Proceedings of the International Multiconference on Computer Science and Information Technology pp. 531–539
ISBN 978-83-60810-27-9
ISSN 1896-7094

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

Jakub Waszczuk*†, Katarzyna Głowińska*, Agata Savary◇*, Adam Przepiórkowski*†
*Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland
◇Université François Rabelais Tours, Laboratoire d'Informatique, 3 pl. Jean-Jaurès, 41000 France
†University of Warsaw, ul. Banacha 2, 02-097 Warsaw, Poland
jw235843@students.mimuw.edu.pl, k.glowinska@gmail.com, agata.savary@univ-tours.fr, adamp@ipipan.waw.pl

***Abstract*—The on-going project aiming at the creation of the National Corpus of Polish assumes several levels of linguistic annotation. We present the technical environment and methodological background developed for the three upper annotation levels: the level of syntactic words and groups, and the level of named entities. We show how knowledge-based platforms Spejd and Sprout are used for the automatic pre-annotation of the corpus, and we discuss some particular problems faced during the elaboration of the syntactic grammar, which contains over 800 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 the automatic data flow and efficient corpus file management. We believe that this environment or substantial parts of it can be reused in or adapted for other corpus annotation tasks.**

I. Introduction

The National Corpus of Polish (Pol. *Narodowy Korpus Języka Polskiego*; NKJP; http://nkjp.pl/) is a 3-year project (2007-2010), involving a consortium of four partners coordinated by the Institute of Computer Science, Polish Academy of Sciences ([1], [2]). The aim of the project is to create a 1-billion (10^9) word corpus of Polish annotated at various levels, with a 300-million balanced subcorpus and a number of annotation tools. The following linguistic annotation layers are distinguished: segmentation (word-level and sentence-level), morphosyntax, word sense disambiguation (limited to around 100 lexemes), syntactic words, syntactic groups and named entities.

A 1-million word balanced subcorpus undergoes manual annotation at all these layers and it serves as a training corpus for various annotation tools (cf., e.g., [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 – layers.

The layer of syntactic words (SWs) builds on top of the morphosyntactic layer. Fine-grained word-level segments are grouped into more traditional words, including reflexive verbs (consisting of two segments: the verb and the reflexive marker), analytical tense and mood forms, etc. [4] Syntactic groups (SGs) are constructed on top of SWs, and include nominal groups, prepositional groups, clause-level groups, etc., but no attempt is made to solve attachment ambiguities. Finally, named entities (NEs), i.e. proper names of persons, geographical objects and organisation, as well as temporal expressions, refer again to the layer of morphosyntactically annotated segments.

This research was funded in 2007-2010 by a research and development grant R17-003-03 from the Polish Ministry of Science and Higher Education.

II. Related Work

Some of the first treebanks were constructed fully manually, by drawing trees for particular sentences; this is the case, for example, for the Penn Treebank (PTB) of English [5], the German Negra/Tiger Treebank [6] and the Prague Dependency Treebank [7]. 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. However, the usual way of developing new treebanks consists in the automatic parsing of texts and the manual selection of the right parse. For example, [8] reports that the ERG grammar [9] 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., [10], [11], as well as [4]), where structural ambiguity is not an issue.[1] Hence, the approach mentioned above, focussing 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 emission reports for the improvement of the grammar, before applying it to the next batch of sentences.

There are many multi-layer corpora developed by now, typically containig morphosyntactic, (deep) syntactic and some semantic and/or discourse representation. For example, the Prague Dependency Treebank mentioned above has these three layers (called morphological, analytical and tectogrammatical), currently further extended with coreference [13] and high-level

[1] 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. [12].

inter-clausal structure [14]. The current project adopts a more fine-grained and conservative approach, with three layers 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.

Also at the level of NEs, the annotation strategies adopted here are rather fine-grained, namely, not only the longest-match occurrences of NEs are annotated, but also all recursively embedded ones, and, moreover, overlappingly coordinated NEs are appropriately marked (cf. [15], [16]).

Let us finally 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 case of prepositional groups, the preposition serves as the syntactic head[2], 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 [17], [4].

III. Annotation Data Flow

The three syntactic annotation layers 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 case of the syntactic annotation (Fig. 1) a shallow parsing system called Spejd is used to extract SWs and SGs from the underlying morphosyntax level (cf. section IV-A). For the similar (from the data flow point of view) task of NE recognition another platform, Sprout, is used (see section IV-B).

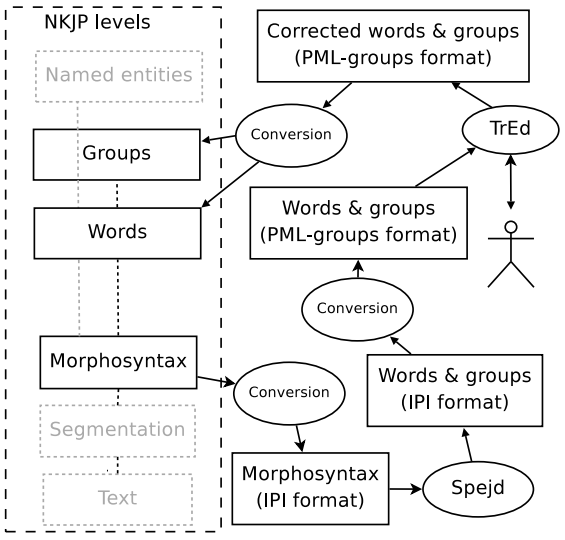

Fig. 1. Data flow in the syntactic annotation task of the NKJP corpus

[2]because it governs the case inflection of its arguments

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). A raw text taken from the corpus repository is processed by lexical resources and grammar rules. The NEs identified in the process, together with their embedded structures, are marked in an XML Sprout-specific output. Since Sprout outputs the cardinal numbers of the beginning and ending characters of each recognized sequence, the converter consults the segmentation level of the text in order to translate text ranges into token identifiers. Moreover, for each token, its morphological tag and lemma are recopied from the morphosyntactic annotation of the text.

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 V) to examine and correct results of automatic annotation. Again, files have to be translated to TrEd-readable formats (PML-groups for syntactic annotation and PML-NE for NEs), which were defined using the Prague Markup Language (PML, http://ufal.mff.cuni.cz/jazz/PML/doc/). Due to the sampling methodology adopted in NKJP, files contained in the 1-million gold standard subcorpus are of a very variable length (from several to several thousand sentences). They do not correspond to complete texts taken from the 1-billion word corpus, but to randomly chosen paragraphs thereof. For the sake of ergonomy, it is important to present the annotator with text portions of a uniform length, thus easily manageable. Therefore, the converter divides each text which 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. Conversely, too small files (of one or several sentences), are organised into file lists and annotated as bigger units.

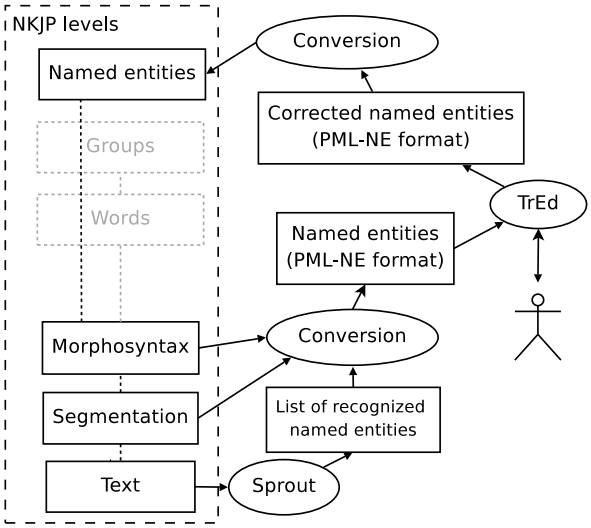

Fig. 2. Data flow in the NEs annotation task of the NKJP corpus

Finally, PML files are transferred to *corpus files management system* (see section V-C) 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 offline with TrEd installed locally, connecting to the subversion repository only to send results of his work, or to download new files. Two TrEd extensions, NKJP_groups and NKJP_names, have been developed to support annotation of PML-groups and PML-NE files. An annotator can download (or upgrade) extensions from within the TrEd application, with no need to run separate installation process. Despite no particular computing background of the annotators, they successfully install and operate the whole annotating platform.

The last stage consists in converting the PML formats of the validated annotations into the final NKJP formats. 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).

IV. Automatic Annotation

A. Shallow Parsing with Spejd

Syntactic annotation in the National Corpus of Polish consists in joining words together into constituents: first at the level of SWs, then at the level of, possibly embedded, SGs. At the former, fine-grained word-level tokens are replaced by coarse-grained SWs (e.g., analytical tense and mood forms, analytical degree forms, reflexive verbs, discontinuous conjunctions, etc.). 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 is presented in [18].

At the SG level, each identified group is annotated with pointers to its syntactic head (SynHead) and semantic head (SemHead). Only those groups that can be recognized with very high accuracy 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 that consists 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*), without an attempt to solve PP-attachment ambiguities. We make an exception for compound prepositions that consist of two prepositions and an interposing noun (e.g., *w odniesieniu do* 'with reference to'), as well as for elective constructions (e.g., *jeden z najlepszych* 'one of the best').

The following SGs are distinguished in NKJP:

- nominal group (NG): *malarz kwiatów* 'a painter of flowers', *nic specjalnego* 'nothing special',
- numeral group (NumG): *stu z nas* 'a hundred of us',
- adjectival group (AdjG): *zbyt długi* 'too long',
- prepositional-nominal group (PrepNG): *za murami miasta* 'beyond city walls',
- prepositional-adjectival group (PrepAdjG): *[wyglądasz] na zmęczonego* '[you look] tired',
- prepositional-numeral group (PrepNumG): *[wakacje] dla dwojga* '[a holiday] for two',
- adverbial group (AdvG): *ładnie*[3] 'nicely',
- discourse group (DisG): *no cóż* 'oh well', *itd.* 'etc.',
- subordinate clause (CG) (with subordinate conjunction): *[powiedział], że nie przyjdzie* '[he said] he wouldn't come',
- interrogative clause (KG): *[nie rozumiem], dlaczego to zrobił* '[I don't understand] why he's done it'.

The manually constructed grammar, for both SWs and SGs, is encoded in the shallow parsing system Spejd (http://nlp.ipipan.waw.pl/Spejd/) [19], 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 rule. 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, so as to avoid excessive matching, and in order to detect errors on the underlying morphosyntactic level. Firstly, as a parser finds a match for a lemma it is usually checked for grammatical class. For example, in the rule for *nie tylko ..., lecz także* (see below), the word *nie* 'not' must be marked as `conj` and not `qub`. Secondly, rules are made maximally specific in that some SGs are divided into several subtypes, e.g., there are 11 types of nominal groups: `NGa` (`Noun`[4]`+Adj`), `NGs` (`Noun+Noun` with the same value of case), `NGg` (`Noun+Noun`$_{\text{gen}}$), `NGk` (`Noun+and+Noun`), `NGn` (`Noun+Num`), `NGb` (`Noun+Brev`[5]), `NGe` (`Noun` as a head of the elective construction), `NGx` (`PPron3+Adj`, e.g. *something special*), as well as special groups `NGadres, NGgodz, NGdata`, for describing addresses, hours and dates. So, instead of the plain `NG`, an alternative of subtypes is given in the rule, e.g., `NGa|NGs|NGk|NGg|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.

```
Rule    "frazeo: nie tylko [lecz także]"
Match:  [base~"nie" && pos~"conj"]
        [base~"tylko" && pos~"conj"];
Right:  []+ ns [base~","]
        [base~"lecz" && pos~"conj"]
        [base~"także"]?;
Eval:   word(Conj1:discr, "nie tylko");
```

The above rule[6] identifies the first part of a discontinuous

[3]Recall that a SG may contain one or several SWs.

[4]In fact it is a nominal group, not a single noun.

[5]`Brev` stands for an abbreviation.

[6]In this rule `ns` stands for "no space" between tokens, the operator `~` means equal, `&&` denotes logical conjunction, `Conj1` is a grammatical class tag for the first part of the discountinuous conjunction and `discr` is a value of the `continuity` attribute.

conjunction *nie tylko . . . , lecz także* 'not only . . . but also' (note that there must be second part of this conjunction in the right context).

Two types of syntactic operations are available: `word`, that joins tokens into SWs, and `group`, that 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 referred to in this way must be unique – it is impossible to inherit information from different components. For example, an 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. Here, the gender, instead of being inherited from the third component, is explicitely fixed to be feminine. Similar rules have thus 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 3 sentences'), has three arguments: 1) the type of the SG, 2) the reference to the SynHead of the phrase (*po*), 3) the reference to its SemHead (*trzech*).

```
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,1,3);
```

The problems encountered in pointing at SynHeads and SemHeads were:

- absent heads
  The SemHead of an interrogative clause is a finite verb. In the sentence *Nie wiem, kiedy i ile.* 'I don't know when and how many.' there is no verb in the subordinate clause. In this case the SemHead is made equal to SynHead, here *kiedy*.
- coordination
  In a coordinated group (e.g., *rząd i parlament* 'government and parliament'), the first element is marked as both the semantic and the syntactic head. If the conjunction were the syntactic head of the group, any information about the part of speech, case, etc., of the conjuncts would be lost, which would render such a conjunction group practically invisible to further syntactic rules.

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

TABLE I
TAXONOMY AND QUANTITIES OF SPEJD RULES

Syntactic words			Syntactic	TOTAL
multiword entities	abbreviations	others	groups	
339	360	122	242	1063

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 lesser 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 matching context data in rules.

B. Named Entity Recognition with Sprout

As discussed in [16], the automatic pre-annotation of named entities in NKJP is done by the general-purpose knowledge-based NLP platform Sprout [20]. 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 [21] 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 or 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. Initial results show the overall precision of 88% and recall of 61%.

V. 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[7], MMAX[8], and GATE [22], before selecting the tree editor TrEd[9] [23] 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

[7] http://www.lat-mpi.eu/tools/synpathy
[8] http://mmax2.sourceforge.net
[9] http://ufal.mff.cuni.cz/~pajas/tred/

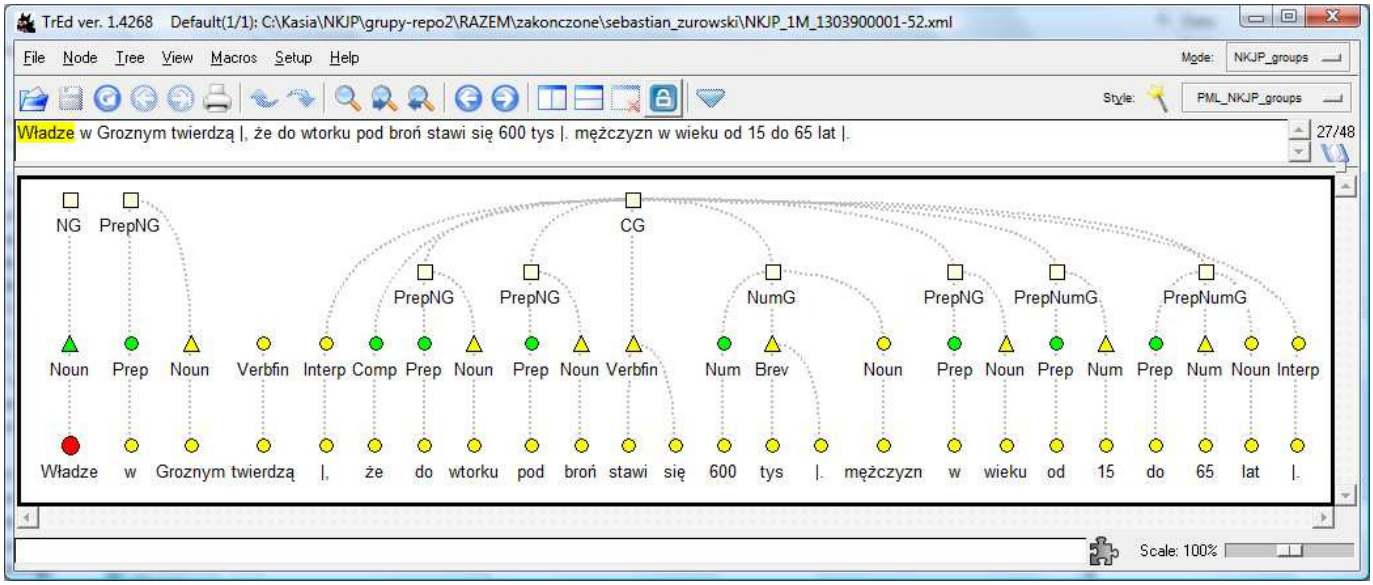

Fig. 3. Syntactic annotation with the use of the TrEd editor for the sentence 'The authorities in Grozny claim that 600 thousand men of 15 to 65 years of age will turn up to arms till Tuesday.'

graphical user's interface, (vi) parallel edition of concurrent annotations, (vii) rich documentation, (viii) technical reliability. TrEd is also widely used by the international community – it scored as the second most used annotation tool on the LREC 2010 map of language resources and tools.

A. Annotator's Workbench

1) Workbench for Syntactic Words and Groups: As shown in Fig. 3, in the central part of a TrEd's window the annotation tree of the sentence is shown. Nodes are situated on three horizontal levels, which represent (from bottom to top) segments, SWs and SGs. The annotator can add or remove nodes, and draw edges between them. Each node has a set of type-specific attributes editable in a separate window on double-clicking the node. Toolbar icons are useful for navigation between sentences (or files), as well as undo and redo actions.

For each annotation level special *TrEd extention* has been prepared. NKJP_groups extention for syntactic annotation supplies a set of macros and keyboard shortcuts for: adding a SW or a SG, adding a regular or a secondary edge[10], pointing at the SynHead or SemHead, grouping multiple nodes at a time, etc. It also provides a PML schema defining PML-groups format and a stylesheet with encoded rules of syntactic tree visualisation. In Fig. 3 the SynHead of each constituent is marked in green and the SemHead is marked with a triangle. Thus, *Władze* is both a SynHead and a SemHead while *do* is a SynHead and *wtorku* a SemHead head only.

When closing a file, a final checkup is done via TrEd in that missing SynHeads and SemHeads of each group are reported.

2) Workbench for Named Entities: The annotator's workbench for named entities is presented in detail in [15]. We recall here its main facilities with respect to the relatively rare

[10]A secondary edge is used in case of overlapping segments.

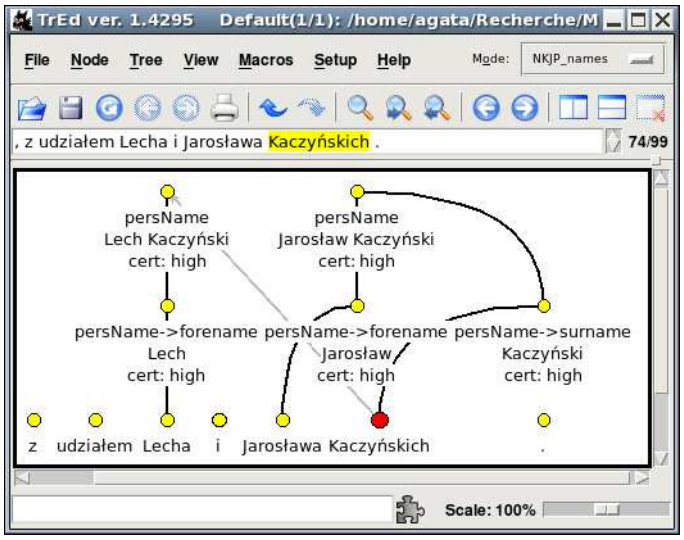

Fig. 4. Annotating coordinated names in the phrase 'with the participation of Lech and Jarosław Kaczyński'

but linguistically difficult phenomenon of overlapping in coordination. Fig. 4 shows a sentence in which two coordinated names have been annotated according to the guidelines. Both of them are personal names with an embedded forename and a common embedded surname *Kaczyńskich*, which appears only once. Since TrEd does not allow one tree node to have two father nodes, the highlighted node representing the surname is assigned to one father by a regular edge, and to another one by a secondary edge (in grey). As in all trees, most attributes of a node are visible below it, and their rapid modification is possible by mouse clicks. These attributes include: the type and subtype (*persName→forename*), the lemma (*Lech Kaczyński*), the derivation type and base, if

any (irrelevant here), and the certainty degree of the annotation (*cert:high*).

B. Workbench for Revision of Annotations

As mentioned above, each text of the gold standard subcorpus is to be annotated at each level by two annotators. Disagreement cases are further reviewed and resolved by an adjudicator (called *super-annotator*), who usually is a person with rich previous experience in annotation at the same 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. Thus, the super-annotator does not review annotations on which the two annotators agree. Another macro exists for an automatic transfer of 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 hasn't been annotated as a NE by *a2*. The nodes corresponding to this discrepancy are highlighted in red in both windows. By a single keyboard shortcut we can transfer the missing node to the lower window, over node *France* and under 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 disrepancies in annotated SWs and SGs.

C. Managment of Corpus Files

Corpus files management system consists of two main components. The first one is the svn[11] 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 his own, private directory in the repository. There he keeps currently annotated files, which he can modify and send back to the `/zakonczone` directory in the subversion repository – 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 – he can edit files in the private directory, but cannot add, move or delete files in the repository. The additional functionality – downloading

[11] Version control system, keeps track of changes made in maintained files.

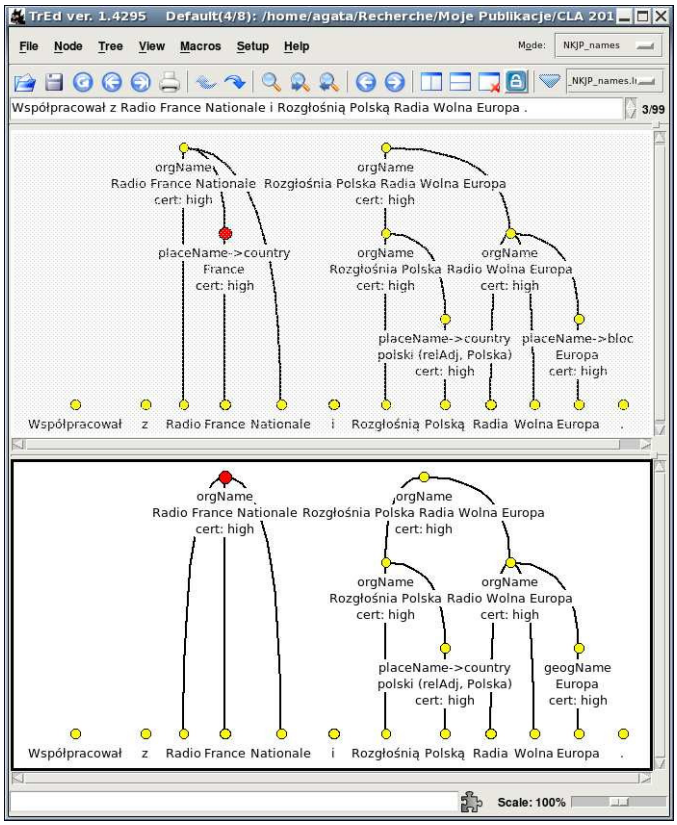

Fig. 5. Comparing two NE annotations in TrEd for the same sentence 'He collaborated with Radio France Nationale and the Polish Station of the Free Europe Radio.'

files for annotation and sending off the completed files – is realised 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 his private directory, the annotator has to add the `checkout = 5` line to the `message.txt` file, and run *svn commit* [12] and *svn update* [13] on the directory. The rest of the work – finding appropriate files and moving them throughout repository – is performed on the server side by means of a post-commit subversion hook[14]. Another command, *checkin* (with `checkin = FILE_NAME` syntax), can be used to send annotated files to the `/zakonczone` directory.

For super-annotation, similar commands, *s_checkout* and *s_checkin*, exist. The *s_checkout* command will download a chosen number of files to compare – every file in two copies, validated by two different annotators. It is guaranteed that the super-annotator will not get files which he has previously seen. The super-annotator corrects one of the downloaded files, using the NKJP_diff TrEd extension to compare it with its second copy (see section V-B), and finally calls *s_checkin* to send corrected version to the `s_zakonczone` directory.

While the `message.txt` file can be modified by the anno-

[12] Sends locally modified files to central subversion repository.
[13] Brings changes from repository into local directory.
[14] Process run on server after every `commit` operation.

tator 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 `/nowe` directory is stored automatically in the database. When files are downloaded or sent by an annotator (*[s_]checkout* and *[s_]checkin* operations), his name and the operation date are also saved in the `db.xml` file. There are two main reasons for saving this kind of information in a separate database 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. Extending repository with database brings about the need for additional integrity-preservation mechanism. Atomicity of generic svn operations is guaranteed, but in case of *[s_]checkout/[s_]checkin* operations the whole process (commit and post-commit) has to be carried out in one transaction. As a solution to this problem, a FIFO[15] with one element has been set up on the server. Every special operation has to borrow an element from the FIFO in advance (and return it when operation is completed), so two post-commit processes will never modify the repository 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 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). Additionaly, the tool can be used to find files left for annotation (that is, files which haven't been downloaded by two annotators yet).

D. Project Managment

Multi-level corpus annotation such as in NKJP is a complex and labor-intensive task. To ensure the coherence of annotations, detailed annotation guides have been edited for both tasks, as well as a Frequently Asked Questions list for the NE task. Additionally, NKJP-proper user's guides have been prepared for TrEd and for the svn client tool SVNTortoise[16] used by the annotators. All these documents are regularly updated and diffused via the repository.

Currently, the team working on the two annotation levels consists of three project managers, one programmer, 15 annotators for the syntactic level, and 6 for the named entities. The project managers and the programmer meet on a monthly basis, while communication with the annotators is mainly maintained via discussion lists. The annotation of SWs and SGs seems particularly challenging, as witnessed by the rather rich activity on the corresponding discussion list. During a sample week about 70 messages have been sent to the list containing: (i) mentions of new multi-word entities to be accounted for, (ii) proposals of new grammar rules, (iii) errors on the underlying morphosyntactic level, (iv) various problems with the scope, type and heads of SWs and SGs such as time expressions, fractions, internet and postal addresses, and unexpected syntactic constructions (e.g., *od kiedy* 'since when' is a group of the *preposition-adverb* type, which is not allowed by traditional grammars). The discussion list for NEs mainly receives questions whether a given sequence should or should not be annotated (e.g., names of animals), and doubts about the type and subtype of a NE (e.g., *Palestyna, Kosowo, Arab*).

[15]Named pipe, inter-process communication method.

[16]`http://tortoisesvn.tigris.org`

VI. Links between Two Annotation Levels

Clearly, SGs are tightly connected to NEs. However, as discussed by [24] and [25], a NE does not necessarily precisely coincide with a nominal group. The following types of mutual relations between NEs and SGs were identified in our corpus:

- a NE coincides with a nominal group, e.g., *Stany Zjednoczone* 'United States',
- a NE is a subsequence of a nominal group, e.g., *[ksiądz biskup [Leszek Głódź]$_{persName}$]$_{NG}$* 'priest bishop Leszek Głódź',
- a NE embraces a sequence of SWs and SGs, e.g., *[[Komisja Badań]$_{NG}$ [na Rzecz Rozwoju Gospodarki]$_{PrepNG}$]$_{orgName}$* 'Research Commission for Economical Development'.

A partial overlapping of a NE and a SG seems infeasible, and we plan to detect such problems during the final corpus consistency check.

With the above typology it is clear, on one hand, that a pipelined processing of the annotations on both levels would not be a satisfactory solution. On the other hand, a completely joint processing of both levels seems rather complex, and inconsistent with pre-existing multi-format resources for Polish. Thus, we think that a parallel processing of both annotation levels is a good solution, even if some knowledge must be encoded twice (e.g., some Spejd rules have to cover most frequent types of NEs, described also in more details by the Sprout grammar). We believe that a common project managment of both tasks, enhanced with shared communication means, as described above, helps in assuring the consistence of the annotations.

VII. Present Outcome and Future Work

The annotation on the SW, SG and NE levels is currently at its zenith. Until mid-August, out of about 85,000 sentences of the gold standard subcorpus over 41,000 have been double-annotated for SWs and SGs, and 73,000 for NEs. Thus, 15% through 52% of the corpus remains to be annotated, including particularly demanding extracts of spoken dialogue data. The revision of annotations is done for 14% of the corpus for SWs and SGs, and is just starting for NEs. Moreover,

the whole corpus needs to be revised by super-annotators. Further, the 1 billion-word main corpus will be annotated first with a morphosyntactic tagger trained on the gold standard subcorpus, then with enhanced Spejd and Sprout grammars, finally additional machine learning tools will complete the automatic NE annotation. We are also currently developing the final conversion tool that will allow to obtain the TEI P5-conformant format (cf. [26], [27]) for both annotation levels out of the TrEd PML format.

In future, we expect the NKJP corpus to be used in advanced linguistic corpus studies of Polish, and as a training or evaluation corpus for various linguistic processors such as taggers, parsers, and information retrieval engines.

VIII. Conclusions

We have presented the methodology and the technical environment developed for the annotation tasks in the National Corpus of Polish on 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 containing currently circa 1063 rules is among the largest chunking grammars for Polish.

An interoperable tree editor TrEd allows for manual correction of annotations by human experts, as well as for the revision of dual annotations by a super-annotator. NKJP-proper extensions, macros, keyboard shortcuts and stylesheets for TrEd enhance the annotator's and super-annotator'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 managment. We think that its architecture is an interesting alternative to web graphical interfaces used in other annotation projects: (i) it limits the server's charge, (ii) the annotators do not have to rely on constant 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.

While the project is still on-going, we think that its solid technical and organisational environment gives it a good chance of success. We also believe that this environment, or substantial parts of it, can be reused or adapted for other annotation tasks.

References

[1] A. Przepiórkowski, R. L. Górski, B. Lewandowska-Tomaszczyk, and M. Łaziński, "Towards the National Corpus of Polish," in *Proceedings of LREC 2008, Marrakech*, 2008.

[2] A. Przepiórkowski, R. L. Górski, M. Łaziński, and P. Pęzik, "Recent Developments in the National Corpus of Polish," in *Proceedings of LREC 2010, Valletta, Malta*, 2010.

[3] S. 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*, ser. LNAI. Berlin: Springer-Verlag, 2010.

[4] A. Przepiórkowski, *Powierzchniowe przetwarzanie języka polskiego*. Warsaw: Akademicka Oficyna Wydawnicza EXIT, 2008.

[5] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, pp. 313–330, 1993.

[6] S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith, "The TIGER treebank," in *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, 2002.

[7] A. Böhmová, J. Hajič, E. Hajičová, and B. Hladká, "The Prague Dependency Treebank: Three-level annotation scenario," in *Treebanks: Building and Using Parsed Corpora*, ser. Text, Speech and Language Technology, A. Abeillé, Ed. Dordrecht: Kluwer, 2003, vol. 20, pp. 103–127.

[8] V. Kordoni and Y. 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*, 2009, pp. 170–173.

[9] D. Flickinger, "On building a more efficient grammar by exploiting types," in *Collaborative Language Engineering*, S. Oepen, D. Flickinger, J. Tsujii, and H. Uszkoreit, Eds. Stanford, CA: CSLI Publications, 2002, pp. 1–17.

[10] S. Abney, "Parsing by chunks," in *Principle-Based Parsing*, R. Berwick, S. Abney, and C. Tenny, Eds. Kluwer, 1991, pp. 257–278.

[11] ——, "Partial parsing via finite-state cascades," *Natural Language Engineering*, vol. 2, no. 4, pp. 337–344, 1996.

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

[13] A. Nedoluzhko, J. Mirovský, and P. 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.

[14] M. Lopatková, N. Klyueva, and P. 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*, Singapore, 2009, pp. 74–81.

[15] A. Savary, J. Waszczuk, and A. Przepiórkowski, "Towards the Annotation of Named Entities in the Polish National Corpus," in *Proceedings of LREC 2010*, Valletta, Malta, 2010.

[16] A. Savary and J. Piskorski, "Lexicons and Grammars for Named Entity Annotation in the National Corpus of Polish," in *Proceeding of IIS'10, Siedlce, Poland*, 2010.

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

[18] K. Głowińska and A. Przepiórkowski, "The Design of Syntactic Annotation Levels in the National Corpus of Polish," in *Proceedings of LREC 2010*, Valletta, Malta, 2010.

[19] A. Buczyński and A. Przepiórkowski, "♠ Demo: An Open Source Tool for Shallow Parsing and Morphosyntactic Disambiguation." in *Proceedings of LREC 2008, Marrakech*, 2008.

[20] M. Becker, W. Drożdżyński, H.-U. Krieger, J. Piskorski, U. Schäfer, and F. Xu, "SProUT - Shallow Processing with Typed Feature Structures and Unification," in *Proceedings of ICON 2002, Mumbay, India*, 2002.

[21] J. Piskorski, "Named-Entity Recognition for Polish with SProUT," in *LNCS Vol 3490: Proceedings of IMTCI 2004, Warsaw, Poland*, 2005.

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

[23] P. Pajas and J. Štěpánek, "Recent Advances in a Feature-Rich Framework for Treebank Annotation," in *Proceedings of COLING'08, Manchester*, 2008.

[24] P. Osenova and S. Kolkovska, "Combining the named-entity recognition task and NP chunking strategy for robust pre-processing," in *Proceedings of the Workshop on Linguistic Theories and Treebanks*, Sozopol, Bulgaria, 2002.

[25] J. R. Finkel and C. D. Manning, "Nested Named Entity Recognition," in *Proceedings of EMNLP-2009*, Singapore, 2009.

[26] A. Przepiórkowski and P. Bański, "Which XML standards for multilevel corpus annotation?" in *Proceedings of the 4th Language & Technology Conference*, Poznań, Poland, 2009.

[27] A. Przepiórkowski, "TEI P5 as an XML Standard for Treebank Encoding," in *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT 8)*, Milan, Italy, 2009, pp. 149–160.