University of Warsaw
Faculty of Psychology

Kamil Tomaszek
Record book number: 432044

Middle Polish Dependency Treebank in Universal Dependencies Format: Design, Implementation, and Analysis

Master's thesis
in COGNITIVE SCIENCE

The thesis was written under the supervision of
Dr. Alina Wróblewska
Institute of Computer Science
Polish Academy of Sciences
Dr. Grzegorz Krajewski
University of Warsaw

Warsaw, November 2025

Summary

This thesis presents a rule-based approach to converting the Middle Polish Dependency Treebank (MPDT), annotated in a Polish-specific scheme, into the Universal Dependencies (UD) format. After introducing the project motivation, data sources, and target standard, the thesis outlines general design assumptions behind the conversion, the mapping strategy, and the validation workflow. It reports overall outcomes of the conversion and sketches applications and extensions, including releasing MPDT-UD and implications for research in historical language processing within cognitive science.

Keywords

Middle Polish, dependency trees, treebank conversion, Universal Dependencies

The title of the thesis in Polish

Średniopolski Bank Drzew Zależnościowych w formacie Universal Dependencies: projekt, implementacja i analiza

Streszczenie

Praca przedstawia podejście regułowe do konwersji Średniopolskiego Banku Drzew Zależnościowych (MPDT), anotowanego w polskim schemacie, do formatu Universal Dependencies (UD). Po krótkim omówieniu motywacji, danych i standardu docelowego zaprezentowano ogólne założenia projektu, strategię odwzorowań oraz schemat walidacji. Przedstawiono ogólne wyniki konwersji oraz możliwe zastosowania i kierunki rozwoju, w tym udostępnienie MPDT-UD i znaczenie dla badań nad przetwarzaniem języka historycznego w kognitywistyce.

Słowa kluczowe

język średniopolski, drzewa zależnościowe, konwersja korpusu, Universal Dependencies

The title of the thesis in English

Middle Polish Dependency Treebank in Universal Dependencies Format: Design, Implementation, and Analysis

Contents

Chapter 1

Introduction

1.1. Motivation

Natural-language preprocessing tools and comparative treebank research have standardized around Universal Dependencies (UD), which enables typologically informed analyses and cross-lingual transfer (Nivre et al. 2020). For Polish texts from the 17th and 18th centuries, however, key resources remain outside UD: texts in the KorBa corpus (Gruszczyński et al. 2022) and the emerging Middle Polish Dependency Treebank (MPDT) are annotated in a Polish-specific scheme (Wieczorek 2025). KorBa is a corpus of historical Polish texts, while MPDT adds a syntactic dependency layer to selected portions of this corpus. However, these resources being annotated in a different format creates challenges for interoperability with UD-based tools and limits straightforward comparative studies with other languages.

A natural solution is to convert these resources to the UD format. From an engineering perspective, however, a faithful, auditable conversion is non-trivial: historical orthography, abbreviations, clitic mobility, numeral complexes, and multiword conjunctions/prepositions interact with head rules and label inventories. Prior conversion experience for contemporary Polish offers valuable guidance (Wróblewska 2018; Wróblewska 2020), yet historical data introduce additional phenomena that require explicit, rule-based handling and transparent traceability.

As Wieczorek (2025) notes, MPDT's current format is well-suited to comparative studies with contemporary Polish syntax; at the same time, she highlights the advantages of moving to UD for cross-linguistic comparability, wider intelligibility, and representational options such as enhanced dependencies for shared dependents and shared governors in coordination—even if some information may be lost in conversion.

This thesis operationalizes that rationale by delivering a documented, UD-oriented converter for MPDT and preparing the current version of MPDT-UD suitable for validation and downstream use. The intended users include historical linguists needing UD-compatible data and NLP practitioners interested in diachronic Polish or cross-lingual experiments.

1.2. Objectives

The thesis pursues the following research goals:

(R1) Design a UD-oriented conversion strategy for MPDT. Specify mapping principles that respect Middle Polish specifics while aligning with UD guidelines.

(R2) Implement an auditable conversion pipeline. Provide modular components for morphosyntax mapping and dependency restructuring, with token-level logging.

(R3) Ensure UD conformance and evaluability. Produce output that passes the official UD validator (on all levels) and supports downstream analysis.

(R4) Document decisions. Record non-obvious mapping choices and edge-case policies to enable maintenance and reuse.

1.3. Contributions

This project delivers concrete, reusable artifacts:

(C1) A rule-based MPDT → MPDT-UD converter. A modular pipeline with fine-grained logging, selectively adapting ideas from Wróblewska (2018) while targeting Middle Polish phenomena. The code will be released in a public repository under an open-source license, together with this thesis, which documents the design and implementation.

(C2) An initial public release of MPDT-UD. A set of MPDT (2018 sentences at the time of writing) converted automatically and validated with the official UD validator.

1.4. Structure of the Thesis

- **Chapter 2: Background.** Introduces dependency grammar and the Polish Dependency Bank (PDB) annotation scheme; outlines the Universal Dependencies (UD) framework, including its layers and relation inventories; and presents the key source resources—KorBa and MPDT—that the conversion operates on.

- **Chapter 3: Linguistic Features of Middle Polish.** Describes linguistic properties of Middle Polish relevant to conversion: orthography and punctuation practices; characteristic morphological categories (`adjb`, `ppasb`, `ppraet`, dual number); evolving masculine gender distinctions; the connective *jako* in its comparative and role uses; and syntactic features such as non-projectivity and predicate ellipsis.
- **Chapter 4: Conversion Design and Implementation.** Details the custom Python pipeline: its modular architecture, core `Sentence` and `Token` classes, and auditable logging system. It explains the two-phase process—(1) rule-based morphosyntactic mapping of POS tags and features, and (2) dependency restructuring—together with label mapping and post-processing that ensure full UD conformance.
- **Chapter 5: Validation and Outcomes.** Introduces the validation workflow based on the official UD validator, including the iterative procedure and validation dataset; reports the conformance results and treatment of remaining edge cases; and presents the final MPDT-UD 1.0 treebank, summarizing its size, data split, licensing, and integration into the UD ecosystem.

Chapter 2

Background

This chapter provides the essential background for understanding the Middle Polish Dependency Treebank conversion to Universal Dependencies. It begins with the theoretical foundations of dependency grammar and its specific Polish manifestation in the Polish Dependency Bank (PDB) scheme (Section 2.1). Then it outlines Universal Dependencies as the target framework, highlighting its advantages for cross-linguistic research (Section 2.2). Finally, it describes the key resources: KorBa as the source corpus and MPDT as the dependency-annotated dataset that forms the input to our conversion pipeline (Section 2.3).

2.1. Dependency Grammar

Dependency grammar is a theory of syntactic structure organized around asymmetric governor–dependent relations. A *dependency* links two lexical items: a *governor* that selects and constrains a dependent, and a *dependent* that is licensed by the governor. One item can be a governor for multiple dependents, but each dependent has a single governor. Sentence structures are modeled as directed trees whose nodes correspond to tokens and whose edges encode these governor–dependent links. The tree has a single *root* (a node with no governor), and every other node is reachable from it along directed edges. In addition to purely structural links, dependency grammar is used here in a morphosyntactic sense, focusing on grammatical relations rather than semantic or prosodic dependency representations.

The dependency scheme used in Middle Polish follows the conventions established for the Polish Dependency Bank (PDB), which is adapted specifically for Polish syntax (Wróblewska 2023). The PDB tagset adapts the NKJP tagset (Przepiórkowski et al. 2012). The PDB annotation scheme uses a comprehensive set of part-of-speech categories and dependency relations designed specifically for Polish morphosyntax.

The PDB tagset includes the following part-of-speech categories:

- **Nouns:** subst (noun), depr (depreciative noun)
- **Pronouns:** ppron12 (non-third person pronoun), ppron3 (third person pronoun), siebie (reflexive pronoun)
- **Adjectives:** adj (adjective), adja (ad-adjectival adjective), adjc (predicative adjective), adjp (prepositional adjective)
- **Verb forms:** fin (finite non-past), praet (past tense), imps (impersonal), impt (imperative), inf (infinitive), aglt (agglutinate of 'być'), bedzie (future form of 'być'), winien (modal verbs like 'winien'), pred (predicative), ger (gerund), pcon (contemporary adverbial participle), pant (anterior adverbial participle), pact (active adjectival participle), ppas (passive adjectival participle)
- **Numerals:** num (cardinal numeral), numcomp (numeral compound)
- **Conjunctions:** comp (subordinating conjunction), conj (coordinating conjunction)
- **Other categories:** adv (adverb), brev (abbreviation), dig (Arabic numeral), romandig (Roman numeral), emo (emoticon), fill (filler), frag (fragment), interj (interjection), interp (punctuation), part (particle), prep (preposition), ign (unrecognized form)

The PDB annotation scheme distinguishes several classes of dependency relations:

- **Core arguments:** subj (subject), obj (direct object), obj_th (thematic object), comp (complement), comp_fin (finite clause complement), comp_inf (open clause [*infinitive*] complement), comp_ag (agent complement)
- **Adjuncts and modifiers:** adjunct with semantic subtypes such as adjunct_temp (temporal), adjunct_loc (locative), adjunct_dur (duration), adjunct_caus (causal), adjunct_mod (manner), adjunct_emph (emphatic particle), adjunct_compar (comparative)
- **Predicate-related:** pd (predicative expression), aux (auxiliary), neg (negation), refl (reflexive)
- **Coordination:** conjunct (coordinated element), pre_coord (pre-coordinator)
- **Multiword expressions:** mwe (multiword expression), ne (named entity), ne_foreign (foreign named entity)

- **Special relations:** `punct` (punctuation), `vocative` (vocative), `orphan` (orphaned dependent), `discourse` (discourse marker), `parataxis` (parataxis), `aglt` (mobile inflection), `imp` (imperative marker), `cond` (conditional clitic), and `root` (sentence root)

The example dependency trees below illustrate the scheme of a PDB-annotated sentence alongside its UD counterpart, showing the structural differences.

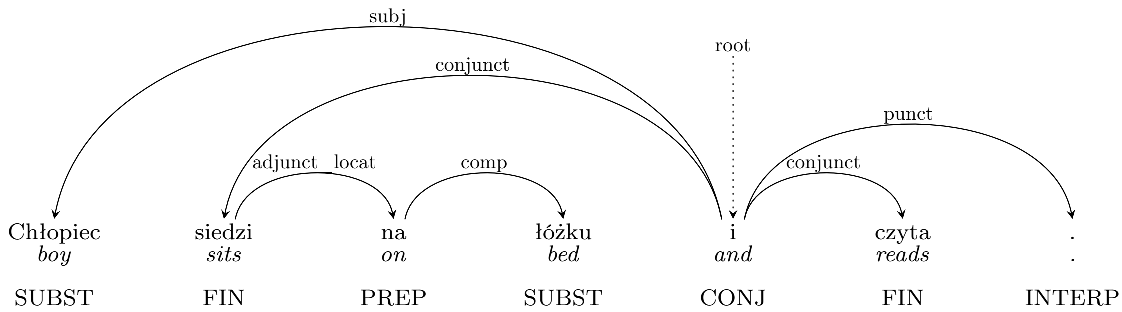

Figure 1: Example dependency tree in the PDB format

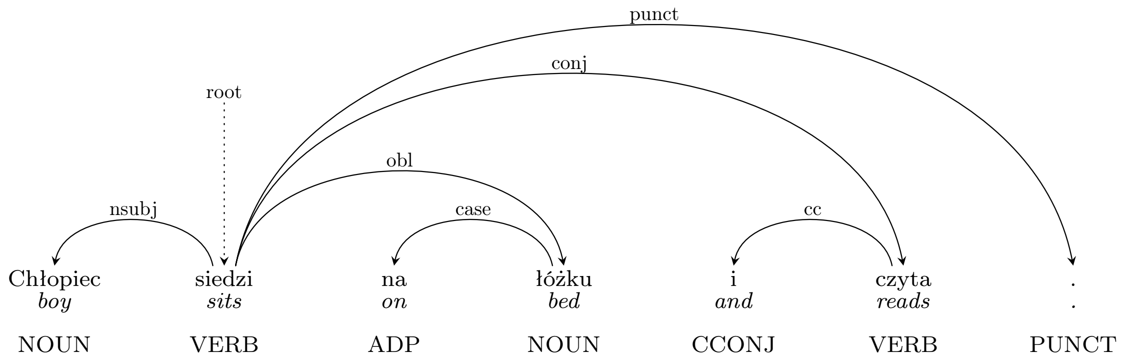

Figure 2: A dependency tree of the same sentence in UD format

Dependency formalisms differ on certain design choices (e.g., whether adpositions are heads or dependents inside adpositional phrases; how to encode coordination; whether and how to mark valency vs. modification). The PDB scheme takes specific positions on these issues, treating prepositions as heads (note the *on*→*bed* relation in Figure 1), using a coordination-centric approach where conjunctions govern coordinated elements (*i* being the `root` of both *siedzi* and *czyta* in Figure 1).

2.2. Universal Dependencies

Universal Dependencies (hereafter UD) is a cross-linguistic annotation framework designed to harmonize morphosyntactic and syntactic representations across languages within a dependency-based, lexicalist model (Nivre et al. 2020; de Marneffe et al. 2021). UD serves as both a theoretical framework and a practical collection of treebanks—currently the largest repository of over 200 treebanks for more than 150 languages.[1] It is widely adopted in NLP and linguistic typology studies, and is maintained by an open community with regular releases.

Annotation scheme: The scheme provides three aligned layers for sentence-level annotation:

1. **Tokenization.** UD defines dependencies between *syntactic words*. To handle orthographic contractions or clitic clusters, it uses *multiword tokens*, ensuring a faithful word-level analysis. A multiword token is a single orthographic unit that is split into multiple syntactic words, each receiving its own morphological analysis and syntactic function.

   For example, Middle Polish *kiedym* (‘when I’) is annotated as:

   ```
   14-15   kiedym    _       _     ...
   14      kiedy     kiedy   ADV   ...
   15      m         być     AUX   ...
   ```

   Here, the single orthographic token *kiedym* (ID 14-15) splits into two syntactic words: *kiedy* ‘when’ (ID 14) and *m* (mobile inflection form of ‘I am’, ID 15).

   Similarly, *jeszcześ* (‘still you are’) becomes:

   ```
   7-8   jeszcześ   _          _      ...
   7     jeszcze    jeszcze    PART   ...
   8     ś          być        AUX    ...
   ```

2. **Morphology.** Each syntactic word is associated with a `LEMMA`, a universal part-of-speech tag (hereafter part-of-speech tag=`POS`; universal part-of-speech tag=`UPOS`) from a fixed 17-tag set, and a bundle of `FEATS` (morphological features). The `UPOS` tags cover open-class words (adjectives `ADJ`, adverbs `ADV`, interjections `INTJ`, nouns `NOUN`, proper nouns `PROPN`, verbs `VERB`), closed-class words (adpositions `ADP`, auxiliary verbs `AUX`, coordinating conjunctions `CCONJ`, determiners `DET`, numerals `NUM`, pronouns `PRON`, particles `PART`, subordinating conjunctions `SCONJ`), and other categories (punctuation `PUNCT`, symbols `SYM`, other `X`). UD v2 standardized features and values across languages and clarified tag boundaries, e.g., extending auxiliary verbs to copulas and tense–aspect–mood particles while narrowing particles. The list of `UPOS` categories is available on the UD webpage.[2]

[1] Universal Dependencies, `https://universaldependencies.org`, accessed 2025-10-10.
[2] Universal Dependencies POS tags: `https://universaldependencies.org/u/pos/index.html`

3. **Syntax.** The syntactic layer is a single-rooted tree with possible 37 universal dependency relations organized according to functional and structural categories. Sentence structures are modeled as directed trees according to the principles of dependency grammar as described in 2.1. Relations include:

    - core arguments (nominal subject `nsubj`, direct object `obj`, indirect object `iobj`, clausal subject `csubj`, clausal complement `ccomp`, open clausal complement `xcomp`),
    - non-core dependents (oblique `obl`, dislocated element `dislocated`, adverbial clause modifier `advcl`, adverbial modifier `advmod`, discourse element `discourse`, auxiliary `aux`, copula `cop`, vocative `vocative`, expletive `expl`, marker `mark`),
    - nominal dependents (nominal modifier `nmod`, numeral modifier `nummod`, adjectival modifier `amod`, determiner `det`, case marker `case`, classifier `clf`, clausal modifier of noun `acl`, appositional modifier `appos`),
    - coordination (conjunct `conj`, coordinating conjunction `cc`),
    - multiword expressions (fixed `fixed`, flat `flat`),
    - special relations (list element `list`, parataxis `parataxis`, orphan `orphan`, punct `punct`, root `root`, overridden disfluency `reparandum`, relation 'goes with' `goeswith`, other dependent `dep`).

    The framework also allows language-specific subtypes (e.g., `nsubj:pass` for passive subjects, `det:poss` for possessive determiners) and defines semi-mandatory subtypes that should be used when the relevant phenomenon exists in the language. A full list of relations and subtypes, along with their descriptions, is available in the UD webpage.[3]

In addition to the *basic* representation, UD also defines an *enhanced* graph that adds extra arcs (and occasionally null nodes) to capture phenomena such as shared dependents in coordination, control and raising, relativization, and ellipsis. In Figure 2, the basic tree structure is shown; an enhanced representation would add an additional edge to represent the dependent (in this case: the subject) of *czyta* ('reads') as also being the *Chłopiec* ('boy'), as shown in figure Figure 3.

[3]Universal Dependencies relations list: `https://universaldependencies.org/u/dep/index.html`

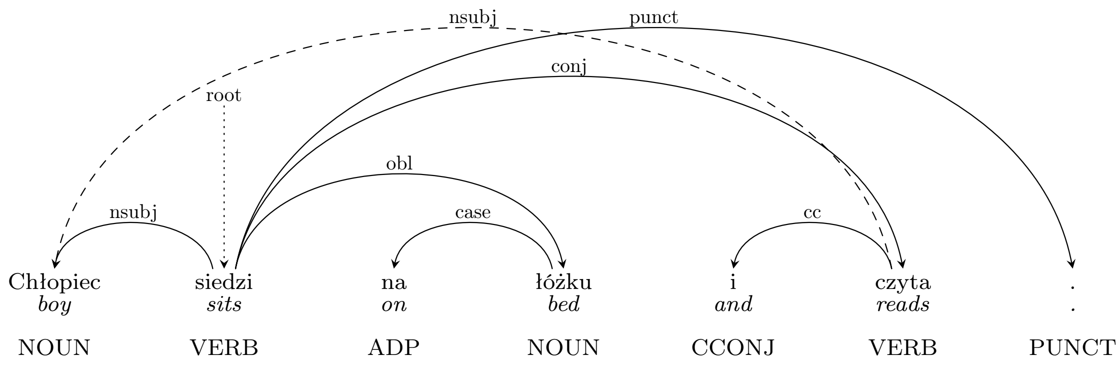

Figure 3: A dependency tree with enhanced dependencies (dashed lines)

Format: For practical implementation and data sharing, UD annotations must be encoded in a standardized format. UD uses the CoNLL-U format, a ten-column tabular specification with the fields:

- `ID` - a syntactic word index (or range for multiword tokens);
- `FORM` - the surface form;
- `LEMMA` - the dictionary form;
- `UPOS` - the universal POS tag;
- `XPOS` - a language-specific POS tag;
- `FEATS` - a pipe (|) separated list of morphological features;
- `HEAD` - the index of the head syntactic word (or 0 for the root);
- `DEPREL` - the dependency relation to the head;
- `DEPS` - for enhanced dependencies;
- `MISC` - for miscellaneous annotations.

Here is a CoNLL-U snippet for the sentence "Chłopiec siedzi na łóżku i czyta.", with the enhanced dependencies.

```
# sent_id = test-sentence
# text = Chłopiec siedzi na łóżku i czyta.
1   Chłopiec   chłopiec   NOUN    subst    Gender=Masc|Number=Sing|Case=Nom                           2   nsubj   _          _
2   siedzi     siedzieć   VERB    fin      Aspect=Imp|Mood=Ind|Tense=Pres|Person=3|Number=Sing        0   root    _          _
3   na         na         ADP     prep     AdpType=Prep|Case=Loc                                      4   case    _          _
4   łóżku      łóżko      NOUN    subst    Gender=Neut|Number=Sing|Case=Loc                           2   obl     _          _
5   i          i          CCONJ   conj     _                                                          2   cc      _          _
6   czyta      czytać     VERB    fin      Aspect=Imp|Mood=Ind|Tense=Pres|Person=3|Number=Sing        2   conj    1:nsubj    _
7   .          .          PUNCT   interp   PunctType=Peri                                             2   punct   _          _
```

2.3. Middle Polish Linguistic Resources

2.3.1. KorBa

KorBa (Gruszczyński et al. 2022) – from Polish *Korpus Barokowy* ('Baroque Corpus') – is a 13.5-million-token corpus of Polish texts from 1601–1772, compiled from over seven hundred sources and annotated morphosyntactically (lemmas, POS, features). It is searchable via MTAS (Multi Tier Annotation Search; Brouwer et al. 2017), and provides parallel transliteration/transcription layers, structural and language markup, and rich metadata (period, region, text type, genre) that enable stratified analyses.

The corpus includes diverse text types ranging from literary works (epic poetry, drama, lyric poetry) to non-literary materials (scientific-didactic texts, persuasive writings, factual literature, official documents, press releases) and biblical texts. Geographically, texts span the Polish-Lithuanian Commonwealth, with approximately 27% of the corpus being of unknown origin. As shown in Figures 4 and 5, the corpus maintains careful balance across regions and text types to ensure representativeness of Middle Polish.

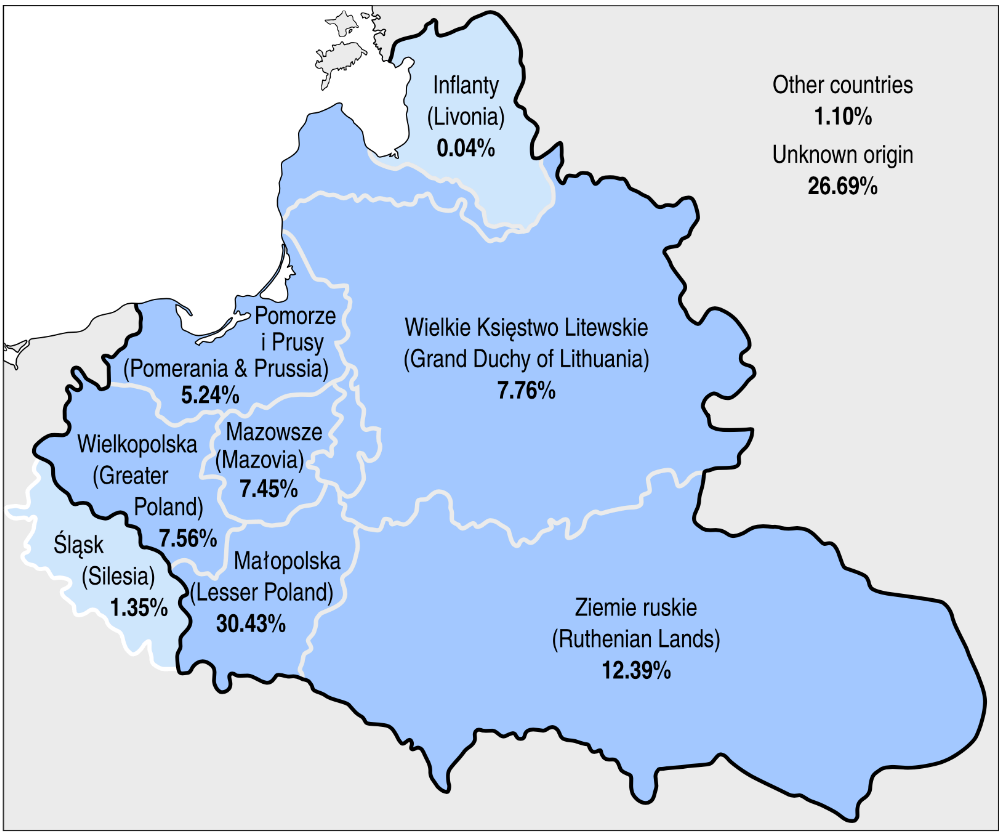

Figure 4: Geographical distribution of texts in the corpus displayed on the map of the Commonwealth after the Union of Lublin of 1569. Source: Gruszczyński et al. (2022), p. 315, CC BY 4.0.

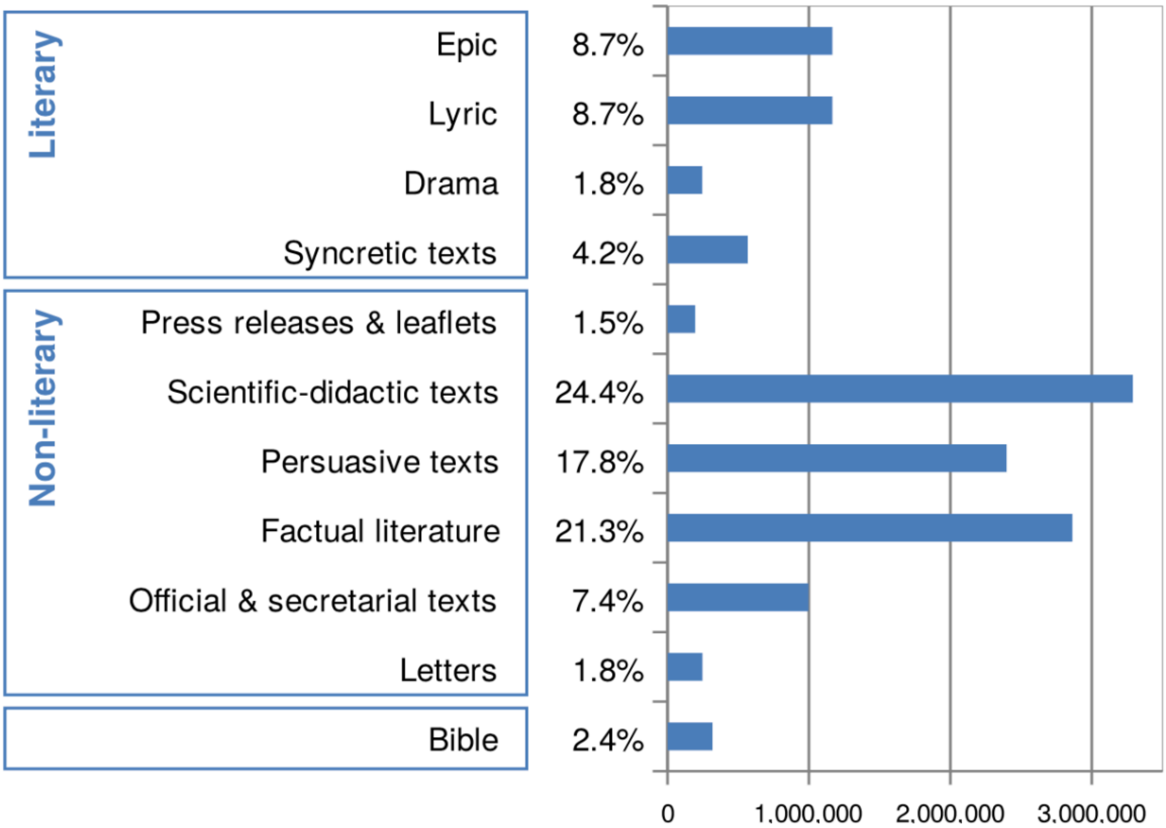

Figure 5: Types of texts in KorBa. Source: Gruszczyński et al. (2022), p. 316, CC BY 4.0.

2.3.2. MPDT

The Middle Polish Dependency Treebank (MPDT) is a manually curated, syntactically annotated subset of the KorBa corpus, capturing key syntactic phenomena of 17th–18th-century Polish texts. The sentences are from the manually annotated part of KorBa, whose careful pre-processing provides reliable morphosyntactic annotation and balanced coverage across genres and periods. In its current form, MPDT represents the first systematic attempt at syntactic annotation of Middle Polish and therefore in the current version excludes poetry and sentences with Latin insertions, while limiting sentence length to 10–50 tokens, with the average sentence length being 23 tokens (Wieczorek 2025).

The annotation workflow consists of the following steps:

1. **Automatic pre-annotation.** Two parsers trained on contemporary PDB data (MaltParser, COMBO) generate initial dependency analyses.

2. **Manual correction.** Two linguist annotators independently revise parser outputs, leveraging complementary error profiles.

3. **Adjudication.** Conflicting annotations are resolved by an adjudicator to produce a single gold-standard tree.

4. **Formatting.** Final annotations are encoded in CoNLL-X [4] with KorBa's extended tagset (e.g., dual number `Dual`).

Corpus statistics

- Total sentences: 2 018
- Total tokens: 47 273
- Distinct POS tags: 45
- Distinct dependency relations: 27
- Non-projective edges: 3 748 across 879 sentences
- Average sentence length: 23.43 tokens

Figure 6 presents the 20 most frequent MPDT POS tags, highlighting the prominence of nouns (`subst`: 11,374 occurrences), punctuation (`interp`: 7,971), adjectives (`adj`: 5,315), and prepositions (`prep`: 4,391).

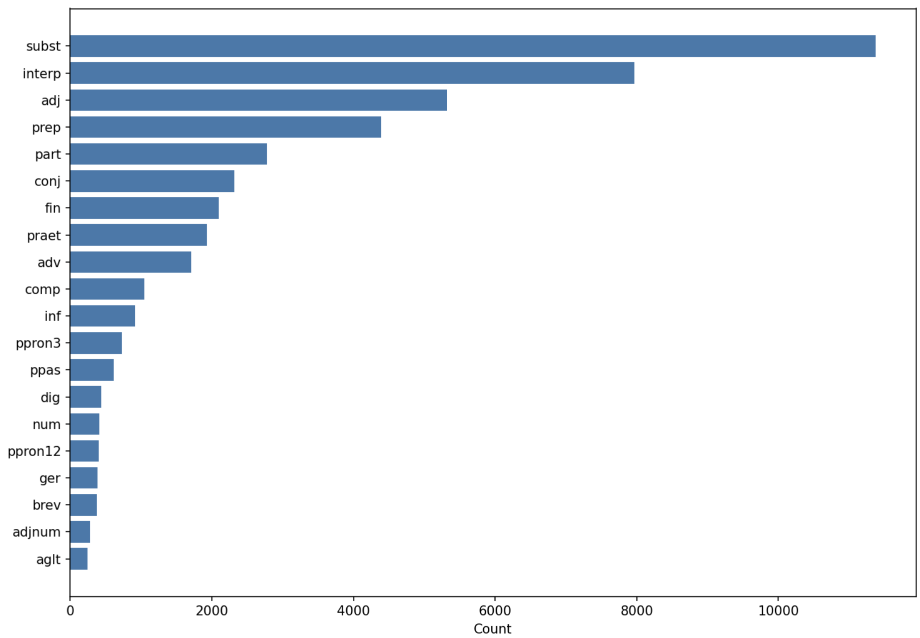

Figure 6: Top 20 MPDT POS tag frequencies

[4]CoNLL-X is the predecessor of the CoNLL-U format (Buchholz and Marsi 2006)

Figure 7 shows the distribution of the top 20 dependency relation types. Adjuncts (`adjunct`: 13,276) and complements (`comp`: 8,539) are most common, followed by punctuation (`punct`: 6,896), coordination elements (`conjunct`: 6,071), and core arguments (`obj`: 3,423; `subj`: 2,286).

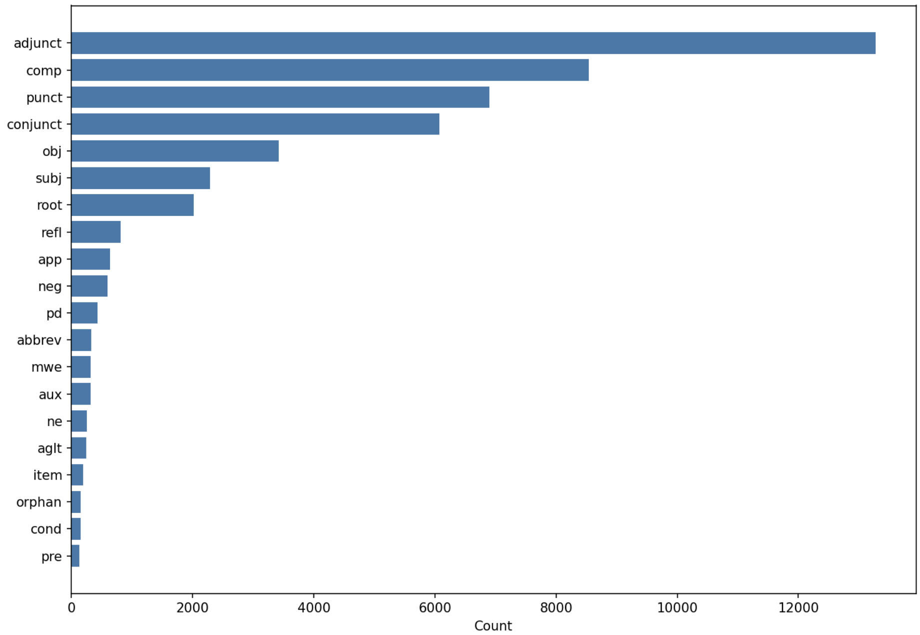

Figure 7: Top 20 MPDT dependency relation base frequencies

Chapter 3

Linguistic Features of Middle Polish

This chapter characterizes the linguistic system of Middle Polish as represented in the KorBa corpus and the Middle Polish Dependency Treebank (MPDT). It outlines key differences from modern Polish orthography and punctuation (Section 3.1), morphology (Section 3.2), and syntax (Section 3.3), emphasizing those that directly affect dependency annotation and conversion to Universal Dependencies (UD).

Note. Unless otherwise indicated, all Examples in this chapter and the chapters that follow (Chapters 3–6) are drawn from the Middle Polish Dependency Treebank (MPDT).

3.1. Orthography and Punctuation

3.1.1. Orthography and Transliteration

The KorBa corpus preserves two parallel orthographic layers: *transliteration* (a faithful rendering of the historical text) and *transcription* (a normalized spelling approximating contemporary Polish orthography). As noted in the KorBa manual, transliteration reflects the original graphic form of 17th–18th-century sources, while the transcription adapts them to modern conventions, keeping key phonetic and morphological features of Middle Polish.

Middle Polish orthography was far from standardized. The same word could appear in several spelling variants, sometimes even within a single text. Graphemes were often used interchangeably (e.g., *i/y*, *u/v*, *ć/ci*, *rz/ż*). Long vowels (*á*, *é*) and palatalization (*ć*, *ź*, *ś*, *ń*) were marked inconsistently. The KorBa transliteration layer preserves this variation, while the transcription layer normalizes it (e.g., *rodźicow* → *rodziców*).[5]

Orthographic conventions also influenced tokenization. Many expressions that are today written separately were then written together, and vice versa. For example:

[5] See Gruszczyński et al. (2022), p. 317.

- Historical joint writing *zchęći*, modern *z chęci* (‘from willingness’)
- Historical separate writing *dla tego*, modern *dlatego* (‘because’, lit. ‘for this’)

Following Wieczorek (2020), the MPDT treats historical spacing as evidence but bases syntactic analysis on function. Two frequent scenarios occur. (i) Historically fused sequences that correspond to ordinary prepositional phrases are split into their syntactic parts; the preposition and its nominal complement are annotated as in regular usage (see Example 1). (ii) Historically separate sequences that function as a single connective or adverbial are kept as two tokens but marked as a fixed multiword unit in MPDT; the unit then receives a single clausal function (see Example 2).

Example 1:

> *Poniewoli musiała zbita na łożu leżeć u rodziców/ ale* ***zchęci*** *obiecała sobie inną okazją/ iż jej od przedsięwzięcia nie oderwą.*
> (‘Against her will she had to lie beaten on the bed at her parents’/ but of her own will she promised herself another occasion/ that they would not tear her away from her undertaking.’)

In Example 1, *zchęci* is analyzed in MPDT as the preposition *z* with the genitive noun *chęci*: the preposition functions as a modifier of the non-finite predicate, and the noun is its complement.

Example 2:

> *Tak jest: i* ***dla tego*** *tak się poniżył.*
> (‘Indeed: and therefore he humbled himself thus.’)

In Example 2, *dla tego* forms a lexicalized connective in MPDT: *dla* serves as the syntactic head of the unit that modifies the clause, and *tego* is linked to it as a multiword element within that unit.

3.1.2. Punctuation

As described by Wieczorek (2025), punctuation in Middle Polish reflected the rhythm and pauses of speech rather than syntactic boundaries. Marks were used inconsistently and sometimes idiosyncratically: slashes (/) often functioned as commas, semicolons as commas, and colons as semicolons or dashes. Conversely, long unpunctuated stretches also occur.

During syntactic annotation, punctuation is interpreted according to its syntactic function, not its original graphic mark. For instance, a slash (/) that introduces a new clause is annotated as `punct`.

Example 3:

Powstawszy raz z bárzo ćięszkiey choroby/ ták rzekł Nie nagorzey śię zemną stáło: Bo mię chorobá vpomniáłá/ ábym się w pychę nie podnośił/ ponieważem iest śmiertelny.
('Having once recovered from a very severe illness/ he said thus: It did not go too badly with me: For the illness reminded me/ that I should not lift myself up in pride/ since I am mortal.')

As noted above, the slash in Example 3 functions as a clause delimiter and is therefore annotated as `punct`.

3.2. Morphology

The morphological system of Middle Polish differs significantly from the modern language, both in its inventory of forms and in category values. These distinctions were codified in the KorBa 2.0 tagset and later adopted in the MPDT.

3.2.1. Additional Parts of Speech and Morphological Features

The Middle Polish tagset introduces several categories and feature values that are rare or absent in contemporary Polish; below those are highlighted with direct impact on UD mapping.

(a) Short-form adjectives (`adjb`). These forms—e.g., *żyw* ('alive'), *godzien* ('worthy')—are indeclinable or partially declined adjectives, often used predicatively without the copula. In UD they are mapped to `UPOS=ADJ` with `Variant=Short`.

Example 4:

Iak długo ia ***żyw*** *iestem, żyie Pán moy poty, Czuię bol y wesołość, czuię y kłopoty.*
('As long as I am alive, my Lord lives likewise; I feel pain and joy, I feel troubles as well.')

Example 5:

Chcesz się zemną rownać: nie ***godzien****eś tego.*
('You want to match yourself with me: you are not worthy of this.')

In modern Polish, the short form *żyw* from Example 4 would be considered archaic or poetic; the modern equivalent is *żywy*. The word *godzien* from Example 5 still exists, along with a few other, like *pewien* ('certain'), however their usage is now limited, and the standard forms are *godny*, and *pewny*.

(b) Short passive participles (`ppasb`). Uninflected short passive participles—e.g., *zbawion* ('saved'), *pisan* ('written')—co-occur with finite forms of *być*. In UD they are annotated as `UPOS=ADJ` with `VerbForm=Part`, `Voice=Pass`, `Variant=Short`.

Example 6:

> *Kto vwierzy, á okrzći się,* ***zbáwion*** *będźie, ále kto nie vwierzy będźie* ***potępion****.*
> ('Whoever believes and is baptized will be saved, but whoever does not believe will be condemned.')

Example 7:

> ***Pisań*** *na zamku pileckim, dnia 23 miesiąca lipca, roku Pańskiego 1620.*
> ('Written at the castle of Pilec, on the 23rd day of July, in the Year of Our Lord 1620.')

In modern Polish, short forms from Examples 6 and 7 are archaic; the standard forms are fully inflected *zbawiony*, *potępiony*, *pisany*.

(c) Past participles (`ppraet`). Forms such as *osłabiałe* ('weakened'), *opuchłymi* ('swollen'), *zasiniałymi* ('bruised/blue-tinged') represent an older stage of adjectival participles derived from past tenses, intermediate between `ppas` and `pact`. In UD they are mapped to `UPOS=ADJ` with `VerbForm=Part` and `Voice=Pass`.

Example 8:

> *Częstokroć ábowiem były widáne z twarzámi* ***opuchłymi****/* ***záśiniáłymi****.*
> ('For often they were seen with swollen/ bruised faces.')

In modern Polish, the past participle forms are still in use, but some are archaic or poetic. Looking at words from Example 8 *opuchłymi* would be rather replaced by *opuchniętymi*, while *zasiniałymi* is still acceptable.

(d) Dual number (`du`). Middle Polish still preserved dual forms for certain nouns, numerals, adjectives, and verbs. The KorBa manual documents the explicit tag `du`. These forms gradually merged with the plural after ca. 1740, though fossilized duals like *ręce*, *oczy* survive in modern Polish (singular *oko* ('eye') → plural *oczy* when referring to the organ, but also pl. *oka* when used in other sense, e.g., *oka w rosole* ('eyes in the broth'); similarly singular *ucho* ('ear') → plural *uszy*, when about body parts, or pl. *ucha*, when referring to cup handles). In UD, dual forms are annotated with `Number=Dual`.

Example 9:

> *6. Po przepędzonych przez* ***dwie lecie*** *tych okrutnych boleściách, pokazał się iey Pan mowiąc: Iż bez lat pięc nie miałaby iadać ani mięsa. ani nabiáłu.*
> (‘6. After two years spent in these cruel pains, the Lord appeared to her saying: That for five years she should not eat either meat or dairy.’)

In Example 9, *dwie lecie* is dual accusative of *dwa* (two) and *lato* (‘summer; year’). In modern Polish, the dual form is archaic; the standard form (both the nominative and accusative) is *dwa lata*.

3.2.2. Gender System and Declension

The masculine gender system in Middle Polish was less differentiated than in the modern language. KorBa distinguishes three values: `m` (general masculine), `manim1` (masculine personal), and `manim2` (masculine non-personal). In early texts, these values overlap; many forms do not yet reflect consistent distinctions in case endings. For example, *ptaki* and *ptacy* (‘birds’) alternate depending on context.

Example 10:

> *6. Vbogáćiłeś ich chybkośćią i lotem nád wszytkie loty prędszym i bystrzeyszym/ i bystrym ták/ iż i strzały/ i* ***ptaki****/ i pioruny poprzedzáć/ á wszytkie rzeczy/ mury/ skáły/ przenikać mogą.*
> (‘You have enriched them with speed and with flight swifter and sharper than all flights/ so that even arrows/ and birds/ and thunder they can outpace/ and penetrate all things/ walls/ rocks.’)

Example 11:

> *122. Czemu* ***ptacy*** *ktorzy ogona nie máią dlugie nogi maią?*
> (‘Why do birds that do not have a tail have long legs?’)

Example 10 illustrates the use of *ptaki* in the general masculine category (`m`), while Example 11 uses the masculine personal value *ptacy* (`manim1`).

3.3. Syntax

3.3.1. Word Order and Non-projectivity

Middle Polish syntax exhibits high flexibility of word order, frequent inversion, and long-distance dependencies. As noted by Wieczorek (2025), discontinuous structures—especially in noun phrases with adjectival modifiers—often yield non-projective trees. The contrast between a linear and a discontinuous configuration is illustrated in Figures 8 and 9.

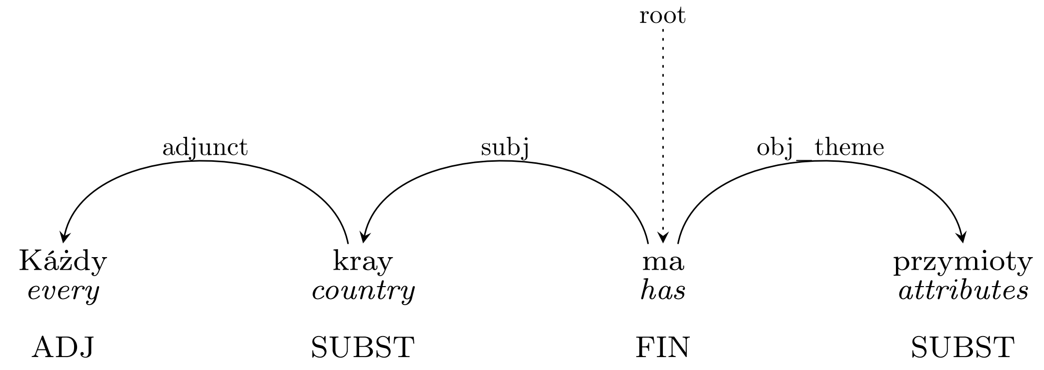

Figure 8: Linear order (no crossing edges)
Source: adapted from Wieczorek (2025), Fig. 6, p. 12.

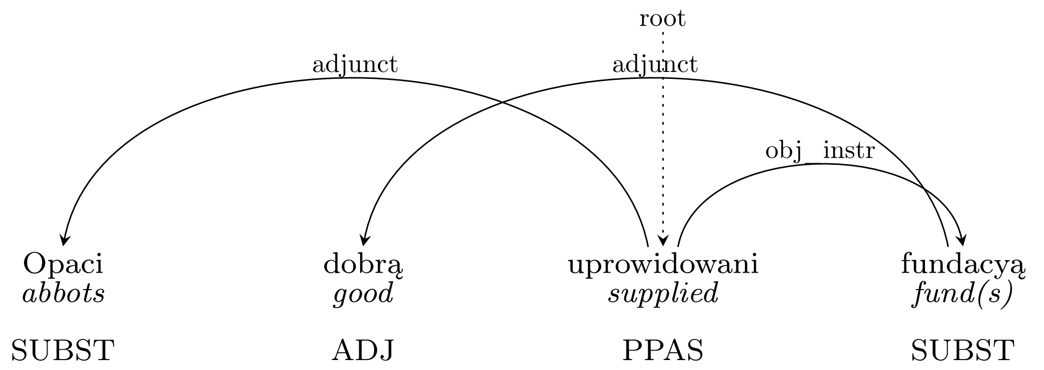

Figure 9: Discontinuous order with crossing edges between *dobrą* and *fundacyą*
Source: adapted from Wieczorek (2025), Fig. 7, p. 12.

These inversions complicate automatic parsing and were one challenge for explicit rule-based conversion to UD.

3.3.2. Predicate Ellipsis

As noted by Wieczorek (2025), it is rare in modern Polish for sentences to lack a predicate (at least in texts written in careful language), but this was quite common in 17th–18th-century Polish. In dependency analysis, the predicate is considered the centre of the sentence (`root`)—most often a finite verb. In the absence of a predicate, another sentence element serves as the centre. Most often, this centre becomes the subject, which receives the label `root` instead of `subj`, as in Figure 10.

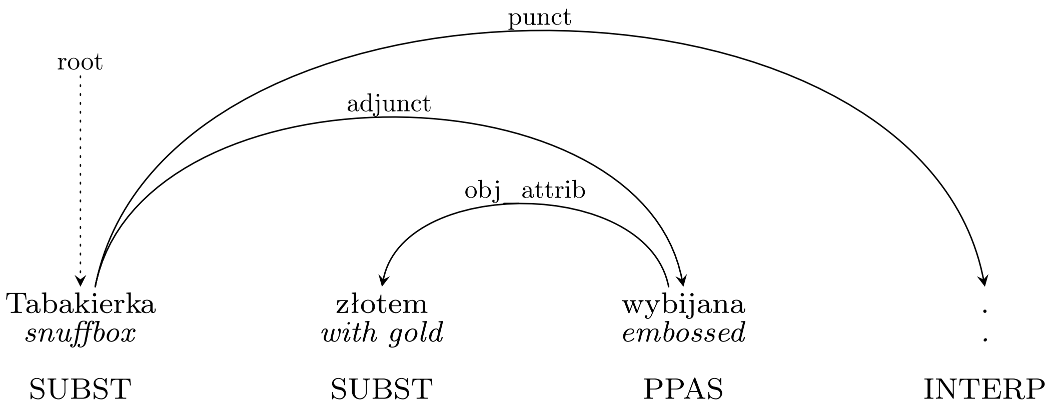

Figure 10: Predicate ellipsis: nominal head as `root`
Source: adapted from Wieczorek (2025), Fig. 8, p. 13.

3.3.3. Clause Linking and Subordination

Middle Polish frequently employs conjunctions that have since changed meaning, an example of which could be the token *jako*. It is frequently employed in two functions: (i) as a comparative/similative marker in the sense of modern *jak* ('like/as; when/how'), and (ii) in the modern-like role/identity sense 'as'. The readings are illustrated in Examples 12–14.

Example 12:

> *Ale iako nowi Obywatele tam przybywać poczeli z Kir, czy Syr Kraiu w Medii leżącego, Kirya, to Syria zwać się poczęła.*
> ('But as/when new inhabitants began to arrive there from the land of Kir, or Syr, lying in Media, Kirya then began to be called Syria.')

Example 13:

> *Iako Roża rozpuszcza z przyrodzenia swego zapach przyiemny, tak Serce dobroczynne wydaie bez przyniewolenia uczynki dobre.*
> ('As a rose by its nature gives off a pleasant fragrance, so a charitable heart produces good deeds without compulsion.')

Example 14:

> *Ja zaś w tych terminach stawam jako mediator, prowadząc do zgody obiedwie strony.*
> ('And I, for my part, in these proceedings stand as a mediator, leading both sides to agreement.')

In Example 12, *jako* functions as a clause linker with temporal meaning (‘when’). In Example 13 it introduces a comparative clause (‘how/as’), and in Example 14 it introduces a role/identity complement (‘as (in the role of)’). In modern Polish, the first two functions are typically expressed with *jak*, while the third remains *jako*.

3.4. Summary

Middle Polish exhibits substantial divergence from modern Polish in orthography, morphology, and syntax:

- Orthography: inconsistent, variable, often merging or splitting tokens differently from modern norms.
- Punctuation: prosodic rather than syntactic, with slashes and colons used irregularly.
- Morphology: additional forms (`adjb`, `ppasb`, `ppraet`), productive dual number, and fluid gender distinctions.
- Syntax: high non-projectivity, frequent inversion, ellipsis, and loose coordination.

These properties directly inform the design of the MPDT → MPDT-UD conversion pipeline, motivating special conversion rules and additional validation layers to preserve linguistic authenticity while ensuring formal compatibility with Universal Dependencies.

Chapter 4

Conversion Design and Implementation

This chapter details the design and implementation of the MPDT→MPDT-UD conversion pipeline. The conversion is a complex, multi-stage process, divided into two primary phases: 1. morphosyntactic mapping (Section 4.3) and 2. dependency tree transformation (Section 4.4). Before detailing them, the chapter first describes the overall architecture of the converter, including its custom data structures and the code repository layout (Section 4.1), the high-level processing workflow (Section 4.2), and at the end–the auditable logging system that fulfills research goal **(R2)** (Section 4.5).

4.1. Converter Architecture and Environment

The entire conversion process is implemented in Python, leveraging a custom-built environment designed for traceability and modularity. The code is publicly available in a GitHub repository: `https://github.com/kvmilos/MPDT-to-UD-converter`.

4.1.1. Core Data Structures

The environment is built around core data structures, `Sentence` and `Token` classes, defined in `utils/classes.py`. A key design choice is that each `Token` object stores both the original MPDT annotation and the new, converted UD annotation in parallel. This allows conversion rules to access the original, unmodified MPDT context at any stage, which is crucial for resolving ambiguity during the complex dependency transformation phase.

The `Token` class is also equipped with numerous helper properties and methods to simplify the writing of conversion rules, such as methods accessing the governor in the new UD tree if it is present, and accessing the old one otherwise, or traversing the tree to find specific dependents or governors via certain relations.

In parallel, the `Sentence` class provides the structural container for all `Token` objects belonging to a single sentence. Beyond simple storage, it offers functionality that is essential for conversion: it maintains a fast token lookup table (`dict_by_id`), stores sentence-level metadata (e.g., `# sent_id`, `# text`), and exposes utility methods for navigating the dependency tree. These include retrieving the root token, or iterating over tokens in different orders. By centralizing this behaviour in a dedicated class, the converter ensures that all modules operate over a consistent and fully accessible representation of sentence structure.

4.1.2. Project Repository Structure

The converter's code is organized into modules based on functionality. The main modules handle the top-level pipeline, morphosyntactic conversion, dependency conversion, and utilities. The structure of the repository is as follows (directories are shown in blue):

```
ud_converter/
├── converter.py
├── morphosyntax/
│   ├── pos_categories/
│   ├── preconversion.py
│   ├── conversion.py
│   ├── postconversion.py
│   └── morphosyntax.py
├── dependency/
│   ├── structures/
│   ├── labels.py
│   ├── edges.py
│   ├── preconversion.py
│   ├── conversion.py
│   └── postconversion.py
├── utils/
│   ├── classes.py
│   ├── constants.py
│   ├── io.py
│   └── logger.py
├── data/
└── logs/
```

The main components are:

- `converter.py`: The main executable script that orchestrates the entire conversion pipeline.
- `morphosyntax`: The package for Phase 1 (morphosyntactic conversion).

  - `pos_categories`: Contains a separate module for most MPDT `XPOS` tags (e.g., `subst.py`, `adj.py`) to handle its specific conversion rules.

- `dependency`: The package for Phase 2 (dependency conversion).
  - `structures`: Contains modules for restructuring specific syntactic constructions (e.g., `coordination.py`, `prepositional.py`).

- `utils`: Utility package with helper modules for the entire application.
  - `classes.py`: Defines the core `Sentence` and `Token` objects.
  - `constants.py`: A central store for all static mappings (e.g., features, lemmas).
  - `io.py`: Handles reading input `.conll` and `.json` files and writing the output `.conllu` file.
  - `logger.py`: Implements the auditable logging system, including the `ChangeCollector` and `LoggingDict` classes.

- `data/`: Default directory for input and output data files.

- `logs/`: Default directory where the detailed conversion logs are saved.

4.2. Conversion Pipeline

The converter is designed as a sequential pipeline, executed from the main `converter.py` script. From the user's perspective, the process consists of four main stages:

(1) **Data Loading:** The pipeline begins by reading two input files: the MPDT treebank in its `.conll` format and a corresponding `.json` metadata file. The data is loaded into the custom `Sentence` and `Token` objects.

(2) **Phase 1: Morphosyntactic Conversion:** The first processing stage performs a rule-based conversion of the MPDT morphosyntactic annotations into their UD counterparts.

(3) **Phase 2: Dependency Conversion:** The second processing stage transforms the syntactic structure of the trees. This highly contextual phase converts the MPDT dependency relations to UD relations and restructures the tree topology to conform to UD guidelines.

(4) **Output Generation:** Finally, the converted `Sentence` and `Token` objects, now populated with UD annotations, are written to a single output `.conllu` file.

4.3. Phase 1: Morphosyntactic Conversion

The first processing phase, handled by the `morphosyntax` module, converts the MPDT `XPOS` tags and morphological features into their `UPOS` and `FEATS` counterparts. This phase is executed as a three-step sub-pipeline for each sentence.

4.3.1. Pre-conversion

First, a set of lemma-based rules are applied to handle specific lexical items whose categorization overrides the more general `XPOS`-based rules. For example:

- Conjunctions like *niż* (‘than’), *jakby* (‘as if’), and *niczym* (‘like’), which introduce comparisons, are unambiguously mapped to `SCONJ` (subordinating conjunction) and assigned the feature `ConjType=Comp` to mark this comparative function.
- The lemma *temu*, when used as a postposition (e.g., *dwa lata temu* ‘two years ago’), is mapped to `ADP` (adposition) and assigned the feature `AdpType=Post` to explicitly mark it as a postposition, distinguishing it from the standard prepositional form.
- Words with an initial capital letter that are not otherwise classified (e.g., as verbs) are provisionally tagged `PROPN` (proper noun). This rule helps correct cases where a proper noun was ambiguously tagged as a common noun (`subst`).

4.3.2. Core POS Conversion

Next, the main conversion logic maps the MPDT `XPOS` tag of each token to its corresponding `UPOS` tag and `FEATS`. The converter dispatches each token to a dedicated function based on its `XPOS` tag.

This design handles both simple and complex conversions. For instance, while `conj` (coordinating conjunction) almost always becomes `CCONJ`, the `subst` (noun) tag requires more logic: most `subst` tokens are mapped to `NOUN`, but the converter first checks for pronominal lemmas (e.g., *kto*, *co*, *nikt*) and maps these to `PRON` (pronoun) with the appropriate `PronType` feature.

This module also handles the specific Middle Polish phenomena described in Chapter 3:

- `adjb` (short adjective) is mapped to `UPOS=ADJ` and given the feature `Variant=Short`.
- `ppasb` (short passive participle) is mapped to `UPOS=ADJ` with the features `VerbForm=Part`, `Voice=Pass`, and `Variant=Short`.
- The MPDT gender system is correctly mapped to the UD features (e.g., `manim1` to `Gender=Masc` and `Animacy=Hum`).

- The Middle Polish `Number=Dual` feature is preserved, as it is a valid feature in Universal Dependencies, even if absent in modern Polish.

4.3.3. Post-conversion

Finally, a sentence-level cleanup function performs two crucial tasks that require the original, non-tokenized text from the metadata:

1. **Reconstructing Multiword Tokens:** This step correctly formats clitic constructions that were already split into syntactic words in the input data. For example, for the Middle Polish word *kiedym* ('when I'), the input `.conll` file contains two separate token lines (*kiedy* and *m*). This function reads the original text, sees they are not space-separated, and inserts the required multiword token entry (e.g., `14-15 kiedym ...`) before the syntactic words it spans, as shown in the example in Chapter 2.2.

2. **Annotating Spaces:** The same function analyzes the original text to add `SpaceAfter=No` to the `MISC` column for any token that is immediately followed by another token or punctuation mark without an intervening space. This is a requirement for the CoNLL-U format since Universal Dependencies v. 2.0.

4.4. Phase 2: Dependency Conversion

The second phase, managed by the `dependency` module, is significantly more complex. Unlike morphosyntax, dependency conversion is not token-local; rules must consider a token's governor, its dependents, and its siblings, often operating on the original MPDT structure, the partially converted UD structure, or both.

Many of the structural transformations were adapted from the principles established for the conversion of the contemporary Polish Dependency Bank (Wróblewska 2018; Wróblewska 2020), but were re-implemented to fit the custom pipeline and handle Middle Polish phenomena. The conversion follows a strict pipeline of restructuring, label mapping, and post-processing.

4.4.1. Structural Restructuring

The first and most critical step is to change the topology of the dependency tree to conform to UD principles. The converter applies a series of modules to handle specific syntactic constructions. The most fundamental transformations are:

- **Prepositional Phrases:** In MPDT, a preposition (`prep`) governs its nominal complement (`comp`). This structure gets inverted: the nominal complement becomes the head of the phrase. This new head inherits the original syntactic function from the preposition (e.g., the `adjunct_locat` relation from *na* in Figure 11 becomes the dependency for *koniu*). This relation is then mapped to a UD relation (e.g., `obl`). Finally, the preposition is re-attached to the noun with the `case` relation. This transformation is illustrated in Figure 11.

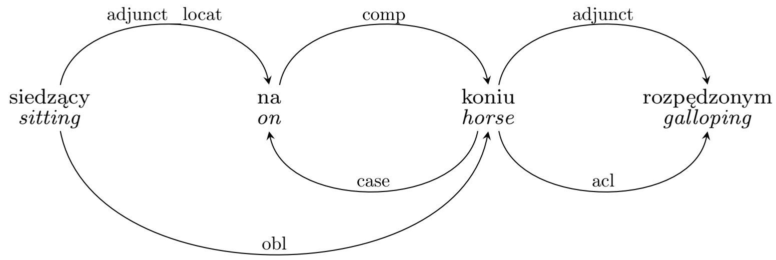

Figure 11: MPDT analysis (arcs above) and converted UD analysis (arcs below) for the fragment *siedzący na koniu rozpędzonym* ('sitting on a galloping horse').

- **Numeral Phrases:** Numeral expressions that govern their nouns in MPDT (e.g., as a `comp`) are restructured. In UD, the noun is promoted to be the head, and the numeral is re-attached as its dependent with the `nummod` relation. The noun phrase then attaches to the verb (e.g., as `obl`), as shown in Figure 12.

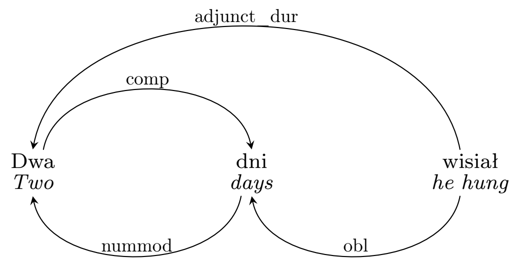

Figure 12: MPDT analysis (arcs above) and converted UD analysis (arcs below) for the clause *Dwa dni wisiał* ('He hung for two days').

- **Predicative expressions:** In MPDT, the copula (e.g., *być* 'to be' or the `pred` token *to*) is the head of the clause, governing the subject (with `subj`) and the non-verbal predicate (with `pd`). To comply with UD, the non-verbal predicate is promoted to be the head. The subject and the copular verb are then re-attached as dependents of this new nominal or adjectival head, receiving the UD relations `nsubj` and `cop`, respectively. This is illustrated in Figure 13.

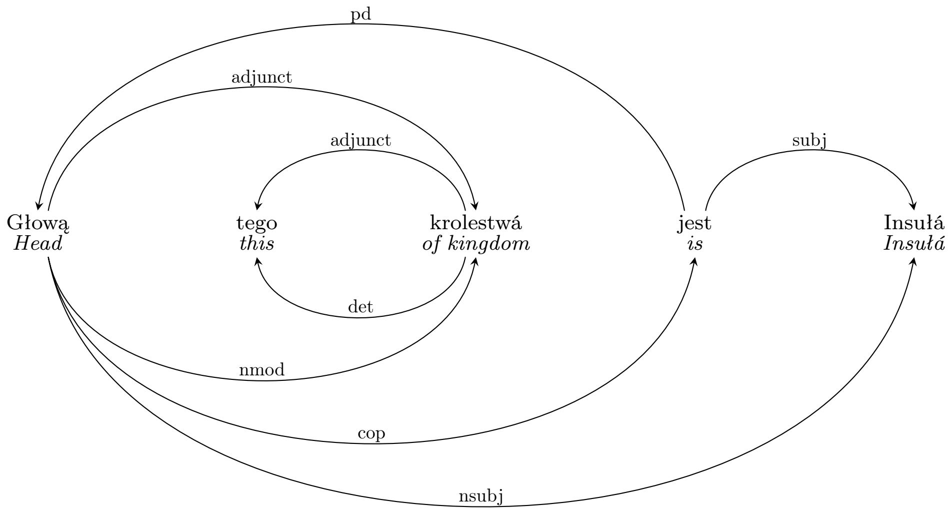

Figure 13: MPDT analysis (arcs above) and converted UD analysis (arcs below) for the fragment *Głową tego krolestwá iest Insułá* (‘The head of this kingdom is Insułá’).

- **Subordinate Clauses:** In MPDT, a subordinating conjunction (`comp`) often governs the predicate of its clause (e.g., as `comp_fin`). The converter inverts this, promoting the subordinate clause’s predicate to be the head (which then attaches to the main clause predicate, or becomes the root), and demotes the conjunction to be a dependent of its clause’s predicate with the `mark` relation. In Figure 14, this corresponds to replacing the MPDT chain *rzekł* $\xrightarrow{\texttt{comp}}$ *że* $\xrightarrow{\texttt{comp\_fin}}$ *było* with the UD configuration *rzekł* $\xrightarrow{\texttt{ccomp}}$ *lekko* and *lekko* $\xrightarrow{\texttt{mark}}$ *że*.

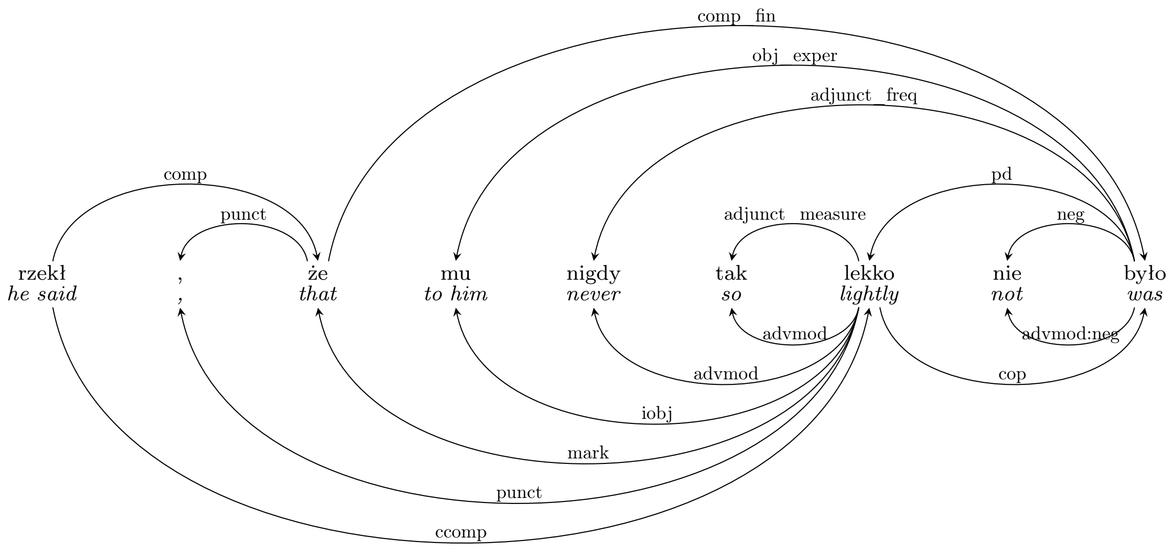

Figure 14: MPDT analysis (arcs above) and converted UD analysis (arcs below) for the fragment *rzekł, że mu nigdy tak lekko nie było* ('he said that it had never been so easy for him').

- **Coordination:** In MPDT, the coordinating conjunction (`conj`) is the head of the coordinated elements (`conjunct`). This structure is rebuilt by promoting the *first* conjunct to be the head of the entire coordination. Subsequent conjuncts are attached to this first conjunct with the `conj` relation. The conjunction itself is re-attached to its *following* conjunct with the `cc` relation. In the upper tree of Figure 15, the coordinator *y* governs both conjuncts *odwagi* and *sercá*, each of which has its own modifier. In the lower tree, the first conjunct *odwagi* becomes the syntactic head of the coordination, the second conjunct *sercá* attaches to it as `conj`, and the conjunction *y* becomes a dependent of *sercá* with relation `cc`.

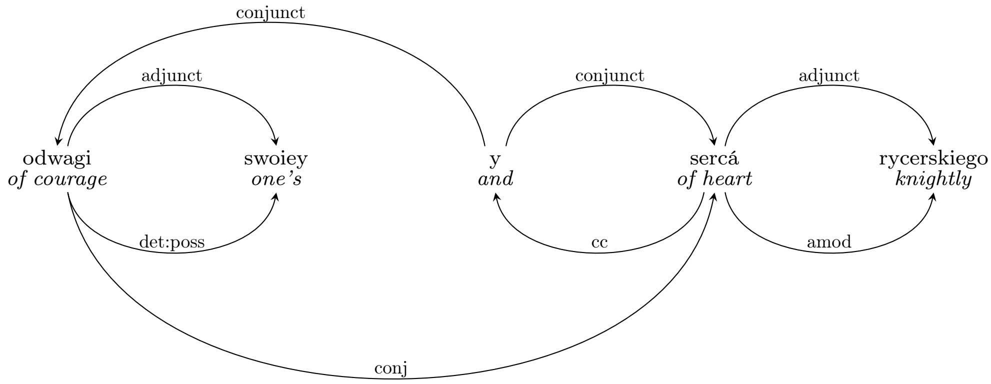

Figure 15: MPDT analysis (arcs above) and converted UD analysis (arcs below) for the fragment *odwagi swoiey y sercá rycerskiego* ('of one's courage and knightly heart').

4.4.2. Label Mapping

After the tree structure is finalized, a dedicated module traverses the tree and assigns a final `UDEPREL` to each token. This mapping is highly context-sensitive, using the `UPOS` of both the token and its new governor, as well as its original MPDT `DEPREL`. For example, the generic MPDT `adjunct` relation is mapped to a variety of UD relations:

- A nominal modifier of a noun (`adjunct` on a `NOUN` dependent) → `nmod`
- An adjectival modifier of a noun (`adjunct` on an `ADJ` dependent) → `amod`
- An adverbial modifier of a verb (`adjunct` on an `ADV` dependent) → `advmod`
- A prepositional phrase modifying a verb (`adjunct` inherited by a `NOUN` from a `prep`) → `obl`
- A clausal modifier of a verb (`adjunct` on a `VERB` dependent) → `advcl`

4.4.3. Correction and Post-processing

Finally, a series of cleanup scripts are run. One module ensures UD validation compliance by removing disallowed dependents (e.g., a `case` token cannot have its own dependents, so any punctuation attached to it is moved to its head).

Another module handles final tasks, such as disambiguating pronouns that are ambiguous between interrogative and relative (e.g., `PronType=Int,Rel` → `PronType=Int` or `PronType=Rel`) based on their new syntactic context. Most importantly for downstream use, this module generates the enhanced dependency graph (`DEPS` column) by propagating shared dependents in coordination. This contributes to research goal **(R3)** by improving evaluability and supporting enhanced UD representations.

4.5. Audibility and Processing Workflow

Beyond the core conversion logic, two key features of the converter are its auditable design and its straightforward user workflow.

4.5.1. Logging and Traceability

A core design principle of the converter is audibility, fulfilling research goal **(R2)**. This is implemented via a custom logging system built into the `utils/logger.py` module.

A central `ChangeCollector` class gathers change events from all modules. To automate this, the core `Token.data` dictionary is implemented as a `LoggingDict`, a dictionary subclass that automatically calls `ChangeCollector.record()` whenever a value is set or changed.

Each log entry records the sentence ID, token ID, the specific module and function that triggered the change, and a message detailing the transformation (e.g., `upos changed from VERB to AUX`). This fine-grained logging (Contribution **C1**) proved invaluable for debugging, as it allows for a step-by-step reconstruction of how a token was processed and which rules fired. It was particularly critical for identifying and resolving rule conflicts during the complex dependency conversion phase.

4.5.2. Processing Workflow

From a user's perspective, the pipeline is executed via a single command. The converter takes the MPDT `.conll` file and the corresponding metadata `.json` file as input.

```
python converter.py input_file.conll output_file.conllu meta_file.json
```

The script processes each sentence and saves the result in the specified `output_file.conllu` in the valid CoNLL-U format, ready for validation and downstream use. The converter can be run in two modes. By default, it executes the complete pipeline, performing both phases of the conversion (morphosyntax and dependencies). However, if the user provides the `-tags-only` command-line flag, the pipeline omits Phase 2. This allows the user to generate a file with only the morphosyntactic conversion applied, leaving the original MPDT dependency structure intact.

4.6. Summary

This chapter has defined and implemented the core of the MPDT $\rightarrow$ MPDT-UD conversion pipeline:

- It specified a UD-oriented conversion strategy for MPDT, including the main structural transformations (prepositional phrases, numeral phrases, predicative expressions, subordination, and coordination), directly addressing research goal **(R1)**.
- It described a modular, auditable implementation in Python, centred on custom `Sentence` and `Token` classes and an integrated logging system, thereby realizing research goal **(R2)** and delivering Contribution **(C1)** (the converter itself).
- It prepared the ground for formal UD conformance and evaluability (**R3**) by enforcing UD-compliant structures, mapping labels to the UD inventory, and generating enhanced dependencies; the quantitative conformance results are presented in Chapter 5.

Together, these elements produce a reusable conversion pipeline and a traceable implementation that underpins the validated MPDT-UD treebank discussed in the following chapter.

Chapter 5

Validation and Outcomes

This chapter presents the final, tangible outcome of the thesis: the initial public release of the **Middle Polish Dependency Treebank in Universal Dependencies format (MPDT-UD)**. Beyond reporting that the treebank passes the official validator, the chapter explains *how* validation shaped the converter's design: validator feedback drove targeted, logged corrections rather than ad-hoc edits, yielding a reproducible and auditable pipeline.

The chapter is structured as follows. Section 5.1 introduces the validation workflow, including the iterative procedure, the validation dataset, and the official UD validator. Section 5.2 reports the conformance results and outlines how remaining edge cases were handled. Section 5.3 presents the final MPDT-UD 1.0 treebank, summarizing its size, data split, licensing, and integration into the UD ecosystem.

5.1. Validation Workflow and Formal Conformance

To fulfill research goal **(R3)**—ensuring that the converted treebank is formally correct and compatible with standard UD tools—a strict validation workflow was adopted. The entire MPDT-UD corpus was checked using the **official Universal Dependencies validator script** `validate.py`.[6] The procedure was implemented as an iterative refinement cycle, applied to a fixed validation dataset and grounded in the behaviour of the official validator.

5.1.1. Iterative Validation Procedure

The validation workflow was integrated into the development process as an iterative, data-driven refinement cycle that directly influenced the converter's architecture. Achieving the final zero-error conformance required repeatedly executing the following

[6]The validator is part of the UD tools repository; available at `https://github.com/UniversalDependencies/tools/blob/master/validate.py`

steps:

1. Running the full MPDT → MPDT-UD conversion pipeline.
2. Executing the `validate.py` script on the resulting `.conllu` file.
3. Collecting and classifying validator messages to identify systematic error types.
4. Implementing targeted adjustments in the relevant module (typically postconversion or label mapping) and repeating the cycle.

Throughout development, validator feedback served as an objective convergence criterion: changes to the pipeline were accepted only if they reduced violations without introducing new ones, and every change was captured by the converter's logging facilities for later audit.

5.1.2. Validation Dataset

The validation was performed on the entire output of the conversion pipeline, i.e. the full MPDT available at the time of writing, converted to UD as MPDT-UD. All mentions in this chapter refer to this complete set.

Concretely, the validation dataset corresponds to the initial UD release of UD_Polish-MPDT (version 2.17). It contains **2,018 sentences**. The material is drawn from the manually annotated portion of the KorBa corpus (as described in Section 2.3) and thus inherits its genre balance and time-span.

Since Middle Polish is not a separate language branch within the UD project, the treebank was validated under the tagset and feature inventory of modern Polish (`pl`). This approach guarantees maximal compatibility with existing UD resources and ensures that the resulting data can be immediately processed by any standard UD-compliant parser or visualization tool.

5.1.3. Official UD Validator

The `validate.py` script performs a comprehensive, multi-layer verification of every sentence in a CoNLL-U file. It is implemented in Python and relies on the official UD feature inventories and relation lists, performing checks in three main categories:

- **Format compliance:** ensures that each sentence adheres to the CoNLL-U specification—exactly ten columns per token, valid multiword token ranges, correct comments, and consistent `SpaceAfter=No` annotation.
- **Morphological validity:** verifies that every `UPOS` tag, `XPOS` tag, and `FEATS` combination is legal according to UD v2.17 inventories (e.g., that `Aspect` features only

occur on verbal categories, and that `PronType` values correspond to pronouns or determiners).

- **Syntactic structure:** checks that the dependency tree is single-rooted, acyclic, and projective when required; that function words (e.g., tokens with relations `case`, `cc`, `mark`) do not have their own dependents; and that all `HEAD` indices refer to valid token IDs.

The validator also provides detailed diagnostic messages for every detected issue, grouped by error type and line number. This makes it suitable not only for final compliance testing but also for iterative debugging during development.

The script must be run with a language code (e.g., `-lang pl`) to check against treebank-specific feature and relation lists. It requires Python 3 and the `regex` and `udapi` modules. A typical invocation looked like:

```
python validate.py --lang pl MPDT-UD.conllu
```

5.2. Conformance Results

The final converted MPDT-UD treebank passes the official UD validator (`validate.py`) with **zero errors**. This 100% conformance fulfills the principal technical objective of the thesis (**R3**) and confirms that the resource is formally compatible with the Universal Dependencies standard.

This outcome is the result of the iterative, data-driven refinement cycle described in Section 5.1.1. Validator feedback was treated as an objective convergence criterion: pipeline changes were accepted only when they reduced the number of violations without introducing new ones, and all modifications were recorded by the converter's logging facilities for later audit.

The emphasis throughout was on *principle-driven* cleanups aligned with UD guidance rather than ad hoc edits—for example, enforcing that function words do not bear dependents, attaching punctuation to appropriate content heads, and canonicalizing order-sensitive multiword relations. In this way, the final state of the converter encodes the corrections required to reach full conformance, rather than relying on one-off manual adjustments to the released data.

5.2.1. Handling Idiosyncratic Edge Cases

Most validation issues were solvable with generalized, context-sensitive rules that can be applied uniformly across the corpus. A small residual set of idiosyncratic cases—rare constructions not easily captured by broad heuristics—were addressed pragmatically to achieve full conformance on the released dataset. These interventions are minimal, logged, and isolated, preserving reproducibility without overfitting the pipeline to particular sentences.

Conversion decisions for **POS** and **FEATS** are documented systematically in the accompanying specification.[7] This document provides a stable reference for future extensions of the pipeline and for users who wish to understand or reuse the morphosyntactic mapping outside the present project.

5.3. Outcomes: the MPDT-UD 1.0 Treebank

The validated conversion pipeline yields the first public release of the Middle Polish Dependency Treebank in Universal Dependencies format (**MPDT-UD 1.0**). This section summarizes the size and internal structure of the treebank and briefly documents its integration into the Universal Dependencies ecosystem, including licensing and availability.

5.3.1. Corpus Size and Data Split

The MPDT-UD 1.0 treebank contains **2,018 sentences**, **46,670 surface tokens**, and **47,273 syntactic words**. The difference between tokens and syntactic words reflects the use of multiword tokens in the CoNLL-U representation. The corpus includes **564 multiword tokens** (across **359** distinct types), with an average of 2.07 syntactic words per multiword token. In addition, **8,217 tokens** (18%) are not followed by a space, which is a direct consequence of historical orthographic conventions such as clitic fusion and non-standard punctuation spacing.[8]

In line with the UD guidelines for treebanks in the 20K–110K word range, approximately 10K words were allocated to the test set, about 10% of the remaining data to the development set, and the rest to the training set. Sentences were randomly shuffled (using seed 42) and then assigned to the three subsets based on token-count quotas. The resulting split is:

- **Training set:** 1,433 sentences, 33,520 tokens.
- **Development set:** 162 sentences, 3,748 tokens.
- **Test set:** 423 sentences, 10,005 tokens.

The treebank does not preserve document boundaries or genre labels at the sentence level; sentences have been shuffled and redistributed purely for the purpose of forming these three subsets. At the corpus level, the metadata assign four broad genres to the resource as a whole: nonfiction, bible, legal, and fiction.

[7] See `https://github.com/kvmilos/MPDT-to-UD-converter/blob/main/MPDT.md`

[8] Summary statistics and tokenization details are publicly available via the UD_Polish-MPDT treebank page: `https://universaldependencies.org/treebanks/pl_mpdt/index.html`.

5.3.2. Release, Licensing, and UD Integration

MPDT-UD 1.0 is distributed within the Universal Dependencies project under the repository name `UD_Polish-MPDT`. It was first released as part of the **UD v2.17** release. The data are available both through the UD GitHub organization and via the standard UD release packages, alongside other Polish resources.

The treebank is released under the **Creative Commons Attribution–ShareAlike 4.0 (CC BY–SA 4.0)** license.[9] In practical terms:

- **You may** share and adapt the data, including for commercial purposes.
- **You must** give appropriate credit, provide a link to the license, and indicate if changes were made.
- **ShareAlike:** if you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
- **No additional restrictions:** you may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

These conditions align with the open-data practices of the UD project and make MPDT-UD 1.0 immediately reusable for historical syntax research, diachronic NLP experiments, and cross-linguistic studies that rely on UD-conformant training and evaluation data.

[9] License text and a plain-language summary are available at `https://creativecommons.org/licenses/by-sa/4.0/`.

Chapter 6

Applications and Cognitive Science Perspective

6.1. Usefulness and Audience

6.1.1. Who benefits and how

6.1.2. Packaging and License

6.1.3. Repository and UD ecosystem integration

6.2. Use Cases

6.2.1. Historical Syntax and Diachrony

6.2.2. Parser Training and Evaluation

6.3. Cognitive Science Perspective

6.3.1. Processing Constraints

6.3.2. Category Change Over Time

6.4. Future Work

6.4.1. Coverage and Phenomena

6.4.2. Generalization and Automation

Bibliography

Brouwer, M., Brugman, H., and Kemps-Snijders, M. (2017). *MTAS: A Solr/Lucene based multi-tier annotation search solution*. CLARIN. URL: http://www.ep.liu.se/ecp/136/002/ecp17136002.pdf.

Buchholz, S. and Marsi, E. (2006). "CoNLL-X Shared Task on Multilingual Dependency Parsing". In: *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. Ed. by Màrquez, L. and Klein, D. New York City: Association for Computational Linguistics, pp. 149–164. URL: https://aclanthology.org/W06-2920/.

De Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). "Universal Dependencies". In: *Computational Linguistics* 47.2, pp. 255–308. ISSN: 0891-2017. DOI: 10.1162/coli_a_00402. URL: https://doi.org/10.1162/coli_a_00402.

Gruszczyński, W., Adamiec, D., Bronikowska, R., Kieraś, W., Modrzejewski, E., Wieczorek, A., and Woliński, M. (2022). "The Electronic Corpus of 17th- and 18th-century Polish Texts". In: *Language Resources and Evaluation* 56.1, pp. 309–332. ISSN: 1574-0218. DOI: 10.1007/s10579-021-09549-1. URL: https://doi.org/10.1007/s10579-021-09549-1.

Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., and Zeman, D. (2020). "Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection". In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Ed. by Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S. Marseille, France: European Language Resources Association, pp. 4034–4043. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.497/.

Przepiórkowski, A., Bańko, M., Górski, R. L., and Lewandowska-Tomaszczyk, B. (2012). *Narodowy Korpus Języka Polskiego*. Warszawa: Wydawnictwo Naukowe PWN. ISBN: 978-83-01-16700-4. URL: https://nkjp.pl/settings/papers/NKJP_ksiazka.pdf.

Wieczorek, A. (2020). *Instrukcja anotowania drzew zależnościowych: Dodatek dla tekstów XVII- i XVIII-wiecznych*. Institute of Polish Language, Polish Academy of Sciences.

Wieczorek, A. (2025). “Towards the Middle Polish Dependency Treebank”. In: *Native Language in the 21st Century: System, Communication Practices and Education*. V & R Unipress.

Wróblewska, A. (2018). “Extended and Enhanced Polish Dependency Bank in Universal Dependencies Format”. In: *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*. Ed. by de Marneffe, M.-C., Lynn, T., and Schuster, S. Brussels, Belgium: Association for Computational Linguistics, pp. 173–182. DOI: `10.18653/v1/W18-6020`. URL: `https://aclanthology.org/W18-6020/`.

Wróblewska, A. (2020). “Towards the Conversion of National Corpus of Polish to Universal Dependencies”. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Ed. by Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S. Marseille, France: European Language Resources Association, pp. 5308–5315. ISBN: 979-10-95546-34-4. URL: `https://aclanthology.org/2020.lrec-1.653/`.

Wróblewska, A. (2023). *Instrukcja anotowania drzew w Polskim Banku Drzew Zależnościowych*.