Towards a Partial Grammar of Polish for Valence Extraction

Adam Przepiórkowski
Institute of Computer Science
Polish Academy of Sciences
Warsaw, Poland
adamp@ipipan.waw.pl

1. Introduction

This paper is a report on work in progress aiming at the construction of a partial grammar of Polish (work supported by the Polish State Research Project, 3 T11C 003 28, *Automatyczna ekstrakcja wiedzy lingwistycznej z dużego korpusu języka polskiego*; English: *Automatic extraction of linguistic knowledge from a large corpus of Polish*; from March 2005 to March 2008). The grammar will be used for the parsing of the IPI PAN Corpus of Polish (Przepiórkowski 2004; http://korpus.pl/) and the parsing results will constitute the empirical basis for the automatic extraction of morphosyntactic valence (argument structure, subcategorisation) information.

The outline of the paper is as follows. Section 2. very briefly describes the motivation behind and the methodology of the automatic acquisition of valence from corpora and Section 3. introduces a new formalism for partial parsing. The main part of this paper, Section 4., sketches a partial grammar of Polish and gives specific examples of grammar rules. Section 5. concludes the paper.

2. Automatic Acquisition of Valence Dictionaries

Although there exists a large valence dictionary of Polish, Polański 1980–1992, and a more recent general dictionary contains explicit valence information (Bańko 2000), there are good reasons for developing automatic valence acquisition systems. First of all, once developed, automatic methods of inferring valence information from corpora may be applied to various collections of texts, thus yielding various specialised (thematic, diachronic) valence dictionaries. Second, automatically constructed dictionaries are bound to be more objective than manually constructed lexica, as the latter are tinted by prescriptive knowledge and potentially conflicting intuitions of a team of lexicographers. Third, the automatic procedure will result in an electronic form of a valence dictionary containing frame probabilities for different valence frames of the same verb. This

information is very useful for probabilistic syntactic parsers.

The usual valence acquisition algorithms may be divided into two stages: a linguistic stage and a statistical stage. The first stage consists in identifying clauses and, within each clause, in finding verbs and top-level phrases. Such phrases, if identified correctly, are dependents (arguments and adjuncts) of the verb. However, in practice, this linguistic stage produces some noise due to errors in the morphosyntactic and syntactic processing; for example, an accusative phrase may be recognised as genitive due to tagging errors, or a single nominal phrase involving a prepositional post-modifier may be recognised by the syntactic parser as two phrases: nominal and prepositional.

For this reason, the results of the linguistic processing are usually subjected to statistical inference rules which help decide which of the linguistic observations may be trusted, and which are consequences of processing errors. Przepiórkowski and Fast 2005 describe this procedure in greater detail and compare the performance of a few basic statistics. In the current paper we will concentrate on the first stage of valence acquisition, i.e., on partial parsing.

3. ♠: Shallow Parsing and Disambiguation Engine

Since the empirical basis for the work reported here is the IPI PAN Corpus of Polish, which has been automatically tagged and whose source version adheres to the XML Corpus Encoding Standard (XCES), a new tool has been developed for the partial parsing of XCES-encoded texts and for the simultaneous correction and disambiguation of morphosyntactic interpretations. The formalism and the tool, called ♠ (pronounced *spade*), is presented in Buczyński and Przepiórkowski 2007 and in articles cited there, so this section only mentions the main characteristics of ♠.

Any ♠ grammar is a cascade of regular grammars, where the output of one regular grammar is the input to the next regular grammar. It is assumed that the input to the cascade is morphosyntactically annotated, although not necessarily disambiguated. Each regular grammar is encoded as one perhaps very complex rule, containing up to five parts: 1) `Rule` (obligatory, but often omitted in the examples below), containing the name of the rule, 2) (optional) `Left`, specifying the left context for the application of the rule, 3) (obligatory) `Match`, specifying the text on which the rule performs, 4) (optional) `Right`, specifying the right context, and 5) (obligatory) `Eval`, containing a list of actions to be performed. For example:

```
Rule    "Po co"
Match:  [pos~~"prep"][base~"co|kto"];
Eval:   unify(case,1,2); group(PG,1,2);
```

This rule, called “Po co”, does not specify any left or right context, i.e., it will be used whenever text matching the `Match` specification is found. Text specifications are based on the query language

of the Poliqarp search engine (Janus and Przepiórkowski 2006) and, in particular, a difference is made between the universal and the existential quantification over morphosyntactic interpretations of tokens. Thus, in the rule above, the specification `[pos~~"prep"]` requires that all (cf. `~~`) interpretations of the token be prepositional, while `[base~"co|kto"]` requires that among base forms of the token there should be a form satisfying the regular expression `co|kto`, i.e., the form *co* or the form *kto*.

Once the match is found, predicates in the `Eval` part are evaluated in turn, until the end or until one of them evaluates to false (e.g., in case of the lack of agreement specified by the predicate `agree`). In the rule above, there are two such predicates: `unify(case,1,2)` deletes any interpretations of the token found by the first specification in `Match` (i.e., by `[pos~~"prep"]`; cf. `1`) and the token found by the second specification (i.e., by `[base~"co|kto"]`) which do not agree in case, while `group(PG,1,2)` creates a syntactic group of type `PG` (Prepositional Group), whose syntactic head is given by the first specification (a preposition), and whose semantic head is given by the second specification (a form of *co* or *kto*).

All predicates currently implemented in ♠ may be divided into two classes: morphosyntactic and syntactic. The former are pure conditions (`agree`), deletion operations (`unify`, `leave`, `delete`) and an operation for adding morphosyntactic interpretations not already present among the interpretations of a token (`add`). The syntactic operations may create syntactic words (i.e., tokens larger than those already present in the input; operation `word`) and syntactic groups (`group`). The use of these operations will be illustrated below.

4. Towards a Partial Grammar of Polish

The grammar is still under development and it currently contains 167 rules (not necessarily unique; some rules are repeated). The grammar is split into a number of sections for the ease of maintenance.

The first section contains rules which correct some of the errors of the morphological analyser used for morphosyntactic annotation, Morfeusz (Woliński 2006), and which delete extremely rare interpretations of some tokens. An example rule of this kind, which deletes all adverbial particle (so-called *qublic;* cf. Przepiórkowski 2004 for the tagset assumed here) interpretations of *mu* (i.e., the onomatopeic / interjection interpretation; it leaves the usual and most probably the intended pronominal interpretations), is given below:

```
Match: [orth~"mu"];
Eval:  delete(pos~qub,1);
```

Another rule of this class adds feminine interpretations to nouns such as *poseł* 'deputy', *minister* 'minister', etc., which only get masculine interpretations from the current version of Morfeusz:

```
Match: [orth~"[Pp]oseł|[Mm]inister|[Pp]rezydent"];
Eval:  add(subst:sg:case*:f,,1);
```

The first argument of `add` specifies the interpretations to be added to the token indicated by the third argument (here: `1`). In this case, 7 substantive singular feminine interpretations are added, for the 7 cases in Polish (cf. the abbreviation `case*`). The middle argument specifies the base form for these interpretations; if it is omitted, as above, the base form of an already existing interpretation is copied to the new interpretations.

The second section contains disambiguation rules. For example, *nie*, which is ambiguous between the negative marker (a *qublic*) and various post-prepositional prenominal interpretations, must be interpreted as the negative marker when occurring at the beginning of a sentence (cf. `sb`):

```
Left:  sb;
Match: [orth~"[Nn]ie"];
Eval:  leave(pos~"qub", 2);
```

Note that `2` above refers to the specification `[orth~"[Nn]ie"]` (`1` would refer to `sb`).

The third section finds abbreviations, which usually are sequences of tokens (with the full stop treated as a separate token), and turns them into syntactic words, so that token specifications in further rules may apply to them. For example, the sequence of *np* or *Np* followed by the full stop but with no space (`ns`) in between (i.e., an abbreviation meaning 'e.g.') is turned into a syntactic word with the single qublic interpretation and the base form *na przykład* 'for example':

```
Match: [orth~"[Nn]p"] ns [orth~"\."];
Eval:  word(qub, "na przykład");
```

The fourth section finds other sequences of tokens which might be reasonably treated as single syntactic words, e.g., *po ciemku* 'in the dark', where *ciemku* is a Polish bound word which always occurs after *po*.

```
Match: [orth~"[Pp]o"] [orth~"[Cc]iemku"];
Eval:  delete(pos~"subst",2); word(qub, "po ciemku");
```

The fifth section recognises various Named Entities and marks them as syntactic words or syntactic groups, as appropriate. For example, the rule below find various forms of *Urząd Regulacji*

Telekomunikacji i Poczty 'Office of Telecommunications and Post Regulation', constructs a syntactic group syntactically and semantically headed by a form of *Urząd* 'Office', and leaves only the genitive singular interpretations of *Regulacji*, *Telekomunikacji* and *Poczty*.

```
Rule    "Urząd Regulacji Telekomunikacji i Poczty"
Match:  [base~"urząd"] [orth~"Regulacji"] [orth~"Telekomunikacji"] [orth~"i"] [orth~"Poczty"];
Eval:   group(NG,1,1);
        leave(case~"gen" && number~"sg", 2);
        leave(case~"gen" && number~"sg", 3);
        leave(case~"gen" && number~"sg", 5);
```

The next four sections are devoted to the recognition of numbers (also written in digits, containing abbreviations such as *mln.* 'million', modified by *około* 'around', etc.), currency expressions, times of day (hours) and dates. Numbers are treated as syntactic words with a special tag (not present in the standard IPI PAN Tagset) called *liczba* 'number', e.g.:

```
Match:  [orth~"[1-9][0-9]*"] (ns [orth~"[,-]"] ns [orth~"[0-9]+"])?;
Eval:   word(liczba,1.orth);
```

This rule marks as *liczba* sequences such as *123*, *12-15* and *123,45*. The specification of the base form above, `1.orth`, indicates — somewhat arbitrarily — that the base form of the whole syntactic word should just be the orthographic form of the first token.

The two rules below are examples of date identification. The first of these rules recognises sequences like *2000 roku* '(in the) 2000 year' and makes them into a group of type `Year_NG` headed by a form of *rok* 'year'. Note that this rules relies on previous rules correctly identifying numbers as syntactic words and assigning them the tag *liczba*. Note also that the regular expression `[12][0-9]{3}` picks out numbers between 1000 and 2999, to increase the probability that the number expresses a year (this range would have to be extended for historical texts).

```
Rule    "2000 roku"
Match:  [pos~"liczba" && orth~"[12][0-9]{3}"] [base~"rok"];
Eval:   group(Year_NG,2,2);
```

```
Rule    "dzień + day + Month_NG + Year_NG?"
Match:  [base~"dzień"] [pos~"liczba" && orth~"[1-9]|[12][0-9]|3[01]"]
        (ns [orth~"-"] ns [orth~"go"])? [type="Month_NG" && synh=[case~"gen"]]
        ([type="Year_NG"] | [pos~"liczba" && orth~"[12][0-9]{3}"])?;
Eval:   group(DayMonth_NG,1,1);
        leave(case~~"gen",4,5);
```

The second date rule above is much more complex and it recognises expressions like *dnia 16 września 2007* '(on the) day (of) 16 September 2007' or *dniu 16-go września 2007 roku* '(on the) day (of) 16th September 2007 year'. Unlike any of the rules given above, this rule contains specifications of groups (which should be recognised by earlier rules, for this rule to work). For example, the specification `[type="Month_NG" && synh=[case~"gen"]]` targets groups of type `Month_NG`, whose syntactic head has a genitive interpretation.

One section is devoted to the proper recognition of affirmative and negative forms of verbs, where the main problem stems from the ambiguity of *nie*, which may be a pronominal form, if occurring after an accusative-taking preposition (see above). The specification of the left context in the simplified rule given below makes sure that *nie* is the negative marker, and `leave(pos~"qub", 2)` removes all other interpretations of this token.

```
Left:   (sb | [case!~"acc"] | [pos!~"prep"]);
Match:  [orth~"[Nn]ie"] [pos~~"praet|fin|impt|imps|inf"];
Eval:   word(3, neg, base); leave(pos~"qub", 2);
```

Note that a different, 3-argument version of the predicate `word` is used here: the first argument points at the token whose interpretations are the basis for the interpretations of the whole syntactic word (here, it is the verbal token specified by `[pos~~"praet|fin|impt|imps|inf"]`), the second argument specifies what must be done to each of these interpretations (information about negation should be added), while the third argument specifies base forms for these interpretations (here, base forms of the corresponding interpretations of the verbal token are copied; cf. `base`).

Finally, the last two sections specify various syntactic groups, including clauses: one section contains rules which are (almost) completely correct, and another consists of heuristic rules. Here is a much simplified version of such a heuristic rule, treating as a contiguous nominal group a sequence of any number of adjectives, a noun and a genitive (nominal or numeral) group, if only this sequence occurs between any groups recognised by earlier rules (the specification `[synh=[]]` in effect puts no conditions on the syntactic head of such a group; the attribute `synh` is only used to indicate that this is a group, and not a token):

```
Left:   [synh=[]];
Match:  [pos~"adj"]* [pos~~"subst"] [synh=[pos~~"subst|num" && case~~"gen"]];
Right:  [synh=[]];
Eval:   unify(case number gender,2,3);
        group(NG,3,3);
```

Note that the specification `unify(case number gender,2,3)` ensures that all adjectives

and the noun simultaneously agree in case, number and gender.

5. Instead of Conclusion

The formalism presented above and illustrated with a number of rules for Polish is rather unique in directly combining partial parsing and morphosyntactic disambiguation operations. A parser implementing this formalism is currently being released on GNU General Public License. While this formalism was designed as a partial parsing formalism and only subsequently extended to morphosyntactic disambiguation, a very similar formalism has been proposed by Doležalová and Petkevič 2007 (and brought to our attention after the presentation of ♠ in Prague, June 2007), by adding syntactic grouping operations to the LanGR morphosyntactic disambiguation language. We hope that future work will involve a detailed comparison of both formalisms and, once the corresponding extensions of LanGR are implemented, the efficiency of the tools.

References

BAŃKO, M. (2000): *Inny słownik języka polskiego*. Warsaw: Wydawnictwo Naukowe PWN.

DOLEŽALOVÁ, J. – PETKEVIČ, V. (2007): Shallow parsing of Czech sentence based on correct morphological disambiguation. I: Kosta, P. and Schürcks, L. (eds.), *Linguistic Investigations into Formal Description of Slavic Languages. Contributions of the Sixth European Conference (FDSL-6)*, Peter Lang: Frankfurt am Main, 53-64.

JANUS, D. – PRZEPIÓRKOWSKI, A. (2007): Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In: *Proceedings of ACL 2007 Demo and Poster Sessions*, 85-88.

POLAŃSKI, K. (1980-1992): *Słownik syntaktyczno-generatywny czasowników polskich*. Wrocław / Cracow : Zakład Narodowy im. Ossolińskich / Instytut Języka Polskiego PAN.

PRZEPIÓRKOWSKI, A. (2004): *The IPI PAN Corpus: Preliminary version.* Warsaw: Institute of Computer Science, Polish Academy of Sciences.

PRZEPIÓRKOWSKI, A. – BUCZYŃSKI, A. (2007): ♠: Shallow Parsing and Disambiguation Engine. In: Vetulani, Z. (ed.), *Proceedings of the 3rd Language & Technology Conference*, Poznań, Poland.

PRZEPIÓRKOWSKI, A. – FAST, J. (2005): Baseline experiments in the extraction of Polish valence frames. In: Kłopotek, M.A., Wierzchoń S. T. and Trojanowski, K. (eds.), *Intelligent Information Processing and Web Mining.* Berlin: Springer-Verlag, 511-520.

WOLIŃSKI, M. (2006): Morfeusz — A Practical Tool for the Morphological Analysis of Polish. In: Kłopotek, M.A., Wierzchoń S. T. and Trojanowski, K. (eds.), *Intelligent Information Processing and Web Mining*. Berlin: Springer-Verlag.