Rule Based Dependency Parser for Polish Language

Marek Korzeniowski and Jacek Mazurkiewicz(✉)

Department of Computer Engineering, Faculty of Electronics,
Wroclaw University of Science and Technology,
ul. Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
borsuczek@gmail.com, Jacek.Mazurkiewicz@pwr.edu.pl

Abstract. The paper presents a dependency parser for Polish language. It uses a simple chain of word combining rules operating on fully morphosyntactically tagged input instead of a formal grammar model or statistical learning. The proposed approach generates robust dependency trees and allows parsing of uncommon texts, such as poetry. This gives a significant advantage over current state-of-the-art dependency parsers.

Keywords: NLP · Text parsing · Dependency parsing · Polish language

1 Introduction

Currently there are two leading dependency parsers available for Polish language: Świgra and "Polish Dependency Parser" [6,8]. The first is based on a formal grammar model of Polish language created by Marek Świdziński [4]. The second represents a data-driven approach created using Maltparser and Składnica – a data-driven parser generator and a data set of fully tagged and parsed Polish sentences [2,7].

These solutions work well for multiple applications but problems arise when one tries to analyse non-standard texts, like poetry. For complex sentences Świgra often does not return a result in reasonable time – the computing process often takes longer than half an hour for a single sentence. "Polish Dependency Parser" used on texts drastically different from the ones it was trained on returns gibberish.

Poetry-parsing capabilities are a completely unnecessary feature for a real-world usage. Another project we worked on required a database of words used in Polish poetry with information about semantic and syntactic relations between them. For such analyses a dependency parser was needed.

We propose a completely different approach to dependency parsing: a method based on a chain of simple heuristic rules, operating on words and their morphological tags. Each rule removes a word from the input and attaches it to a different word – effectively joining them into a parent-child pair. During the final step the last word is removed and returned as the root of the dependency tree. If no rule can be applied to a given word the input is discarded as unrecognised.

L. Rutkowski et al. (Eds.): ICAISC 2017, Part II, LNAI 10246, pp. 498–508, 2017.
DOI: 10.1007/978-3-319-59060-8_45

The proposed parser can analyse any kind of grammatically correct texts and return correctly parsed sentences only. Its operating time is insignificant when compared to the time of morphosyntactic tagging – the preliminary step of this process. For morphosyntactic tagging the WCRFT2 tagger was used [3].

The output trees are quite crude and carry much less information compared with trees returned by other parsers. Nevertheless, this method can be used to quickly find relations in complex texts, especially when other methods would fail to generate any useful results.

2 Preliminary Definitions

2.1 Parts of Speech Archetypes

The morphological tagset used by the WCRFT2 tagger defines over thirty parts of speech [5]. The following archetypes were introduced to reduce the complexity of the parser:

- noun (*subst*, *depr*, *ppron12*, *ppron3*, *siebie* and *ger*) representing parts of speech that can become the subject of a sentence,
- verb (*fin*, *bedzie*, *praet*, *impt*, *imps* and *winien*) representing parts of speech that can become the predicate,
- adjective (*num*, *numcol*, *adj*, *pact* and *pant*) representing parts of speech that describe nouns,
- adverb (*adja*, *adjp*, *adjc*, *adv*, *inf*, *pcon*, *pant* and *pred*) representing parts of speech that describe verbs.

2.2 Dependency Trees

The roots of the dependency trees returned by the parser are always a verb. In Polish language a simple sentence can only have a single verb and this verb is always the predicate. In most cases a compound sentence can easily be devised into sub-sentences, each contacting a single verb being the predicate. Such approach can not be used for verb-less sentences which are quite common in Polish. Such sentences are rejected by the algorithm as unrecognised.

Interjections in sentences present another problem. In most cases such sentences are rejected as unrecognised, but sometimes they can also yield incorrect results. For example, the sentence *"Znalazłem, choć nie szybko, króla."* can be word-by-word translated to "I found, but not fast, the king." with "but not fast" as the interjection informing that we looked for the king for quite some time. The parser will recognise "I found" as a correct sub-sentence and discard the rest, while the correct sub-sentence should be "I found the king".

The root (verb) of the dependency tree can have child nodes being nouns, adverbs and prepositions. Prepositions always have one child which needs to be a noun. Adverbs can have children being other adverbs or adjectives. Nouns can have children being other nouns, adjectives or prepositions. Adjectives can have

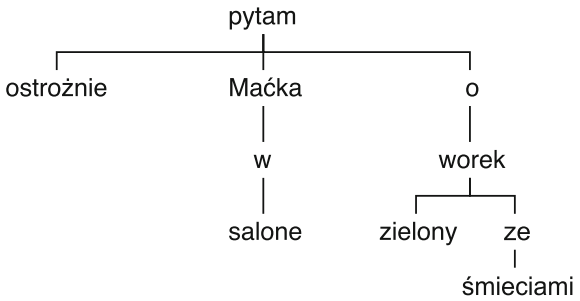

Fig. 1. Graphical representation of the dependency tree of the sentence *"Ostrożnie pytam się Maćka w salonie o zielony worek ze śmieciami."*

a single child – also being an adjective – only if they represent a compound numeral.

Parts of speech not included in the archetypes, or not being a preposition, are simply ignored and they are not included in the output.

In the paper a compact notation will be used to represent the dependency trees. A node will be represented by a word. If the node is not a leaf node, its children will be presented as a comma-separated list enclosed in square brackets. For example: the sentence *"Ostrożnie pytam się Maćka w salonie o zielony worek ze śmieciami."* which roughly translated to "Cautiously I ask Mike in the showroom about the green bag with trash" has dependency tree shown in Fig. 1 that can be written as:

```
pytam[ ostrożnie, Maćka[ w[ salonie ] ], o[ worek[ zielony,
ze[ śmieciami ] ] ] ] ]
```

English version:

```
(I) ask[ cautiously, Mike[ in[ showroom ] ], about[ bag[
green, with[ trash ] ] ] ] ]
```

3 Parsing Rule Chain

To demonstrate the state of the input text after each step of the parser's rule chain the following example sentence will be used: *"Ostrożnie zapytałem bardzo czerwoną dziewczynkę z kapturkiem równie czerwonym o zapałki, a następnie niechętnie kupiłem od niej dwadzieścia dwa pomidory i jedną marchew"*. Word-by-word translation: "(I) carefully asked (a) very red girl with (a) hood equally red about matches, and afterwards reluctantly bought from her twenty-two tomatoes and (a) single carrot". Words in parenthesis have been inserted into the translation to make it grammatically correct.

The complete parsing rule-chain is summarized in Fig. 2.

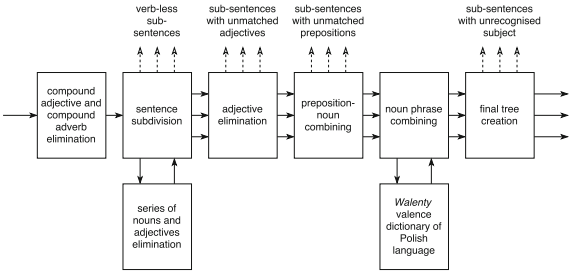

Fig. 2. The parser's rule-chain illustrated on a single diagram. The input is in form of a single sentence. The output is in form of multiple dependency trees – one for each sub-sentence that the input was composed from. While traversing the rule-chain, sub-sentences can get discarded as unrecognised for numerous reasons – shown on the diagram in form of dashed lines.

3.1 Compound Adjective and Compound Adverb Elimination

The first step of the parsing rule chain can be described as a simple iteration over all input words in search of compound numerals, adverb-adjective or adverb-adverb combinations.

If a compound numeral is found, it is replaced by a tree having the least significant numeral in its root. For example a numeral *"sto dwadzieścia dwa"* ("one hundred and twenty-two") would be replaced by "`dwa[ dwadzieścia[ sto ] ]`" (in English "`two[ twenty[ one hundred ] ]`").

If an adverb followed by a different adverb or adjective is found, the two words are replaced by a two-element tree with the second word as the root. This rule transforms phrases like *"niewiele szybciej"* ("slightly faster") or *"niesamowicie zielony"* ("amazingly green") into trees "`szybciej[ niewiele ]`" and "`zielony[ niesamowicie ]`" (in English "`faster[ slightly ]`" and "`green[ amazingly ]`").

The state of the example sentence can be written as:

```
Ostrożnie zapytałem czerwoną[ bardzo ] dziewczynkę z
kapturkiem czerwonym[ równie ] o zapałki, a następnie
niechętnie kupiłem od niej dwa[ dwadzieścia ] pomidory i
jedną marchew.
```

English version:

```
(I) carefully asked (a) red[ very ] girl with (a) hood red[
equally ] about matches, and afterwards reluctantly bought
from her two[ twenty ] tomatoes and (a) single carrot.
```

It should be noted that all trees inserted into the text are treated as single words – equivalent with their roots. All rules described in the paper can by applied to trees as if they were words.

3.2 Sentence Subdivision

In Polish language every two verbs need to be separated by a comma or a conjunction – they create different sub-sentences. The problem is that commas and conjunctions can also be used to separate series of nouns and adjectives. A state machine was created to resolve this issue. Its task was to join series of nouns and adjectives into special trees. For example the phrase *"Znalazłem psa i kota"* would be replaced by "`Znalazłem @[ psa, kota ]`" (in English "`I found @[ cat, dog ]`") where '@' is a placeholder plural noun with appropriately derived case and gender. The machine's state diagram is shown in Fig. 3.

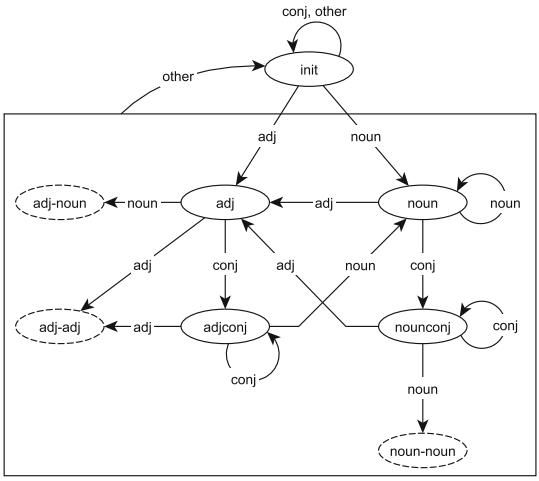

Fig. 3. State diagram of the finite-state machine responsible for finding series of nouns and adjectives. "conj" is an abbreviation of conjunction.

The state machine starts at the initial state "init" and takes whole words with their tags as input. If the machine stops in an accepting state a corresponding action is taken and the whole process is repeated. There are three accepting states:

noun-noun two nouns are joined into a tree with a '@' in its root. All words between them are removed.

adj-adj two adjectives are accumulated into a temporary list. All words between them are removed. This list needs to be joined with a noun in one of the following state machine passes.

adj-noun a noun and an adjective (or an adjective list) are joined into a tree. All words between them are removed.

The process ends when the machine is not in an accepting state after processing the last word of the input. Then the input can be safely divided into separate sub-sentences on every comma and conjunction.

There are some rare cases where the state machine will incorrectly detect series of nouns. Consider the following sentence: *"Znalazłem psa i kota tym psem poszczułem"* which can be translated to "I found a dog and I let it loose on a cat". To demonstrate the problem a word-by-word translation is more suitable: "(I) found (a) dog and cat this dog (I let) loose". The first part of the sentence "(I) found (a) dog and cat" simply means "I found a dog and a cat", but when the ending "this dog (I let) loose" is added, the meaning changes to "I found a dog and I let it loose on a cat".

The state of the example sentence can be written as:

```
Ostrożnie zapytałem dziewczynkę[ czerwoną[ bardzo ] ] z
kapturkiem czerwonym[ równie ] o zapałki,
a następnie niechętnie kupiłem od niej @[ pomidory[ dwa[
dwadzieścia ] ], marchew[ jedną ] ].
```

English version:

```
(I) carefully asked (a) girl[ red[ very ] ] with (a) hood
red[ equally ] about matches,
and afterwards reluctantly bought from her @[ tomatoes[
two[ twenty ] ], carrot[ single ] ].
```

3.3 Adjective Elimination

The previous step eliminates practically all adjectives by connecting them with nouns. In some rare cases adjectives can appear not directly before the nouns described by them. For example: *"W czarodziejskiej cię odwiedziłem wieży"* ("I visited you in a magicians tower") with a word-by-word translation "In magicians you (I) visited tower". Such constructions are not used in common spoken or written language, but they can be found in a poetry. The third step of the parsing rule chain was created to handle such cases.

For each adjective in a sub-sentence the closest noun with a compatible morphological tag (same gender, case and number) is found and the two are joined together or the sub-sentence is discarded as unrecognised.

In Polish adjectives can also be subjects, like in the sentence *"Szczęśliwi nie znają litości"* ("The happy have no mercy"). Such constructions unfortunately get discarded as unrecognised in this step.

The state of the example sentence can be written as:

```
Ostrożnie zapytałem dziewczynkę[ czerwoną[ bardzo ] ] z
kapturkiem[ czerwonym[ równie ] ] o zapałki,
a następnie niechętnie kupiłem od niej @[ pomidory[ dwa[
dwadzieścia ] ], marchew[ jedną] ].
```

English version:

```
(I) carefully asked (a) girl[ red[ very ] ] with (a) hood[
red[ equally ] ] about matches,
and afterwards reluctantly bought from her @[ tomatoes[
two[ twenty ] ], carrot[ single ] ].
```

3.4 Preposition-Noun Combining

For each preposition in the input the next word is checked. If it is a noun, the two words are joined together into a tree with the preposition in its root. If the next word is not a noun the entire sub-sentence is discarded as unrecognised.

The state of the example sentence can be written as:

```
Ostrożnie zapytałem dziewczynkę[ czerwoną[ bardzo ] ] z[
kapturkiem[ czerwonym[ równie ] ] ] o[ zapałki ],
a następnie niechętnie kupiłem od[ niej ] @[ pomidory[ dwa[
dwadzieścia ] ], marchew[ jedną] ].
```

English version:

```
(I) carefully asked (a) girl[ red[ very ] ] with[ hood[
red[ equally ] ] ] about[ matches ],
and afterwards reluctantly bought from[ her ] @[ tomatoes[
two[ twenty ] ], carrot[ single ] ].
```

3.5 Noun Phrase Combining

In Polish every noun in a sentence can to be connected with the predicate or with an adjacent noun. The connection can be direct or via a preposition.

Two adjacent nouns that are not separated by a preposition should be connected only if the second noun is in genitive case. This rule is true in most of the cases. If the two nouns are separated by a preposition there is no simple way to determine if they should be connected or not.

This problem was solved using Walenty – a valence dictionary of Polish language [1]. Two nouns separated by a preposition are allowed to be joined only if the predicate cannot join with the preposition separating the nouns. This is determined by looking up semantic frames bound with the predicate in Walenty.

Many verbs in Walenty have incomplete entries. This led to a quite high error rate of this step. To radically decrease the number of incorrectly joined nouns a special table of allowed noun-preposition-noun semantic frames was created. Two nouns separated by a preposition can be joined together only if they represent a construct listed in this table. The table has a purely heuristic nature and was created empirically from a list of most common noun phrases in Polish language. It's content is shown in Table 1. To summarise – two adjacent nouns are joined together if:

1. They are not separated by a preposition and the second one is in genitive case.
2. They are separated by a preposition and according to Walenty the predicate cannot join with the preposition. They also represent a construct listed in the table of allowed semantic frames.

Table 1. Table of allowed noun-preposition-noun semantic frames.

Prep.	Second noun case	Example phrase
do	Genitive	*maszyna do pisania* (machine for typing)
o	Locative	*książka o ptakach* (a book about birds)
o	Accusative	*walka o życie* (a fight for life)
z	Ablative	*worek z kotem* (a bag with a cat)
z	Genitive	*człowiek z miasta* (a man from a city)
na	Accusative	*worek na śmieci* (a bag for trash)

In every other case the two nouns are not joined together.

Nouns are processed from right to left. The whole noun-phrase combining can be described as applying a right associative binary operator to each adjacent noun pair in the input text.

The state of the example sentence can be written as:

```
Ostrożnie zapytałem dziewczynkę[ czerwoną[ bardzo ], z[
kapturkiem[ czerwonym[ równie ] ] ] ] o[ zapałki ],
a następnie niechętnie kupiłem od[ niej ] @[ pomidory[ dwa[
dwadzieścia ] ], marchew[ jedną] ].
```

English version:

```
(I) carefully asked (a) girl[ red[ very ], with[ hood[ red[
equally ] ] ] ] about[ matches ],
and afterwards reluctantly bought from[ her ] @[ tomatoes[
two[ twenty ] ], a carrot[ single ] ].
```

3.6 Final Tree Creation

The predicate is placed in the root of the output tree. All adverbs, preposition and nouns left in the input are added as its children. All other words are discarded.

The subject of the sentence is verified before returning the result. In Polish the subject is a noun in nominative case. If such noun is not found – the sentence is accepted as correct (is has a "default" subject). If there is exactly one such noun, the sentence is also accepted. In all other cases the sentence is discarded.

The example sentence after the complete rule-chain has the form of the following two trees:

```
zapytałem[ ostrożnie, dziewczynkę[ czerwoną[ bardzo ], z[
kapturkiem[ czerwonym[ równie ] ] ] ], o[ zapałki ] ]
kupiłem[ następnie, niechętne, od[ niej ], @[ pomidory[
dwa[ dwadzieścia ] ], marchew[ jedną] ] ]
```

English version:

```
asked[ carefully, girl[ red[ very ], with[ hood[ red[
equally ] ] ] ], about[ matches ] ]
bought[ afterwards, reluctantly, from[ her ], @[ tomatoes[
two[ twenty ] ], carrot[ single ] ] ]
```

4 Results

To test the created parser a corpus of Polish poetry was prepared. It was build from the work of classical Polish poets such as Adam Mickiewicz and Bolesław Leśmian. An initial preprocessing stage was applied to remove or replace all characters not being a letter, comma, punctuation mark, question mark or an exclamation mark. The text was tagged using the WCRFT2 tagger and supplied as input to the parser. In 24219 input sentences 71051 sub-sentences were found and:

- 36052 were accepted,
- 27882 were discarded as verb-less,
- 3670 were discarded in the adjective elimination step,
- 3448 were discarded for other reasons.

The parsing took 0.45 s on a single 3 GHz core of an Intel i7 processor. 4253 sentences were discarded before parsing because of words unknown to the tagger.

A comparison of results obtained from other parsers was not conducted as currently there are no alternatives for the constructed parser. The parser's purpose was to process poetry – texts that are practically unparsable by other parsers. Only simple texts could be used for a meaningful comparison. Such tests were considered unnecessary as they would not provide any useful information.

A few examples of the implemented parser's output are as follows:

"Chłopiec dorósł młodzieńca, w obce pokolenia, w dalekie zbłądził kraje i pod wschodnim słońcem poił duszę płomieniem, on wiecznym był gońcem na lądzie i na morzu." — Adam Mickiewicz *Sen z Lorda Byrona*

```
dorósł[ chłopiec, młodzieńca ]
/discarded/
zbłądził[ w[ kraje[ dalekie ] ] ]
poił[ pod[ słońcem [ wschodnim ] ], duszę, płomieniem ]
był[ on, gońcem[ wiecznym ], na[ lądzie ] ]
/discarded/
```

"Tu mnóstwo scen okropnych snuło się od razu, i ów chłopiec był cząstką każdego obrazu." — Adam Mickiewicz *Sen z Lorda Byrona*

```
snuło[ tu, mnóstwo[ scen[ okropnych ] ], od[ razu ] ]
był[ chłopiec[ ów ], cząstką[ obrazu [ każdego ] ]
```

"Te przypomnienia Kirem śmiertelnej zasłony oddzieliły na wieki młodzieńca od żony, po cóż w takiej godzinie takie przypomnienia?" — Adam Mickiewicz *Sen z Lorda Byrona*

```
oddzieliły[
  przypomnienia[ te ],
  Kirem[ zasłony[ śmiertelnej ] ],
  na[ wieki[ młodzieńca ] ],
  od[żony ] ]
/discarded/
```

5 Conclusions

The paper proposes a new approach to dependency parsing. A parser has been implemented and tested on Polish poetry – texts that are mostly unparsable by current state-of-the-art dependency parsers.

The largest drawback of the parser is the lack of support for verb-less sentences which leads to approximately 40% loss of the test corpus. This problem should be resolved before the parser can be considered as a fully functional tool. However, even in its current state it can be used for crude dependency analysing of texts which so far could not have been analysed at all.

Another problem of the parser is high sensitivity to tagging errors. In most cases a miss-tagged word leads to discarding a whole sub-sentence but sometimes in can lead to generating incorrect output. This usually happens when the tagger assigns a wrong part-of-speech to a word.

Even though the described solution was created strictly for Polish, it could be applied to other languages. Especially those from the Slavic group, which prove to be hard to formalize. The parser's rule-chain would need to be modified to accommodate the target grammar, but the approach described in this paper would remain the same.

References

1. Andrejewicz, J., et al.: Walenty: CLARIN-PL Digital Repository (2016). http://hdl.handle.net/11321/251
2. Nivre, J., Hall, J., Nilsson, J.: Maltparser: a data-driven parser-generator for dependency parsing. In: Proceedings of LREC, vol. 6, pp. 2216–2219 (2006)
3. Radziszewski, A., Warzocha, R.: WCRFT2: CLARIN-PL Digital Repository (2014). http://hdl.handle.net/11321/36
4. Swidzinski, M.: Formal Grammar of Polish Language. No. 349, Warsaw University (1992). (in Polish)
5. Wolinski, M.: System of morphosyntactic markers in IPI PAS frame. Polonica XXII-XXIII, pp. 39–55 (2003). (in Polish)
6. Wolinski, M.: Swigra: CLARIN-PL Digital Repository (2016). http://hdl.handle.net/11321/258

7. Wolinski, M., Glowinska, K., Swidzinski, M.: A preliminary version of skladnica-a treebank of Polish. In: Proceedings of the 5th Language & Technology Conference, Poznan, pp. 299–303 (2011)
8. Wroblewska, A.: Polish Dependency Parser Trained on an Automatically Induced Dependency Bank. Ph.D. dissertation, Institute of Computer Science, Polish Academy of Sciences, Warsaw (2014)