}$. Zostały one następnie przetestowane osobno na częściach testowych obu korpusów. Model PDB został dodatkowo poddany ewaluacji na całości danych średniopolskich, ponieważ nie były one w żaden sposób wykorzystane do jego trenowania. Trzecim i ostatnim scenariuszem było uczenie wyłącznie na danych KORBA. W tym przypadku, ze względu na mały rozmiar korpusu, przeprowadzona została 10-krotna walidacja krzyżowa z podziałem na zbiory treningowy, walidacyjny oraz testowy w proporcjach 8:1:1 w każdym przebiegu. Uzyskane wyniki zostały zebrane w tabeli 4.7.

Rezultaty eksperymentów wyraźnie wskazują, że udział nawet ograniczonej liczby przykładów z korpusu historycznego w trenowaniu jest istotny dla jakości przetwarzania tekstów dawnych. Model uczony wyłącznie na danych $\text{KORBA}_{\text{DEP}}$ radzi sobie z nimi lepiej niż model „współczesny". Różnica w wynikach wynosi odpowiednio około 2 i 3,5 pp. dla miar UAS oraz LAS. Najlepszym rozwiązaniem okazuje się jednak połączenie obu dostępnych zasobów. Warto też zwrócić uwagę na wyniki ewaluacji na zbiorze testowym PDB — są one niemal identyczne dla modeli PDB i PDB +$\text{KORBA}_{\text{DEP}}$ — udział danych średniopolskich w uczeniu modelu nie wpłynął negatywnie na jego skuteczność w analizie składniowej wypowiedzeń pochodzących z tekstów współczesnych.

[12] Źródło skanu: https://cbdu.ijppan.pl/id/eprint/4540/.

dane testowe	dane treningowe	UAS	LAS
PDB	PDB	96,00%	90,78%
	PDB +KORBA$_{DEP}$	**96,16%**	**90,95%**
KORBA$_{DEP}$	PDB	84,33%	74,68%
	KORBA$_{DEP}$	86,52% $\sigma = 0,43\%$	78,16% $\sigma = 0,52\%$
	PDB +KORBA$_{DEP}$	**89,14%**	**81,27%**
KORBA$_{DEP}$ (całość)	PDB	83,60%	73,91%

Tabela 4.7: Skuteczność parsowania zależnościowego danych historycznych z XVII i XVIII w.

4.2.4. Parsowanie składnikowe

Aby porównać jakość rozbiorów składnikowych produkowanych przez Hydrę w ramach analizy hybrydowej z wynikami innego narzędzia, został przeprowadzony eksperyment z użyciem parsera Berkeley Neural Parser (BeNePar, Kitaev *et al.* 2019). Porównanie takie nie było jednak możliwe przy bezpośrednim wykorzystaniu danych ZALEŻNICA oraz TIGER, ponieważ BeNePar dokonuje jedynie rozbiorów wypowiedzeń na ciągłe drzewa składnikowe. Z obu korpusów zostały zatem wybrane (z zachowaniem podziału pierwotnego na części treningową, walidacyjną oraz testową) wyłącznie drzewa niezawierające nieciągłości. Na powstałych zbiorach danych, które oznaczymy odpowiednio ZALEŻNICA-C oraz TIGER-C, wyuczone zostały modele obu porównywanych narzędzi. BeNePar nie wyznacza również centrów składniowych, dlatego nie została w jego przypadku uwzględniona miara *składniki+centra*. Miara *krawędzie* z kolei ma sens tylko w przypadku drzew hybrydowych.

Tabela 4.8 podsumowuje uzyskane wyniki. W pierwszym wierszu dla każdego z języków powtórzone są dla przypomnienia wartości uzyskane w podstawowym eksperymencie z punktu 4.2.2. Następnie podane są wyniki otrzymane w sposób opisany powyżej. Dodatkowo, w wierszach oznaczonych symbolem $\star$, przytaczamy skuteczność BeNePara raportowaną przez jego autorów. Wartości te nie są jednak wprost porównywalne z pozostałymi, ponieważ dotyczą eksperymentów przeprowadzonych na innych danych, pochodzących z kampanii ewaluacyjnej SPMRL 2013[13] (Seddah *et al.*, 2013). W przypadku eksperymentów z użyciem danych ZALEŻNICA-C

[13] Dane te odpowiadają ówczesnej wersji treebanku Składnica. Ze względu na historię wykorzystywanych w tej pracy zasobów polskojęzycznych, można je w przybliżeniu uznać za podzbiór danych ZALEŻNICA-C, zawierający mniej więcej połowę drzew. Jednocześnie są to raczej zdania „łatwiejsze" w analizie, akceptowane przez mniej rozbudowaną wersję gramatyki Świgra.

wyniki odpowiadające obu systemom są bardzo zbliżone, z przewagą BeNePara na poziomie około 0,2 pp. Biorąc dodatkowo pod uwagę fakt, że możliwości Hydry wykraczają poza generowanie rozbiorów ciągłych, uznaję, że mój system sprawdza się bardzo dobrze jako parser stricte składnikowy dla polszczyzny. Dla danych niemieckojęzycznych ograniczonych do drzew ciągłych wyniki Hydry są wyraźniej niższe, niż te dla porównywanego systemu, ale różnice w miarach *nawiasy* oraz *składniki* nie przekraczają 1,5 pp.

język	parser	dane	nawiasy	składniki	składniki +centra	krawędzie
PL	**Hydra**	ZALEŻNICA	97,62%	97,58%	97,05%	95,73%
	Hydra	ZALEŻNICA-C	98,03%	97,93%	97,46%	96,02%
	BeNePar	ZALEŻNICA-C	98,20%	98,10%	—	—
⋆	BeNePar	SPMRL	—	97,15%	—	—
DE	**Hydra**	TIGER	93,77%	92,58%	92,16%	94,80%
	Hydra	TIGER-C	94,43%	93,11%	92,64%	94,65%
	BeNePar	TIGER-C	95,59%	94,41%	—	—
⋆	BeNePar	SPMRL	—	92,10%	—	—

Tabela 4.8: Parsowanie składnikowe: wyniki ewaluacji i porównanie z innymi systemami.

W przypadku języka polskiego warto przytoczyć również badania nad statystycznym ujednoznacznianiem lasów rozbiorów parsera Świgra (Woliński i Rogozińska, 2013, 2016; Woliński, 2019). W eksperymentach na wersji treebanku Składnica liczącej około 11 tysięcy wypowiedzeń autorzy tych badań uzyskali wynik 95,6% dla miary *składniki*. Należy w tym miejscu podkreślić, że wspomniane prace dotyczyły inaczej postawionego problemu — chodziło o wybór drzewa ze zbioru wyznaczonego wcześniej przez parser regułowy (warunkiem koniecznym było zatem zaakceptowanie zdania przez gramatykę parsera). Zadanie postawione przed Hydrą polega natomiast na wygenerowaniu od podstaw dokładnie jednego drzewa dla każdej podanej na wejściu sekwencji segmentów.

Stworzenie i ewaluacja modelu Hydry ograniczonego do komponentu składnikowego jest mniej oczywiste niż wariant wyłącznie zależnościowy z punktu 4.2.3.

Narzędzie zostało bowiem pomyślane do operowania na drzewach z oznaczeniem centrów składniowych: centra składniowe determinują kręgosłupy, na które należy podzielić drzewo. Obecność takiego elementu anotacji w bankach drzew składnikowych (lub istnienie spójnej anotacji zależnościowej) nie jest niestety powszechna. Z kolei wprowadzenie informacji o centrach wymaga znajomości języka oraz schematu anotacji każdego zasobu.

Pomimo wspomnianych ograniczeń, eksperyment dla drzew składnikowych bez centrów uznajemy za wart przeprowadzenia. Aby uniknąć uwikłania w specyfikę poszczególnych treebanków i zachować uniwersalność proponowanego podejścia, centra składniowe można wyznaczyć heurystycznie. Użyta heurystyka powinna być „nieświadoma" konkretnego schematu anotacji składniowej, występującego w nim zestawu kategorii składniowych czy stojącej za nim teorii lingwistycznej oraz od nich niezależna. Oznacza to, że wyznaczone w ten sposób „centra" przynajmniej częściowo nie będą nimi w sensie gramatycznym. Nie stanowi to jednak problemu samo w sobie: ich wyróżnienie jest w tym scenariuszu krokiem pomocniczym, czysto technicznym, pozwalającym na wydzielenie z drzew jednostek, na jakich operuje parser Hydra. Istotna jest jedynie poprawność powstających w ten sposób drzew wyrażona miarami *nawiasy* oraz *składniki* (zob. 4.2.1).

Rozważane były cztery heurystyczne metody wyznaczania centrów:

pierwsze — wybór pierwszego dziecka każdego nieterminala jako centralnego,

środkowe — wybór środkowego dziecka,

najpłytsze — wybór dziecka, przez które prowadzi najkrótsza ścieżka do dowolnego liścia,

najczęstsze — dla nieterminala o kategorii X wybór dziecka, którego kategoria składniowa najczęściej występuje wśród dzieci węzłów typu X w całym zbiorze danych.

W przypadku trzech ostatnich heurystyk, jeśli istnieje więcej niż jedno dziecko spełniające warunek (lub liczba dzieci jest parzysta w przypadku heurystyki *środkowy*), wybierane jest to spośród pasujących, które występuje pierwsze w porządku drzewa.

Przed przystąpieniem do właściwych eksperymentów z parsowaniem zostały zbadane efekty zastąpienia oznaczeń centrów składniowych obecnych w danych ZALEŻNICA oraz TIGER wynikami zastosowania wymienionych heurystyk. W każdym przypadku obliczona została liczba różnych powstałych w ten sposób kręgosłupów składniowych. Wybór centrów składniowych determinuje relacje nadrzędnik/podrzędnik występujące pomiędzy terminalami, można zatem sprawdzić, jak zmienia się średnia odległość zależnościowa[14] między segmentami. Ostatnią braną pod

[14] Średnia odległość zależnościowa drzewa to średni dystans (liczony w segmentach uporządkowanych w kolejności zdaniowej) między końcami jego krawędzi zależnościowych.

uwagę statystyką jest dokładność (*accuracy*), z jaką każdej heurystyce udaje się trafić w faktyczne centrum składniowe frazy, tj. odsetek występujących w danych składników, którym heurystyka przypisuje centrum zgodnie z anotacją treebanku. Uzyskane wartości zostały zebrane w tabeli 4.9. Dla pełnego porównania w jej ostatnim wierszu dla każdego zbioru danych umieszczono statystyki dla scenariusza *składniowe*, w którym centra pozostawione są bez zmian (jest to oczywiście nieosiągalne bez istnienia odpowiedniej anotacji).

dane	**heurystyka**	**kręgosłupy**	**śr. odległość zależnościowa**	**dokładność**
ZALEŻNICA	pierwsze	1161	3,47	57,60%
	środkowe	805	2,48	73,31%
	najpłytsze	339	2,60	61,72%
	najczęstsze	245	2,97	58,13%
	składniowe	109	2,65	100,00%
TIGER	pierwsze	1227	4,45	33,50%
	środkowe	692	2,95	53,40%
	najpłytsze	159	3,14	51,11%
	najczęstsze	430	3,55	57,88%
	składniowe	281	3,49	100,00%

Tabela 4.9: Porównanie podziałów drzew na kręgosłupy uzyskanych dla poszczególnych heurystyk wyznaczania centrów.

Intuicyjnie największy potencjał wydają się mieć heurystyki *najpłytsze* oraz *najczęstsze*. Pierwsza z nich dąży do wyznaczania jak najkrótszych kręgosłupów, co powinno się przekładać na stosunkowo niewielką liczbę ich typów. Rzeczywiście: dla polskich danych jest ich ponad 3 razy mniej niż dla najmniej „oszczędnej" pod tym względem heurystyki *pierwsze*, dla danych niemieckich najmniej spośród analizowanych scenariuszy. Heurystyka *najczęstsze* stanowi próbę znalezienia w danych powtarzających się schematów, dając szansę na pewne usystematyzowanie sposobu oznaczania pomocniczych centrów. Jej wadą jest konieczność wykonania dodatkowych, wstępnych obliczeń na całych przetwarzanych danych, aby wyznaczyć najczęściej występujące typy dzieci dla każdej kategorii składniowej. Scenariusze *pierwsze* i *środkowe* są z kolei bardzo proste i działają lokalnie: każde rozstrzygnięcie odbywa się w kontekście pojedynczego rozgałęzienia w drzewie. Heurystyka

pierwsze wydaje się przy tym najsłabsza: produkuje najwięcej różnych kręgosłupów i w oczywisty sposób skutkuje największą średnią odległością zależnościową, czyli potencjalnie trudniejszymi dla parsera, dalej sięgającymi krawędziami.

Jako pierwsze do przetestowania zostały wybrane heurystyki *pierwsze* — jako najbardziej naiwna, a więc być może skutkująca najgorszymi wynikami — oraz *najpłytsze* — bardziej szczegółowa, ale niewymagająca dodatkowego przygotowania. Ewaluację otrzymanych w ten sposób modeli podsumowuje tabela 4.10. Obliczone zostały tylko miary *nawiasy* oraz *składniki*, ponieważ w rozważanym eksperymencie zakładamy, że faktyczne centra składniowe nie są znane i nie interesuje nas ich wyznaczenie. Dla porównania w ostatnim wierszu tabeli przypomniane zostały wartości uzyskane w podstawowym eksperymencie z punktu 4.2.2. Okazuje się, że zastosowanie heurystyki i jej wybór ma niewielki wpływ na skuteczność parsowania ograniczonego do wyznaczania składników. Co ciekawe, prostsza heurystyka *pierwsze* daje nawet lepszy rezultat. Wobec niewielkich zaobserwowanych różnic dwie pozostałe heurystyki nie były już testowane.

Przeprowadzony eksperyment wskazuje, że parser hybrydowy Hydra można łatwo i skutecznie zaadaptować do roli analizatora składnikowego dla drzew, których schemat anotacji nie obejmuje centrów składniowych. Wymaga to prostej operacji na danych z treebanku, a jakość uzyskanych rozbiorów pozostaje niewiele niższa niż w scenariuszu z centrami. Porównanie to dotyczy oczywiście tylko tych aspektów informacji składniowej, które są dostępne w rozważanych danych — znakowanie zależnościowe i struktura kręgosłupów pozostają tu poza zasięgiem modelu.

heurystyka	**nawiasy**	**składniki**
pierwsze	97,45%	97,43%
najpłytsze	97,13%	96,97%
składniowe	97,62%	97,58%

Tabela 4.10: Skuteczność parsowania składnikowego dla języka polskiego w zależności od użytej heurystyki wyznaczania centrów.

4.3. Znakowanie fleksyjne

4.3.1. Teksty współczesne: NKJP 1M i PolEval

Standardem *de facto* w ewaluacji narzędzi do automatycznego znakowania fleksyjnego polszczyzny jest użycie ręcznie znakowanej części Narodowego Korpusu Języka Polskiego (Degórski i Przepiórkowski, 2012). (Pod)korpus ten, który będziemy skrótowo nazywać NKJP 1M (a dane dla modelu zeń pochodzące — NKJP),[15] został wykorzystany w podstawowym eksperymencie mającym na celu ocenę jakości tagowania, segmentacji i lematyzacji wykonywanej przez narzędzie Hydra. Podział korpusu na części treningową, walidacyjną oraz testową jest zgodny z wykorzystywanym w platformie NLPre-PL.

Eksperyment na danych NKJP został poszerzony przez zaczerpnięcie dodatkowych danych testowych z innego źródła niż korpus treningowy oraz walidacyjny. Aby wyniki takiej ewaluacji były miarodajne, schemat anotacji (w szczególności repertuar znaczników fleksyjnych) zewnętrznego korpusu testowego powinien być spójny z danymi, na których model został wyuczony. Przydatnym zasobem okazał się w tym kontekście zbiór testowy przygotowany na potrzeby konkursu PolEval 2017 (Kobyliński i Ogrodniczuk, 2017). Na korpus ten składają się teksty pochodzące ze źródła innego niż NKJP (zarówno część 1M, jak i pełny korpus), a jego anotacja odbyła się kilka lat później. Jednocześnie jego znakowanie jest spójne z NKJP 1M. Rozmiary zbiorów danych anotowanych fleksyjnie wykorzystywanych w eksperymentach opisanych w tym oraz kolejnych punktach, z uwzględnieniem liczby zdań, segmentów oraz średnich długości zdań, umieszczone zostały w tabeli 4.11.

Model Hydry wykonujący segmentację, lematyzację oraz tagowanie został wytrenowany na części uczącej danych NKJP oraz przetestowany na dwóch zbiorach danych: części testowej NKJP oraz danych POLEVAL. Ewaluacja została przeprowadzona w dwóch wariantach: z segmentacją wzorcową oraz automatyczną. W pierwszym wariancie model otrzymywał na wejściu teksty podzielone na segmenty zgodnie z anotacją korpusu i jedynie przypisywał im formy hasłowe oraz znaczniki. W wariancie drugim ten sam model dodatkowo samodzielnie wyznaczał segmentację każdego zdania podanego jako ciągły napis. W przypadku segmentacji automatycznej zamiast miary *accuracy* zastosowana została miara F_1.[16] Otrzymane wyniki

[15] Określenie 1M bierze się z rozmiaru podkorpusu. Nieoznakowane (w szczególności niepoddane segmentacji) teksty wylosowane do NKJP 1M liczyły łącznie około miliona tradycyjnie rozumianych słów tekstowych. W wyniku segmentacji niektóre słowa zostały podzielone i ostatecznie zaanotowany korpus zawiera około 1,2 miliona segmentów.

[16] W przypadku segmentacji sens tej miary jest jasny: istotne jest, aby model wykrył jak najwięcej faktycznie występujących jednostek (tu: segmentów) oraz aby wskazał jak najmniej jednostek

dane	podzbiór	zdania	segmenty	średnio segm./zdanie
NKJP	treningowy	68 943	978 339	14,19
	walidacyjny	7 755	112 454	14,50
	testowy	8 964	125 051	13,95
	łącznie	85 662	1 215 844	14,19
POLEVAL	testowy	3 307	55 033	16,64
XIX	treningowy	25 839	524 127	20,28
	walidacyjny	2 827	61 360	21,70
	testowy	3 214	64 972	20,22
	łącznie	31 880	650 459	20,40
KORBA	treningowy	24 234	662 339	27,33
	walidacyjny	2 898	81 970	28,29
	testowy	3 007	82 260	27,36
	łącznie	30 139	826 569	27,43

Tabela 4.11: Rozmiar i podział zbiorów danych polskojęzycznych anotowanych fleksyjnie.

przedstawione zostały w tabeli 4.12. Miary dla tagowania oraz lematyzacji zostały dodatkowo obliczone przy uwzględnieniu heurystycznej korekty opisanej w punkcie 3.4.2.

Jakość znakowania fleksyjnego wykonanego przez Hydrę jest wysoka. W trudniejszym dla modelu scenariuszu automatycznej segmentacji, bez korekty, wartości F_1 dla tagowania i hasłowania przekraczają odpowiednio 97% oraz 98% dla obu zestawów testowych. Poprawność samej segmentacji również jest satysfakcjonująca. Jeśli porównamy wyniki na danych NKJP oraz POLEVAL, liczby otrzymane dla tego drugiego korpusu są zbliżone przy segmentacji wzorcowej i systematycznie niższe przy automatycznej, przy czym w tym drugim przypadku różnice wynoszą około

nieprawidłowych. Dla lematyzacji oraz tagowania miara *accuracy* przestaje być adekwatna, ponieważ w przypadku błędnego podziału na segmenty przez model mogą różnić się sekwencje jednostek, których etykiety porównujemy. Poprawnie oznakowany segment to taki, który został prawidłowo wyznaczony oraz opatrzony właściwą etykietą — w tej sytuacji większy sens ma obliczenie miary F_1 na podstawie *precision* oraz *recall*.

dane testowe	seg. automat.	segmentacja	tagowanie	(korekta)	lematyzacja	(korekta)
NKJP		—	97,75%	97,76%	98,62%	99,09%
	✓	99,82%	97,60%	97,62%	98,48%	98,94%
POLEVAL		—	97,73%	97,73%	98,66%	99,07%
	✓	99,54%	97,29%	97,30%	98,22%	98,62%

Tabela 4.12: Wyniki ewaluacji modelu do znakowania fleksyjnego.

0,3 pp. Model dobrze poradził sobie zatem z tekstami z zewnętrznego źródła. Na szczególną uwagę zasługuje bardzo dobra jakość lematyzacji, dodatkowo rosnąca po zastosowaniu korekty heurystycznej. Przy zadanej z góry poprawnej segmentacji właściwe lematy przypisywane są ponad 99% segmentów. W bardziej realistycznym scenariuszu, kiedy model sam wyznacza podział na segmenty, miara F_1 pozostaje wysoka: odpowiednio 98,94% i 98,62% dla danych NKJP i POLEVAL.

W tabeli 4.13 przytoczono dla porównania wyniki osiągane przez inne narzędzia w tym samym zadaniu, raportowane w leaderboardzie NLPre-PL (dla danych testowych pochodzących z NKJP 1M) oraz podsumowaniu konkursu PolEval 2017.[17] Hydra wypada na tym tle zdecydowanie pozytywnie, osiągając wartości miary F_1 przy automatycznej segmentacji wyższe o około 2 pp dla tagowania i 1 pp. dla lematyzacji niż najlepszy system notowany w tabeli (jeśli uwzględnimy korektę wyników za pomocą analizatora fleksyjnego, przewaga w lematyzacji dodatkowo wzrasta o ca. 0,5 pp). Przy segmentacji wzorcowej odpowiednie różnice, również na korzyść Hydry, wynoszą odpowiednio ≈1,2 oraz ≈0,3 pp.

4.3.2. Korpusy tekstów historycznych

Znakowany fleksyjnie korpus tekstów polskich z lat 1830-1918[19] (Kieraś i Woliński, 2018) powstał w wyniku ręcznej anotacji danych zgromadzonych we wcześniej-

[17] Anotacja zbioru danych ewaluacyjnych PolEval została od czasu przeprowadzenia konkursu odświeżona do postaci spójnej ze zmianami w systemie znaczników NKJP (zob. Kieraś *et al.* 2021). W korpusach wykorzystywanych do naszych eksperymentów trzymamy się aktualnej wersji tagsetu NKJP. Biorąc pod uwagę niewielką skalę różnic między systemami znakowania uznajemy jednak, że porównanie z wynikami konkursu ma sens.

[18] https://spacy.io/

[19] https://chronofleks.nlp.ipipan.waw.pl/korpus19/

	system	seg. aut.	segmentacja	tagowanie	lematyzacja
NLPre-PL	COMBO		—	96,54%	98,35%
	(Klimaszewski i Wróblewska, 2021)	✓	99,16%	95,65%	97,44%
	Stanza		—	94,10%	97,43%
	(Qi *et al.*, 2020)	✓	99,77%	93,59%	96,91%
	spaCy[18]		—	96,03%	97,44%
		✓	99,56%	94,55%	95,94%
	Concraft		—	90,22%	97,15%
	(Waszczuk, 2012)	✓	98,55%	89,88%	96,79%
	UDPpipe		—	90,87%	96,13%
	(Straka *et al.*, 2019)	✓	99,73%	90,60%	95,84%
	Trankit		—	91,69%	93,15%
	(Nguyen *et al.*, 2021)	✓	98,24%	89,59%	91,02%
PolEval	Toygger		—	94,63%	—
	(Krasnowska-Kieraś, 2017)	✓	—	—	—
	KRNNT		—	93,81%	97,83%
	(Wróbel, 2017)	✓	—	92,98%	96,91%
	NeuroParser		—	93,61%	97,11%
	(Rychlikowski *et al.*, 2017)	✓	—	91,59%	97,00%
	Concraft		—	91,61%	95,65%
		✓	—	90,08%	94,72%
	WCRFT		—	91,17%	94,29%
	(Radziszewski, 2013)	✓	—	89,69%	93,79%
	WMBT		—	90,67%	94,07%
	(Radziszewski i Śniatowski, 2011)	✓	—	89,06%	93,62%

Tabela 4.13: Jakość znakowania fleksyjnego polszczyzny osiągana przez inne systemy.

szym projekcie (Bilińska *et al.*, 2016). Zasób ten, nazywany dalej korpusem XIX,[20] liczy około 600 tysięcy segmentów. Drugim wykorzystanym w eksperymentach ręcznie znakowanym zasobem dla polszczyzny historycznej był Elektroniczny Korpus Tekstów Polskich XVII i XVIII w.[21] (Gruszczyński *et al.*, 2022), zwyczajowo nazywany KorBa.[22]

Systemy znaczników korpusów NKJP 1M, XIX oraz KorBa są podobne, ale nie identyczne. Tagsety korpusów historycznych wywodzą się z używanego w NKJP tagsetu analizatora Morfeusz 2 i zostały zmodyfikowane tak, aby odzwierciedlać pewne zjawiska obecne w tekstach z epoki. Dzięki temu podobieństwu wszystkie trzy zestawy znaczników można było dość łatwo sprowadzić do jednej, spójnej postaci na potrzeby trenowania i ewaluacji modeli. Przekształcenie to wymagało pewnego uproszczenia poszczególnych tagsetów, a w efekcie zmniejszenie liczebności ich elementów. Należy więc zaznaczyć, że dane NKJP wykorzystane w tym punkcie są z tego powodu różne od tych z punktów poprzednich. Szczegółowy wykaz zmian wprowadzonych w znacznikach poszczególnych korpusów znaleźć można w tabeli A.5.

Eksperyment z wykorzystaniem znakowanych morfoskładniowo tekstów dawnych miał podobny cel i metodologię jak w przypadku testów na średniopolskim treebanku zależnościowym i PDB opisanych w punkcie 4.2.3. Wytrenowane zostały trzy modele Hydry przewidujące segmentację, lematy oraz znaczniki fleksyjne. Dla każdego modelu połączono części uczące i walidacyjne odpowiednich zbiorów danych tak, aby obejmowały one teksty z coraz dawniej sięgających czasów. Uwzględnione kombinacje składały się wyłącznie z danych współczesnych (NKJP), ich połączenia z danymi XIX-wiecznymi (NKJP +XIX) oraz wszystkich dostępnych zbiorów (NKJP +XIX +KORBA). Każdy z modeli został przetestowany osobno na częściach testowych poszczególnych korpusów. Wyniki zebrane zostały w tabeli 4.14.

Ogólna tendencja wyłaniająca się z wyników eksperymentu nie jest zaskakująca: rozszerzanie zbioru treningowego o starsze dane skutkuje lepszą jakością analizy dla odpowiednio starszych danych testowych. Co interesujące, w przypadku danych testowych NKJP rozbieżności między wartościami poszczególnych miar dla wszystkich trzech modeli są w zasadzie zaniedbywalne. Materiał uczący z wcześniejszych epok, łącznie zawierający więcej segmentów niż dane współczesne, zdaje się nie mieć wpływu na jakość modelu mierzoną dla danych NKJP.

Dla tekstów XIX-wiecznych oba modele „historyczne" znów radzą sobie bardzo

[20] Określenie to jest umowne: korpus nie obejmuje tekstów sprzed powstania listopadowego, jednocześnie zostały do niego włączone teksty z początku XX w.

[21] https://korba.edu.pl

[22] Od „Korpus Barokowy" — określenia pierwszej wersji zasobu, obecnie poszerzonej o teksty z epoki oświecenia.

dane testowe	dane treningowe	seg. automat.	segmentacja	tagowanie	lematyzacja
NKJP	NKJP		—	98,15%	98,62%
		✓	99,83%	97,99%	98,48%
	NKJP +XIX		—	98,16%	98,69%
		✓	99,83%	**98,02%**	**98,55%**
	NKJP +XIX +KORBA		—	98,10%	98,67%
		✓	99,82%	97,95%	98,52%
XIX	NKJP		—	95,14%	97,39%
		✓	98,83%	94,05%	96,34%
	NKJP +XIX		—	96,76%	98,55%
		✓	98,89%	95,73%	97,48%
	NKJP +XIX +KORBA		—	96,82%	98,63%
		✓	98,88%	**95,74%**	**97,54%**
KORBA	NKJP		—	90,84%	94,69%
		✓	97,98%	89,30%	93,17%
	NKJP +XIX		—	92,70%	96,04%
		✓	98,01%	91,20%	94,52%
	NKJP +XIX +KORBA		—	95,36%	97,91%
		✓	98,02%	**93,53%**	**96,02%**

Tabela 4.14: Wyniki ewaluacji modeli do znakowania fleksyjnego danych historycznych.

podobnie, ale tym razem model „współczesny” wypada już zauważalnie gorzej. Jego jakość jest przy tym nadal dość wysoka. Najbardziej wyraziste różnice można zaobserwować na danych z XVII i XVIII wieku. W tym przypadku każde rozszerzenie danych uczących wnosi wyraźną poprawę.

Z uzyskanych wyników wyciągnąć można wniosek, że o ile akceptowalne (lub wręcz pożądane) jest uproszczenie zestawów znaczników konieczne do ich uspójnienia, do przetwarzania danych polskojęzycznych pochodzących z różnych epok dobrym wyborem może być pojedynczy model trenowany na połączonym materiale

korpusowym. Taki właśnie model uzyskał w opisywanym eksperymencie najlepsze lub zbliżone do najlepszych rezultaty dla każdego ze zbiorów testowych.

4.4. Łączenie i ważenie danych

Architektura systemu Hydra pozwala na wytrenowanie jednego modelu wykonującego równolegle różne rodzaje anotacji – na przykład fleksyjną i składniową. Dane uczące znakowane na wszystkich potrzebnych poziomach mogą jednak nie istnieć lub mieć niewielki rozmiar, podczas gdy dostępne są zasoby anotowane na poszczególnych poziomach (jednym lub więcej). O ile znakowanie danego typu jest spójne pomiędzy korpusami, interesująca i przydatna wydaje się możliwość maksymalnego wykorzystania zawartej w nich informacji poprzez ich połączenie. Możliwą korzyścią z takiego podejścia, weryfikowaną eksperymentalnie w tym miejscu rozprawy, jest lepsze wykorzystanie dostępnych danych pomimo ich niepełności.

Z technicznego punktu widzenia przygotowanie danych z niepełną anotacją wymaga zastąpienia brakujących etykiet specjalną wartością (maską, `MASK`). Zamaskowane etykiety są ignorowane w procesie uczenia modelu – nie jest dla nich obliczana funkcja straty, zatem nie mają wpływu na zmianę parametrów modelu podczas propagacji wstecznej. Na przykład włączenie korpusu anotowanego wyłącznie składniowo do materiału uczącego dla modelu, który ma dodatkowo przypisywać znaczniki fleksyjne, wymaga przypisania każdemu segmentowi w miejsce tagu wartości `MASK`.

W tym punkcie przedstawione zostaną wyniki eksperymentu polegającego na wykorzystaniu połączonych zbiorów zależnica (znakowanego fleksyjnie i składniowo) oraz nkjp (znakowanego wyłącznie fleksyjnie). Zasoby te współdzielą pewną liczbę wypowiedzeń, co wynika z historii treebanków stanowiących punkt wyjścia dla stworzenia danych hybrydowych (zob. punkty 1.2.1–1.2.3). W przypadku elementów występujących tylko w jednym z korpusów procedura łączenia była oczywista — znakowanie oraz przydział danego przykładu do podzbioru treningowego, walidacyjnego lub testowego pozostawały bez zmian. Niektóre elementy należące do części wspólnej wymagały natomiast dwojakiego rodzaju rozstrzygnięć. W nielicznych sytuacjach, w których segmentacja, lematyzacja lub znaczniki fleksyjne różniły się pomiędzy anotacjami w obu korpusach,[23] przyjmowano wersję z treebanku. Drugim nietrywialnym przypadkiem były zdania, które w zbiorach zależnica oraz nkjp znalazły się w różnych częściach (na przykład odpowiednio

[23] Dla wypowiedzeń, które trafiły do Składnicy lub PDB z korpusu NKJP 1M, twórcy banków drzew dopuszczali korektę anotacji fleksyjnej w przypadku wykrycia błędu, zmiany te jednak nie były przenoszone do korpusu źródłowego.

walidacyjnej i testowej) — wówczas również decydujący był podział z korpusu składniowego.

Wielkości poszczególnych części tak powstałego zbioru danych, z wyszczególnieniem liczby przykładów, które trafiły do nich z odpowiednich zbiorów wyjściowych, zostały zestawione w tabeli A.6. Wszystkie przykłady w połączonych danych opatrzone są informacją wykorzystywaną przez moduły lematyzacji oraz tagowania (obecną zarówno w Zależnicy, jak w NKJP 1M), natomiast tylko te pochodzące z treebanku posiadają dodatkowo informację składniową. Wypowiedzenia opatrzone kompletem anotacji składają się przy tym na około jedną czwartą danych.

Na opisanych powyżej danych został przeprowadzony szereg eksperymentów. W każdym z nich trenowano model parsera wykorzystujący HerBERT large, zmieniając przy tym wagę, jaką przy obliczaniu funkcji straty dla całego modelu przypisywano modułowi składniowemu. W podstawowym scenariuszu trenowania udział każdego klasyfikatora w sumie składającej się na funkcję straty jest taki sam. Przypisując klasyfikatorom wagi, można ten udział zmieniać, skłaniając tym samym model do silniejszego dopasowania do wyjść z „cięższych" klasyfikatorów, nawet kosztem „lżejszych". W opisywanym eksperymencie składowe funkcji straty pochodzące z klasyfikatorów odpowiedzialnych za przewidywanie struktury składniowej mnożone były przez dodatkowy czynnik. Ewaluacji poddane zostały modele trenowane z czynnikiem (wagą komponentu składniowego) 1, 2, 4, 8 oraz 16. Dla części danych testowych pochodzącej z Zależnicy możliwe było sprawdzenie jakości wszystkich poziomów znakowania wykonanego przez parser. Dla części posiadającej wyłącznie anotację fleksyjną z NKJP 1M ewaluacja obejmowała tylko lematyzację i tagowanie.

Celem eksperymentów było zbadanie, czy i w jakim stopniu wzmacnianie sygnału uczącego związanego ze strukturą składniową poprawi jakość modułu parsującego wytrenowanego modelu. Jednocześnie istotne było, aby nie stracić skuteczności znakowania fleksyjnego, której spadek był spodziewany wraz ze wzrostem wagi komponentu składniowego. Wyniki empiryczne zostały zawarte w tabeli 4.15. Rysunek 4.4 przedstawia dodatkową wizualizację otrzymanych wartości dla wybranych miar w zależności od przyjętej wagi. Kształt wykresów był bardzo podobny dla wszystkich miar dotyczących parsowania, dlatego dla przejrzystości rysunku uwzględnione zostały tylko dwie z nich: *UAS* oraz *składniki+centra*.

Spośród przetestowanych wariantów najlepszym kompromisem pomiędzy jakością analizy fleksyjnej i składniowej okazuje się zastosowanie wag 4 lub 6. Dalsze zwiększanie wagi ma niewielki wpływ na poprawność parsowania, powoduje natomiast spadek wartości miar dla lematyzacji oraz tagowania. Warto tutaj zwrócić uwagę na pewną korelację z odsetkiem obecnych w danych segmentów opatrzonych pełną anotacją, który wynosi około 25%. Czterokrotne „wzmocnienie" informa-

waga	dane testowe	tagowanie	lematyzacja	UAS	LAS	nawiasy	składniki	składniki+centra	krawędzie
1	ZALEŻNICA	96,66%	98,44%	95,89%	90,73%	97,55%	97,59%	97,06%	95,79%
	NKJP	97,27%	98,58%	—	—	—	—	—	—
2	ZALEŻNICA	96,60%	98,56%	96,08%	90,80%	97,60%	97,68%	97,16%	95,86%
	NKJP	97,33%	98,70%	—	—	—	—	—	—
4	ZALEŻNICA	96,51%	98,31%	96,19%	91,03%	97,67%	97,69%	97,18%	95,89%
	NKJP	97,16%	98,51%	—	—	—	—	—	—
6	ZALEŻNICA	96,59%	98,38%	96,21%	91,08%	97,74%	97,80%	97,28%	95,99%
	NKJP	97,12%	98,53%	—	—	—	—	—	—
8	ZALEŻNICA	96,16%	97,46%	96,17%	90,92%	97,62%	97,64%	97,12%	95,88%
	NKJP	96,70%	97,71%	—	—	—	—	—	—
16	ZALEŻNICA	96,07%	97,40%	96,20%	91,15%	97,68%	97,72%	97,21%	95,98%
	NKJP	96,53%	97,57%	—	—	—	—	—	—

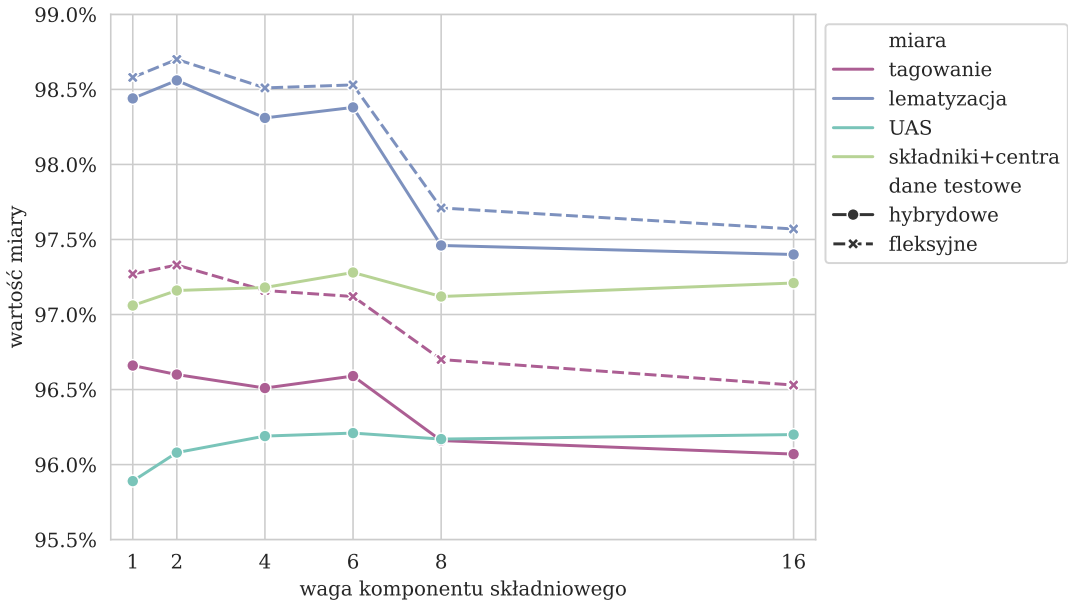
Tabela 4.15: Ewaluacja modeli składniowo-fleksyjnych trenowanych z różnymi wagami komponentu składniowego.

cji składniowej przekazywanej modelowi podczas uczenia można zatem uznać za swego rodzaju wyrównanie tej proporcji do 100%.

4.5. Rozpoznawanie jednostek nazewniczych

Model Hydry służący do rozpoznawania nazw własnych został wytrenowany na 90% danych NKJP 1M (zob. 3.4.3) — 10% zostało użyte do walidacji. Ewaluacja przeprowadzona została na korpusie testowym kampanii PolEval 2018 (Wawer i Małek, 2018). W tabeli 4.16 znaleźć można uzyskane wyniki. Dla porównania przetestowane zostało inne narzędzie do rozpoznawania nazw własnych o strukturze hierarchicznej, dla którego dostępny jest model trenowany na NKJP 1M: PolDeepNer2.[24] W tabeli

[24] https://github.com/CLARIN-PL/PolDeepNer2.



Rysunek 4.4: Skuteczność modeli składniowo-fleksyjnych w zależności od wagi komponentu składniowego.

	system	**precyzja**	**czułość**	**F_1**
	Hydra	88,61%	87,54%	88,07%
	PolDeepNer2	89,77%	90,43%	90,10%
⋆	PolDeepNer2			89,90%
⋆	Dadas i Protasiewicz (2020)			87,00%
◇	Borchmann *et al.* (2018)			82,60%
◇	PolDeepNer (Marcińczuk *et al.*, 2018)			82,20%
◇	Liner2 (Marcińczuk *et al.*, 2017)			77,80%

Tabela 4.16: NER: wyniki ewaluacji i porównanie z innymi systemami.

przytoczono także wyniki uzyskiwane przez kilka innych systemów, raportowane przez ich autorów (oznaczone symbolem ⋆) oraz zamieszczone na liście wyników zadania NER w ramach konkursu PolEval 2018 (symbol ◇). Szczegółowe wyniki ewaluacji modelu Hydry, z podziałem na typy jednostek nazewniczych, zawiera tabela B.3.

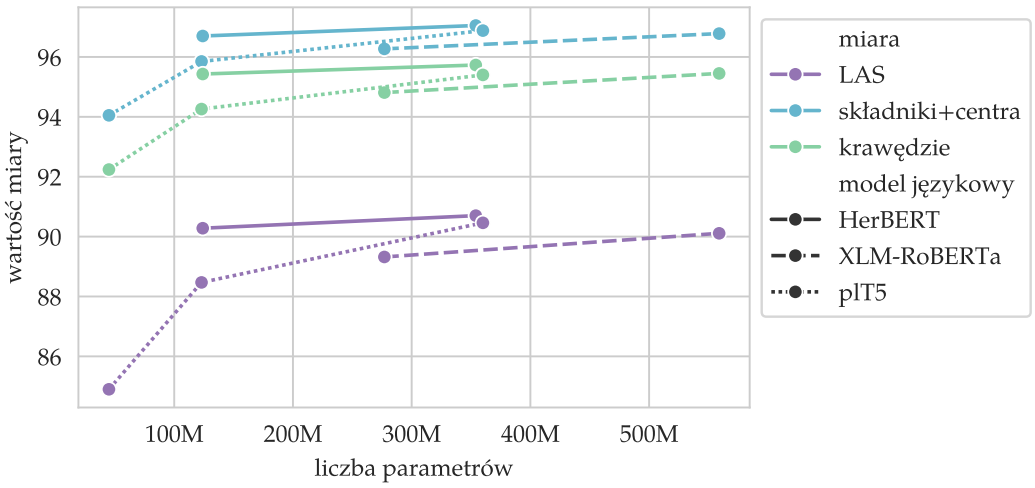
Jakość oznaczania jednostek nazewniczych za pomocą Hydry, wyrażona miarą F_1, jest o około 2 pp. niższa niż w przypadku najlepszego z przytoczonych narzędzi — PolDeepNer2, aczkolwiek plasuje się wyżej niż te dla pozostałych wymienionych systemów. Wynik ten zdecydowanie pozostawia obszar do poprawy. Biorąc

jednak pod uwagę, że NER nie jest podstawowym zastosowaniem przewidzianym dla Hydry, uznajemy uzyskany rezultat za zadowalający. W połączeniu z efektami eksperymentów opisanych w punkcie 4.3 wskazuje on również, że zaprojektowana dla prezentowanego parsera architektura może być z powodzeniem stosowana do innych zadań z zakresu anotacji tekstu.

4.6. Modele językowe

Obok modeli HerBERT w rozmiarach BASE[25] oraz LARGE, jako komponent Hydry dostarczający wektorowych reprezentacji segmentów wypróbowane zostały również wielojęzyczne XLM-RoBERTa (Conneau *et al.*, 2020) BASE[26] i LARGE[27] oraz wytrenowany na polskich danych plT5 (Chrabrowa *et al.*, 2022) — model o architekturze T5 (Raffel *et al.*, 2020). W przypadku tej ostatniej grupy modeli, użyte zostały wyłącznie moduły kodujące (*encoder*) z trzech wariantów: SMALL,[28] BASE[29] oraz LARGE.[30] Tabela 4.17 zawiera zestawienie wyników uzyskanych dla danych polskojęzycznych, natomiast wykres na rysunku 4.5 przedstawia wartości wybranych miar ewaluacyjnych w zależności od typu modelu językowego oraz jego rozmiaru (liczby parametrów).

[25] https://huggingface.co/allegro/herbert-base-cased
[26] https://huggingface.co/FacebookAI/xlm-roberta-base
[27] https://huggingface.co/FacebookAI/xlm-roberta-large
[28] https://huggingface.co/allegro/plt5-small
[29] https://huggingface.co/allegro/plt5-base
[30] https://huggingface.co/allegro/plt5-large



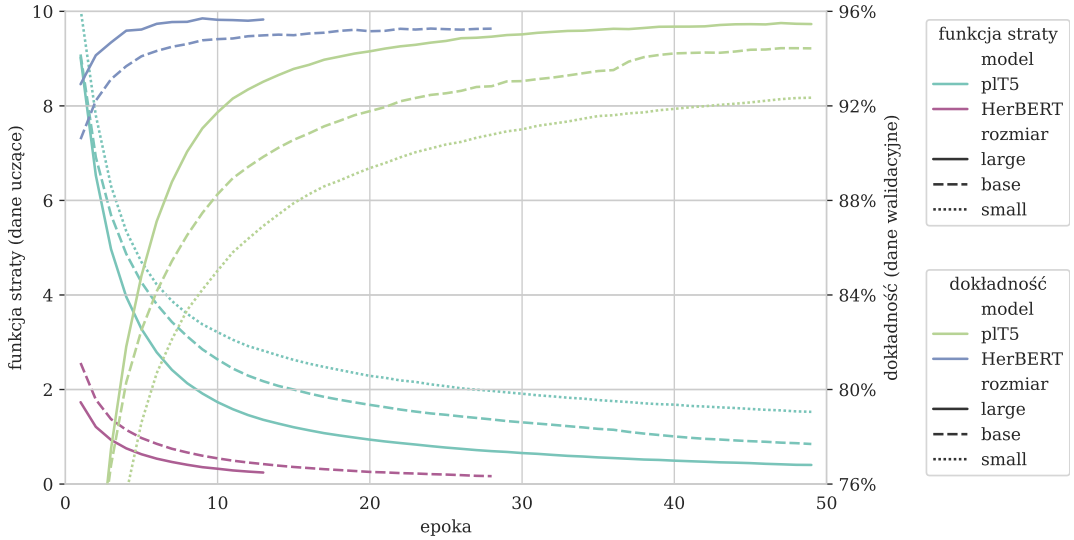
Rysunek 4.5: Zależność skuteczności parsowania dla języka polskiego od liczby parametrów użytego modelu językowego.

model		UAS	LAS	nawiasy	składniki	składniki +centra	krawędzie
HerBERT	BASE	95,60%	90,28%	97,27%	97,26%	96,70%	95,43%
	124M		nieciągłe:	45,61%	44,48%	44,22%	
HerBERT	LARGE	95,99%	90,70%	97,62%	97,58%	97,05%	95,73%
	354M		nieciągłe:	48,92%	48,00%	47,85%	
XLM-RoBERTa	BASE	95,06%	89,32%	96,96%	96,89%	96,27%	94,81%
	277M		nieciągłe:	39,94%	39,55%	39,44%	
XLM-RoBERTa	LARGE	95,77%	90,11%	97,34%	97,32%	96,78%	95,45%
	559M		nieciągłe:	46,73%	45,60%	45,24%	
plT5	SMALL	93,26%	84,90%	95,32%	94,91%	94,05%	92,24%
	45M		nieciągłe:	19,51%	18,83%	18,75%	
plT5	BASE	95,04%	88,47%	96,67%	96,48%	95,85%	94,26%
	123M		nieciągłe:	35,18%	34,27%	34,15%	
plT5	LARGE	95,95%	90,46%	97,53%	97,42%	96,88%	95,40%
	360M		nieciągłe:	46,93%	45,73%	45,66%	

Tabela 4.17: Wyniki ewaluacji parsowania hybrydowego dla języka polskiego z użyciem różnych modeli językowych.

Układy widocznych na wykresie punktów są bardzo podobne dla poszczególnych miar. Spośród testowanych modeli najskuteczniejszy okazał się HerBERT. Zwłaszcza przy mniejszej liczbie parametrów dystans pomiędzy nim a plT5 jest wyraźniejszy. Relatywnie najmniej użytecznym modelem okazała się XLM-RoBERTa, co nie jest zaskakujące, biorąc pod uwagę fakt, że pozostałe modele były przygotowane specyficznie dla polskiego. Utrata skuteczności przy zmianie modelu na wielojęzyczny nie jest jednak drastyczna, co wskazuje, że dostępność modelu językowego przeznaczonego *stricte* dla danego języka nie jest warunkiem koniecznym dla wytrenowania dobrej jakości modelu parsera Hydra.

Rysunek 4.6 ilustruje dodatkowe spostrzeżenie. Przedstawione zostały na nim przebiegi uczenia uzyskane przy wykorzystaniu modeli HerBERT oraz plT5 w różnych rozmiarach: na wykresie pokazane są zmiany wartości funkcji straty (na danych treningowych) oraz łącznej dokładności (na danych walidacyjnych) w kolejnych



Rysunek 4.6: Krzywe uczenia przy wykorzystaniu modeli językowych plT5 oraz HerBERT.

epokach,[31] W przypadku drugiej z wymienionych rodzin modeli proces uczenia przy tym samym parametrze *learning rate* (zob. 4.1) przebiega dużo wolniej. Dla modeli plT5 w rozmiarach SMALL oraz BASE wyraźnie widać, że dokładność na danych walidacyjnych pod koniec trenowania nadal wykazuje tendencję malejącą — zatrzymanie uczenia spowodowane było dla tych modeli przekroczeniem limitu 50 epok, a nie brakiem poprawy tej miary.

[31] Krzywe dla modeli XLM-RoBERTa zostały pominięte jako bardzo zbliżone do tych odpowiadających modelom HerBERT. Nie jest to zaskakujące, ponieważ architektury te są bardzo podobne.

Podsumowanie

W rozprawie przedstawiłam nową metodę automatycznej analizy składniowej tekstów polskich. W moim podejściu łączę grafową metodę parsowania zależnościowego z techniką typu *parsing as tagging*, formułującą problem parsowania jako przypisywanie etykiet poszczególnym segmentom. Kluczowymi dla zaprezentowanej metody elementami są kręgosłupy składniowe, które łączone są w wynikową strukturę odpowiadającą budowie składniowej wypowiedzenia. Istotnie nowym na gruncie maszynowego przetwarzania polszczyzny aspektem mojej techniki jest hybrydowa natura generowanych rozbiorów składniowych. Drzewa uzyskiwane przez zastosowanie mojego algorytmu łączą w sobie mianowicie struktury składnikowe i zależnościowe, pozwalając na elastyczne wykorzystanie zalet obu reprezentacji. Zaletą zaprezentowanego przeze mnie podejścia do parsowania jest również możliwość produkowania rozbiorów zawierających składniki nieciągłe.

W ramach pracy stworzyłam implementację prezentowanej techniki parsowania w postaci głębokiej sieci neuronowej — narzędzie Hydra — oraz przeprowadziłam szereg eksperymentów, które wykazały wysoką skuteczność znakowania tekstu przy użyciu mojej metody. Dotyczy to zarówno zadania stanowiącego podstawowy cel, czyli anotacji składniowej, jak i dodatkowych zadań, które mogą realizować modele o zaproponowanej przeze mnie architekturze: znakowania fleksyjnego oraz wyszukiwania w tekście jednostek nazewniczych. Prowadzone prace skupione były na automatycznym przetwarzaniu polszczyzny, ale możliwość zastosowania opracowanej metody do danych w innym języku (pod warunkiem dostępności odpowiedniego materiału uczącego) została potwierdzona eksperymentem z korpusem niemieckojęzycznym.

Należy nadmienić, że wspomniana implementacja sposobu parsowania stanowiącego przedmiot niniejszej pracy jest w pełni funkcjonalna i znalazła już zastosowanie w projektach i zadaniach realizowanych w Zespole Inżynierii Lingwistycznej PAN. Głównym praktycznym zastosowaniem parsera Hydra planowanym podczas jego tworzenia było automatyczne oznakowanie Korpusu Współczesnego Języka Polskiego (KWJP, Kieraś *et al.* 2025) obejmującego teksty z dekady 2011–2020. Cel ten zo-

stał osiągnięty: teksty składające się na korpus zostały za pomocą Hydry podzielone na segmenty (ponad 1,5 miliarda) i opatrzone anotacją w warstwach fleksyjnej oraz składniowej. Opracowane narzędzie sprawdziło się zatem również pod względem użytkowym i wydajnościowym przy przetwarzaniu dużego zasobu językowego.

Znakowanie KWJP wykonane przy użyciu Hydry zostało zindeksowane w wyszukiwarce korpusowej. Zapytania, za pomocą których można przeszukiwać teksty, mogą odwoływać się do form hasłowych, znaczników morfoskładniowych, funkcji oraz nadrzędników zależnościowych segmentów, a także składników wypowiedzeń. Drzewa wygenerowane przez parser są udostępniane użytkownikom w postaci dwóch typów wizualizacji graficznych: pełnej struktury hybrydowej oraz tylko jej części zależnościowej.

Parser Hydra służy również jako narzędzie wspierające rozbudowę treebanku Składnica tak, aby objął cały zbiór pierwotnie wyznaczonych wypowiedzeń (zob. 1.2.1). Około 6 tysięcy zdań niemożliwych do przetworzenia za pomocą gramatyki Świgry zostało automatycznie zanalizowane modelem Hydry. Uzyskane struktury składniowe są obecnie poddawane ręcznej weryfikacji i korekcie.

Możliwość uzyskiwania za pomocą Hydry rozbiorów nieciągłych jest szczególnie istotna w kontekście trwających prac nad potokiem przetwarzania dla polskich tekstów historycznych. Jak wskazuje Wieczorek (2023), zagęszczenie konstrukcji nieciągłych w badanych zdaniach z XVII i XVIII wieku jest kilkakrotnie większe, niż w materiale współczesnym — 8,9% w porównaniu do 1,8% krawędzi nieprojekcyjnych w drzewach zależnościowych.

Model Hydry wykonujący znakowanie fleksyjne oraz rozpoznawanie nazw własnych został włączony w potok przetwarzania danych systemu do wykrywania neologizmów NeoN (Tomaszewska *et al.*, 2025). Automatyczna anotacja wykonana za pomocą modelu stanowi jedno ze źródeł informacji, na podstawie których w stale aktualizowanej bazie tekstów pochodzących z internetu wyszukiwane są nowe jednostki leksykalne pojawiające się w polszczyźnie.

Podsumowując, przedstawiona przeze mnie w niniejszej rozprawie metoda automatycznej analizy składniowej, poddana eksperymentalnej ewaluacji, okazała się bardzo skuteczna. Jej istotną zaletą jest możliwość praktycznego użycia w projektach z dziedziny przetwarzania języka naturalnego.

Dodatek A

Dodatkowe informacje: dane

kat. składniowa	liczba wystąpień		kat. składniowa	liczba wystąpień	
NP	109273	26.08%	CAP	2317	0.55%
PP	91172	21.76%	CVP	1603	0.38%
S	72393	17.28%	CPP	1295	0.31%
VROOT	45214	10.79%	CO	389	0.09%
VP	35816	8.55%	DL	337	0.08%
AP	15099	3.60%	CH	228	0.05%
PN	12951	3.09%	CAVP	212	0.05%
CNP	12570	3.00%	AA	121	0.03%
CS	5905	1.41%	ISU	37	0.01%
AVP	4604	1.10%	MTA	37	0.01%
VZ	4416	1.05%	CAC	30	0.01%
NM	3001	0.72%	CVZ	23	0.01%

Tabela A.1: Kategorie składniowe występujące w banku drzew TIGER.

ZALEŻNICA			TIGER		
kręgosłup	**liczba wystąpień**		**kręgosłup**	**liczba wystąpień**	
NP → NP → N	57 711	16,56%	—	459 724	53,15%
Punct	52 952	15,20%	NP	148 810	17,20%
NP → N	50 953	14,62%	PP	88 756	10,26%
AdjP → Adj	35 446	10,17%	VROOT → S	37 184	4,30%
PrepNP → Prep	33 726	9,68%	S	31 560	3,65%
ROOT → S → VP → V	16 472	4,73%	VP	30 771	3,56%
VP → VP → V	12 450	3,57%	AP	13 336	1,54%
S → VP → V	12 076	3,47%	CNP	8 546	0,99%
AdvP → Adv	8 897	2,55%	PN	7 350	0,85%
AdjP → AdjP → Adj	8 470	2,43%	AVP	4 463	0,52%
Part	6 411	1,84%	VP → VZ	4 163	0,48%
Part → Refl	5 854	1,68%	VROOT → CS	3 666	0,42%
CP → Comp	5 609	1,61%	NP → CNP	3 005	0,35%
Neg	4 001	1,15%	NODE	2 525	0,29%
NP → Conj	3 970	1,14%	CAP	20 73	0,24%
ROOT → S → Conj	3 293	0,95%	CS	1 808	0,21%
—	3 160	0,91%	NP → PN	1 736	0,20%
NP → NumP → Num	2 952	0,85%	NP → PN → NP	1 697	0,20%
CP → S → VP → V	2 537	0,73%	CVP	1 539	0,18%
AdvP → AdvP → Adv	2 129	0,61%	CPP	1 201	0,14%

Tabela A.2: Najczęstsze typy kręgosłupów w danych hybrydowych.

ZALEŻNICA			TIGER		
kat. składniowa	**liczba wystąpień**		**kat. składniowa**	**liczba wystąpień**	
NP	180 691	17,75%	NP	531 550	33,73%
Conj	136 315	13,39%	VP	404 281	25,65%
AdvP	135 900	13,35%	AP	383 669	24,35%
N	109 215	10,73%	PP	89 505	5,68%
VP	60 303	5,92%	S	71 734	4,55%
AdjP	54 180	5,32%	VROOT	43 066	2,73%
Punct	52 952	5,20%	CNP	12 125	0,77%
V	45 159	4,44%	PN	11 837	0,75%
Adj	43 957	4,32%	CS	5 802	0,37%
PrepNP	37 799	3,71%	AVP	4 560	0,29%
Prep	36 992	3,63%	VZ	4 384	0,28%
S	36 750	3,61%	NODE	3 933	0,25%
ROOT	22 093	2,17%	NM	2 975	0,19%
Part	13 312	1,31%	CAP	2 276	0,14%
Adv	11 122	1,09%	CVP	1 585	0,10%
CP	8 745	0,86%	CPP	1 282	0,08%
Comp	7 488	0,74%	CO	380	0,02%
Refl	5 855	0,58%	DL	318	0,02%
NumP	4 765	0,47%	CH	224	0,01%
Num	4 544	0,45%	CAVP	209	0,01%

Tabela A.3: Najczęstsze kategorie składniowe w danych hybrydowych.

ZALEŻNICA			TIGER		
relacja	**liczba wystąpień**		**relacja**	**liczba wystąpień**	
comp	58 376	16,75%	NK	271 235	31,36%
punct	51 581	14,80%	MO	112 245	12,98%
adjunct	45 867	13,16%	–	109 421	12,65%
conjunct	29 576	8,49%	SB	63 984	7,40%
root	22 093	6,34%	CJ	51 139	5,91%
subj	22 064	6,33%	root	46 265	5,35%
obj	16 021	4,60%	OC	37 336	4,32%
adjunct_locat	7 904	2,27%	OA	32 546	3,76%
adjunct_emph	6 363	1,83%	MNR	25 119	2,90%
refl	6 134	1,76%	AG	23 398	2,71%
adjunct_temp	6 014	1,73%	PNC	10 531	1,22%
comp_fin	5 750	1,65%	PD	9 404	1,09%
pd	5 339	1,53%	CP	7 977	0,92%
mwe	4 999	1,43%	OP	7 553	0,87%
comp_inf	4 981	1,43%	RC	6 704	0,78%
ne	4 977	1,43%	NG	5 190	0,60%
adjunct_mod	4 189	1,20%	DA	5 76	0,60%
neg	3 656	1,05%	SVP	5 070	0,59%
adjunct_other	3 193	0,92%	PM	4 513	0,52%
adjunct_rc	2 799	0,80%	APP	3 750	0,43%

Tabela A.4: Najczęstsze relacje zależnościowe w danych hybrydowych.

zmiana znacznika	wyjaśnienie
NKJP	
num:*:congr|rec:*→num:*	usunięcie kategorii akomodacyjności liczebnika
ppron12:*:f|m_|n:*→ppron12:*	usunięcie kat. rodzaju zaimka 1. i 2. os.
:sg:acc:m1|m2:→*:manim2:*	zmiana wartości rodzajów męskich
:sg:acc:m3→:m:*	
:sg::m_:*→*:m:*	
:pl:nom|acc|voc:m1→:manim1:*	
:pl:nom|acc|voc:m2|m3→:m:*	
*:pl:*m_:*→*:m:*	
num:*:col:*→numcol:*	zastąpienie klasy liczebnika zbiorowego
:col|ncol:→*:*	usunięcie kat. przyrodzaju
adjc→adjb:sg:nom:m	zastąpienie klasy przymiotnika predykatywnego
adjp:*→adjb:sg:*:n	zastąpienie klasy. przymiotnika poprzyimkowego
pact:*→pact:*:pos	dodanie kat. stopnia imiesłowów
ppas:*→ppas:*:pos	
XIX	
:sg:acc:m1|m2:→*:manim2:*	zmiana wartości rodzajów męskich
:sg:acc:m3→:m:*	
:sg::m_:*→*:m:*	
:pl:nom|acc|voc:m1→:manim1:*	
:pl:nom|acc|voc:m2|m3→:m:*	
*:pl:*m_:*→*:m:*	
num:*:col:*→numcol:*	zastąpienie klasy liczebnika zbiorowego
:col|ncol:→*:*	usunięcie kat. przyrodzaju
adjc→adjb:sg:nom:m	zastąpienie klasy przymiotnika predykatywnego
adjp:*→adjb:sg:*:n	zastąpienie klasy. przymiotnika poprzyimkowego
pact:*→pact:*:pos	dodanie kat. stopnia imiesłowów
ppas:*→ppas:*:pos	
adjnum:*→adj:*	zastąpienie klasy liczebnika przymiotnikowego
advnum:*→adv:*	zastąpienie klasy liczebnika przysłówkowego
fut:*→bedzie:*	zastąpienie klasy. być cz. przyszłego
plusq:*→praet:*	zastąpienie klasy być cz. zaprzeszłego
KORBA	
adjnum:*→adj:*	zastąpienie klasy liczebnika przymiotnikowego
advnum:*→adv:*	zastąpienie klasy liczebnika przysłówkowego
fut:*→bedzie:*	zastąpienie klasy być cz. przyszłego
plusq:*→praet:*	zastąpienie klasy być cz. zaprzeszłego
ppraet:*:asp:neg→adj:*	zastąpienie klasy imiesłowu przeszłego

Tabela A.5: Modyfikacje znaczników korpusów NKJP, XIX oraz KORBA. Objaśnienia kategorii fleksyjnych występujących w poszczególnych tagsetach oraz ich możliwych wartości znaleźć można w dokumentacjach odpowiednich korpusów.

podzbiór		**tylko** ZALEŻNICA	ZALEŻNICA ∩ NKJP	**tylko** NKJP	**łącznie**
treningowy	wypowiedzenia	6 242	11 433	57 442	75 117
		8.31%	15.22%	76.47%	
	segmenty	105 450	175 016	803 436	1 083 902
		9.73%	16.15%	74.12%	
walidacyjny	wypowiedzenia	781	1 431	6 466	8 678
		9.00%	16.49%	74.51%	
	segmenty	13 236	21 344	92 711	127 291
		10.40%	16.77%	72.83%	
testowy	wypowiedzenia	777	1 429	7 461	9 667
	8.04%	14.78%	77.18%		
	segmenty	13 157	20 222	102 903	136 282
		9.65%	14.84%	75.51%	
łącznie	wypowiedzenia	7 800	14 293	71 369	93 462
		8.35%	15.29%	76.36%	
	segmenty	131 843	216 582	999 050	1 347 475
		9.78%	16.07%	74.14%	

Tabela A.6: Rozmiar i podział zbioru danych powstałego z połączenia korpusów NKJP 1M oraz Zależnica. Wyszczególnione zostały elementy występujące wyłącznie w jednym z łączonych zasobów oraz należące do ich części wspólnej.

Dodatek B

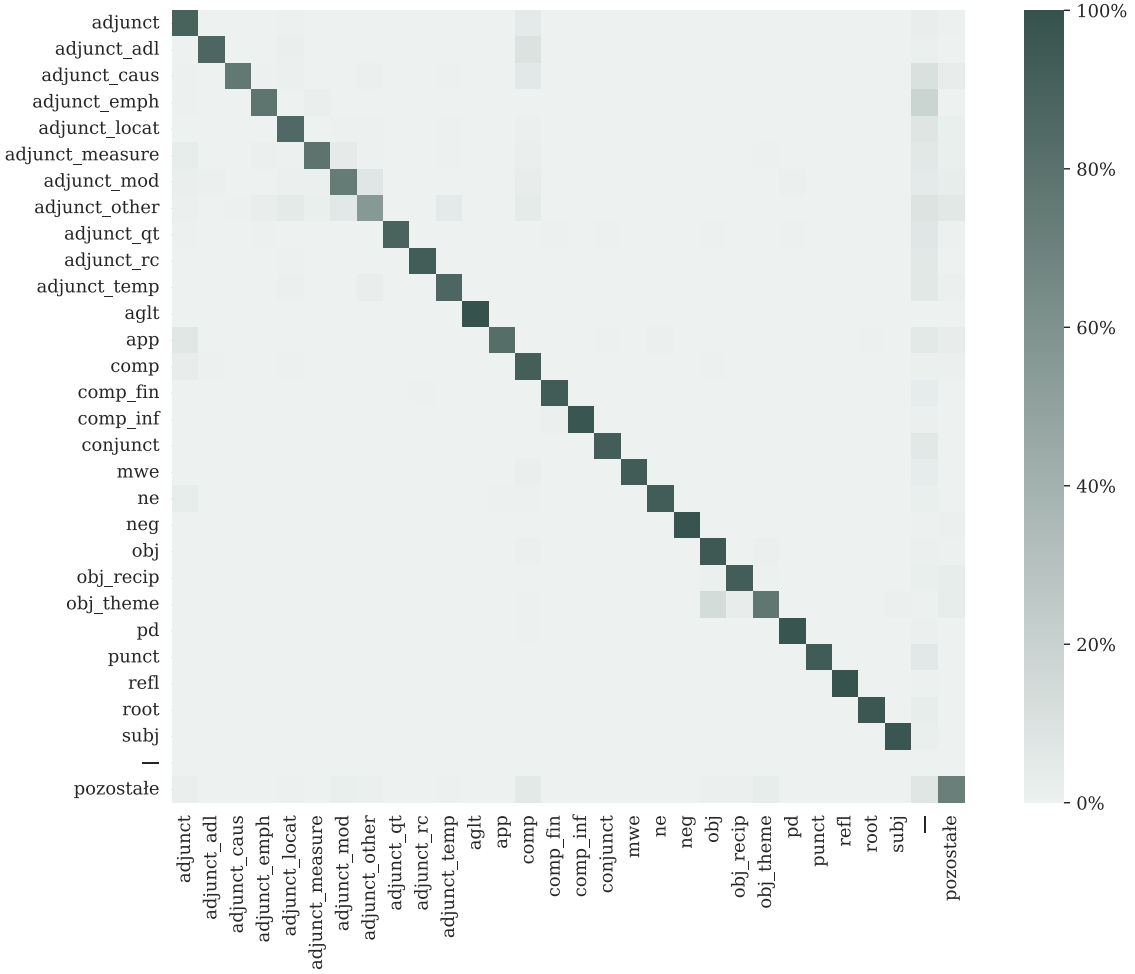
Dodatkowe informacje: ewaluacja

kat. składniowa	precyzja	czułość	F1	wystąpienia	
Refl	100,00%	100,00%	100,00%	573	0,77%
ROOT	100,00%	100,00%	100,00%	2206	2,98%
Aux	100,00%	100,00%	100,00%	54	0,07%
Prep	99,91%	99,94%	99,93%	3495	4,72%
N	99,77%	99,70%	99,74%	10450	14,12%
V	99,73%	99,73%	99,73%	4401	5,95%
Adj	99,44%	99,58%	99,51%	4083	5,52%
Neg	99,47%	99,47%	99,47%	376	0,51%
Punct	99,07%	99,05%	99,06%	5162	6,98%
Adv	97,84%	98,68%	98,26%	1057	1,43%
AdjP	97,52%	97,40%	97,46%	5009	6,77%
Comp	97,22%	97,51%	97,36%	682	0,92%
NP	97,29%	96,82%	97,05%	17269	23,34%
Num	96,29%	97,57%	96,92%	452	0,61%
Part	98,19%	94,99%	96,57%	1317	1,78%
Conj	95,53%	96,65%	96,09%	1372	1,85%
PrepNP	96,20%	95,88%	96,04%	3594	4,86%
Brev	96,59%	94,74%	95,65%	209	0,28%
Compar	94,44%	95,97%	95,20%	124	0,17%
NumP	94,33%	95,53%	94,93%	470	0,64%
VP	94,69%	94,88%	94,79%	5805	7,85%
AdvP	94,83%	94,39%	94,61%	1301	1,76%
S	92,77%	94,29%	93,52%	3605	4,87%
PrepAdjP	92,38%	88,18%	90,23%	110	0,15%
CP	88,05%	88,71%	88,38%	806	1,09%
Interj	91,67%	78,57%	84,62%	14	0,02%

Tabela B.1: Precyzja, czułość oraz miara F1 wykrywania poszczególnych typów składników dla języka polskiego.

relacja	**precyzja**	**czułość**	**F1**	**wystąpienia**	
aglt	99,55%	100,00%	99,77%	219	0,66%
refl	99,33%	99,33%	99,33%	593	1,78%
neg	98,49%	98,49%	98,49%	331	0,99%
abbrev_punct	97,46%	99,14%	98,29%	116	0,35%
comp_inf	98,55%	97,54%	98,04%	487	1,46%
aux	99,07%	96,36%	97,70%	110	0,33%
subj	96,68%	97,33%	97,01%	2097	6,28%
pd	95,45%	98,09%	96,75%	470	1,41%
root	95,95%	96,60%	96,27%	2206	6,61%
punct	93,88%	93,89%	93,88%	5044	15,11%
obj	92,28%	95,33%	93,78%	1543	4,62%
mwe	93,95%	93,55%	93,75%	465	1,39%
ne	92,48%	93,12%	92,80%	436	1,31%
conjunct	92,41%	92,70%	92,56%	2878	8,62%
comp_fin	90,07%	93,97%	91,98%	531	1,59%
comp	91,14%	92,48%	91,80%	5571	16,69%
adjunct_rc	90,04%	93,25%	91,62%	252	0,75%
adjunct	91,86%	90,02%	90,93%	4398	13,18%
adjunct_qt	87,76%	89,12%	88,43%	193	0,58%
adjunct_compar	88,50%	86,21%	87,34%	116	0,35%
obj_recip	81,61%	92,81%	86,85%	153	0,46%
app	88,27%	82,78%	85,43%	209	0,63%
adjunct_temp	83,48%	87,36%	85,38%	538	1,61%
adjunct_adl	79,79%	87,72%	83,57%	171	0,51%
adjunct_locat	79,08%	85,33%	82,09%	784	2,35%
adjunct_emph	80,14%	78,41%	79,26%	602	1,80%
adjunct_caus	81,82%	75,90%	78,75%	166	0,50%
adjunct_measure	73,64%	79,28%	76,36%	222	0,67%
obj_instr	75,64%	76,62%	76,13%	77	0,23%
adjunct_cond	76,56%	72,06%	74,24%	68	0,20%
adjunct_freq	70,89%	77,78%	74,17%	72	0,22%
adjunct_mod	73,32%	74,28%	73,79%	381	1,14%
obj_theme	70,38%	77,10%	73,59%	262	0,78%
adjunct_abl	67,12%	67,12%	67,12%	73	0,22%
adjunct_purp	72,38%	55,47%	62,81%	137	0,41%
adjunct_comment	62,50%	59,70%	61,07%	67	0,20%
adjunct_other	62,16%	55,76%	58,79%	330	0,99%
adjunct_recip	56,25%	47,37%	51,43%	76	0,23%
adjunct_attrib	42,50%	34,69%	38,20%	98	0,29%

Tabela B.2: Precyzja, czułość oraz miara F1 wykrywania poszczególnych relacji zależnościowych dla języka polskiego. Prawidłowo wykryta relacja oznacza właściwie wyznaczoną krawędź zależnościową opatrzoną właściwą etykietą. W zestawieniu pominięte zostały 32 typy relacji przypisane w danych testowych do mniej niż 0,2% krawędzi.



Rysunek B.1: Macierz błędów wykrywania relacji zależnościowych dla języka polskiego. Wiersze odpowiadają relacjom faktycznie występującym w danych testowych, kolumny — decyzjom parsera. Przekątna macierzy odpowiada prawidłowemu wskazaniu relacji. Pozostałe komórki oznaczają poprawne wykrycie krawędzi z przypisaniem niewłaściwej etykiety lub niewykrycie krawędzi (kolumna '—').

typ jednostki	**precyzja**	**czułość**	**F1**	**wystąpienia**	
`persName.forename`	92,99%	93,24%	93,12%	6877	17,17%
`persName.surname`	95,10%	91,33%	93,17%	6054	15,11%
`persName`	91,70%	90,83%	91,26%	10127	25,28%
`placeName.settlement`	92,74%	91,63%	92,18%	3430	8,56%
`placeName.country`	95,60%	89,97%	92,70%	3451	8,61%
`date`	87,45%	88,39%	87,92%	1869	4,67%
`orgName`	80,21%	83,41%	81,78%	4727	11,80%
`time`	75,67%	79,92%	77,73%	249	0,62%
`geogName`	75,28%	75,67%	75,48%	1755	4,38%
`placeName.district`	69,17%	58,45%	63,36%	142	0,35%
`placeName.region`	65,67%	55,65%	60,24%	354	0,88%
`persName.addName`	33,65%	31,63%	32,61%	781	1,95%
`placeName.bloc`	69,77%	12,35%	20,98%	243	0,61%
`placeName`	1,78%	100,00%	3,49%	3	0,01%

Tabela B.3: Precyzja, czułość oraz miara F1 wykrywania poszczególnych typów jednostek nazewniczych.

Dodatek C

Architektura kodująca Transformer

W następującym opisie architektury sieci neuronowej Transformer (Vaswani *et al.*, 2017) dla uproszczenia pomijamy kwestię przekazywania danych do modelu partiami (*batching*) i zakładamy, że tekst do przetworzenia został już podzielony na t tokenów stanowiących sekwencję wejściową. Tokenom zostały następnie przyporządkowane wstępne reprezentacje wektorowe długości m. Reprezentacje takie pochodzą najczęściej z trenowanej wraz z całym modelem Transformer macierzy o wymiarach $|V| \times m$, gdzie V to zestaw (słownik, *vocabulary*) możliwych tokenów. i-ty wiersz tej macierzy stanowi zanurzenie wektorowe dla i-tego tokenu w słowniku. Można również wyobrazić sobie scenariusz, w którym wektory zostają zaczerpnięte z zewnętrznego zasobu, na przykład typu Word2Vec czy FastText (zob. 2.1). Pochodzenie wektorów nie jest kluczowe dla samej architektury, tak naprawdę nie one muszą nawet reprezentować fragmentów tekstu, ale jakieś inne dane sekwencyjne.

Wejściowe zanurzenia wektorowe tworzą macierz E o wymiarach $t \times m$, która stanowi wejście dla dowolnej liczby działających równolegle modułów uwagi (*attention heads*). Każdy moduł uwagi zaczyna swoje obliczenia od pomnożenia E przez trzy macierze parametrów: $\mathbf{W}_q$, $\mathbf{W}_k$ oraz $\mathbf{W}_v$[1] o wymiarach odpowiednio $m \times l_1$, $m \times l_1$ oraz $m \times l_2$ (l_1 oraz l_2 są ustalonymi parametrami modelu). W wyniku tych przekształceń powstają macierze nazywane standardowo Q, K oraz V (z ang. *query* — zapytanie, *key* — klucz oraz *value* — wartość):

$$Q := E \cdot \mathbf{W}_q \tag{C.1}$$

$$K := E \cdot \mathbf{W}_k \tag{C.2}$$

$$V := E \cdot \mathbf{W}_v \tag{C.3}$$

Macierze te mają wymiary $t \times l_1$, $t \times l_1$ oraz $t \times l_2$

Iloczyn macierzy zapytań oraz kluczy, poddany dodatkowemu przekształceniu f_a, daje w wyniku macierz uwagi (*attention*) o wymiarach $t \times t$:

[1] Przyjmujemy konwencję, według której macierze trenowane (optymalizowane) w ramach uczenia modelu oznaczamy $\mathbf{W}$ i rozróżniamy indeksami dolnymi.

$$A := f_a(Q \cdot K^{\mathrm{T}}) \tag{C.4}$$

Zgodnie z intuicją stojącą za architekturą Transformer, macierz ta reprezentuje stopień wzajemnego powiązania, istotność tokenów względem siebie w kontekście tekstu wejściowego. Wartość A_{ij} stanowi iloczyn skalarny i-tego wiersza Q z j-tym wierszem K i wskazuje, jak wiele „uwagi" należy poświęcić tokenowi j jako kontekstowi dla tokenu i. Przekształcenie f_a obejmuje przeskalowanie wartości w macierzy oraz zastosowanie funkcji *softmax* do poszczególnych wierszy — wartości w każdym z nich sumują się w efekcie do jedności.

Ostatecznym wynikiem działania modułu uwagi jest iloczyn macierzy A oraz V, który oznaczymy X. Ma on wymiar $t \times l_2$, a jego i-ty wiersz $(1 \leq i \leq t)$ obliczany jest w następujący sposób:

$$X_i := A_i^{\mathrm{T}} \cdot V \tag{C.5}$$

i stanowi średnią wierszy macierzy V (odpowiadających poszczególnym tokenom) ważoną i-tym wierszem A (określającym istotność tych tokenów w kontekście tokenu i). Powstała tak macierz X zawiera zatem t nowych, wektorowych reprezentacji dla rozważanych tokenów, wzbogaconych o kontekst całej sekwencji wejściowej dobrany z użyciem macierzy uwagi A.

Jak wspomniano wcześniej, w architekturze Transformer występuje pewna liczba n modułów uwagi działających równolegle. Powstaje zatem n macierzy takich, jak opisana powyżej X: $X_1, ... X_n$ które zostają skonkatenowane wierszami. Do powstałej w ten sposób macierzy o wymiarach $t \times n \cdot l_2$ dodawana jest wejściowa macierz E (wartości m, n oraz l_2 muszą być dobrane tak, aby zachodziło $n \cdot l_2 = m$), a wynik dodawania poddawany jest normalizacji. Krok ten, nazywany *residual connection*, zapewnia, że informacja zawarta w E nie jest całkowicie zastępowana, ale „aktualizowana" o nowo obliczoną informację w X:

$$X' := \mathit{normalise}(E + [X_1; ...; X_n]) \tag{C.6}$$

Macierz X' przechodzi następnie przez dwie warstwy gęste z aktywacją nieliniową, odpowiednio zwiększające jej wymiar poziomy (*expansion*) oraz przywracające go do pierwotnego (*compression*). Wynik tych dwóch operacji ponownie łączony jest z X' mechanizmem *residual connection*. Powstała macierz Y ma wymiar $t \times m$ identyczny z wymiarem macierzy wejściowej E i koduje zanurzenia wektorowe tokenów wzbogacone o kontekst całej sekwencji. Opisana architektura sieci neuro-

nowej, przyjmującej na wejściu E i produkującej Y, nazywana jest warstwą kodującą (*encoder*) typu Transformer.[2]

Wynik działania jednej warstwy Transformer może zostać przekazany jako wejście do kolejnej. Modele typu BERT składają się typowo z kilkunastu takich warstw następujących jedna po drugiej i produkujących kolejne reprezentacje wektorowe wejściowych tokenów.

[2] Warstwa dekodująca (*decoder*) wykorzystuje podobne mechanizmy. Pomijamy jej szczegółowy opis, ponieważ nie jest wykorzystywana w modelach językowych typu BERT.

Bibliografia

Acedański, S. (2010). A morphosyntactic brill tagger for inflectional languages. W: H. Loftsson, E. Rögnvaldsson, i S. Helgadóttir, red., *Advances in Natural Language Processing*, str. 3–14, Berlin, Heidelberg, Niemcy. Springer Berlin Heidelberg. 81

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S. i Collins, M. (2016). Globally normalized transition-based neural networks. W: K. Erk i N. A. Smith, red., *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, str. 2442–2452, Berlin, Niemcy. Association for Computational Linguistics. 47

Attardi, G. (2006). Experiments with a multilanguage non-projective dependency parser. W: L. Màrquez i D. Klein, red., *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, str. 166–170, Nowy Jork, USA. Association for Computational Linguistics. 47

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference. W: *Information Processing, Proceedings of the 1st International Conference on Information Processing*, str. 125–131. 39

Ballesteros, M. i Carreras, X. (2015). Transition-based spinal parsing. W: *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, str. 289–299, Pekin, Chiny. Association for Computational Linguistics. 53

Ballesteros, M. i Carreras, X. (2017). Arc-standard spinal parsing with stack-LSTMs. W: Y. Miyao i K. Sagae, red., *Proceedings of the 15th International Conference on Parsing Technologies*, str. 115–121, Piza, Włochy. Association for Computational Linguistics. 53

Bangalore, S. i Joshi, A. K. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2), 237–265. 42

Batsuren, K., Goldman, O., Khalifa, S., Habash, N., Kieraś, W., Bella, G., Leonard, B., Nicolai, G., Gorman, K., Ate, Y. G., Ryskina, M., Mielke, S., Budianskaya, E., El-Khaissi, C., Pimentel, T., Gasser, M., Lane, W. A., Raj, M., Coler, M., Samame, J. R. M., Camaiteri, D. S., Rojas, E. Z., López Francis, D., Oncevay, A.,

López Bautista, J., Villegas, G. C. S., Hennigen, L. T., Ek, A., Guriel, D., Dirix, P., Bernardy, J.-P., Scherbakov, A., Bayyr-ool, A., Anastasopoulos, A., Zariquiey, R., Sheifer, K., Ganieva, S., Cruz, H., Karahóǧa, R., Markantonatou, S., Pavlidis, G., Plugaryov, M., Klyachko, E., Salehi, A., Angulo, C., Baxi, J., Krizhanovsky, A., Krizhanovskaya, N., Salesky, E., Vania, C., Ivanova, S., White, J., Maudslay, R. H., Valvoda, J., Zmigrod, R., Czarnowska, P., Nikkarinen, I., Salchak, A., Bhatt, B., Straughn, C., Liu, Z., Washington, J. N., Pinter, Y., Ataman, D., Wolinski, M., Suhardijanto, T., Yablonskaya, A., Stoehr, N., Dolatian, H., Nuriah, Z., Ratan, S., Tyers, F. M., Ponti, E. M., Aiton, G., Arora, A., Hatcher, R. J., Kumar, R., Young, J., Rodionova, D., Yemelina, A., Andrushko, T., Marchenko, I., Mashkovtseva, P., Serova, A., Prud'hommeaux, E., Nepomniashchaya, M., Giunchiglia, F., Chodroff, E., Hulden, M., Silfverberg, M., McCarthy, A. D., Yarowsky, D., Cotterell, R., Tsarfaty, R. i Vylomova, E. (2022). UniMorph 4.0: Universal Morphology. W: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, i S. Piperidis, red., *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, str. 840–855, Marsylia, Francja. European Language Resources Association. 80

Bengio, Y., Ducharme, R., Vincent, P. i Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb), 1137–1155. 43

Bień, J., Szafran, K. i Woliński, M. (2001). Experimental parsers of Polish. W: G. Zybatow, U. Junghanns, G. Mehlhorn, i L. Szucsich, red., *Current Issues in Formal Slavic Linguistics*, tom 5 serii *Linguistik International*, str. 185–190, Frankfurt nad Menem, Niemcy. Peter Lang. 50

Bień, J. S. (1996). Komputerowa weryfikacja opisu składni polskiej. Raport techniczny 96-06 (227), Instytut Informatyki UW, Warszawa. 50

Bień, J. S. (1997). Komputerowa weryfikacja formalnej gramatyki Świdzińskiego. *Biuletyn Polskiego Towarzystwa Językoznawczego*, LII, 147–164. 50

Bień, J. S. (2009). *Problemy formalnego opisu składni polskiej*. BEL Studio, Warszawa. 50

Bilińska, J., Derwojedowa, M., Kieraś, W. i Kwiecień, M. (2016). Mikrokorpus polszczyzny 1830-1918. *Komunikacja specjalistyczna*, 11, 149–161. 105

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. W: C.-R. Huang i D. Jurafsky, red., *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, str. 89–97, Pekin, Chiny. Coling 2010 Organizing Committee. 49

Bojanowski, P., Grave, E., Joulin, A. i Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146. 43

Booth, T. L. (1969). Probabilistic representation of formal languages. W: *10th Annual Symposium on Switching and Automata Theory*, str. 74–81. 40

Borchmann, L., Gretkowski, A. i Graliński, F. (2018). Approaching nested named entity recognition with parallel LSTM-CRFs. W: *Proceedings of the PolEval 2018 Workshop*, str. 63–73, Warszawa. Instytut Podstaw Informatyki PAN. 110

Brants, S., Dipper, S., Eisenberg, P., Hansen, S., König, E., Lezius, W., Rohrer, C., Smith, G. i Uszkoreit, H. (2004). TIGER: Linguistic interpretation of a German corpus. *Journal of Language and Computation*, 2, 597–620. 33

Bresnan, J., red. (1982). *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, USA. 41

Carreras, X. (2007). Experiments with a higher-order projective dependency parser. W: J. Eisner, red., *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, str. 957–961, Praga, Czech y. Association for Computational Linguistics. 48, 49

Carreras, X., Collins, M. i Koo, T. (2008). TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. W: A. Clark i K. Toutanova, red., *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, str. 9–16, Manchester, Wielka Brytania. Coling 2008 Organizing Committee. 53

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. W: *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, str. 598–603, Menlo Park, USA. AAAI Press. 40

Charniak, E. (2000). A maximum-entropy-inspired parser. W: *1st Meeting of the North American Chapter of the Association for Computational Linguistics*. 40

Chen, D. i Manning, C. (2014). A fast and accurate dependency parser using neural networks. W: A. Moschitti, B. Pang, i W. Daelemans, red., *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, str. 740–750, Doha, Qatar. Association for Computational Linguistics. 47

Choe, D. K. i Charniak, E. (2016). Parsing as language modeling. W: J. Su, K. Duh, i X. Carreras, red., *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, str. 2331–2336, Austin, USA. Association for Computational Linguistics. 43

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(3), 113–124. 39

Chrabrowa, A., Dragan, Ł., Grzegorczyk, K., Kajtoch, D., Koszowski, M., Mroczkowski, R. i Rybak, P. (2022). Evaluation of transfer learning for Polish with

a text-to-text model. W: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, i S. Piperidis, red., *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, str. 4374–4394, Marsylia, Francja. European Language Resources Association. 111

Chu, Y.-J. i Liu, T.-H. (1965). On the shortest arborescence of a directed graph. *Science Sinica*, XIV(10), 1396–1400. 48

Coavoux, M. i Cohen, S. B. (2019). Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. W: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, str. 204–217, Minneapolis, USA. Association for Computational Linguistics. 45

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. W: *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, str. 16–23, Madryt, Hiszpania. Association for Computational Linguistics. 40

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4), 589–637. 40

Collobert, R. (2011). Deep learning for efficient discriminative parsing. W: G. Gordon, D. Dunson, i M. Dudík, red., *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, tom 15 serii *Proceedings of Machine Learning Research*, str. 224–232, Fort Lauderdale, USA. PMLR. 42, 43

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. i Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(76), 2493–2537. 42, 43

Colmerauer, A. (1978). Metamorphosis grammars. W: *Natural Language Communication with Computers*, tom 63 serii *Lecture Notes in Computer Science*, str. 133–189, Berlin, Heidelberg, Niemcy. Springer-Verlag. 41

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L. i Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. W: D. Jurafsky, J. Chai, N. Schluter, i J. Tetreault, red., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, str. 8440–8451, Online. Association for Computational Linguistics. 111

Corro, C. (2020). Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from $O(n^6)$ down to $O(n^3)$. W: B. Webber, T. Cohn, Y. He, i Y. Liu, red., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, str. 2753–2764. Association for Computational Linguistics. 45

Covington, M. (2001). A fundamental algorithm for dependency parsing. W: *Proceedings of the 39th Annual ACM Southeast Conference*. 47

Cross, J. i Huang, L. (2016). Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. W: J. Su, K. Duh, i X. Carreras, red., *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, str. 1–11, Austin, USA. Association for Computational Linguistics. 43

Dadas, S. i Protasiewicz, J. (2020). A bidirectional iterative algorithm for nested named entity recognition. *IEEE Access*. 110

Dalrymple, M., red. (2023). *Handbook of Lexical Functional Grammar*. Nr 13 serii Empirically Oriented Theoretical Morphology and Syntax. Language Science Press, Berlin, Niemcy. 41

de Marneffe, M.-C., Manning, C. D., Nivre, J. i Zeman, D. (2021). Universal Dependencies. *Computational Linguistics*, 47(2), 255–308. 23

Dębowski, Ł. (2004). Trigram morphosyntactic tagger for Polish. W: M. A. Kłopotek, S. T. Wierzchoń, i K. Trojanowski, red., *Intelligent Information Processing and Web Mining*, str. 409–413, Berlin, Heidelberg, Niemcy. Springer Berlin Heidelberg. 81

DeepSeek-AI (2025). Deepseek-v3 technical report. preprint arXiv:2412.19437. 60

Degórski, L. i Przepiórkowski, A. (2012). Ręcznie znakowany milionowy podkorpus NKJP. W: Przepiórkowski *et al.* (2012), chapter 5, str. 51–58. 101

Derwojedowa, M. (2011). *Składnia liczebników we współczesnym języku polskim. Zarys opisu zależnościowego*. Wydawnictwo Wydziału Polonistyki UW, Warszawa. 51

Derwojedowa, M., Rudolf, M. i Świdziński, M. (2003). Two formal approaches to Polish numeral phrases implemented. W: M. Gębka-Wolak, I. Kaproń-Charzyńska, i M. Urban, red., *Studia z gramatyki i leksykologii języka polskiego*, str. 93–108. Uniwersytet Mikołaja Kopernika, Toruń. 50

Devlin, J., Chang, M.-W., Lee, K. i Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. W: J. Burstein, C. Doran, i T. Solorio, red., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, str. 4171–4186, Minneapolis, USA. Association for Computational Linguistics. 43, 59

Dozat, T. i Manning, C. D. (2017). Deep biaffine attention for neural dependency parsing. W: *International Conference on Learning Representations*. 49, 61, 77, 78

Dyer, C., Ballesteros, M., Ling, W., Matthews, A. i Smith, N. A. (2015). Transition-based dependency parsing with stack long short-term memory. W: C. Zong i M. Strube, red., *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural*

Language Processing (*Volume 1: Long Papers*), str. 334–343, Pekin, Chiny. Association for Computational Linguistics. 47

Dyer, C., Kuncoro, A., Ballesteros, M. i Smith, N. A. (2016). Recurrent neural network grammars. W: K. Knight, A. Nenkova, i O. Rambow, red., *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, str. 199–209, San Diego, USA. Association for Computational Linguistics. 43

Earley, J. (1968). *An Efficient Context-Free Parsing Algorithm*. Rozprawa doktorska, Carnegie Mellon University, Pittsburgh, USA. 40

Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2), 94–102. 40

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B(4), 233–24. 48

Eisner, J. (2000). Bilexical grammars and their cubic-time parsing algorithms. W: *Advances in Probabilistic and Other Parsing Technologies*, str. 29–61. Springer, Dordrecht, Holandia. 48

Eisner, J. M. (1996). Three new probabilistic models for dependency parsing: An exploration. W: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. 48

Falenska, A., Czesznak, Z., Jung, K., Völkel, M., Seeker, W. i Kuhn, J. (2020). GRAIN-S: Manually annotated syntax for German interviews. W: *Proceedings of the Twelfth Language Resources and Evaluation Conference*, str. 5169–5177, Marsylia, Francja. European Language Resources Association. 33

Fernández-González, D. i Gómez-Rodríguez, C. (2021). Reducing discontinuous to continuous parsing with pointer network reordering. W: M.-F. Moens, X. Huang, L. Specia, i S. W.-t. Yih, red., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, str. 10570–10578, On-line/Punta Cana, Dominikana. Association for Computational Linguistics. 45

Fernández-González, D. i Gómez-Rodríguez, C. (2022). Multitask Pointer Network for multi-representational parsing. *Knowledge-Based Systems*, 236, 107760. 50

Fernández-González, D. i Martins, A. F. T. (2015). Parsing as reduction. W: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (*Volume 1: Long Papers*), str. 1523–1533, Pekin, Chiny. Association for Computational Linguistics. 45

Firth, J. R. (1957). A synopsis of linguistic theory 1930–1955. *Studies in Linguistic Analysis*, str. 1–31. 42

Graliński, F., Jassem, K. i Junczys-Dowmunt, M. (2013). Psi-toolkit: A natural language processing pipeline. W: *Computational Linguistics: Applications*, tom 458

serii *Studies in Computational Intelligence*, str. 27–39. Springer Berlin Heidelberg, Berlin, Heidelberg, Niemcy. 52

Graliński, F. (2002). Wstępujący parser języka polskiego na potrzeby systemu POLENG. tom 6 serii *Speech and Language Technology*, str. 263–276. 50

Graliński, F. (2007). *Formalizacja nieciągłości zdań przy zastosowaniu rozszerzonej gramatyki bezkontekstowej*. Rozprawa doktorska, Uniwersytet im. Adama Mickiewicza w Poznaniu. 50

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A. i in. (2024). The llama 3 herd of models. preprint arXiv:2407.21783. 60

Grünewald, S., Friedrich, A. i Kuhn, J. (2021). Applying occam's razor to transformer-based dependency parsing: What works, what doesn't, and what is really necessary. W: S. Oepen, K. Sagae, R. Tsarfaty, G. Bouma, D. Seddah, i D. Zeman, red., *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies* (*IWPT 2021*), str. 131–144. Association for Computational Linguistics. 49

Gruszczyński, W., Adamiec, D., Bronikowska, R., Kieraś, W., Modrzejewski, E., Wieczorek, A. i Woliński, M. (2022). The electronic corpus of 17th- and 18th-century Polish texts. *Language Resources and Evaluation*, 56(1), 309–332. 105

Gómez-Rodríguez, C. i Vilares, D. (2018). Constituent parsing as sequence labeling. W: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, str. 1314–1324, Bruksela, Belgia. Association for Computational Linguistics. 42

Hajnicz, E., Patejuk, A., Przepiórkowski, A. i Woliński, M. (2016). Walenty: słownik walencyjny języka polskiego z bogatym komponentem frazeologicznym. W: K. Skwarska i E. Kaczmarska, red., *Výzkum slovesné valence ve slovanských zemích*, str. 71–102. Slovanský ústav AV ČR, Praga, Czechy. 50

Hashimoto, K., Xiong, C., Tsuruoka, Y. i Socher, R. (2017). A joint many-task model: Growing a neural network for multiple NLP tasks. W: M. Palmer, R. Hwa, i S. Riedel, red., *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, str. 1923–1933, Kopenhaga, Dania. Association for Computational Linguistics. 49

Hinton, G., Srivastava, N. i Swersky, K. (2012). Lecture 6e. rmsprop: Divide the gradient by a running average of its recent magnitude. Przezrocza do wykładu. 85

Hochreiter, S. i Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. 43

Jackendoff, R. (1977). *X-bar Syntax: A Study of Phrase Structure*. Nr 2 serii Linguistic Inquiry Monographs. MIT Press. 53

Jassem, K. (2006). *Przetwarzanie tekstów polskich w systemie tłumaczenia automatycznego POLENG*. Wydawnictwo Naukowe UAM, Poznań. 50

Jelinek, F. i Lafferty, J. D. (1991). Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3), 315–353. 40

Joshi, A. K. (1985). Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? W: D. R. Dowty, L. Karttunen, i A. Zwicky, red., *Natural Language Parsing. Psychological, Computational, and Theoretical Perspectives*, Studies in Natural Language Processing, str. 206–250. Cambridge University Press. 40

Joshi, A. K. i Bangalore, S. (1994). Disambiguation of super parts of speech (or supertags): Almost parsing. W: *COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*, Kyoto, Japan. 42

Jurafsky, D. i Martin, J. H. (2009). *Speech and Language Processing*. Prentice-Hall, Upper Saddle River, USA, wydanie drugie. 39

Jurafsky, D. i Martin, J. H. (2024). *Speech and Language Processing*. Wydanie trzecie (wersja wstępna on-line). 39

Kaplan, R. M. (1973). A general syntactic processor. *Natural Language Processing*, str. 193–241. 39

Karlsson, F., Voutilainen, A., Heikkila, J. i Anttila, A. (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin, Niemcy. 45

Kasami, T. (1965). An efficient recognition and syntax analysis algorithm for context-free languages. Raport techniczny AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedfords, USA. 40

Kay, M. (1982). Algorithm schemata and data structures in syntactic processing. *Text Processing: Text Analysis and Generation, Text Typology and Attribution*, str. 327–358. 39

Kieraś, W. i Woliński, M. (2017). Morfeusz 2 — analizator i generator fleksyjny dla języka polskiego. *Język Polski*, XCVII(1), 75–83. 29

Kieraś, W. i Woliński, M. (2018). Manually annotated corpus of Polish texts published between 1830 and 1918. W: N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, i T. Tokunaga, red., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, str. 3854–3859, Miyazaki, Japonia. European Language Resources Association (ELRA). 103

Kieraś, W., Woliński, M. i Nitoń, B. (2021). Nowe wielowarstwowe znakowanie lingwistyczne zrównoważonego Narodowego Korpusu Języka Polskiego. *Język*

Polski, CI(2), 59–70. 103

Kieraś, W., Marciniak, M., Łaziński, M., Woliński, M., Bojałkowska, K., Eźlakowski, W., Kobyliński, L., Komosińska, D., Krasnowska-Kieraś, K., Rudolf, M., Tomaszewska, A., Wołoszyn, J. i Zawadzka-Paluektau, N. (2025). Korpus Współczesnego Języka Polskiego 2011–2020. *Język Polski*, CV(2), 5–20. 23, 115

Kiperwasser, E. i Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4, 313–327. 47, 49

Kitaev, N. i Klein, D. (2018). Constituency parsing with a self-attentive encoder. W: I. Gurevych i Y. Miyao, red., *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, str. 2676–2686, Melbourne, Australia. Association for Computational Linguistics. 44

Kitaev, N., Cao, S. i Klein, D. (2019). Multilingual constituency parsing with self-attention and pre-training. W: A. Korhonen, D. Traum, i L. Màrquez, red., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, str. 3499–3505, Florencja, Włochy. Association for Computational Linguistics. 44, 96

Klemensiewicz, Z. (1969). *Zarys składni polskiej*. PWN, Warszawa. 51

Klimaszewski, M. i Wróblewska, A. (2021). COMBO: State-of-the-art morphosyntactic analysis. W: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, str. 50–62, On-line/Punta Cana, Dominikana. Association for Computational Linguistics. 49, 51, 61, 78, 82, 104

Kobyliński, L. i Kieraś, W. (2016). Part of speech tagging for Polish: State of the art and future perspectives. W: *Proceedings of the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*, Konya, Turcja. 81

Kobyliński, L. i Ogrodniczuk, M. (2017). Results of the PolEval 2017 competition: Part-of-speech tagging shared task. W: Z. Vetulani i P. Paroubek, red., *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, str. 362–366, Poznań. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu. 101

Kondratyuk, D. i Straka, M. (2019). 75 languages, 1 model: Parsing Universal Dependencies universally. W: K. Inui, J. Jiang, V. Ng, i X. Wan, red., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, str. 2779–2795, Hong Kong, Chiny. Association for Computational Linguistics. 49

Krasnowska-Kieraś, K. (2017). Morphosyntactic disambiguation for Polish with bi-LSTM neural networks. W: Z. Vetulani i P. Paroubek, red., *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for*

Computer Science and Linguistics, str. 367–371, Poznań. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu. 81, 104

Krasnowska-Kieraś, K. i Kobyliński, L. (2019). Part of speech tagging for Polish. *Poznań Studies in Contemporary Linguistics*, 55(2), 211–237. 81

Krasnowska-Kieraś, K. i Woliński, M. (2023). Constituency parsing with spines and attachments. W: J. Mikyška, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, i P. M. Sloot, red., *Computational Science — ICCS 2023. 23rd International Conference, Prague, Czech Republic, July 3–5, 2023, Proceedings, Part I*, tom 14073 serii *Lecture Notes in Computer Science*, str. 191–205, Cham, Szwajcaria. Springer Nature Switzerland. 10

Krasnowska-Kieraś, K. i Woliński, M. (2024). Parsing headed constituencies. W: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, i N. Xue, red., *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, str. 12633–12643, Turyn, Włochy. ELRA and ICCL. 10

Kulmizev, A., de Lhoneux, M., Gontrum, J., Fano, E. i Nivre, J. (2019). Deep contextualized word embeddings in transition-based and graph-based dependency parsing - a tale of two parsers revisited. W: K. Inui, J. Jiang, V. Ng, i X. Wan, red., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, str. 2755–2768, Hong Kong, Chiny. Association for Computational Linguistics. 47

Kupść, A. (2001). *An HPSG Grammar of Polish Clitics*. Rozprawa doktorska, Instytut Podstaw Informatyki PAN i Université Paris 7, Warszawa. 51

Kübler, S., McDonald, R. i Nivre, J., red. (2009). *Dependency Parsing*. Nr 13 serii Synthesis Lectures on Human Language Technologies. Springer Nature Switzerland, Cham, Szwajcaria. 21

Lafferty, J., Sleator, D. i Temperley, D. (1992). Grammatical trigrams: A probabilistic model of link grammar. W: *Proceedings of the 1992 AAAI Fall Symposium*. 45

Marcińczuk, M., Kocoń, J. i Oleksy, M. (2017). Liner2 — a generic framework for named entity recognition. W: *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*, str. 86–91. Association for Computational Linguistics. 110

Marciniak, M. (2001). *Algorytmy implementacyjne syntaktycznych reguł koreferencji zaimków dla języka polskiego w terminach HPSG*. Rozprawa doktorska, Instytut Podstaw Informatyki PAN, Warszawa. 51

Marciniak, M., Kieraś, W., Bojałkowska, K., Borkowski, P., Borys, M., Eźlakowski, W., Guz, W., Kobyliński, L., Komosińska, D., Krasnowska-Kieraś, K., Łaziński, M., Miernecka, M., Nitoń, B., Ogrodniczuk, M., Rudolf, M., Tomaszewska, A., Woliński, M., Wołoszyn, J., Wójtowicz, B., Wróblewska, A. i Zawadzka-Paluektau,

N. (2023). Korpus Współczesnego Języka Polskiego. Instytut Podstaw Informatyki PAN, Warszawa. `https://kwjp.pl`. 23

Marcińczuk, M., Kocoń, J. i Gawor, M. (2018). Recognition of named entities for polish — comparison of deep learning and conditional random fields approaches. W: M. Ogrodniczuk i L. Kobyliński, red., *Proceedings of the PolEval 2018 Workshop*, str. 77–92, Warszawa. Instytut Podstaw Informatyki PAN. 110

Marcus, M. P., Santorini, B. i Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2), 313–330. 44

McDonald, R. i Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. W: D. McCarthy i S. Wintner, red., *11th Conference of the European Chapter of the Association for Computational Linguistics*, str. 81–88, Trydent, Włochy. Association for Computational Linguistics. 48, 49

McDonald, R., Pereira, F., Ribarov, K. i Hajič, J. (2005a). Non-projective dependency parsing using spanning tree algorithms. W: R. Mooney, C. Brew, L.-F. Chien, i K. Kirchhoff, red., *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, str. 523–530, Vancouver, Kanada. Association for Computational Linguistics. 48

McDonald, R., Crammer, K. i Pereira, F. (2005b). Online large-margin training of dependency parsers. W: K. Knight, H. T. Ng, i K. Oflazer, red., *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, str. 91–98, Ann Arbor, USA. Association for Computational Linguistics. 48

McDonald, R., Lerman, K. i Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. W: L. Màrquez i D. Klein, red., *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, str. 216–220, Nowy Jork, USA. Association for Computational Linguistics. 48, 49

Mel'čuk, I. A. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press, Albany, USA. 45

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. i Dean, J. (2013). Distributed representations of words and phrases and their compositionality. W: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, i K. Weinberger, red., *Advances in Neural Information Processing Systems*, tom 26. Curran Associates Inc. 43

Mroczkowski, R., Rybak, P., Wróblewska, A. i Gawlik, I. (2021). HerBERT: Efficiently pretrained transformer-based language model for Polish. W: *Proceedings of the 8th Workshop on Balto-Slavic Natural Language Processing*, str. 1–10, Kijów, Ukraina. Association for Computational Linguistics. 85

Mykowiecka, A. (1999). *Opis składniowy polskich konstrukcji względnych w formalizmie HPSG*. Rozprawa doktorska, Instytut Podstaw Informatyki PAN, Warszawa. 51

Müller, S., Abeillé, A., Borsley, R. D. i Koenig, J.-P., red. (2024). *Head-Driven Phrase Structure Grammar*. Nr 9 serii Empirically Oriented Theoretical Morphology and Syntax. Language Science Press, Berlin, Niemcy, wydanie drugie. 42

Naur, P., Backus, J. W., Bauer, F. L., Green, J., Katz, C., McCarthy, J., Perlis, A. J., Rutishauser, H., Samelson, K., Vauquois, B., Wegstein, J. H., van Wijngaarden, A. i Woodger, M. (1960). Report on the algorithmic language ALGOL 60. *Communications of the ACM*, 3(5), 299–311. 39

Nguyen, M. V., Lai, V. D., Pouran Ben Veyseh, A. i Nguyen, T. H. (2021). Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. W: D. Gkatzia i D. Seddah, red., *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, str. 80–90. Association for Computational Linguistics. 49, 70, 104

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. W: *Proceedings of the Eighth International Conference on Parsing Technologies*, str. 149–160, Nancy, Francja. 47

Nivre, J. i Nilsson, J. (2005). Pseudo-projective dependency parsing. W: K. Knight, H. T. Ng, i K. Oflazer, red., *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, str. 99–106, Ann Arbor, USA. Association for Computational Linguistics. 47

Nivre, J., Hall, J. i Nilsson, J. (2006). MaltParser: A data-driven parser-generator for dependency parsing. W: N. Calzolari, K. Choukri, A. Gangemi, B. Maegaard, J. Mariani, J. Odijk, i D. Tapias, red., *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genua, Włochy. European Language Resources Association (ELRA). 47

Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F. i Zeman, D. (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. W: N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, i S. Odijk, Jan andc Piperidis, red., *Proceedings of the Twelfth Language Resources and Evaluation Conference*, str. 4034–4043, Marsylia, Francja. European Language Resources Association. 32

Obrębski, T. (2002). *Automatyczna analiza składniowa języka polskiego z wykorzystaniem gramatyki zależnościowej*. Rozprawa doktorska, Instytut Podstaw Informatyki PAN, Warszawa. 51

Obrębski, T. (2003). MTT-compatible computationally effective surface-syntactic parser. W: *Proceedings of First International Conference on Meaning-Text Theory*, str. 259–268. 51

Ogrodniczuk, M. (2005). An extension of Świdziński's grammar of Polish. *Archives of Control Sciences*, 15(3), 393–402. 50

Ogrodniczuk, M. (2006). *Weryfikacja korpusu wypowiedników polskich (z wykorzystaniem gramatyki formalnej Świdzińskiego)*. Rozprawa doktorska, Wydział Neofilologii Uniwersytetu Warszawskiego, Warszawa. 50

OpenAI (2024). Gpt-4 technical report. preprint arXiv:2303.08774. 60

Osgood, C. E., Suci, G. J. i Tannenbaum, P. H. (1957). *The Measurement of Meaning*. University of Illinois Press. 42

Patejuk, A. (2015). *Unlike Coordination in Polish: An LFG Account*. Rozprawa doktorska, Instytut języka Polskiego PAN, Kraków. 51

Patejuk, A. (2016). Integrating a rich external valency dictionary with an implemented XLE/LFG grammar. W: D. Arnold, M. Butt, B. Crysmann, T. H. King, i S. Müller, red., *The Proceedings of the Joint 2016 Conference on Head-driven Phrase Structure Grammar and Lexical Functional Grammar*, str. 520–540, Stanford, USA. CSLI Publications. 51

Patejuk, A. i Przepiórkowski, A. (2012). Towards an LFG parser for Polish: An exercise in parasitic grammar development. W: *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, str. 3849–3852, Stambuł, Turcja. European Language Resources Association (ELRA). 51

Patejuk, A. i Przepiórkowski, A. (2014). Synergistic development of grammatical resources: A valence dictionary, an LFG grammar, and an LFG structure bank for Polish. W: V. Henrich, E. Hinrichs, D. de Kok, P. Osenova, i A. Przepiórkowski, red., *Proceedings of the Thirteenth International Workshop on Treebanks and Linguistic Theories (TLT 13)*, str. 113–126, Tybinga, Niemcy. Department of Linguistics (SfS), University of Tübingen. 51

Patejuk, A. i Przepiórkowski, A. (2017). POLFIE: współczesna gramatyka formalna języka polskiego. *Język Polski*, XCVII(1), 48–64. 51

Patejuk, A. i Przepiórkowski, A. (2018). *From Lexical Functional Grammar to Enhanced Universal Dependencies: Linguistically Informed Treebanks of Polish*. Instytut Podstaw Informatyki PAN, Warsaw. 32, 51

Pennington, J., Socher, R. i Manning, C. (2014). GloVe: Global vectors for word representation. W: A. Moschitti, B. Pang, i W. Daelemans, red., *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, str. 1532–1543, Doha, Katar. Association for Computational Linguistics. 43

Pereira, F. C. i Warren, D. H. (1980). Definite clause grammars for language analysis — a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3), 231–278. 41

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. i Zettlemoyer, L. (2018). Deep contextualized word representations. W: M. Walker, H. Ji, i A. Stent, red., *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, str. 2227–2237, Nowy Orlean, USA. Association for Computational Linguistics. 43

Piasecki, M. (2007). Polish tagger TaKIPI: rule based construction and optimisation. *TASK Quarterly*, 11(1/2), 151–167. 81

Pollard, C. i Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, USA. 42

Przepiórkowski, A. (1999). *Case Assignment and the Complement-Adjunct Dichotomy: A Non-Configurational Constraint-Based Approach*. Rozprawa doktorska, Universität Tübingen, Tybinga, Niemcy. 51

Przepiórkowski, A. (2008). *Powierzchniowe przetwarzanie języka polskiego*. Akademicka Oficyna Wydawnicza EXIT, Warszawa. 52

Przepiórkowski, A. i Patejuk, A. (2020). From Lexical Functional Grammar to enhanced Universal Dependencies: The UD-LFG treebank of Polish. *Language Resources and Evaluation*, 54, 185–221. 51

Przepiórkowski, A., Kupść, A., Marciniak, M. i Mykowiecka, A. (2002). *Formalny opis języka polskiego: Teoria i implementacja*. Akademicka Oficyna Wydawnicza EXIT, Warszawa. 51

Przepiórkowski, A., Bańko, M., Górski, R. L. i Lewandowska-Tomaszczyk, B., red. (2012). *Narodowy Korpus Języka Polskiego*. Wydawnictwo Naukowe PWN, Warszawa. 18, 27, 135, 145

Przepiórkowski, A., Hajnicz, E., Andrzejczuk, A., Patejuk, A. i Woliński, M. (2017). Walenty: gruntowny składniowo-semantyczny słownik walencyjny języka polskiego. *Język Polski*, XCVII(1), 30–47. 50

Qi, P., Zhang, Y., Zhang, Y., Bolton, J. i Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. W: A. Celikyilmaz i T.-H. Wen, red., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, str. 101–108, Online. Association for Computational Linguistics. 49, 104

Radziszewski, A. (2013). A tiered CRF tagger for Polish. W: *Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions*, str. 215–230. Springer Berlin Heidelberg, Berlin, Heidelberg, Niemcy. 81, 104

Radziszewski, A. i Acedański, S. (2012). Taggers gonna tag: An argument against evaluating disambiguation capacities of morphosyntactic taggers. W: P. Sojka, A. Horák, I. Kopeček, i K. Pala, red., *Text, Speech and Dialogue*, str. 81–87, Berlin, Heidelberg, Niemcy. Springer Berlin Heidelberg. 81

Radziszewski, A. i Śniatowski, T. (2011). A memory-based tagger for Polish. W: *Proceedings of the 5th Language & Technology Conference*. 81, 104

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. i Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140), 1–67. 111

Resnik, P. (1992). Probabilistic Tree-Adjoining Grammar as a framework for statistical natural language processing. W: *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, str. 418–424, Nantes, Francja. 41

Rybak, P. i Wróblewska, A. (2018a). Semi-supervised neural system for tagging, parsing and lematization. W: D. Zeman i J. Hajič, red., *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, str. 45–54. Association for Computational Linguistics. 49

Rybak, P. i Wróblewska, A. (2018b). Semi-supervised neural system for tagging, parsing and lemmatization. Addendum. W: M. Ogrodniczuk i L. Kobyliński, red., *Proceedings of the PolEval 2018 Workshop*, str. 49–51, Warszawa. Instytut Podstaw Informatyki PAN. 49

Rychlikowski, P., Zapotoczny, M. i Chorowski, J. (2017). Character-based neural POS tagger. W: Z. Vetulani i P. Paroubek, red., *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, str. 383–385, Poznań. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu. 104

Salomaa, A. (1969). Probabilistic and weighted grammars. *Information and Control*, 15(6), 529–544. 40

Saloni, Z. i Świdziński, M. (2001). *Składnia współczesnego języka polskiego*. Wydawnictwo Naukowe PWN, Warszawa, wydanie czwarte. 13

Saloni, Z., Woliński, M., Wołosz, R., Gruszczyński, W. i Skowrońska, D. (2015). *Słownik gramatyczny języka polskiego*. Warszawa, wydanie trzecie. 69, 80

Savary, A., Chojnacka-Kuraś, M., Wesołek, A., Skowrońska, D. i Śliwiński, P. (2012). Anotacja jednostek nazewniczych. W: Przepiórkowski *et al.* (2012), chapter 9, str. 129–159. 74

Schabes, Y. (1992). Stochastic lexicalized tree-adjoining grammars. W: *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*, str. 426–433. 41

Schabes, Y., Abeillé, A. i Joshi, A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. W: *Coling Budapest 1988 Volume 2: International Conference on Computational Linguistics*, str. 578–583. 41

Seddah, D., Tsarfaty, R., Kübler, S., Candito, M., Choi, J. D., Farkas, R., Foster, J., Goenaga, I., Gojenola Galletebeitia, K., Goldberg, Y., Green, S., Habash, N.,

Kuhlmann, M., Maier, W., Nivre, J., Przepiórkowski, A., Roth, R., Seeker, W., Versley, Y., Vincze, V., Woliński, M., Wróblewska, A. i Villemonte de la Clergerie, E. (2013). Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. W: *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, str. 146–182, Seattle, USA. Association for Computational Linguistics. 96

Sgall, P., Hajicová, E. i Panevová, J. (1986). *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. Springer, Dordrecht, Holandia. 45

Sleator, D. i Temperley, D. (1993). Parsing English with a link grammar. W: H. Bunt, R. Berwick, K. Church, A. Joshi, R. Kaplan, M. Kay, B. Lang, M. Nagao, A. Nijholt, M. Steedman, H. Thompson, M. Tomita, K. Vijay-Shanker, Y. Wilks, i K. Wittenburg, red., *Proceedings of the Third International Workshop on Parsing Technologies*, str. 277–292, Tilburg, Holandia i Durbuy, Belgia. Association for Computational Linguistics. 45

Socher, R., Bauer, J., Manning, C. D. i Ng, A. Y. (2013). Parsing with compositional vector grammars. W: H. Schuetze, P. Fung, i M. Poesio, red., *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), str. 455–465, Sofia, Bułgaria. Association for Computational Linguistics. 43

Stanojević, M. i Cohen, S. B. (2021). A root of a problem: Optimizing single-root dependency parsing. W: M.-F. Moens, X. Huang, L. Specia, i S. W.-t. Yih, red., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, str. 10540–10557, On-line/Punta Cana, Dominikana. Association for Computational Linguistics. 62

Stern, M., Andreas, J. i Klein, D. (2017). A minimal span-based neural constituency parser. W: R. Barzilay i M.-Y. Kan, red., *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (*Volume 1: Long Papers*), str. 818–827, Vancouver, Kanada. Association for Computational Linguistics. 43

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2), 165–201. 40

Straka, M. (2018). UDPipe 2.0 prototype at CoNLL 2018 UD shared task. W: D. Zeman i J. Hajič, red., *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, str. 197–207, Bruksela, Belgia. Association for Computational Linguistics. 49

Straka, M., Straková, J. i Hajic, J. (2019). UDPipe at SIGMORPHON 2019: Contextualized embeddings, regularization with morphological categories, corpora merging. W: G. Nicolai i R. Cotterell, red., *Proceedings of the 16th Workshop on Computational*

Research in Phonetics, Phonology, and Morphology, str. 95–103, Florencja, Włochy. Association for Computational Linguistics. 49, 104

Strzyz, M., Vilares, D. i Gómez-Rodríguez, C. (2019). Sequence labeling parsing by learning across representations. W: A. Korhonen, D. Traum, i L. Màrquez, red., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, str. 5350–5357, Florencja, Włochy. Association for Computational Linguistics. 50

Świdziński, M. (1989). A dependency syntax of Polish. *Metataxis in practice. Dependency syntax for multilingual machine translation*, str. 69–88. 51

Świdziński, M. (1992). *Gramatyka formalna języka polskiego*. Rozprawy Uniwersytetu Warszawskiego, Warszawa. 50

Świdziński, M. i Woliński, M. (2010). Towards a bank of constituent parse trees for Polish. W: P. Sojka, A. Horák, I. Kopeček, i K. Pala, red., *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic*, nr 6231 serii Lecture Notes in Artificial Intelligence, str. 197–204, Heidelberg, Niemcy. Springer-Verlag. 27

Szpakowicz, S. (1978). *Automatyczna analiza składniowa polskich zdań pisanych*. Rozprawa doktorska, Instytut Informatyki UW, Warszawa. 50

Szpakowicz, S. (1983). *Formalny opis składniowy zdań polskich*. Wydawnictwa Uniwersytetu Warszawskiego, Warszawa. 50

Tarjan, R. E. (1977). Finding optimum branchings. *Networks*, 7(1), 25–35. 62

Tesnière, L. (1959). *Éléments de syntaxe structurale*. Klincksieck, Paryż, Francja. 45

Tian, Y., Xia, F. i Song, Y. (2024). Large language models are no longer shallow parsers. W: L.-W. Ku, A. Martins, i V. Srikumar, red., *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, str. 7131–7142, Bangkok, Tajlandia. Association for Computational Linguistics. 60

Tomaszewska, A., Czerski, D., Żuk, B. i Ogrodniczuk, M. (2025). NeoN: A tool for automated detection, linguistic and LLM-driven analysis of neologisms in Polish. W: M. H. Lees, W. Cai, S. A. Cheong, Y. Su, D. Abramson, J. J. Dongarra, i P. M. A. Sloot, red., *Computational Science – ICCS 2025*, Lecture Notes in Computer Science, str. 318–326, Cham, Szwajcaria. Springer Nature Switzerland. 116

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. i Polosukhin, I. (2017). Attention is all you need. W: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, i R. Garnett, red., *Advances in Neural Information Processing Systems*, tom 30. Curran Associates, Inc. 43, 59, 127

Vetulani, Z. (2004). *Komunikacja człowieka z maszyną. Komputerowe modelowanie kompetencji językowej*. Akademicka Oficyna Wydawnicza EXIT, Warszawa. 50

Vetulani, Z., Marciniak, J., Obrębski, T., Vetulani, G., Dąbrowski, A., Kubis, M., Osiński, J., Walkowska, J., Kubacki, P. i Witalewski, K. (2010). *Zasoby językowe*

i technologie przetwarzania tekstu. POLINT-112-SMS jako przykład aplikacji z zakresu bezpieczeństwa publicznego. Wydawnictwo Naukowe UAM, Poznań. 50

Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I. i Hinton, G. (2015). Grammar as a foreign language. W: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, i R. Garnett, red., *Advances in Neural Information Processing Systems*, tom 28. Curran Associates, Inc. 43

Warner, B., Chaffin, A., Clavié, B., Weller, O., Hallström, O., Taghadouini, S., Gallagher, A., Biswas, R., Ladhak, F., Aarsen, T., Adams, G. T., Howard, J. i Poli, I. (2025). Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. W: W. Che, J. Nabende, E. Shutova, i M. T. Pilehvar, red., *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, str. 2526–2547, Wiedeń, Austria. Association for Computational Linguistics. 60

Waszczuk, J. (2012). Harnessing the CRF complexity with domain-specific constraints. the case of morphosyntactic tagging of a highly inflected language. W: M. Kay i C. Boitet, red., *Proceedings of COLING 2012*, str. 2789–2804, Bombaj, Indie. The COLING 2012 Organizing Committee. 81, 104

Wawer, A. i Małek, E. (2018). Results of the PolEval 2018 Shared Shared Task 2: Named Entity Recognition. W: *Proceedings of the PolEval 2018 Workshop*, str. 53–62, Warszawa. Instytut Podstaw Informatyki PAN. 109

Wieczorek, A. (2022). Zastosowanie parsera zależnościowego do analizy składniowej tekstów średniopolskich. Prezentacja na konferencji „Język ojczysty w XXI wieku – system, edukacja, perspektywy", Kielce. 94

Wieczorek, A. (2023). Discontinuous noun phrases containing adjective or adjective-like modifiers in Middle Polish texts. Preliminary research conducted on an experimental dependency treebank. Prezentacja na konferencji „26th International Conference on Historical Linguistics", Heidelberg, Niemcy. 116

Wiącek, M., Rybak, P., Pszenny, L. i Wróblewska, A. (2024). NLPre: A revised approach towards language-centric benchmarking of natural language preprocessing systems. W: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, i N. Xue, red., *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, str. 12271–12287, Turyn, Włochy. ELRA and ICCL. 93

Woliński, M. (2004). *Komputerowa weryfikacja gramatyki Świdzińskiego*. Rozprawa doktorska, Instytut Podstaw Informatyki PAN, Warszawa. 50

Woliński, M. (2005). An efficient implementation of a large grammar of Polish. *Archives of Control Sciences*, 15(3), 251–258. 50

Woliński, M. (2015). Deploying the new valency dictionary Walenty in a DCG parser of Polish. W: M. Dickinson, E. Hinrichs, A. Patejuk, i A. Przepiórkowski, red., *Proceedings of the Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*, str. 221–229, Warszawa. Instytut Podstaw Informatyki PAN. 50

Woliński, M. (2019). *Automatyczna analiza składnikowa języka polskiego*. Wydawnictwa Uniwersytetu Warszawskiego, Warszawa. 27, 50, 97

Woliński, M. i Hajnicz, E. (2021). Składnica: a constituency treebank of Polish harmonised with the Walenty valency dictionary. *Language Resources and Evaluation*, 55, 209–239. 27

Woliński, M. i Rogozińska, D. (2013). First experiments in PCFG-like disambiguation of constituency parse forests for Polish. W: Z. Vetulani, red., *Proceedings of the 6th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, str. 343–347, Poznań. Wydawnictwo Poznańskie, Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu. 97

Woliński, M. i Rogozińska, D. (2016). Experiments in PCFG-like disambiguation of constituency parse forests for Polish. W: Z. Vetulani, H. Uszkoreit, i M. Kubis, red., *Human Language Technology. Challenges for Computer Science and Linguistics: 6th Language and Technology Conference, LTC 2013. Revised Selected Papers*, nr 9561 serii Lecture Notes in Artificial Intelligence, str. 146–158, Cham, Szwajcaria. Springer International Publishing. 50, 97

Woliński, M., Hajnicz, E. i Bartosiak, T. (2018). A new version of the Składnica treebank of Polish harmonised with the Walenty valency dictionary. W: N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis, i T. Tokunaga, red., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, str. 1839–1844, Paryż, Francja. European Language Resources Association (ELRA). 27

Wróbel, K. (2017). KRRNT: Polish recurrent neural network tagger. W: Z. Vetulani i P. Paroubek, red., *Proceedings of the 8th Language & Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*, str. 386–391, Poznań. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu. 104

Wróblewska, A. (2012). Polish Dependency Bank. *Linguistic Issues in Language Technology*, 7(1). 51

Wróblewska, A. (2014). *Polish Dependency Parser Trained on an Automatically Induced Dependency Bank*. Rozprawa doktorska, Instytut Podstaw Informatyki PAN, Warszawa. 31, 51

Wróblewska, A. (2018). Extended and enhanced Polish dependency bank in Universal Dependencies format. W: M.-C. de Marneffe, T. Lynn, i S. Schuster,

red., *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, str. 173–182. Association for Computational Linguistics. 31, 32, 51

Wróblewska, A. i Woliński, M. (2012). Preliminary experiments in Polish dependency parsing. W: P. Bouvry, M. A. Kłopotek, F. Leprevost, M. Marciniak, A. Mykowiecka, i H. Rybiński, red., *Security and Intelligent Information Systems: International Joint Conference, SIIS 2011, Warsaw, Poland, June 13-14, 2011, Revised Selected Papers*, nr 7053 serii Lecture Notes in Computer Science, str. 279–292. Springer-Verlag. 31, 51

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Łukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. i Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. preprint arXiv:1609.08144. 67

Yamada, H. i Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. W: *Proceedings of the Eighth International Conference on Parsing Technologies*, str. 195–206, Nancy, Francja. 47

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2), 189–208. 40

Zhang, X., Cheng, J. i Lapata, M. (2017). Dependency parsing as head selection. W: M. Lapata, P. Blunsom, i A. Koller, red., *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, str. 665–676, Walencja, Hiszpania. Association for Computational Linguistics. 49

Zhou, J. i Zhao, H. (2019). Head-Driven Phrase Structure Grammar parsing on Penn Treebank. W: A. Korhonen, D. Traum, i L. Màrquez, red., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, str. 2396–2408, Florencja, Włochy. Association for Computational Linguistics. 50

Zhou, J., Li, Z. i Zhao, H. (2020). Parsing all: Syntax and semantics, dependencies and spans. W: *Findings of the Association for Computational Linguistics: EMNLP 2020*, str. 4438–4449. Association for Computational Linguistics. 50

Zmigrod, R., Vieira, T. i Cotterell, R. (2020). Please mind the root: Decoding arborescences for dependency parsing. W: B. Webber, T. Cohn, Y. He, i Y. Liu, red., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, str. 4809–4819, On-line. Association for Computational Linguistics. 64