Annotation Tools for Syntax and Named Entities in the National Corpus of Polish

Jakub Waszczuk

Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland
Institute of Informatics, University of Warsaw,
ul. Banacha 2, 02-097 Warsaw, Poland
E-mail: jw235843@students.mimuw.edu.pl

Katarzyna Głowińska

Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland
E-mail: k.glowinska@gmail.com

Agata Savary

Laboratoire d'Informatique, Université François Rabelais Tours,
3 pl. Jean-Jaurès, 41000 Blois, France
E-mail: agata.savary@univ-tours.fr

Adam Przepiórkowski

Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland
Institute of Informatics, University of Warsaw,
ul. Banacha 2, 02-097 Warsaw, Poland
E-mail: adamp@ipipan.waw.pl

Michał Lenart

Institute of Computer Science, Polish Academy of Sciences,
ul. Ordona 21, 01-237 Warsaw, Poland
Institute of Informatics, University of Warsaw,
ul. Banacha 2, 02-097 Warsaw, Poland
E-mail: michal.lenart@gmail.com

Abstract: The ongoing *National Corpus of Polish* project assumes several levels of linguistic annotation. We present the technical environment and methodological background developed for the three upper annotation levels: the levels of syntactic words, syntactic groups and named entities. We show how knowledge-based platforms Spejd and Sprout are used for the automatic pre-annotation of the corpus and discuss some particular problems faced during the preparation of the parser grammar, which contains over 1000 rules and is one of the largest chunking grammars for Polish. We also show how the tree editor

TrEd has been customized for manual post-editing of annotations and for further revision of discrepancies. Our XML format converters and customized archiving repository ensure an automatic data flow and efficient corpus file management. We discuss the inter-annotator agreement in the manually annotated data, and present the first results of a CRF classifier trained on these data.

Keywords: corpus annotation, National Corpus of Polish, shallow parsing, chunking, named entity recognition

Biographical notes:

Jakub Waszczuk is an MSc student in Computer Science at the University of Warsaw. He works as a programmer at the Institute of Computer Science, Polish Academy of Sciences. His current scientific interests include algorithms, machine learning methods and natural language processing.

Katarzyna Głowińska obtained a PhD in Linguistics from the University of Warsaw, Poland in 1999. She has participated in various research project in the field of computational linguistics, dedicated to information extraction, machine translation and corpus annotation. Her current scientific interests are morphology and syntax of Polish, as well as semantics.

Agata Savary received an MSc in Computer Science from the University of Warsaw and a PhD in Computer Science from the University of Marne-la-Vallée, France in 2000. She is currently associate professor at the François Rabelais University in Tours, France. Her research interests include language resources, multi-lingual processing of multi-word expressions, named-entity recognition and XML document correction.

Adam Przepiórkowski obtained an MSc in Computer Science from the University of Warsaw and a PhD in Linguistics at the University of Tübingen, Germany. He is currently an associate professor at the Institute of Computer Science of the Polish Academy of Sciences, where he is the head of the Linguistic Engineering Group, and at the Institute of Informatics at the University of Warsaw. His research interests include corpus linguistics, syntactic parsing and formal linguistics.

Michał Lenart studies Computer Science at the University of Warsaw. He currently works as a programmer at the Institute of Computer Science, Polish Academy of Sciences.

1 Introduction

The National Corpus of Polish (Pol. *Narodowy Korpus Języka Polskiego*; NKJP; `http://nkjp.pl/`) is a 3.5-year project (2007–2011), involving a consortium of four partners coordinated by the Institute of Computer Science, Polish Academy of Sciences ([21, 24]). The aim of the project is to create a one billion (10^9) word corpus of Polish annotated at various levels, with a 300-million-word balanced subcorpus and a number of annotation tools. The following linguistic

annotation levels are distinguished: segmentation (word-level and sentence-level), morphosyntax, word sense disambiguation (limited to around 100 lexemes), syntactic words, syntactic groups and named entities.

The level of syntactic words (SWs), such as reflexive and analytical verb forms [23], is built on top of the morphosyntactic level. Syntactic groups (SGs; also called syntactic phrases), such as nominal, prepositional or clause-level groups, are constructed on top of SWs. Finally, named entities (NEs), i.e. proper names of persons, geographical features and organisations, as well as temporal expressions, refer, again, to the level of morphosyntactically annotated segments (cf. section 3).

A 1-million-word balanced subcorpus has undergone, at all these levels, automatic pre-annotation and manual correction of the pre-annotation results. We refer to this process, in short, as *manual annotation*. Further, the 1-million-word manually annotated subcorpus serves as a training corpus for various annotation tools (e.g., for the PANTERA tagger; cf. [3]). The current paper gives an overview of methodologies and tools used for the semi-manual annotation of the 1-million corpus at the last three – broadly syntactic – levels. It also presents the first results of a machine learning method for named-entity recognition trained on the manually revised data.

2 Related Work

Some of the first treebanks were constructed completely manually, by drawing trees for particular sentences; this is the case, for example, for the Penn Treebank (PTB) of English [17], the German Negra/Tiger Treebank [7] and the Prague Dependency Treebank [9]. Some treebanks were created by converting existing treebanks to the new linguistic theory; for example, parts of roughly Chomskyan PTB were converted to Head-driven Phrase Structure Grammar, Lexical Functional Grammar and Constraint Categorial Grammar. Nowadays, however, a frequent way of developing new treebanks consists in the automatic parsing of texts and the manual selection of the right parse. If such a parse does not exist, the best result may be manually modified and included in the treebank, and the grammar may be appropriately extended. For example, [14] reports that the ERG grammar [12] covers around 80% of the Wall Street Journal part of PTB, with sentences not adequately covered by the grammar serving as the basis for further grammar development.

The outcome of the effort reported here will not constitute a typical treebank, as the annotation in NKJP stops at the partial (or shallow) syntactic markup (cf., e.g., [1, 2], as well as [23]), where structural ambiguity is not an issue; in fact, there is a separate project carried out at the same institute, aiming at the construction of a full constituency treebank on the basis of the same 1-million word subcorpus; cf. [29]. Hence, the approach mentioned above, focusing on disambiguation, is not directly applicable to the task at hand, but the general semi-manual iterative methodology is similar: parse sentences using a manually constructed grammar, ask annotators to correct the results of parsing by hand, and use error and omission reports for the improvement of the grammar, before applying it to the next batch of sentences.

There are many multi-level corpora developed by now, typically containing morphosyntactic, (deep) syntactic and some semantic and/or discourse representation. For example, the Prague Dependency Treebank mentioned above has these three levels (called morphological, analytical and tectogrammatical), currently extended further to include coreference [18] and high-level inter-clausal structure [15]. The current project adopts a more fine-grained and conservative approach, with three levels between morphosyntax and deep syntax proper: possibly multi-segment SWs, NEs and possibly partial SGs. We claim that this gradual procedure makes it possible to better control the quality of the linguistic annotation.

Furthermore, at the level of NEs, the annotation strategies adopted here are rather fine-grained, namely, not just the longest-matching occurrences of NEs are annotated, but also all recursively embedded ones, and, moreover, overlappingly coordinated NEs (as in *North and South America*) are appropriately marked (cf. [27, 26]).

Finally, let us note that, while only partial syntactic structures are annotated here, syntactic groups contain the kind of information not usually found in treebanks, namely, they mark both syntactic and semantic heads. For example, in the case of prepositional groups, the preposition serves as the syntactic head (because it governs the case inflection of its arguments), but the semantic head is the most meaningful word within the argument of this preposition. Arguments for the usefulness of this kind of annotation, and further details, may be found, e.g., in [22, 23].

3 Annotation Data Flow

The three levels of syntactic annotation in the NKJP are organised into two parallel data flows: one for syntactic words and syntactic groups (henceforth, *syntactic annotation* in the narrower sense), and the other for named entities.

The main differences between the data flows show up during the pre-processing step – different tools, with different input specifications, are used for automatic pre-annotation. In the case of syntactic annotation (Fig. 1) a shallow parsing system called Spejd is used to extract SWs and SGs from the underlying morphosyntax level (cf. section 4.1). For the task of NE recognition, another platform, Sprout, is used (see section 4.2).

Spejd takes structured text with segmentation and morphosyntax information as input. It requires a specific input format (called *IPI format* in Fig. 1) that can be automatically obtained from the NKJP morphosyntax level. Conversely, Sprout requires pure text as input, which complicates the whole process of data conversion (see Fig. 2). The NEs, identified by means of lexical resources and grammar rules, are marked in a Sprout-specific output together with cardinal numbers of the beginning and ending characters. The converter consults the segmentation level in order to translate text ranges into token identifiers.

After the pre-processing step, files have to be prepared for manual annotation. In both data flows, annotators use a tree editor TrEd (see section 5) to examine and correct results of automatic annotation. Again, files have to be translated to TrEd-readable PML formats (`http://ufal.mff.cuni.cz/jazz/PML/doc/`). For

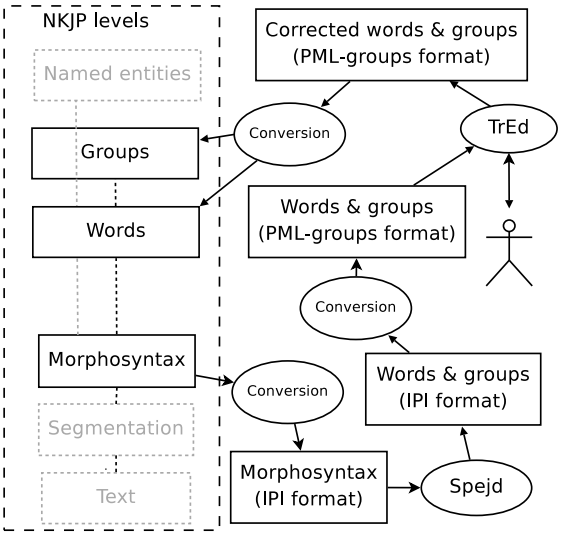

Figure 1 Data flow in the syntactic annotation task of the NKJP corpus

ergonomic reasons, it is important to present the annotator with text portions of uniform length, thus easily manageable. Therefore, the converter divides every text that is too large into files of a limited number of sentences corresponding to roughly 1 hour of human annotation effort. Text splitting is designed so as to keep together all sentences appearing in one paragraph.

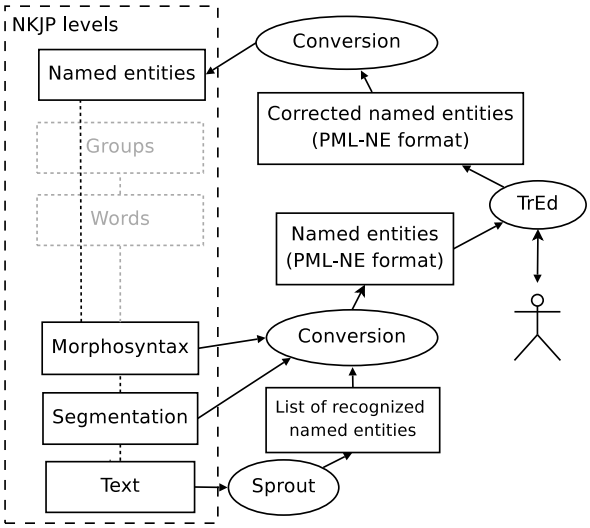

Figure 2 Data flow in the NEs annotation task of the NKJP corpus

Finally, PML files are transferred to *corpus files management system* (see section 5.3) which is responsible for distributing files between annotators and for storing results of consecutive annotation steps.

Two annotators work on each corpus fragment. An adjudicator reviews any cases of disagreement and chooses the correct annotation. Each annotator and adjudicator works off-line with TrEd installed locally, connecting to the subversion repository only to send results of his or her work or to download new files.

The last stage consists of converting the PML files with validated annotation into the final TEI-P5-conformant NKJP format. Here, the subfiles have to be merged into files corresponding to the initial texts and embedded XML elements (NEs and SGs) get transformed into pointers (for stand-off annotation). Entities at different annotation levels are marked as ⟨`seg`⟩ments, and their inclusions as ⟨`ptr`⟩ links. The SW annotation level (L_{words}) is based on the level of morphosyntactic items ($L_{morphosyntax}$). The SG level (L_{groups}) builds on L_{words}, i.e., the scope of a given SG in text is defined by its ⟨`ptr`⟩ references to segments defined at the levels L_{words} and L_{groups} (the latter in the case of embedded SGs). The same should hold for the final version of the named-entity level (L_{named}). However, since the SW, SG and NE levels are being developed simultaneously, the current version of L_{named} relates to $L_{morphosyntax}$ instead of L_{words}. For instance, in Fig. 3, the organisation name *Irlandzka Armia Republikańska* 'Irish Republican Army' points to ⟨`seg`⟩ments `morph_1.2-seg` (*Armia* 'Army') and `morph_1.3-seg` (*Republikańska* 'Republican') at the $L_{morphosyntax}$ level (in file `ann_morphosyntax.xml`), and to ⟨`seg`⟩ment `named_1.34-s_n3` (*Irlandzka* 'Irish') defined just below at the L_{named} level. Both named entities have a set of attributes defining their types (`ne_type`), subtypes, if any (`ne_subtype`), corpus occurrence forms (`orth`), lemmas (`base`), and the degree of annotator's certainty with respect to this annotation (`certainty`). Additionally, the derivational adjective *Irlandzka* 'Irish' is assigned its type of derivation (`derivType`) and its derivational base *Irlandia* 'Ireland' (`derivedFrom`).

4 Automatic Annotation

4.1 Shallow Parsing with Spejd

Syntactic annotation of the National Corpus of Polish consists of combining words into constituents: first at the level of SWs, then at the level of SGs (possibly embedded). At the former, fine-grained word-level tokens are replaced by coarse-grained SWs (e.g., analytical tense and mood forms, reflexive verbs, discontinuous conjunctions). The tagset at this level differs somewhat from the tagset of word-level segments in order to allow for broader grammatical classes and more traditional grammatical categories, such as tense, mood and reflexivity. The complete tagset for SWs and the list of SGs distinguished in NKJP are presented in [13].

At the SG level, for every identified group a syntactic head (SynHead) and a semantic head (SemHead) are selected. Only accurately recognizable groups are marked, so that the shallow grammar resulting from the manual correction process can be reliably applied to the whole 1-billion-word corpus. For example, a nominal phrase consisting of a noun and a prepositional phrase, e.g., *dom z ogrodem* 'a house with a garden', is always treated as two SGs (*dom* and *z ogrodem*), with no attempt to solve PP-attachment ambiguities. An exception is

```
<teiCorpus xmlns:xi="http://www.w3.org/2001/XInclude" xmlns="http://www.tei-c.org/ns/1.0">
<xi:include href="NKJP_1M_header.xml"/>
<TEI>
 <xi:include href="header.xml"/>
 <text><body>
 <p xml:id="named_1-p" corresp="ann_words.xml#words_1-p">
  <s xml:id="named_1.34-s" corresp="ann_words.xml#words_1.34-s">
   <seg xml:id="named_1.34-s_n2">
    <fs type="named">
     <f name="ne_type"><symbol value="orgName"/></f>
     <f name="orth"><string>Irlandzka Armia Republikańska</string></f>
     <f name="base"><string>Irlandzka Armia Republikańska</string></f>
     <f name="certainty"><symbol value="high"/></f>
    </fs>
    <ptr target="named_1.34-s_n3"/>                    <!-- Irlandzka -->
    <ptr target="ann_morphosyntax.xml#morph_1.2-seg"/> <!-- Armia -->
    <ptr target="ann_morphosyntax.xml#morph_1.3-seg"/> <!-- Republikańska -->
   </seg>
   <seg xml:id="named_1.34-s_n3">
    <fs type="named">
     <f name="derived">
      <fs type="derivation">
       <f name="derivType"><symbol value="relAdj"/></f>
       <f name="derivedFrom"><string>Irlandia</string></f>
      </fs>
     </f>
     <f name="ne_type"><symbol value="placeName"/></f>
     <f name="ne_subtype"><symbol value="country"/></f>
     <f name="orth"><string>Irlandzka</string></f>
     <f name="base"><string>irlandzki</string></f>
     <f name="certainty"><symbol value="high"/></f>
    </fs>
    <ptr target="ann_morphosyntax.xml#morph_1.1-seg"/>
   </seg>
  </s>
 </p>
 </body></text></TEI>
</teiCorpus>
```

Figure 3 TEI-P5-conformant annotation for the named entity *Irlandzka Armia Republikańska* 'Irish Republican Army'

made for compound prepositions consisting of two prepositions and an interposed noun (e.g., *w odniesieniu do* 'in reference to') and for elective constructions (e.g., *jeden z najlepszych* 'one of the best').

The manually constructed grammar, for both SWs and SGs, is encoded in the shallow parsing system Spejd (`http://nlp.ipipan.waw.pl/Spejd/`) [8], a novel open source tool for simultaneous morphological disambiguation (this functionality is not used in this project) and partial parsing with unification.

Spejd rules form a cascade, with the output of one rule constituting the input of the next. Therefore, rule ordering is crucial. For example, since nominal groups

are embedded in prepositional-nominal groups, the rules for the former precede those for the latter.

Spejd rules are created in a conservative fashion, in order to avoid excessive matching and detect errors at the underlying morphosyntactic level. First, as the parser finds a match for a lemma, it is usually checked for grammatical class. Second, rules are made maximally specific in that some SGs are divided into several subtypes. For example there are 11 types of nominal groups, e.g., `NGa` (`Noun`+`Adj`), `NGk` (`Noun`+and+`Noun`), `NGn` (`Noun`+`Num`), `NGb` (`Noun`+`Brev`), `NGx` (`PPron3`+`Adj`, e.g., *something special*). Thus, instead of the plain `NG`, a disjunction of subtypes is given in the rule, i.e., `NGa|NGk|NGn|NGb`.

A Spejd rule may consist of five elements: `Rule` (rule identifier), `Left` (left context), `Match` (specification), `Right` (right context), `Eval` (conditions and operations). Context specification is optional.

Two types of syntactic operations are available: `word`, which joins tokens into SWs, and `group`, which joins SWs into SGs.

The `word` operation has two mandatory arguments: 1) information about a token in accordance with the tagset (i.e., grammatical class and grammatical category values; pieces of information are separated by colons), 2) the base form of the resulting SW. These two arguments may be preceded by an optional argument: reference to the token which provides some morphological information for the whole SW. In this case, the second argument determines how this information should be modified. In Spejd, the token to which the rule refers in this way must be unique – it is impossible to inherit information from different components. For example, the analytical future tense (e.g., *będę szedł* 'I will walk') is a combination of future auxiliary (`będzie`) and past participle (`praet`). All the information is taken from the `bedzie` form, except for the gender, which should be taken from the `praet` form. A solution to this problem is a multiplication of rules. An example of a rule for future tense forms in the feminine (`f`) is presented below, where the operator `~` means equal, `&&` denotes logical conjunction. Here, the gender, instead of being inherited from the third component, is explicitly fixed to be feminine. Similar rules have to be created for all other possible genders.

```
Rule     "analytical future tense:
         bedzie + się + praet (f)"
Match:   [pos~"bedzie"] [base~"się"]
         [pos~"praet" && aspect~"imperf"
         && gender~"f"];
Eval:    word(1,Verbfin:fut:ind:refl:f,3.base);
```

The `group` operation (as in example below corresponding to, e.g., *po tych trzech zdaniach* 'after these three sentences'), has three arguments: 1) the type of the SG, 2) the reference to the SynHead of the phrase (*po* 'after'), 3) the reference to its SemHead (*zdaniach* 'sentences').

```
Rule     "PrepNumG: Prep + Adj + Num + Noun"
Match:   [pos~"Prep"] [pos~"Adj|Pact|Ppas"]
         [pos~"Num|Numcol"]
         ([pos~"Noun"] | [type="NG"]);
Eval:    unify(case number gender,2,3,4);
         unify(case,1,3);
         group(PrepNumG,1,4);
```

See Tab. 1 for breakdown of Spejd rules into various types.

Syntactic words			Syntactic	TOTAL
multiword entities	abbreviations	others	groups	
339	383	123	247	1092

Table 1 Taxonomy and quantities of Spejd rules

Spejd rules are applied to a corpus when its underlying morphosyntactic level has already been disambiguated manually. We fully benefit from this fact in our rules. The information about context is used to a smaller degree. Rules are based mainly on morphological information of the matched items themselves. As a result, our grammar performs very well on a good-quality disambiguated corpus. However, if applied to a non- or poorly disambiguated corpus, it would require more context-specific rules.

4.2 Named Entity Recognition with Sprout

As discussed in [26], the automatic pre-annotation of named entities in NKJP is done by the general-purpose knowledge-based NLP platform Sprout [5]. This tool offers several convenient features such as: (i) a rather rich grammar formalism with finite-state operators, unification and cascading, (ii) a very fast gazetteer lookup, (iii) an XML-based output, called Sproutput, in the form of typed feature structures whose type hierarchy can be defined by the user. Existing Polish named entity grammar and resources for Sprout [20] have been extended and adapted for the annotation task in NKJP. They include a gazetteer of about 300,000 inflected forms (55,000 lemmas), and 120 grammar rules for 6 types and 8 subtypes: (i) personal names (`persName`) with subtypes `forname`, `surname`, and additional name (`addName`), (ii) names of organisations (`orgName`), (iii) names of geographical objects such as rivers, mountains, etc. (`geogName`), (iv) names of geo-political units (`placeName`) with subtypes `district`, `settlement`, `region`, `country`, and `bloc`, (v) `date` expressions, (vi) `time` expressions. For types (ii), (iii) and (iv) relational adjectives (`relAdj`), as well as inhabitants or member names (`persDeriv`) are also covered. The latter categories are systematically attached to their derivation base names (e.g. *warszawski* 'Warsaw-related' and *warszawiak* 'inhabitant of Warsaw' are marked as derived from *Warszawa* 'Warsaw'). Note that this attachment is context-dependent and cannot always be unambiguously done by an external lexicon. For instance *ostrowski* is an adjective related to several Polish towns: *Ostrów Wielkopolski*, *Ostròw Mazowiecka*, etc., while *europejski* 'European' can refer to *Europa* 'Europe' or to *Unia Europejska* 'European Union'.

The results of the Sprout grammar application to NKJP show the overall precision of 71%, recall of 35%, and F-measure of 47% when all attributes of a NE are taken into account. If only the types and subtypes are considered, the corresponding results come up to 78%, 38% and 51% for precision, recall and F-measure, respectively (see section 7).

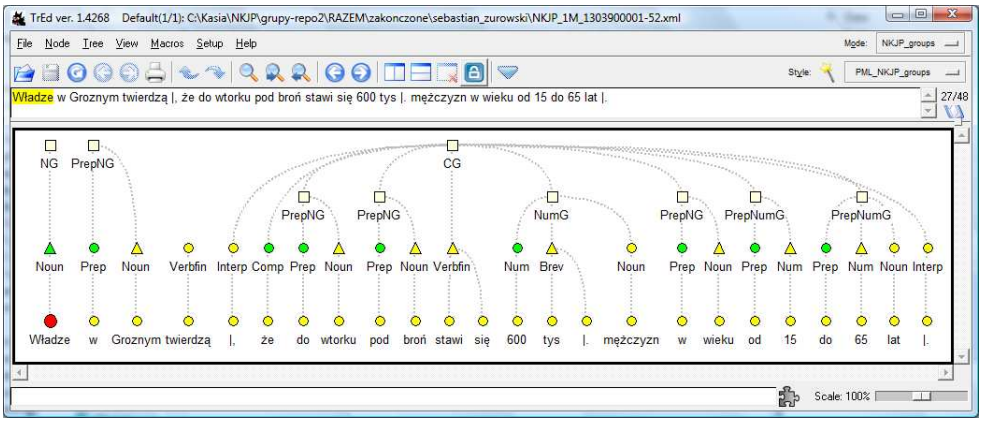

Figure 4 Syntactic annotation with the use of the TrEd editor for the sentence 'The authorities in Grozny claim that 600 thousand men of 15 to 65 years of age will turn up to arms till Tuesday.'

5 Manual Post-Editing

Manual post-editing of annotations is the most labor-intensive subtask and requires efficient and user-friendly tools. We have evaluated several annotation platforms such as Synpathy (`http://www.lat-mpi.eu/tools/synpathy`), MMAX (`http://mmax2.sourceforge.net`), and GATE [28], before selecting the tree editor TrEd (`http://ufal.mff.cuni.cz/~pajas/tred/`) [19] for the following reasons: (i) admitting pre-annotated input and multi-level annotation, (ii) customizable open XML-based abstract data format (PML), (iii) easy manipulation of tree representations, (v) ergonomic customizable graphical user's interface, (vi) parallel edition of concurrent annotations, (vii) rich documentation, (viii) technical reliability.

5.1 Annotator's Workbench

Two separate annotator's workbenches have been prepared within TrEd, one for SWs and SGs (Fig. 4), and the other for NEs. Both workbenches can be used either in the annotation mode or in the adjudication mode for revision of previous annotations. In the central part of a TrEd's window, the annotation tree of the sentence is shown. Nodes are situated on horizontal levels, the lowest of which corresponds to segments of the morphosyntactic annotation. Higher level nodes represent: (i) SWs dominated by possibily embedded SGs, or (ii) NEs possibly embedded in other NEs, respectively. Native TrEd facilities allow the annotator to add or remove nodes, draw edges between them, edit type-specific node attributes, and rapidly navigate between sentences and files. For both workbenches an NKJP-proper *TrEd extension* allows for grouping multiple nodes at a time, adding nodes of specific types and subtypes, rapidly modifying attribute values, etc. It also provides a PML schema defining the corresponding PML format and a stylesheet with tree visualisation rules. For instance, in Fig. 4 the SynHead of each constituent is marked in green and the SemHead is marked with a triangle. Thus,

Władze ('authorities') is both a SynHead and a SemHead while *do* ('till') is a SynHead and *wtorku* ('Tuesday') a SemHead only.

When closing a reviewed file, final checkups are performed via TrEd: (i) in the SW/SG workbench groups with missing SynHeads and SemHeads are reported, (ii) in the NE workbench the format and completeness of attributes are checked.

5.2 Revision of Annotations

As mentioned above, each text of the gold standard subcorpus is to be annotated at each level by two annotators. Cases of disagreement are further reviewed and resolved by an adjudicator (called *super-annotator*), who is usually a person with rich experience in annotation at that particular level. In order to maximize the objectivity of judgement, the general principle is that: (i) the two annotators of the same text know nothing about each other's results, except what they may learn via the discussion list, (ii) a super-annotator cannot review any portion of the corpus that he or she has previously annotated.

In order for the annotator's work to be most effective, a set of macros and keyboard shortcuts were developed to automatically find discrepancies in two annotations of the same text, as well as to automatically transfer an annotation between two files. Fig. 5 shows a TrEd screenshot with two NE annotations of the same sentence, containing recursively embedded organisation and location names. The lower window, corresponding to the annotator *a2*, was chosen as the final version of the annotation. However, the upper window, corresponding to the annotator *a1*, contains a node for the country name *France* that has not been annotated as a NE by *a2*. Using a single keyboard shortcut we can transfer the missing node to the lower window, over the node *France* and under the node *Radio France Nationale*, so that the remaining nodes remain intact. The automatic detection and transfer of discrepancies act not only on missing or dislocated nodes, but also on a node's attributes. In Fig. 5 the next difference to be highlighted will be the node over *Europa* that has been assigned different types (here *a2* chose the correct type, thus the annotation by *a1* will not be transferred). The same types of macros exist for the revision of discrepancies in annotated SWs and SGs.

5.3 Management of Corpus Files

Corpus files management system consists of two main components. The first one is the svn repository, where all files earmarked for annotation are stored. The second element is a textual database (versioned XML file), which contains all information regarding the current state of annotation.

Every annotator has access to their own private directory in the repository. There they keep currently annotated files, which they can modify and send back to a target directory for completed files. As a rule, every file will be examined by two different annotators. The annotator does not have the necessary permissions to run all svn operations – they can edit files in the private directory, but cannot add, move or delete files in the repository. The additional functionality – downloading files for annotation and sending off the completed files – is performed by a special `message.txt` file, placed in the private directory.

This file works as an interface between the annotator and the subversion server. For example, in order to download five files to their private directory,

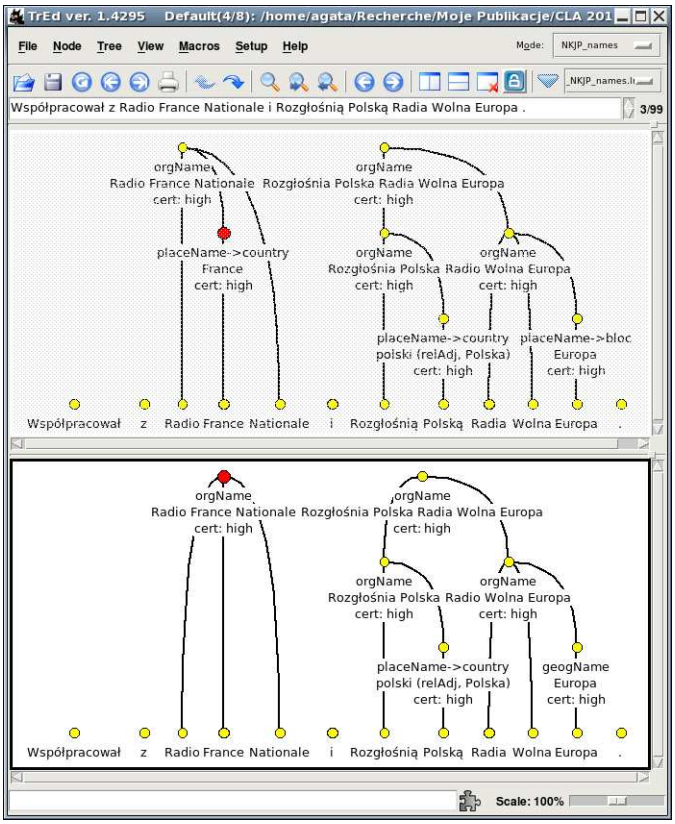

Figure 5 Comparing two NE annotations in TrEd for the same sentence 'He collaborated with Radio France Nationale and the Polish Station of the Free Europe Radio.'

the annotator has to add the `checkout = 5` line to the `message.txt` file, and run *svn commit* and *svn update* on the directory. The rest of the work – finding appropriate files and moving them throughout the repository – is performed on the server side by means of a post-commit subversion hook (process run on server after every *commit* operation). Another command, *checkin*, can be used to send off annotated files. For super-annotation, similar commands, *s_checkout* and *s_checkin*, exist. The *s_checkout* command will download a chosen number of files to compare, while *s_checkin* will send the corrected version back to the target directory for super-annotated files. While the `message.txt` file can be modified by the annotator directly, a client-side GUI application – with *[s_]checkout* and *[s_]checkin* functionality – has been developed for annotators' convenience. It fills out the `message.txt` file automatically, thus saving the annotator's effort of editing additional commands manually.

The database – a versioned `db.xml` file – keeps track of every important repository operation. The information about every new file placed in the repository is stored automatically in the database. When files are downloaded or sent off by annotators, their name and the operation date are also saved in the database. There are two main reasons for saving this kind of information in a separate file.

First, it allows to quickly find the information about the current state of the annotation, which is important, e.g., for the implementation of the server-side part of the *[s_]checkout*/*[s_]checkin* operations. Second, it simplifies searching the repository – most of the important information can be obtained from the database, without looking into the repository itself. Integrity between the repository and the database is ensured – no two post-commit processes which modify the repository or the database can run simultaneously.

To simplify querying the database, another tool has been developed. It takes, as command-line arguments, a number of various searching parameters – annotator's name and file name (as regular expressions), file status (checked in or checked out), checkin date range, etc. Another option can be used to extract the number of sentences and words from particular files (in this case the tool has to consult the repository, because files statistics are not stored in the database). Additionally, the tool can be used to find files left for annotation (that is, files which haven't been downloaded by two annotators yet).

6 Inter-Annotator Agreement

The inter-annotator agreement is a classical quality indicator in an annotation task: (i) the clearer and the more detailed the annotation guidelines are, the fewer ambiguities, underspecifications and contradictions need to be resolved by the annotators, (ii) the better the project methodology is, the clearer the annotation procedures and requirements are, and the better the chance of coherent actions among independent annotators. This indicator also allows to estimate the cost of the super-annotation: the higher it is, the less discrepancies need to be revised by an adjudicator.

The inter-annotator agreement, despite its intuitive simplicity, is a rather complex notion in an annotation task like ours. The weighted kappa measure ([10]) is not easily applicable here because it assumes that the units to be annotated are known beforehand. In our task, annotators first have to identify the boundaries of existing syntactic words, groups and names before they categorize them, thus there is no a priori list of units for which different annotations are to be compared (see [6] for similar considerations on annotating multi-word expressions). Therefore, we use simpler classical information retrieval measures. If annotators *a1* and *a2* have annotated the same corpus text, we admit that annotations produced by *a1* constitute the reference corpus and calculate the precision and the recall of *a2* with respect to this reference. Note that if we invert the roles of both annotators, we obtain complementary results: the precision (recall) of *a2* with respect to *a1* is equal to the recall (precision) of *a1* with respect to *a2*. Thus, the F-measure of *a1* wrt *a2* is equal to the F-measure of *a2* wrt *a1*.

Estimating the precision/recall, however, is not quite straightforward in our task. As mentioned before, each unit (SW, SG or NE) is assigned an annotation tree whose leaves are segments from the morphosyntactic annotation level, the tree's height can exceed 1 and its every node obtains a set of attributes (syntactic and semantic head, lemma, derivational base etc.). We assume that an annotation tree node is correct with respect to the reference corpus if the latter contains a node which:

- has the same attributes
- covers the same, possibly non-adjacent, segments on the morphosyntactic level

Thus, a parent node may be correct even if its subnodes are incorrect or incomplete. Consider, for instance, the named entity *Instytutu Podstaw Informatyki Polskiej Akademii Nauk* 'Institute of Computer Science of the Polish Academy of Sciences - genitive' and suppose that:

- annotator *a1* has created a node of type *orgName* covering all 6 words with the lemma *Instytut Podstaw Informatyki Polskiej Akademii Nauk*, and an embedded node of type *orgName* covering the last 3 words with the lemma *Polska Akademia Nauk* 'Polish Academy of Sciences':
  $[Instytutu\ Podstaw\ Informatyki\ [Polskiej\ Akademii\ Nauk]_{orgName}]_{orgName}$
- annotator *a2* has created the same nodes as *a1* but, additionally, he also created an embedded *orgName* node covering the first three words with the lemma *Instytut Podstaw Informatyki* 'Institute of Computer Science'
  $[[Instytutu\ Podstaw\ Informatyki]_{orgName}\ [Polskiej\ Akademii\ Nauk]_{orgName}]_{orgName}$

We assume then that two out of three named entities have been correcly annotated by *a1* wrt *a2* ($P_1 = 1$, $R_1 = 2/3$). If, however, *a1* made a mistake in the lemma of a name (e.g. **Polska Akademia Nauka*) then only one name will be considered correct ($P_1 = 1/2$, $R_1 = 1/3$).

Results of the inter-annotator agreement based on the above assumptions are given in Tab. 2. *Persons*, which are the most numerous NEs, correspond to all NEs of type `persName`, and possibly any of its 3 subtypes. *Locations* represent all NEs of types `geogName` or `placeName` (and any of its 5 subtypes). *Organizations* relate to type `orgName`. *Temporal expressions* designate types `date` and `time`. Finally, *derivations* embrace items having the attribute `derivType` (equal to either `relAdj` or `persDeriv`), and relative to any type and/or subtype. Unsurprisingly, the inter-annotator agreement is much higher for the syntactic annotation (SWs and SGs) than for the NEs, the annotation of which is largely of a semantic nature. Note also that the admitted agreement measure is rather severe as only fully equivalent nodes are considered as correctly annotated. We are considering a more fine-grained weighted measure which would allow for a partial agreement on segments or arguments within two nodes. Thus, nodes covering partially overlapping segments or having partially equivalent attributes would not be considered as totally unrelated and the agreement ratio would certainly increase.

Syntactic words	Syntactic groups	Named entities					
		Persons	Locations	Organi-zations	Temporal expr.	Derivations	Overall
0.99	0.91	0.89	0.78	0.69	0.88	0.71	0.83

Table 2 Inter-annotator agreement results

7 Machine Learning-Based Named Entity Recognition

While machine learning-based (ML-based) named entity recognition (NER) is not a new subject, the interest in automatic recognition of nested NEs has been rather limited. In [4] a few simple methods for recognition of such nested NEs have been described (all reduce the problem to layered sequence tagging). [11] proposed a more refined discriminative constituency parser, with constituents for each named entity. The prototype described here is intended to be a baseline system for nested named entity recognition in NKJP corpus. Thus, following [4], a relatively simple *joined label tagging* method has been chosen for the prototype implementation. Additionally, a comparison of rule-based and machine learning-based methods has been performed. It reveals some possibilities of incorporating both methods in a future more comprehensive NER system.

7.1 Prototype

The prototype has been constructed on the basis of the joined label tagging method described in [4]. This method involves modelling and recognizing nested NEs by means of a conventional sequence tagger. Other methods have been proposed in the article – e.g. *layering*, *cascading* – but only the cascading method outperformed the joined label tagging method. Furthermore, the cascading method proved to be better only for some particular configurations of cascade layers. Therefore, the simplest method seemed most appropriate for our baseline system.

In the joined label tagging method, the nested NER task is reduced to the task of sequence tagging in the following way:

- Each named entity is represented by its subtype (or type, when subtype is not present). Optional information about derivation type is also preserved.
- Standard BIO-encoding method [25] is used to represent NEs on each level of nesting.
- Labels from all levels of nesting are joined to form one, complex label.

For example, the named entity structure for the phrase *Polska Akademia Nauk* 'Polish Academy of Sciences':

$$[[Polska]_{placeName->country(relAdj)}\ Akademia\ Nauk]_{orgName}$$

will be translated into a list of joined tags:

[B_country@relAdj#B_orgName, I_orgName, I_orgName]

Linear-chain Conditional Random Fields (CRFs) formalism has been used for modelling and recognizing joined labels. In CRFs each word is represented with a list of *observations*. The choice of types of observations is important for the quality of modelling and should be well adapted to the character of the specific task – here, NER in Polish language. For now, the following list of observations has been used to obtain initial results: orthographic form, lemma, previous lemma, before-previous lemma, next lemma, capitalization, part of speech, prefixes of length 3 to 5, suffixes of lenght 3 to 5. It could be further extended with: word case, presence of one-word NEs in the gazetteer, additional orthographic information, etc.

For each word observation, two types of *features* have been used in our experiments – $(observation, label_{i-1}, label_i)$ and $(observation, label_i)$, where i represents the position of a particular word in the sentence.

7.2 Preliminary results

Prototype validation has been performed on the part of 1-million-word subcorpus manually annotated with NEs. For sentences not yet super-annotated, one of the two manually constructed NE structures was arbitrarily selected. The validation corpus consisted of 82,212 sentences, i.e., 1,159,970 words and 81,176 NEs. Many of the observation types listed above depend on morphosyntactic annotation. Therefore, results described below relate only to the situation when morphosyntactic annotation (of similar, manual-annotation quality) is available. For the prototype validation, a K-fold cross-validation has been used (with $K = 5$). First, the validation corpus was split at the level of files into 5 samples of similar size, i.e., with both number of sentences and number of NEs greater than 15,000. During each phase of the validation process:

- $K - 1$ corpus samples are used to train the CRF model.
- The remaining sample is randomly divided into two parts of similar size – *dev* and *eval*.
- Each model acquired throughout the iterative training process is evaluated on the *dev* part.
- The model with the highest F-measure on the *dev* part constitutes the result of the training algorithm.

Final evaluation of the respective *eval* parts yielded average precision of 80.5%, recall of 74.5%, and F-measure of 77.3%. All statistics have been computed on the level of NEs, not of labels. A recognized NE was considered to be correct if the following properties were the same as in the manual annotation version: (i) its text range (i.e. the list of the segments from the morphosyntactic level assigned to this NE), (ii) its type and subtype, if any, (iii) its derivation type (`relAdj` or `persDeriv`), if any.

To compare Sprout-based and CRF-based NER, a filtered subcorpus was prepared. It contained only those files, which had been transferred to the files management repository after the last correction of Sprout rules. Evaluation of Sprout performed on the entire corpus would be biased, since earlier files constituted a basis for later improvement of rules. Sprout with the last version of rules was run again on every file from the selected corpus. Sprout's results for each file were compared to the gold standard version of the same file, i.e. the super-annotated version if it already existed, or one of the two human-annotated versions otherwise. Then, the precision, recall and F-measure were calculated with respect to the same rules as in case of CRF model's evaluation.

Tab. 3 shows the compared results of both Sprout and the average performances of the 5 CRF models obtained by the method described above. The basic units in both cases are not individual tokens but whole NEs, possibly multi-word or embedded. The CRF method significantly outperforms Sprout for

recall in all NE categories. In terms of precision, Sprout obtains slightly better results for temporal expressions only. Note that Sprout rules are intended to cover not only the categorisation of NEs into types and sybtypes, but also more subtle morphosyntactic and semantic phenomena: lemmatization, normalization (for dates) and assignment of semantically motivated derivation bases. In the context of the fully automatic annotation of the 1-billion main corpus we think that a hybrid – data-based and rule-based annotation method – would be the most useful. In such a model, the CRF-based tool would be mainly responsible for the identification and categorisation of NEs, while the grammar rules could provide lemmas, normal forms and derivation bases of recognized NEs. The development of such a module is one of our major plans.

NE category	Sprout			CRF (average of 5 models)		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Persons	0.80	0.33	0.47	0.85	0.80	0.82
Locations	0.76	0.46	0.57	0.77	0.70	0.73
Organizations	0.62	0.24	0.35	0.66	0.60	0.63
Temporal expr.	0.85	0.57	0.68	0.82	0.78	0.80
Derivations	0.79	0.58	0.67	0.80	0.72	0.76
Overall	0.78	0.38	0.51	0.80	0.74	0.77

Table 3 Compared results of NE annotation by Sprout and CRF

8 Present Outcome and Future Work

The manual revision of annotation at the SW, SG and NE levels has recently been completed. The revised files have been converted to the final TEI-P5 format. The final revised version of the 1-million-word gold-standard subcorpus contains about 990,000 SWs, 305,000 SGs and 75,000 NEs.

The 1-billion-word main corpus will be further annotated, first with a morphosyntactic tagger trained on the gold standard subcorpus, then with enhanced Spejd grammar. Its annotation on the NE level will be done by machine learning tools trained on the 1-million gold standard subcorpus (cf. section 7).

Tools for verifying the consistency between different annotation levels are being developed. They will allow to check if: (i) references from one annotation level point at segments existing in other levels, (ii) orthographic forms (`orth`) and base forms (`base`), as well as paragraphs are non empty, (iv) morphosyntactic tags are coherent with the admitted tagsets.

Upon accomplishment of all annotation levels the NKJP 1-milion-word gold standard subcorpus will be published integrally under an open licence. The 1-billion-word corpus, however, is subject to various source text copyright constraints and thus can be publicly accessible via a web concordancer only (`http://nkjp.pl/index.php?page=6&lang=1`).

In the future, we hope to extend the scope of annotation in NKJP. First of all, new NE categories like products, events, quantities and measures, etc., should be

taken into account. Second, new annotation levels will be added, starting with the coreference level, which will build upon the levels of SGs and NEs.

9 Conclusions

We have presented the methodology and the technical environment developed for the annotation tasks in the National Corpus of Polish at three levels: syntactic words and syntactic groups (annotated jointly), as well as named entities. Two rule-based annotation platforms, Spejd and Sprout, are used to automatically pre-annotate the corpus. The Spejd grammar developed within this study and currently containing about 1090 rules is among the largest chunking grammars for Polish. The Sprout grammar for named entity recognition is a result of the adaptation of an existing information extraction grammar to the annotation task.

An interoperable tree editor TrEd allows for manual correction of annotations by human experts, as well as for the revision of dual annotations by an adjudicator. NKJP-proper extensions, macros, keyboard shortcuts and stylesheets for TrEd have been developed. Although no particular usability tests have been performed, the feedback from annotators shows that these custom TrEd components enhance both the annotator's and the adjudicator's workbench. Several XML-to-XML converting tools enable the necessary processing chains between Spejd, Sprout, TrEd and the NKJP TEI-P5-conformant encoding standard.

A central versioning repository with custom facilities is responsible for the corpus file management. We think that its architecture is an interesting alternative to web graphical interfaces used in other annotation projects: (i) it reduces the server's load, (ii) the annotators do not have to rely on high-capacity Internet connections – they only connect to the server for down- and uploading the files to be annotated. Here, again, automatic procedures have been developed, such as repository access statistics, automatic creation of file lists, coherence verification, etc. They reduce the annotators' efforts with respect to the manipulated files, facilitate the project management and ensure the security of the corpus.

A prototype of a machine-learning tool for named entity recognition has been developed, trained and evaluated on the gold-standard double-annotated and partially revised NE level. Its initial results show a precision of 80% and a recall of 74%. It is one of the first methods in the international community dedicated to recursively embedded NEs, and probably the first comprehensive machine-learning method for Polish NER (see also [16]).

While the project is still ongoing, we think that its solid technical and organisational environment gives it a good chance to succeed. This assumption has been confirmed by good results of the inter-annotator agreement, particularly in the two syntactic levels (SWs and SGs). We also believe that this environment, or substantial parts of it, can be reused or adapted for other annotation tasks.

Acknowledgements

This research was funded in 2007–2011 by a research and development grant R17-003-03 from the Polish Ministry of Science and Higher Education.

References

[1] Steven Abney. Parsing by Chunks. In Robert Berwick, Steve Abney, and Carol Tenny, editors, *Principle-Based Parsing*, pages 257–278. Kluwer, 1991.

[2] Steven Abney. Partial parsing via finite-state cascades. *Natural Language Engineering*, 2(4):337–344, 1996.

[3] Szymon Acedański. A Morphosyntactic Brill Tagger with Lexical Rules for Inflectional Languages. In *Proceedings of the 7th International Conference on Natural Language Processing, IceTAL 2010, Reykjavík, Iceland*, LNAI, Berlin, 2010. Springer-Verlag.

[4] Beatrice Alex, Barry Haddow, and Claire Grover. Recognising nested named entities in biomedical text. In *BioNLP '07: Proceedings of the Workshop on BioNLP 2007*, pages 65–72, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

[5] Markus Becker, Witold Drożdżyński, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. SProUT - Shallow Processing with Typed Feature Structures and Unification. In *Proceedings of ICON 2002, Mumbay, India*, 2002.

[6] Eduard Bejček and Pavel Straňák. Annotation of multiword expressions in the Prague dependency treebank. *Language Resources and Evaluation*, 44(1–2):7–21, 2010.

[7] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, 2002.

[8] Aleksander Buczyński and Adam Przepiórkowski. ♠ Demo: An Open Source Tool for Shallow Parsing and Morphosyntactic Disambiguation. In *Proceedings of LREC 2008, Marrakech*, 2008.

[9] Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. The Prague Dependency Treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*, pages 103–127. Kluwer, Dordrecht, 2003.

[10] Jacob Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20:37–46, 1060.

[11] Jenny R. Finkel and Christopher D. Manning. Nested named entity recognition. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 141–150, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[12] Dan Flickinger. On building a more efficient grammar by exploiting types. In Stephan Oepen, Dan Flickinger, Jun'ichi Tsujii, and Hans Uszkoreit, editors, *Collaborative Language Engineering*, pages 1–17. CSLI Publications, Stanford, CA, 2002.

[13] Katarzyna Głowińska and Adam Przepiórkowski. The Design of Syntactic Annotation Levels in the National Corpus of Polish. In *Proceedings of LREC 2010*, Valletta, Malta, 2010.

[14] Valia Kordoni and Yi Zhang. Annotating Wall Street Journal Texts Using a Hand-Crafted Deep Linguistic Grammar. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III) at ACL-IJCNLP 2009, Singapore*, pages 170–173, 2009.

[15] Markéta Lopatková, Natalia Klyueva, and Petr Homola. Annotation of Sentence Structure; Capturing the Relationship among Clauses in Czech Sentences. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III) at ACL-IJCNLP 2009*, pages 74–81, Singapore, 2009.

[16] Michał Marcińczuk and Maciej Piasecki. Study on Named Entity Recognition for Polish Based on Hidden Markov Models. *Lecture Notes in Computer Science*, 6231:142–149, 2010. Springer.

[17] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, 1993.

[18] Anna Nedoluzhko, Jiří Mirovský, and Petr Pajas. The Coding Scheme for Annotating Extended Nominal Coreference and Bridging Anaphora in the Prague Dependency Treebank. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III) at ACL-IJCNLP 2009*, Singapore, 2009.

[19] Petr Pajas and Jan Štěpánek. Recent Advances in a Feature-Rich Framework for Treebank Annotation. In *Proceedings of COLING'08, Manchester*, 2008.

[20] Jakub Piskorski. Named-Entity Recognition for Polish with SProUT. In *LNCS Vol 3490: Proceedings of IMTCI 2004, Warsaw, Poland*, 2005.

[21] Adam Przepiórkowski, Rafał L. Górski, Barbara Lewandowska-Tomaszczyk, and Marek Łaziński. Towards the National Corpus of Polish. In *Proceedings of LREC 2008, Marrakech*, 2008.

[22] Adam Przepiórkowski. On Heads and Coordination in a Partial Treebank. In *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2006)*, Prague, 2006.

[23] Adam Przepiórkowski. *Powierzchniowe przetwarzanie języka polskiego*. Akademicka Oficyna Wydawnicza EXIT, Warsaw, 2008.

[24] Adam Przepiórkowski, Rafał L. Górski, Marek Łaziński, and Piotr Pęzik. Recent Developments in the National Corpus of Polish. In *Proceedings of LREC 2010, Valletta, Malta*, 2010.

[25] Lance A. Ramshaw and Mitchell P. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third Workshop on Very Large Corpora (ACL 1995)*, pages 82–94, 1995.

[26] Agata Savary and Jakub Piskorski. Lexicons and Grammars for Named Entity Annotation in the National Corpus of Polish. In *Proceeding of IIS'10, Siedlce, Poland*, 2010.

[27] Agata Savary, Jakub Waszczuk, and Adam Przepiórkowski. Towards the Annotation of Named Entities in the Polish National Corpus. In *Proceedings of LREC 2010*, Valletta, Malta, 2010.

[28] Graham Wilcock. *Introduction to Linguistic Annotation and Text Analytics*. Morgan & Claypool, 2009.

[29] Marek Świdziński and Marcin Woliński. Towards a bank of constituent parse trees for Polish. In *Text, Speech and Dialogue: 13th International Conference, TSD 2010, Brno, Czech Republic*, Lecture Notes in Artificial Intelligence, Berlin, 2010. Springer-Verlag.