**L**inguistic **I**ssues in **L**anguage **T**echnology – **LiLT**
Submitted, January 2012

# Polish Dependency Bank

**Alina Wróblewska**

Published by CSLI Publications

**LiLT** volume 7, issue 2 January 2012

# Polish Dependency Bank

Alina Wróblewska, *Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland*

## Abstract

The paper outlines the first Polish dependency bank derived from constituent trees. The conversion is a fully automatic and unambiguous process. The converter takes manually disambiguated constituent trees encoded in the XML format as input and produces dependency structures encoded in the column-based CoNLL format. The conversion is a relatively straightforward process, since constituents have their syntactic centres marked in most cases. However, a certain amount of reorganising is necessary, in order to make the dependency structures meet annotation principles. The main part of the paper will be devoted to a characteristics of Polish dependency types. The Polish dependency bank can be used for training or evaluation of Polish parsers.

**LiLT** Volume 7, Issue 2, January 2012.
*Polish Dependency Bank.*

## 1 Introduction

The Polish dependency bank consists of about 7500 syntactically annotated sentences derived from the Polish constituency treebank.[1] A bank of constituent trees is under development at the Institute of Computer Science PAS (Świdziński and Woliński 2010). The planned size of the treebank is 20,000 sentences taken from the hand-annotated balanced subcorpus of the National Corpus of Polish (Przepiórkowski et al. 2010). However, as the project is still ongoing, trees for about 7500 sentences are currently available. The structure of constituent trees is designed with convertibility in mind. In particular, each constituent has its syntactic centre marked, which makes it relatively straightforward to convert constituent trees into dependency structures.

Sentences in the Polish dependency bank are annotated as graphs with arcs representing directed binary relations between lexical nodes (tokens). Every token in a sentence is linked with a dependency type. One of related tokens is regarded as the head of the dependency relation, while the other one is its dependent. Arcs linking lexical nodes are named with dependency labels. All nodes in a dependency structure correspond to terminal nodes of a constituent tree and are assigned an unique index. The ROOT node is always assigned the index 0. All other nodes are assigned the index corresponding to the position of the token in a sentence.

The conversion is a fully automatic and unambiguous process. The converter takes manually disambiguated constituent trees encoded in the XML format as input and outputs dependency structures encoded in the column-based CoNLL format (Buchholz and Marsi 2006). The choice of the output format has been determined by available dependency parsing systems, which we are going to use, and formats admitted by them.

The annotation schema of constituent trees requires encoding some morpho-syntactic information for each constituent. We make use of this information and transfer surface forms, lemmas, part-of-speech tags and morphological features of each token into the appropriate dependency structure, without the need for additional language processing tools. Except for the morpho-syntactic information, which is essential to derive dependency structures form constituent trees, we make use of phrasal categories and types of phrase structure rules

[1] The Polish constituent treebank is being developed in a semi-automatic manner. First, candidate parse trees are automatically generated for a sentence. Than, they are validated by human annotators, i.e., the candidate parse tree best corresponding to constraints of the Świdziński's formal definition of Polish (Świdziński 1992) is selected. If no correct tree exists, the sentence and its parse trees are rejected.

used to build and annotate constituent trees. Some dependency types have been also encoded in constituent trees, e.g. subject. The subject relation is directly transferred onto dependency structures. Many other dependencies are possible to extract, as constituent trees contain information about the head of a large part of constituents. An undoubted advantage of the constituency annotation schema is the distinction between *required phrases* (Pol. 'fraza wymagana') and *free phrases* (Pol. 'fraza luźna'). Required phrases correspond to arguments subcategorised by verbs, adjectives, adverbs, nouns and prepositions. The morpho-syntactic information and types of phrase structure rules enable the identification of dependency types for these arguments.

As the main goal of our work is to give a detailed description of the schema used to annotate dependency structures, section 2 is devoted to the characteristics of Polish dependency types. In order to meet annotation principles, we will present the necessary reordering of derived dependency structures in section 3. In this document, we put emphasis on the presentation of the annotation schema of dependency structures. However, we also perform some experiments (section 4). The derived dependency bank will be used to train and to evaluate a dependency parser for Polish. Section 5 concludes the paper.

## 2 Polish Dependency Types

Our goal is to convert the source constituent treebank to a bank of labelled dependency structures. The idea behind the conversion is to cover all language-specific syntactic phenomena encoded in the available Polish constituent trees and to annotate them with correctly chosen dependencies. Therefore, the precise definition of dependency relations seems to be crucial. Based on our research, we compile a list of valid Polish dependency types, which are discussed below. We start with the presentation of argument types, continue with defining non-argument dependencies and finish with the treatment of coordinating constructions.

### 2.1 Arguments

1. *comp* – complement
   The *comp* function may have diverse realisations and may be governed by differently realized heads.
   The **adjectival complement** may be governed by a verb form, e.g. *uczynić kogoś silnym*$_{ADJ}$ (Eng. 'to make sb. strong').
   Similarly, the **adverbial complement** may be governed by a verb form, e.g. *zeskoczyć skądś*$_{ADV}$ (Eng. 'to jump from somewhere').

The **nominal complement**, in turn, may be governed by an adjective, e.g. *pełny mleka*$_{NOM}$ (Eng. 'full of milk'), a preposition, a verb form or a numeral. A nominal *comp* governed by a verb form has not to be promoted to the subject during passivisation, therefore it may be distinguished from *obj*. Furthermore, it may fulfil different semantic roles Location, Instrument, Goal, etc. (except for Recipient, Experiencer, etc., reserved for *obj_th*).
Regarding numeral phrases, we annotate numerals as governors of depending noun phrases. According to Saloni and Świdziński (1998), who argue for treating numerals as heads in Polish numeral phrases, it is the numeral that is governed by the verb form. The case of the dependent noun phrase, in turn, either agrees with the case of the governing numeral (dative, instrumental, locative) or is determined as genitive in case of nominative, accusative, vocative or genitive numerals.
The **prepositional complement** may be governed by a verb form, an adjective, e.g. zdolny do$_{PREP}$ (Eng. 'able to, capable of') or an adverb, e.g. właśnie przez$_{PREP}$ (Eng. 'just by).

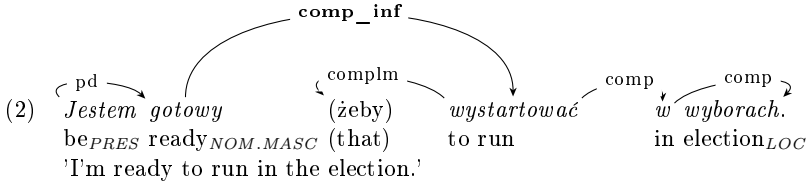
2. *comp_fin* – clausal complement
The *comp_fin* function is fullfiled by a closed complement clause (declarative, interrogative, or exclamatory) with an internal subject that may be realized as a pro-drop pronoun. The *comp_fin* argument may be governed by a verb form, a subordinating conjunction or a noun (see (1)).

(1)


'She did not answer the question what happened to her.'

3. *comp_inf* – infinitival clausal complement
The *comp_inf* function realized as an infinitival clausal complement (non-finite clause) may be governed by an adjective phrase (see (2)), a noun phrase, e.g. *(mieć) prawo coś zrobić* (Eng. '(to have) the right to do sth') or a verb form. Different control verbs or quasi-verbs[2] may bear the *comp_inf* complement, e.g. *chcieć* (Eng. 'to want'), *kazać* (Eng. 'to order, to tell'), *można* (Eng. 'to be allowed'), *trzeba* (Eng. 'it's necessary').

[2] The impersonal and subjectless Polish quasi-verbs may be conjugated for tense and mood, but not for person, e.g. *można* (Eng. 'to be allowed'), *trzeba* (Eng. 'it's necessary').

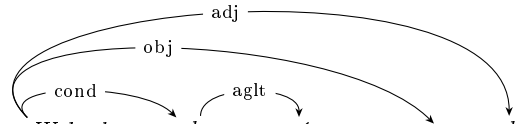



4. *complm* – complementizer
   The *complm* function is fulfilled by a complementizer, e.g. *że, iż* (Eng. 'that'), *żeby, aby, by* (Eng. 'so as to'). A complementizer introduces a complement clause, the predicate of which is its governor. In some contexts, a complementizer may be realized optionally (see (2)).
5. *obj* – direct object
   The *obj* argument governed by a verb form is realized as a noun phrase marked for the accusative, genitive, instrumental or even dative case, e.g. *zagrażać czemuś*$_{DAT}$ (Eng. 'to threaten sth'). The principal feature of *obj* is its ability to transform into the subject in passive constructions. This feature distinguishes *obj* from other verb arguments realized as noun phrases.
6. *obj_th* – dative object
   The *obj_th* function fulfilled by a dative noun phrase is governed by a verb form. Apart from the dative case marking, there are some additional properties distinguishing *obj_th* from other nominal verb arguments. First, *obj_th* must fulfil the semantic role of Recipient, Experiencer, Beneficiary, etc. Second, it cannot be promoted to the subject during passivisation or change its status to the direct object through any argument-structure alternation (e.g. 'dative shift', Kibort 2008).
7. *pd* – predicative complement
   Any element (verbal or small clause, adjective phrase, noun phrase, etc.) in the predicative position in a sentence is annotated as *pd*. The *pd* function may be governed by a form of the copula verb *być* (Eng. 'to be') or copula-like verbs, e.g. *stać się* (Eng. 'to become'), *nazywać się* (Eng. 'be called').
8. *subj* – subject
   The *subj* argument is subcategorised by the sentence predicate. If *subj* takes the form of a nominative noun phrase, it must morphologically agree with the predicate in person, number and gender. If it takes the form of a noun phrase marked for a case other than the nominative, the predicate is realized as a 3rd person singular verb form marked for the neuter gender. The *subj* function

may be also realized as a sentential clause, an adjectival phrase, a numeral phrase, etc. Furthermore, the Polish subject may take the form of an elliptic pro-drop pronoun *pro*. *pro* is not encoded in a dependency structure consisting of nodes that correspond to tokens in a sentence and not any additional (artificial) nodes. In contrast to other complements governed by a predicate, *subj* is responsible for binding anaphoric expressions (reflexive and reciprocal pronouns) and may control adverbial participles in Polish.

### 2.2 Non-arguments

9. *adjunct* – adjunct
   Adjunct is a non-subcategorised dependent with the modifying function. It may be realized as an *adjective* depending on a noun or a numeral, an *adverb* depending on a verb form, an another adverb, an adjective or a prepositional phrase, an *attributive noun* marked for genitive with a nominal or numeral head, a *noun phrase* with the temporal, locative, etc. meaning and a verbal head, a *number* depending on a noun, a *past* or *present adverbial participle* with a verbal head, an *active* or *passive adjectival participle* with a nominal head, a *prepositional phrase* depending on a noun, a verb, an adverb or a participle, a *subordinate clause* with a head realized as a noun, a numeral or a verb form, a *conditional subordinate clause* depending on the sentence predicate of a matrix clause, *the question particle* (*czy*) with a verb form as its head, etc.
10. *aglt* – mobile inflection
    The *aglt* function fulfilled by a 'mobile' affix (verbal enclitic) is marked for number, person and gender. A mobile inflection may depend on a verb form or a conditional clitic *by* appended to a verb form (see (3)).




(3) *Walnął- -by- -ś go ręką?*
hit$_{PAST.PART}$ would CLITIC$_{2.SG.MASC}$ him hand$_{INSTR}$
'Would you hit him with your hand?'

11. *app* – apposition
    Apposition *app* is most commonly realized as a noun phrase depending on an immediately preceding noun or as the second noun in a noun-noun compound, e.g. *strażak-ratownik* (Eng. 'fireman-

rescuer') depending on the first one. As we currently do not distinguish named entities, a last name is annotated as the dependent of a first name.

12. *aux* – auxiliary verbs
The *aux* function is fulfilled by conjugated auxiliary verbs *być* or *zostać* (Eng. 'to be'). It depends on the main verb form (participle, infinitive) in analytical future tense constructions, analytical past conditional constructions or passive constructions.

13. *cond* – conditional clitic
The *cond* function is fulfilled by the clitic particle *by*, which may be appended to the verb form (see (3)) or may appear anywhere in a sentence. Regardless of its location, the conditional clitic depends on the verbal head.

14. *imp* – imperative particle
The *imp* function is fulfilled by the particle *niech* (Eng. 'let, may') and depends on the main verb form in analytical imperative constructions.

(4)


'May the force be with you!'

15. *mwe* – multiword expression
The successive tokens of a multiword expression are annotated according to their linear order, i.e., the first token constitutes the head of the second token which is, in turn, the head of the next token. The following tokens combinations are annotated as multiword expressions: preposition-adjectival phrases, e.g. *po prostu*$_{ADJ}$ (Eng. 'simply'), *co gorsza* (Eng. 'what is worse'), preposition-adverbial phrases, e.g. *na pewno*$_{ADV}$ (Eng. 'for sure'), *na zewnątrz*$_{ADV}$ (Eng. 'outside'), *co najmniej* (Eng. 'at least'), adverb-prepositional phrases,[3] e.g. *wraz z* (Eng. 'along with'), *zgodnie z* (Eng. 'in accordance with') (see(5)), complex conjunctions, e.g. *a więc* (Eng. 'therefore'), *nie tylko* (Eng. 'not only'), *ale także* (Eng. 'but also'), *jak i* (Eng. 'and'), *mimo że* (Eng. 'although'), *podczas gdy* (Eng. 'whereas'), adjective compounds, e.g. *biało-czerwona* (Eng. 'white-red').

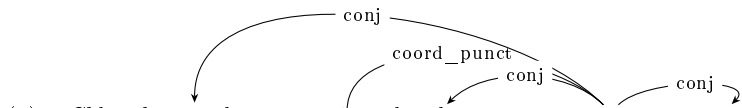
[3] Several combinations of adverbs and prepositions are regarded by Milewska (2003) as complex prepositions (Pol. 'przyimki wtórne').


(5) *Głosować będziemy zgodnie z zasadami.*
to vote be$_{1.PL.FUT}$ in accordance with rules$_{INSTR}$
'We will vote in accordance with the rules.'

16. *neg* – negation marker (negator)
The *neg* function is fulfilled by the negation marker *nie* (Eng. 'not') mostly with a verbal head immediately following it.
17. *pred* – sentence predicate (or a nominal predicate)
The *pred* function may be fulfilled by a verb form (finite verb, *-no/-to*-impersonal, infinitive) or a 'main' noun in independently annotated noun phrases. It always depends on the ROOT node.
18. *punct* – punctuation mark
The *punct* type fulfilled by a punctuation mark, e.g. *.,:;?!()"*- depends on the element which it delimits.
19. *abbrev_punct* – abbreviation marker
The *abbrev_punct* function fulfilled by a full stop depends on the preceding abbreviation.
20. *refl* – reflexive marker
The reflexive marker depends on a verb with reflexive meaning or another verb. The *refl* function is realized as the particle *się* in Polish.

### 2.3 Coordination

21. *conjunct* – coordinated conjunct
Coordinated conjuncts depend on a coordinating conjunction.
22. *coord* – coordinating conjunction
The conjunction coordinating two sentences/clauses is annotated as *coord* (see (6)). The conjunction coordinating elements other than clauses, e.g. nouns in nominal coordination, is annotated with an appropriate dependency type.
23. *coord_punct* – punctuation conjunction
If no coordinating conjunction is used to coordinate two elements, a punctuation mark, e.g. comma, colon, is used as the coordinating element (see (6)).
24. *pre_coord* – pre-conjunction
The first part of a two-part correlative conjunction, e.g. *albo... albo...* (Eng. 'either... or...'), *ani... ani...* (Eng. 'neither... nor...') depends on the second part of the conjunction.




(6) *Chłopak zerwał się , złapał posążek i ścisnął mocno.*
boy$_{NOM}$ jump up REFL , grab$_{PAST}$ statue$_{ACC}$ and press$_{PAST}$ tightly
'The boy jumped up, grabbed for a statue and pressed it tightly.'

## 3 Reordering in Dependency Structures

The conversion is a relatively straightforward process, since constituents have their syntactic centre marked in most cases. However, a certain amount of reorganising is necessary, in order to make dependency structures meet the principles of the dependency graph theory.

### 3.1 Head Selection

According to theoretical principles of the dependency graph, each token may have only one head. In Polish constituent trees, there is a centre marked in most constituents. We make use of this information and annotate each central element in a constituent tree as the head in the equivalent dependency structure. However, there are some cases where several elements have been marked as centres and we have to decide which one should be annotated as the head in the dependency structure. We identify multi-headed scenarios and define some head-selection heuristics.

The first case concerns a conditional verb form consisting of a verbal stem and a conditional clitic *by*. As both of them have been annotated as central elements of a constituent, we decide to select the verb form as the head of the conditional particle and label the relation *cond* (see (3)). Similarly, a mobile inflection, which is regarded as an independent syntactic element in Polish, may be appended to a verb form or a conditional verb form (see (3)) and all elements are regarded centres of a constituent. We annotate a verbal stem or a conditional particle as the head of the mobile inflection *aglt*.

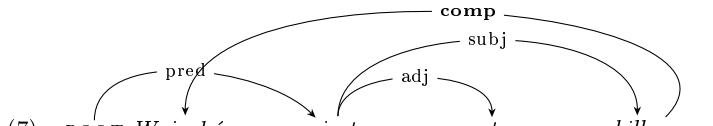
The second case of multi-headed constituents concerns analytical verb and quasi-verb forms. An analytical verb form consists of an auxiliary verb and a main verb form (infinitive, participle). The auxiliary verb fulfils some grammatical functions and constitutes a morpho-syntactic extension of the main verb. There are two auxiliary verbs in Polish: *zostać* (Eng. 'to be') used only in passive constructions and *być* (Eng. 'to be') used to build the imperfective future tense, the analytical past conditional, analytical forms of quasi-verb, e.g. *będzie trzeba* (Eng. 'it will be necessary'), analytical forms of predicative *to* and pas-

sive constructions. All parts of analytical verb forms[4] have been marked as central elements of a constituent. We convert the main verb form as the head of the auxiliary verb (*aux*).

We also have to select the head of complex subordinating or coordinating conjunctions, all parts of which have been annotated as constituent centres. In two-part subordinating conjunctions, e.g. *mimo że* (Eng. 'although'), the first token is converted as the head of the second one and the relation is labelled *mwe*. In two-part coordinating conjunctions, e.g. *albo... albo...* (Eng. 'either... or...'), the first element depends on the second one and the relation is labelled *pre_ coord*.
In case of other multi-headed constituents, we select the first element as the head of the second element, which is, in turn, the head of the next one, and so on, e.g. abbreviation, multiword expressions, series of punctuation marks.

### 3.2 Discontinuous Constituents

As the constituent annotation schema does not admit discontinuous constituents, unconnected constituent parts are encoded as 'independent' constituents. The straightforward conversion of discontinuous constituents results in dependency structures which are incompatible with annotation principles outlined in section 2. That is why we decide to reorder the dependency structure and annotate it in accordance with annotation principles, even if it results in a non-projective dependency structure (see (7)). Currently, we only identify and reorganise discontinuous numeral phrases.




### 3.3 Passive Construction

The passive voice is indicated in Polish by a conjugated auxiliary verb combined with the past or present adverbial participle. In constituent trees, auxiliary verbs constitute centres of passive constructions and participles have been annotated as adjectival phrases required by auxiliaries. Nevertheless, we annotate passive constructions by analogy to

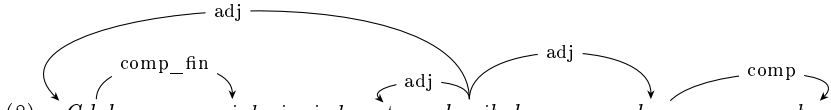
[4] We will consider passive constructions in Section 3.3, as they have been annotated differently to analytical verb forms in constituent trees.

analytical future or past conditional constructions. The participle is governed by the ROOT node and the relation is labelled *pred*. The auxiliary verb depends on the participle and the relation is labelled *aux*. Required arguments and non-subcategorised adjuncts depend on the sentence predicate.

### 3.4 Subordinate Clauses

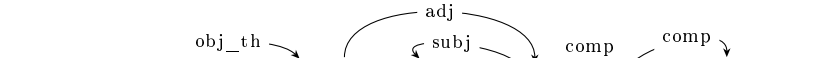
Different types of subordinate clauses with adjectival status are distinguished in the constituent treebank. We found out that annotations of subordinate clauses of the same type differ, which may violate our conversion rules.[5] We recognise particular clause types by categories of phrase structure rules and convert them uniformly.

In subordinate clauses introduced by a conjunction, e.g. *albowiem, bo, gdyż* (Eng. 'because, since'), the conjunction constitutes the head of a subordinate clause and depends on the sentence predicate of the matrix clause. Similarly, conjunctions *jeśli, gdyby* (Eng. 'if') introduce subordinate clauses and depend on the sentence predicate of the matrix clause. However, they may be accompanied by an optional particle *to* (Eng. 'then') at the beginning of the matrix clause. The particle *to* depends on the sentence predicate of the matrix clause and is labelled *adjunct* (see (8)).




(8) *Gdyby -m miał pieniądze, to kupił -by -m bez namysłu.*
if CLITIC have money$_{ACC}$ then buy COND CLITIC without reflection
'If I had money I would buy it immediately.'

In relative clauses depending on noun phrases, the clause predicate constitutes the head of the entire relative clause (see (9)). The relative pronoun is governed by the head verb and labelled according to the annotation schema.

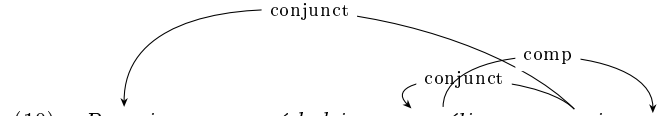



(9) *Po prostu` pomaga ludziom, którzy są w potrzebie.*
after simple help$_{PRES}$ people$_{DAT}$ who$_{NOM}$ be$_{PRES}$ in need$_{LOC}$
'He just helps people who are in need.'

[5] At the same time as the Polish constituent grammar improves, the annotation rules change and next sentences are annotated according to the new rules. However, there are still some previously annotated trees that are incompatible with the present grammar version.

### 3.5 Incorporated Conjunction

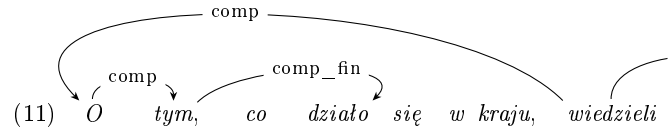
Polish admits complex sentences with a conjunction incorporated into a clause, instead of taking an initial position in this clause. We may distinguish between constructions with the coordinating conjunction, e.g. *przeto, więc, zatem* (Eng. 'therefore, then') incorporated into the second of coordinated clauses and constructions with the subordinating conjunction, e.g. *bowiem* (Eng. 'since, as'), which may appear anywhere in the subordinate clause. In constituent trees, the incorporated conjunction depends on the immediately preceding constituent, e.g. verb, adverb, noun. Even if the conversion may result in a non-projective dependency structure, we annotate complex sentences with incorporated conjunctions according to our annotation schema. The incorporated conjunction constitutes either the head of coordinated clauses (see (10)) or the head of a subordinate clause.




(10) *Pragnie pomagać ludziom, myśli więc o medycynie.*
want$_{PRES}$ to help people$_{DAT}$ think$_{PRES}$ so about medicine$_{INSTR}$
'He wants to help people, so he's thinking about studying medicine.'

### 3.6 Correlation-based Interpolation

The correlation-based interpolation Pol. *korelat* (Świdziński 1992) is a pronoun (or a pronoun in a prepositional phrase) that correlates with a subordinate clause. In constituent trees, it is the subordinate clause that is annotated as the constituent centre and the pronoun is its correlation-based interpolation. As we do not want to introduce any additional dependency type, we attempt to manage such constructions with our annotation schema. We convert the pronoun depending on the sentence predicate as the governor of the subordinate clause (see (11)).




(11) *O tym, co działo się w kraju, wiedzieli tylko notable.*
about this$_{LOC}$ what happen REFL in country know$_{PAST}$ only notables$_{NOM}$
'Only notables knew what has happened in the country.'

Besides described reordering, we do not interfere in the internal structure of constituent trees and we take syntactic facts encoded in trees as they are.

## 4 Polish Dependency Parser

The Polish dependency bank could be used as the gold standard for evaluation of Polish parsers. However, we are not aware of any publicly available Polish dependency parser. That is why we decided to train a Polish dependency parser on the part of the dependency bank using a publicly available parser-generation system. The induced parser is evaluated against two sets of dependency structures using labelled and unlabelled accuracy metrics.

### 4.1 Parsing System

The Polish dependency parser is trained with the publicly available parsing system – MaltParser[6] (Nivre et al. 2006), which uses a transition-based parsing method. A transition-based dependency parser uses a deterministic parsing algorithm that builds a dependency structure of an input sentence based on transitions (shift-reduce actions) predicted by a classifier. The classifier learns to predict the next transition given training data and the parse history. The architecture of an induced deterministic parser consists of three main components: a **parsing algorithm** deriving a labelled dependency structure from an input sentence, a **feature model** helping in prediction of the next parser action, and a treebank-induced **classifier** deterministically predicting the optimal next action given a feature representation of a parser configuration in the current state.
We have carried out a series of experiments aiming at the identification of settings of the best performing parser. We select the built-in parsing algorithm `stackeager` (Nivre 2009) designed for non-projective dependency structures. A feature model is defined in terms of token attributes, i.e., word form (FORM), coarse-grained part-of-speech tag (CPOS), part-of-speech tag (POS), morphological features (FEATS), and lemma (LEMMA) available in input data, or dependency types (DEPREL) extracted from partially built dependency graphs and updated during parsing. The chosen classifier is trained with the LIBSVM library (Chang and Lin 2001), which is an implementation of support vector machines.

### 4.2 Experiment and Results

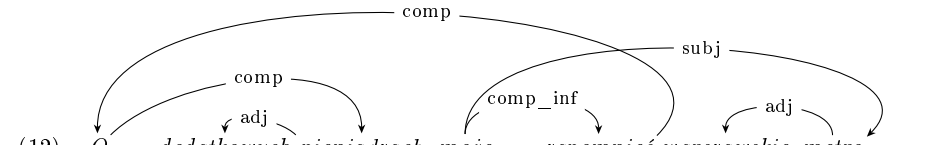
We randomly split the entire dependency bank into a training set with 6832 sentences and a validation set containing 759 sentences. It is worth mentioning that sentences in our data set are not long and consist of 9.8 tokens on average. Therefore, we assume that they have relatively

[6] We use MaltParser 1.4.1 downloaded from http://maltparser.org .

simple syntactic structures in most cases. That is why we also decide to validate the parsing quality in the realistic parsing scenario. We evaluate the parsing quality against the set of 50 manually annotated sentences (17.8 tokens/sentence) taken from two Polish magazines and the National Corpus of Polish (NKJP).[7] The performance of the Polish MaltParser is evaluated with the following metrics: *labelled attachment score* (LAS)[8] and *unlabelled attachment score* (UAS)[9].

According to our results, the Polish MaltParser achieves 88.4% LAS and 91.4% UAS if tested against the validation set (759 sentences) and 71% LAS/75.2% UAS if tested against the set of manually annotated sentences. We also evaluate individual labels in terms of precision, recall and f-measure. It is worth noting that we obtain balanced precision and recall values. It follows that if the parser finds a dependency relation between two tokens there is a great chance to label it correctly.

The cursory error analysis of the parsed sentences selected from newspapers and NKJP shows that the parser had considerable problems with analysing some syntactic phenomena, especially coordination, subordination and apposition. Furthermore, it did not find any of five multiword expressions. It follows that such constructions may be too sparsely represented in the training corpus. As Polish is a relatively free order language, sentences such as the one in (12) taken from our manually annotated bank are quite commonly used. However, such sentences are not present in the constituent bank and as a result they might not be converted to the Polish dependency bank. Training of a dependency parser on the bank lacking in dependency structures of free-ordered sentences leads to decrease in the parsing performance.




[7] We have annotated two excerpts from *Newsweek Polska* (13 sentences) and from *Zwierciadło* (Eng. 'Mirror') (4 sentences). Furthermore, we have randomly selected 33 sentences from the National Corpus of Polish.

[8] Labelled attachment score (LAS) – the percentage of tokens that are assigned a correct head and a correct dependency type.

[9] Unlabelled attachment score (UAS) – the percentage of tokens that are assigned a correct head.

## 5 Conclusions

The Polish dependency bank derived from the constituency treebank contains sentences annotated with dependency structures. As the choice of a dependency relation seems to have a crucial impact on the annotation quality, we conducted a detailed analysis of Polish dependency types. The conversion was a fully automatic and unambiguous process. However, reordering in the final dependency structures was necessary to make them meet the principles of the dependency graph theory.
As the final experiment shows, it is possible to train a Polish dependency parser on the derived dependency bank. We achieved parsing results of 88.4% LAS if evaluated against the validation set of sentences randomly selected from the entire dependency bank and results of 71% LAS in more realistic scenario of parsing newspapers excerpts. The results are quite optimistic and encourage us to continue the development of the Polish dependency bank.

## Acknowledgments

This research is supported by the POIG.01.01.02-14-013/09 project which is cofinanced by the European Union under the European Regional Development Fund.

## References

Boguslavsky, Igor, Ivan Chardin, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Iomdin, Leonid Kreidlin, and Nadezhda Frid. 2002. Development of a Dependency Treebank for Russian and its possible Applications in NLP. In *In Proceedings of the 3rd International Conference on Language Resources and Evaluation, Las Palmas, Gran Canaria*, pages 852–856.

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The PDT: a 3-level annotation scenario. In A. Abeillé, ed., *Treebanks: Building and Using Parsed Corpora*, vol. 20 of *Text, Speech and Language Technology*, chap. 7. Dordrecht: Kluwer Academic Publishers.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER Treebank. In *Proceedings of TLT-02*. Sozopol, Bulgaria.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164. Association for Computational Linguistics.

Bunt, Harry, Paola Merlo, and Joakim Nivre, eds. 2010. *Trends in Parsing Technology*. Text, Speech and Language Technology. Dordrecht: Springer.

Chang, Chih-Chung and Chih-Jen Lin. 2001. *LIBSVM: a library for support vector machines.* Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Daum, Michael, Kilian A. Foth, and Wolfgang Menzel. 2004. Automatic Transformation of Phrase Treebanks to Dependency Trees. In *In Proceedings LREC-04*, pages 1149–1152. Lisbon, Portugal.

de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating Typed Dependency Parses from Phrase Structure Parses. In *In LREC 2006*. Genoa, Italy.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8. Manchester, UK: Coling 2008 Organizing Committee.

Forst, Martin, Núria Bertomeu, Berthold Crysmann, Frederik Fouvry, Silvia Hansen-Schirra, and Valia Kordoni. 2004. Towards a Dependency-Based Gold Standard for German Parsers. The TIGER Dependency Bank. In S. Hansen-Schirra, S. Oepen, and H. Uszkoreit, eds., *COLING 2004 5th International Workshop on Linguistically Interpreted Corpora*, pages 31–38. Geneva, Switzerland: COLING.

Hajič, Jan and Eva Hajičová. 1997. Syntactic tagging in the Prague Dependency Treebank. In R. Marcinkeviciene and N. Volz, eds., *Proceedings of the Second European Seminar Language Applications for a Multilingual Europe*, pages 55–68. Kaunas, Lithuania: TELRI.

Hajič, Jan. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In E. Hajičová, ed., *Issues of Valency and Meaning.*

*Studies in Honor of Jarmila Panevová*, pages 12–19. Prague Karolinum, Charles University Press.

Kibort, Anna. 2008. On the syntax of ditransitive constructions. In *Proceedings of the LFG08 Conference*, pages 312–332.

King, Tracy Holloway, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *In Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8.

Kübler, Sandra, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Mel'čuk, Igor. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Milewska, Beata. 2003. *Przyimki wtórne we współczesnej polszczyźnie*. Gdańsk: Wydawnictwo Uniwersytetu Gdańskiego.

Nivre, Joakim. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, ACL '09, pages 351–359. Association for Computational Linguistics.

Nivre, Joakim, Igor M. Boguslavsky, and Leonid L. Iomdin. 2008. Parsing the SynTagRus treebank of Russian. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 641–648.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2006. MaltParser: a data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219.

Obrębski, Tomasz. 2002. *Automatyczna analiza składniowa języka polskiego z wykorzystaniem gramatyki zależnościowej*. Phd thesis, Institute of Computer Science, Polish Academy of Sciences, Warsaw.

Obrębski, Tomasz. 2003. MTT-compatible computationally effective surface-syntactic parser. In *Proceedings of First International Conference on Meaning-Text Theory*, pages 259–268. Paris.

Przepiórkowski, Adam, Rafał L. Górski, Marek Łaziński, and Piotr Pęzik. 2010. Recent developments in the National Corpus of Polish. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation, LREC2010*. ELRA, Valetta, Malta.

Saloni, Zygmunt and Marek Świdziński. 1998. *Składnia współczesnego języka polskiego*. Warszawa: Państwowe Wydawnictwo Naukowe.

Świdziński, Marek. 1989. A Dependency Syntax of Polish. In D. Maxwell and K. Scubert, eds., *Metataxis in Practice. Dependency Syntax for Multilingual Machine Translation*, pages 69–88. Dordrecht: Foris Publications.

Świdziński, Marek. 1992. *Gramatyka formalna języka polskiego*. Rozprawy Uniwersytetu Warszawskiego. Warszawa: Wydawnictwa Uniwersytetu Warszawskiego.

Świdziński, Marek and Marcin Woliński. 2010. Towards a Bank of Constituent Parse Trees for Polish. In P. Sojka, ed., *Text, Speech and Dialogue, 13th International Conference, TSD 2010, Brno, September 2010, Proceedings*, vol. 6231 of *LNAI*, pages 197–204. Heidelberg: Springer.

Woliński, Marcin. 2010. Dendrarium – an Open Source Tool for Treebank Building. In M. A. Kłopotek, M. Marciniak, A. Mykowiecka, W. Penczek, and S. T. Wierzchoń, eds., *Intelligent Information Systems*, pages 193–204.