1. Introduction

The aim of this paper is to present the Open Source (under GNU GPL version 3) release of Spejd\(^1\) (pronounced as spade and abbreviated to ♠, i.e., the Unicode character 0x2660) a tool described in (Przepiórkowski and Buczyński, 2007). Spejd is a parser for cascades of, essentially, regular constituency grammars over morphosyntactically annotated, but not necessarily disambiguated, texts. Contrary to the common pipeline approach, where morphosyntactic tagging is fully accomplished before partial (or shallow) parsing, we argue that both tasks are best approached in parallel. This has been suggested before, and formalisms which allow for the interweaving of partial parsing and morphosyntactic disambiguation have been proposed (e.g.: (Neumann et al., 2000), (Marimon and Porta, 2000) and (Aït-Mokhtar et al., 2002)). Our approach is novel in that a fully uniform formalism is presented, and the engine is fully language-independent and we hope it will also be useful in the processing of other languages.

2. Formalism

Each rule consists of up to 5 parts marked as Rule, Left, Match, Right and Eval:

Rule: "some rule id here"
Left: ;
Match: [pos~~"prep"] [base~"co|kto"]; # a form of CO or KTO
Right: ;
Eval: unify(case,1,2); group(PG,1,2);

The rule means:

1. find a sequence of two tokens such that the first token is an unambiguous preposition ([pos~~"prep"], and the second token is a form of the lexeme CO 'what' or KTO 'who' ([base~"co|kto"]);
2. if there exist interpretations of these two tokens with the same value of case, reject all interpretations of these tokens which do not agree in case (cf. unify(case,1,2));
3. mark thus identified sequence as a syntactic group (group) of type PG (prepositional group), whose syntactic head is the first token (1) and whose semantic head is the second token (2; cf. group(PG,1,2)).\(^2\)

The Left and Right parts of a rule, specifying the context of the match, may be empty; in such a case they may be omitted. The other fields, i.e., Rule, Match and Eval are obligatory. Note that, apart from Rule, all fields end in a semicolon, and also particular actions in Eval are separated by semicolons. Comments may be added to rules, starting with the hash character "#", and fields may be split across lines, so a rule fully equivalent to the rule above may look as follows:

# a trivial rule for the purpose
# of this article only
Rule: "some rule id here"
Match: [pos~~"prep"] # a sure preposition
[base~"co|kto"]; # a form of CO or KTO
Eval: unify(case,1,2); # unify cases
group(PG,1,2); # Prepositional Group

\(^1\) The previous name, SPADE (Shallow Parsing and Disambiguation Engine), is disused because of the existence of an earlier parsing system with the same acronym, Sentence-level Parsing for Discourse Evaluation. http://www.isi.edu/licensed-sw/spade/.

\(^2\) A rationale for distinguishing these two kinds of heads is given in (Przepiórkowski, 2007a).
Although the Rule part, specifying the identifier of the rule, is obligatory, we will omit it below in the interest of brevity.

2.1. Matching (Left, Match, Right)

The contents of parts Left, Match and Right have the same syntax and semantics. Each of them may contain a sequence of the following specifications:

1. **token specification**, e.g., \([\text{pos}==\text{prep}]\) or \([\text{base}==\text{co|kto}]\); these specifications adhere to segment specifications of the Poliarp (Przepiórkowski et al., 2004; Janus and Przepiórkowski, 2007) corpus search engine as defined in (Przepiórkowski, 2004); in particular, a specification like \([\text{pos}==\text{subst}]\) says that all morphosyntactic interpretations of a given token are nominal (substantive), while \([\text{pos}==\text{subst}]\) means that there exists a nominal interpretation of a given token;

2. **group specification**, extending the Poliarp query language as proposed in (Przepiórkowski, 2007a), e.g., \([\text{semh}==\text{pos}==\text{subst}]\) specifies a syntactic group whose semantic head is a token whose all interpretations are substantive (i.e., nominal);

3. one of the following special specifications: ns: no space; sb: sentence beginning; se: sentence end;

4. an alternative of such sequences in parentheses, e.g., \([[\text{pos}==\text{subst}] | [\text{synh}==\text{pos}==\text{subst}]])

Additionally, each such specification may be modified with one of the three regular expression quantifiers: ?, * and +. The default matching strategy for such quantifiers is greedy, but an advanced user can change it to reluctant (lazy) or possessive, either globally in configuration file, or locally for selected rules.

An example of a possible value of Left, Match or Right might be:

\([\text{pos}==\text{adv}] ([\text{pos}==\text{prep}] \text{pos}==\text{subst}) \text{ns} | \text{pos}==\text{interp}? \text{se} | \text{synh}==\text{pos}==\text{prep}]\)

The meaning of this specification is: find an adverb followed by a prepositional group, where the prepositional group is specified as either a sequence of an unambiguous preposition and a possible noun at the end of a sentence, or an already recognised prepositional group.

2.2. Conditions and Actions (Eval)

The Eval part contains a sequence of Prolog-like predicates evaluating to true or false; if a predicate evaluates to false, further predicates are not evaluated and the rule is aborted. Almost all predicates have side effects, or actions. In fact, many of them always evaluate to true, and they are ‘evaluated’ solely for their side effects.

There are two types of actions: morphosyntactic and syntactic. While morphosyntactic actions delete or add some interpretations of specified tokens, syntactic actions group entities into syntactic words (consecutive tokens which syntactically behave like single words, e.g., multi-token named entities, etc.) or syntactic groups.

2.3. Examples

The formalism is more formally introduced in (Przepiórkowski and Buczyński, 2007). In the present article we will only give some examples of specific rules (from a grammar of Polish alluded to in §5).

We start with an example of a disambiguation rule, for the token nie (and Nie), which happens to be ambiguous between the negative marker (called qublic) and various post-prepositional pronominal interpretations. Such a token must be interpreted as the negative marker when occurring at the beginning of a sentence (cf. sb):

Left: \(\text{sb}\);
Match: \(\text{orth}==\text{[Nn]ie}\);
Eval: \(\text{leave(\text{pos-qub}, 2)}\);

Note that 2 above refers to the specification \(\text{orth}==\text{[Nn]ie}\) (1 would refer to \(\text{sb}\), i.e., to sentence beginning).

An example of a rule which simultaneously disambiguates nie to verbal negation and marks a negated verb (i.e., a sequence of two tokens) as a single syntactic word, is given below:

Left: \((\text{sb} | \text{[case!=-acc]} | \text{pos}==\text{prep}))\)
Match: \(\text{orth}==\text{[Nn]ie}\)
\([\text{pos}==\text{praet|fin|impt|imps|inf}]\);
Eval: \(\text{word(3, neg, base)}\)
\(\text{leave(\text{pos-qub}, 2)}\);

The specification of the left context in the simplified rule above makes sure that \(\text{nie}\) is the negative marker (it does not occur after an accusative-taking preposition, in which case it could have been a pronoun), and \(\text{leave(\text{pos-qub}, 2)}\) removes all other interpretations of this token. Moreover, the word predicate is used to create a new syntactic word and calculate its morphosyntactic interpretations: 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 \([\text{pos}==\text{praet|fin|impt|imps|inf}]\)), 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).

The third example rule is a much simplified version of a heuristic (uncertain) rule which finds a sequence of any number of adjectives, a noun and a genitive (nominal or numeral) group, between any two groups recognised by earlier rules (the specification \([\text{synh}==\{\}]\) in effect puts no conditions on the syntactic head of such a group and only serves to make sure it is a group and not a token). Such a sequence is marked as a nominal group (NG) whose both heads, the syntactic head and the semantic head, are the noun identified by the specification \([\text{pos}==\text{subst}]\):

Left: \([\text{synh}==\{\}]\);
Note that the specification `unify(case number gender, 2, 3)` ensures that all adjectives and the noun simultaneously agree in case, number and gender. The same formalism can be also used to capture idiomatic multiword expressions, for example `brac nogi za pas` ('to take off', 'to run away', literally: 'to take legs behind the waist').

```
Match: [base~"brac|wzac"]*[pos~adv]
[orth~nogi][orth~za][orth~pas];
Eval: leave(pos~verb, 1);
leave(number~pl && case~acc,3);
leave(case~acc,5);
word(1,,base "nogi za pas");
```

The rule takes case of both aspects (the imperfective `BRA\'C` and the perfective `WZIA\'C`), as well as of optional adverbs (usually just `szybko` ('quickly') after the verb. On successful match, possible non-verbal interpretations of the first segment (such as the nominative `BRA\'C` — 'brotherhood') are discarded, as well as any non-accusative interpretations of `NOGI` and `PAS`. The syntactic word created in the process inherits grammatical class and categories (number, person, gender, etc.) from the verb, with `nogi za pas` appended to its base form.

In (Moszczyński, 2006) two drawbacks of using the Poliqarp query language for encoding multiword expressions were identified: lack of permutation operator, forcing one to explicitly indicate word order variations by listing all possible realisations, and no support for unification, which is necessary to handle agreement. The Spejd formalism is largely based on the Poliqarp query language, and does not have a permutation operator either, but it supports agreement and unification, allowing for precise (if not compact) description of multiword expressions.

### 3. Adding Lexical Resources

Spejd can make use of two distinct kinds of lexical resources, operating at the level of orthographic forms of segments and their morphosyntactic interpretations, named gazetteers and dictionaries, respectively. They are especially useful in areas where one needs to amend or extend the morphological analysis of a segment, for example add interpretations or assign values to a category not covered by the analyser.

3.1. Gazetteers

Gazetteers look for specific orthographic forms of segments or sequences of segments. Upon finding them, specified Spejd actions are applied to the segments constituting the form. In fact, gazetteers are just a short way of writing many similar, simple rules. For example, the following Spejd rule makes up for the fact that some male profession titles can be also used in a noninflective manner to title a woman:

```
Rule "Female prof"
Match: [orth~"minister|pose\'l|prezydent|profesor"]; Eval: add(subst:sg:case~f,1);
```

In a gazetteer, the same rule would look like:

```
Enter "minister|pose\'l|prezydent|profesor" = add(subst:sg:case~f,1);
```

3.2. Dictionaries

Dictionaries are different, because they operate at the level of specific interpretations of a segment. Instead of adding a new interpretation to the segment, they may modify some of the existing interpretations (i.e., those matching the dictionary entry key). An example of such a resource is a sentiment dictionary. Words are assigned positive or negative sentiment polarity, according to their base form. Let us consider a simple dictionary entry:

```
obraza = sneg
```

The entry defines that the lexeme `OBRAZA` ('insult') has a negative sentiment polarity. Reading the text input, Spejd may find a segment with the orthographic form `obrazy`, which may be a form of either the aforementioned lexeme `OBRAZA`, or another lexeme, `OBRAZ` (image, painting). The morphological analyser returns the following interpretations:

- `obraz subst:pl:nom:m3`
- `obraz subst:pl:acc:m3`
- `obraz subst:pl:voc:m3`
- `obraza subst:sg:gen:f`
- `obraza subst:pl:nom:f`
- `obraza subst:pl:acc:f`
- `obraza subst:pl:voc:f`

Applying the sentiment dictionary results in adding a negative sentiment tag (`sneg`) to interpretations 4–7, leaving interpretations 1–3 unchanged (neutral sentiment).

Note that a similar result could be achieved by a Spejd rule:

---

3Actually, in the IPI PAN tagset there is no such grammatical class as `verb`. The real rule checks instead for an alternative of verbal parts of speech, like `praet, fin, impt, imps, inf, pant, pcon and ger.`
Polish called Morfeusz (Woliński, 2006) is installed in the IPI PAN Corpus of Polish (Przepiórkowski, 2004)), or just plain text. In the latter case, the IPI PAN Corpus of Polish (http://korpus.pl/), an XML corpus, contains over 350 different rules created, currently containing over 350 different rules (Przepiórkowski, 2004)). A Spejd grammar has been constructed, currently containing over 350 different rules (Przepiórkowski, 2007b; Przepiórkowski, 2008). The grammar relies on the full functionality of ♠ and it consists of the following parts:

- purely morphosyntactic rules, countering the known deficiencies of the morphological analyser used to tag the IPI PAN Corpus, Morfeusz (Woliński, 2006),
- simple disambiguation rules,
- rules creating syntactic words, including synthetic verbs, abbreviations (as in the original segmentation the full stop ending an abbreviation is treated as a separate segment), number ranges, simple proper names, etc.,
- rules creating syntactic groups, further split into:
  - lexicalised rules, containing references to particular lexical items; such rules find more complex named entities, dates, various idioms, etc.,
  - general syntactic rules, e.g., identifying noun groups as certain sequences of adjectives and nouns, etc.,
- coda, i.e., various rules logically belonging to the first groups of rules (morphosyntactic rules, disambiguation rules, etc.), but relying on the presence of syntactic groups, identified by subsequent rules.

4.1. Input Format
As for now, the parser implementing the specification above can take as input either the version of the XML Corpus Encoding Standard (Ide et al., 2000), as assumed in the IPI PAN Corpus of Polish (http://korpus.pl/; Przepiórkowski (2004)), or just plain text. In the latter case, it is currently assumed that the morphological analyser for Polish called Morfeusz (Woliński, 2006) is installed in the case of the dictionary approach sketched above, such morphosyntactic disambiguation would also disambiguate the sentiment value of the segment.

4.2. Efficiency
Since the formalism described above is novel and to some extent still evolving, its implementation had to be not only reasonably fast, but also easy to modify and maintain. The system has been implemented in Java, as a prototype, and not much effort has been devoted to efficiency, yet. In brief, the Left, Match and Right parts of a rule are compiled into regular expressions over an internal, compact representation of texts, and then matched using non-deterministic finite automata (currently each rule is processed separately, with the exception of gazetteers and dictionaries, requiring time logarithmic in their size).

4.3. Output Format
The parser in itself does not provide any visualisation of the produced structures. However, as the output is well-formed and valid XML, several XSLT stylesheets have been created to visualise various aspects of the parsing results, either putting more emphasis on the detected structures or on the ambiguities resolution. The stylesheets are included with the release of Spejd.

5. Current Applications
Spejd may be applied in any tasks involving partial parsing or rule-based disambiguation, but its two main current applications are valence acquisition and sentiment analysis. For task of the automatic learning of subcategorisation frames from the morphosyntactically annotated IPI PAN Corpus of Polish (http://korpus.pl/; Przepiórkowski (2004)), a Spejd grammar has been constructed, currently containing over 350 different rules (Przepiórkowski, 2007b; Przepiórkowski, 2008).\(^{5}\) The grammar relies on the full functionality of ♠ and it consists of the following parts:

- purely morphosyntactic rules, countering the known deficiencies of the morphological analyser used to tag the IPI PAN Corpus, Morfeusz (Woliński, 2006),
- simple disambiguation rules,
- rules creating syntactic words, including synthetic verbs, abbreviations (as in the original segmentation the full stop ending an abbreviation is treated as a separate segment), number ranges, simple proper names, etc.,
- rules creating syntactic groups, further split into:
  - lexicalised rules, containing references to particular lexical items; such rules find more complex named entities, dates, various idioms, etc.,
  - general syntactic rules, e.g., identifying noun groups as certain sequences of adjectives and nouns, etc.,
- coda, i.e., various rules logically belonging to the first groups of rules (morphosyntactic rules, disambiguation rules, etc.), but relying on the presence of syntactic groups, identified by subsequent rules.

\(^{5}\)In the full grammar, some of the rules are repeated, so it currently contains over 450 rule tokens.
The final output of the grammar for any sentence is the set of maximal constituents in the sentence, i.e., an observed valence of the main verb of this sentence. Obviously, such observed valences are noisy, so the set of observations obtained this way constitutes an input to the usual statistical filtering stage (Brent, 1993; Manning, 1993; Korhonen, 2002; Fast and Przepiórkowski, 2005). The results of this procedure of valence acquisition for Polish are currently under evaluation, but it is already clear that they are at least comparable to the application of a deep parser to the same task and the same data (Dębowski and Woliński, 2007; Dębowski, 2007).

Another practical application of ♠ is the automatic recognition of sentiment polarity in Polish product reviews (Buczyński and Wawer, 2008a). Shallow parsing can help the sentiment analysis in many ways, including rule-based disambiguation between morphosyntactic interpretations carrying different sentiment polarity, capturing multiword units with a non-compositional sentiment value (the value of such unit as a whole might be charged emotionally, positive or negative, all the individual words being neutral), and detection of sentiment reversing constructions, for example negation or nullification (‘lack of . . .’) (Buczyński and Wawer, 2008b). Adding a small set of 12 relatively simple sentiment rules to an earlier system based on the baseline bag-of-words approach made it possible to increase the accuracy of sentiment recognition from 75% to 78% on a noisy dataset of 4175 product or service reviews downloaded from various Polish Internet shops.

6. Conclusion

The system presented above and to be demonstrated at LREC 2008, ♠, is perhaps unique in allowing the grammar developer to encode morphosyntactic disambiguation and shallow parsing instructions in the same unified formalism, possibly in the same rule. The formalism is more flexible than either the usual shallow parsing formalisms, which assume disambiguated input, or the usual unification-based formalisms, which couple disambiguation (via unification) with structure building. While a rule set is currently prepared for the parsing of the IPI PAN Corpus of Polish, ♣ is fully language-independent and we hope it will also be useful in the processing of other languages.

7. References


