The Packaged TEI P5-based Stand-off Annotation Format[⋆]

Maciej Ogrodniczuk

Institute of Computer Science, Polish Academy of Sciences

Abstract. This document describes the "packaged" version of the National Corpus of Polish-based TEI P5 stand-off annotation format, recently tested with the linguistic Web Service for Polish. It is intended to be used for multilevel, language-independent interlinked representation and interchange of linguistic properties.

Keywords: linguistic annotation, language representation, treebank representation, linguistic interchange

1 Introduction

The National Corpus of Polish (PL: Narodowy Korpus Języka Polskiego, hence NKJP) proposed a stand-off, TEI P5-based [2] format for representation of linguistic information which stores different annotation levels in separate, interlinked files [1, 5]. For interchange, a certain method of packaging them must be applied. Among several possible methods (e.g. with additional minimalistic linkbase file referencing the annotation files and bundled together – similarly to OpenDocument representation) following the TEI path seems the most reasonable choice. It allows maintenance of compatibility with the current standard while keeping encoding overhead not substantially different from competitive interchange formats (not taking into account the extensive metadata description by TEI).

The following sections present the target format and initial conclusions from its usage in the ICS PAS Web Service (later referred to as *multiservice*[1]) offering annotation of Polish texts basing on individual offline tools, made available in a common online interface.

2 The Package Structure

Four important decisions in the process of selecting the annotation model for a linguistic corpus (which could be later extended to any representation of linguistic data) are related to:

[⋆] The work reported here was carried out within the Common Language Resources and Technology Infrastructure (CLARIN) project co-funded by the European Commission under the 7th Framework Programme (Grant Agreement No 212230).

[1] See `http://chopin.ipipan.waw.pl:8083/WSWebClient/`.

1. basic representation of text encoding and format — made more or less indisputable by the advantages offered by XML and UTF-8, confirmed by widespread use of these standards,
2. concrete encoding method within a consistent framework — in our case facilitated by earlier decision of using TEI P5 in the process of encoding NKJP[2],
3. packaging method — initially commented in the introductory section and resulting in the easiest possible option of creating a corpus out of separate annotation layers,
4. representation of linguistic information — here: NKJP-based, but potentially still open to improvements with respect to the richness of TEI and generality of feature structures [3].

Of the above, the third and fourth decisions seem to need certain explanation. A competitive method of encoding a collection of annotations could rely on `<group>` elements (*grouping together a sequence of distinct texts (or groups of such texts) which are regarded as a unit for some purpose*, as indicated in the TEI Guidelines[3]), however, this idea should be turned down due to numerous reasons: different practices of usage of this element (for *collected works of an author, a sequence of prose essays, etc.*[3]), extensive overhead in representation of annotation descriptions (such as necessity of using `<front>` for composite `<text>`s), impossibility of encoding different headers for annotation components (which is useful for separate processing of annotation layers) and, last but not least, incompatibility with NKJP encoding (inability to store `<teiCorpus>` elements inside any other element than `<teiCorpus>` itself).

As for representation of linguistic information and linking it to the base text, TEI offers numerous, often much simpler methods for this purpose (e.g. by means of `<text>`s with `<interp>`retations of a certain `type` linked to the base text with `inst` attribute). After serious consideration the NKJP format was selected for practical reasons: no significant benefit could be offered by any new format with respect to the number of existing tools and practices already applied in the currently mature NKJP project.

Summarizing, the adopted annotation model consists of:

– a single corpus (`<teiCorpus>`) as a means of representation of the collection of source text and annotations,
– several subcorpora (second-level `<teiCorpus>` elements) each representing a different level of annotation (sentence-level segmentation, tokenization etc.),
– individual texts in every subcorpus (`TEI`), each representing a variant of annotation on a given level (e.g. results of tagging coming from different taggers).

[2] The comparison of this annotation model to other existing encoding standards such as SynAF, XCES or PAULA can be found in [4].

[3] See [2], description of `<group>` element in Appendix C: Elements (`http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-group.html`).

The structure of the output package is therefore:

```
<teiCorpus>
  <teiHeader>
    <!-- header information for the package -->
  </teiHeader>
  <teiCorpus>
    <!-- single annotation level -->
    <teiHeader>
      <!-- header information for the annotation level -->
    </teiHeader>
    <TEI>
      <!-- single annotation variant -->
      <teiHeader>
        <!-- header information for the variant annotation -->
      </teiHeader>
      <text>
        <body>
          <!-- annotation details -->
        </body>
      </text>
    </TEI>
    <!-- possibly more annotation variants of the current level
         in the form of <TEI> elements -->
  </teiCorpus>
  <!-- possibly more annotation levels in <teiCorpus> elements -->
</teiCorpus>
```

As required by the TEI document model, the corpus package must always contain at least one annotation level with at least one annotation variant. This assumption is satisfied in very straightforward manner since every annotation model should contain at least basic segmentation in the form of the TEI-encoded source (see 4.2).

3 TEI Headers

All three types of headers in the package (the package header, headers for subcorpora representing annotation variants and headers for individual variants) follow the TEI recommendation for a minimal header[4], supplying only the minimal level of encoding required which results in the following form, discussed in detail in the following subsections:

[4] See [2], *2.6 – Minimal and Recommended Headers*, `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/HD.html#HD7`.

```
<teiHeader>
  <fileDesc>
    <titleStmt>
      <title>Treebank representation of the text</title>
      <respStmt>
        <resp>provided by</resp>
        <name type="webService">ICS PAS Multiservice</name>
      </respStmt>
    </titleStmt>
    <publicationStmt>
      <distributor>ICS PAS Multiservice</distributor>
      <date when="2010-11-17">
        <time when="15:30:00" dur="PT0.02S"/>
      </date>
    </publicationStmt>
    <sourceDesc>
      <p>Text retrieved from <ptr target="http://example.edu/
                                          test.txt"/>.</p>
    </sourceDesc>
  </fileDesc>
</teiHeader>
```

3.1 Package Header

The `title` element in the package header is used to provide some descriptive information on the type of returned content. The responsibility statement (`<respStmt>`) refers to the generator (in our test case, the multiservice, without listing its components used in the annotation process; their names and types[5] are provided in nested headers) assigning it the role of text encoder (of the data analysed by constituent tools, referred to in the corpus substructure).

The publication statement (`<publicationStmt>`) contains information about the process of electronic delivery of the data. `date` and `time` fields are used to represent the moment of issuing the output "corpus" to the end user (in a normalized format, `YYYY-MM-DD` and `HH:MM:SS` respectively) and duration of the packaging (in a formalized W3C format[6]) Annotation generator is listed as the `distributor` of the data. As `address` is intended to be used for postal addresses only, it was omitted in the electronic case (although storing information on URLs of the automated encoder could be helpful).

[5] Note that `type` must be a legal XML name (see `http://www.w3.org/TR/REC-xml/#dt-name`), which results in using e.g. camelCase instead of whitespace in the attribute value.

[6] See `http://www.tei-c.org/release/doc/tei-p5-doc/en/html/ref-data.duration.w3c.html`.

The source description (`<sourceDesc>`) is intended to provide the basic reference to the source of processed input — text retrieved from the Web form, supplied file or external URL, pointed to with `<ptr>` in the latter case.

3.2 Annotation-Level and Variant Annotation Headers

At the annotation level the `<title>` stores information about the name and/or purpose of the layer (e.g. „morphological analysis"); information `<distributor>` is still the annotation generator, also responsible (by means of `<respStmt>`) for TEI P5-encoding of data. `<date>` and `<time>` fields refer to the execution of the processing at the given annotation level.

TEI headers stored inside `<TEI>` elements representing variant annotation for a given level (within the same `<teiCorpus>`) provide information on the tool responsible for the concrete annotation. `<respStmt>` and `<distributor>` fields contain the tool name while `<date>`/`<time>` fields contain the timestamp of the end of the analysis process and the information about its duration.

4 Annotation Components

Annotations are represented as TEI `<text>`s. Formally, the `<TEI>` element of a particular annotation variant contains a header information followed by a single `<text>`/`<body>` with `<p>`aragraphs grouping generic `<seg>`ments corresponding to annotation elements. Following NKJP concept, this structure is preserved and made parallel at all annotation levels (except for segmentation where `<s>`entences can be marked additionally, see 4.2).

To enhance human-readability of the text, segments are preceded with additional XML comments storing fragments of source related to the annotated unit. Linguistic features are preserved using TEI-embedded feature structure formalism.

4.1 Identifiers and Links

Each corpus segment intended to be further referenced is given a unique identifier. To maintain consistency, identifiers take a standardized form of the following:

1. one-letter level symbol:
   – `s` – segmentation,
   – `l` – lemmatization,
   – `m` – morphological analysis,
   – `t` – tagging,
   – `w` – syntactic words,
   – `g` – syntactic groups,

   - `p` – deep parsing,
2. the number of the variant of particular layer (since there may be more than one e.g. tagging layer present, produced by rival taggers),
3. a minus sign,
4. a compound tag index (without intention to be further processed) composed of dot-separated paragraph number, token number, lemma number and MSD number (when respective segments are present).

For overlapping segments (e.g. resulting from different tokenization variants produced by a single annotation tool) additional substructure of the given part of the identifier is expressed with a minus-sign, so that e.g. `t1-seg2.3.4-5-6.7.8` identifier marks the segment representing eighth morphosyntactic interpretation attached to the seventh lemma corresponding to the sixth segment of a fifth variant of a fourth token coming from the third sentence in a second paragraph from the first segmentation-level annotation variant.

References to corresponding elements are stored in `corresp` attributes of `<seg>`ments and take form of XML element identifiers (in contrast to NKJP, no string ranges are used to simplify usage of the format).

4.2 Segmentation

The segmentation model combines both the text structure and paragraph-, sentence- and token-level descriptions into a single layer. This differs from the NKJP approach, but seems to follow the assumption[7] that the data being annotated is never acquired as plain text so mimicking its "original" form is of little value. The segmentation layer can have varied, application-dependent complexity (with just text as paragraph or sentence content when no further annotation is needed – or with the full paragraph-sentence-token set). Storing original text fragments directly in segmentation layer offers major benefits for performance of further processing, based entirely on XML identifiers and never on string ranges. The only drawback is possible duplication of the source text when different variants of segmentation[8] are needed. This seems unusual for paragraph- and sentence-level segmentation and rare for tokenization, but the case of comparing different tokenization variants is difficult even with string-range references to any running text. In most cases the segmentation level would be a single reference layer for further annotation levels.

Paragraphs are adopted from the source text or an artificial single paragraph for shorter or unstructured texts is introduced. To maintain consistency and simplicity, paragraphs are used instead of anonymous blocks which could probably be more appropriate for this purpose, especially in cases when the complete

[7] See discussion in Chapter 4 – *Text Structure* of [5].

[8] This refers to variants produced by different tools and stored in separate variant layers of a segmentation level and not to variants of segmentation suggested by a single tool, which can be easily represented with `<choice>` tags; see tokenization description further in this section.

analyzed text corresponds to a unit lower than sentence[9]. When paragraph-level annotation is sufficient, `<p>` tags can contain plain text.

Sentences are represented with an `<s>` tag nested directly in paragraphs. Similarly to paragraph-level segmentation, sentences can contain token-level annotation or plain text when it is requested to split text into sentences without performing any further analysis. When the text being analyzed does not constitute a complete sentence, but tokenization is needed, sentence tag is omitted.

Individual tokens are represented by `<seg>`ments of `token` type. Variant tokens resulting in overlapping text are stored in `<choice>` elements with variant `<seg>`ment subelements[10]. A token which was attached to the previous token (by means of not being separated with whitespace) is carrying a `subtype` attribute with `attached` value. In the following example it is used in two roles: for punctuation and subsequent (here: last) segment of a compound token.

```
<body>
  <p xml:id="s1-p1">
    <!-- Miałem list. (EN: I had a letter.) -->
    <s xml:id-"s1-s1.1">
      <choice>
        <seg xml:id="s1-seg1.1.1-1" type="token">Miałem</seg>
        <seg xml:id="s1-seg1.1.1-2">
          <seg xml:id="s1-seg1.1.1-2-1" type="token">Miał</seg>
          <seg xml:id="s1-seg1.1.1-2-2" type="token"
               subtype="attached">em</seg>
        </seg>
      </choice>
      <seg xml:id="s1-seg1.1.2" type="token">list</seg>
      <seg xml:id="s1-seg1.1.3" type="token"
           subtype="attached">.</seg>
    </s>
  </p>
</body>
```

4.3 Lemmatization

Segments in lemmatization layer refer to segmentation layer and contain definitions of feature structures which in turn store interpretations as individual features. Value of each feature is a string preserving lemma for a given token; alternative values are grouped in value alternation elements (`<vAlt>`).

[9] The definition of `<ab>` element from the *Elements* chapter of the TEI Guidelines comes with the following explanation: *an anonymous block contains any arbitrary component-level unit of text, acting as an anonymous container for phrase or inter level elements analogous to, but without the semantic baggage of, a paragraph.*

[10] The semantically neutral bracket `<seg>`ment carrying `s1-seg1.1.1-2` identifier (represented in NKJP as `<nkjp:paren>`) became here a fully-fledged content unit (a *meta-token*) which can be referenced in further annotation layers.

```
<body>
  <p xml:id="l1-p1" corresp="s1-p1">
    <!-- ...  -->
    <!-- list (EN: letter or list) -->
    <seg xml:id="l1-seg1.1.2" type="lemma" corresp="s1-seg1.1.2">
      <fs type="lex">
        <f name="base">
          <vAlt>
            <!-- EN: letter -->
            <string xml:id="l1-string1.1.2.1">list</string>
            <!-- EN: list -->
            <string xml:id="l1-string1.1.2.2">lista</string>
          </vAlt>
        </f>
      </fs>
    </seg>
    <!-- ... -->
  </p>
</body>
```

4.4 Morphological Analysis

Segments in morphological layer also refer to segmentation layer with individual features indicating respective lemmata (feature values storing them). Multiple analyses associated with a single lemma are linked by repeating its identifier in the reference attribute.

Morph type feature structure serves as a wrapper to lexical interpretations; each of them contains reference to the base form existing in lemmatization layer (using `fval` pointer), POS tag information and morphosyntactic tag(s). Feature values are represented as symbols rather than strings since they come from a definite set. Such notation can be used for tools with arbitrary tagsets.

```
<body>
  <p xml:id="m1-p1" corresp="s1-p1">
    <!-- ... -->
    <!-- list -->
    <seg xml:id="m1-seg1.1.2" type="morph" corresp="s1-seg1.1.2">
      <fs type="morph">
        <f name="interps">
          <vAlt>
            <fs type="lex">
              <f name="base" fval="l1-string1.1.2.1"/>
              <f name="ctag">
                <symbol value="subst"/>
              </f>
```

```
                <f name="msd">
                  <symbol xml:id="m1-symbol1.1.2.1.1"
                          value="sg:nom:m3"/>
                  <symbol xml:id="m1-symbol1.1.2.1.2"
                          value="sg:acc:m3"/>
                </f>
              </fs>
              <fs type="lex">
                <f name="base" fval="l1-string1.1.2.2"/>
                <f name="ctag">
                  <symbol value="subst"/>
                </f>
                <f name="msd">
                  <symbol xml:id="m1-symbol1.1.2.2.1"
                          value="pl:gen:f"/>
                </f>
              </fs>
            </vAlt>
          </f>
        </fs>
      </seg>
      <!-- ... -->
  </p>
</body>
```

4.5 Tagging

The process of tagging results in selection of a single lemma and a single morphosyntactic description for each token. This produces a new annotation layer referring to morphological level (which, in consequence, results in selection of corresponding lemma and token variant, if alternations are present).

In case of multiple tokenization variants, only one is selected which is reflected in the presence of a single `<seg>`ment for a given fragment of the source text. Similarly, when multiple morphological analyses were present, only one is referred to from respective segment.

In contrast to NKJP where disambiguation information is a complex structure, listing dates and responsibilities, here it is stored in a single feature value since it is assumed that the disambiguation process is performed by a single tool referred to in the header.

```
<body>
  <p xml:id="t1-p1" corresp="s1-p1">
    <!-- ... -->
    <!-- list -->
    <seg xml:id="t1-seg1.1.2" type="tag" corresp="t1-seg1.1.2">
```

```
      <fs type="morph">
        <f name="disamb" fVal="m1-symbol1.1.2.1.1"/>
      </fs>
    </seg>
    <!-- ... -->
  </p>
</body>
```

4.6 Syntactic Words

Syntactic word is a non-empty sequence of tokens and/or constituent syntactic words (which amounts to a non-empty sequence of tokens) which can be referenced from the shallow parsing layer.

Syntactic word layer is intended to help with syntactic parsing; since the process of morphological analysis (and, in turn, tagging) could result in identification of segments shorter than space-to-space words (such as "*biało*": en. *white* and "-" and "*czerwony*": en. *red* for a compound adjective "*biało-czerwony*": en. *white-and-red*), construction of parsing rules would be much more complex than with "traditional" understanding of segments.

```
<body>
  <p xml:id="w1-p1" corresp="s1-p1">
    <!-- ... -->
    <!-- biało-czerwony (EN: white-and-red) -->
    <seg xml:id="w1-seg2.1" type="sword">
      <!-- biało -->
      <ptr type="nonhead" target="s1-seg2.1.1" />
      <!-- - -->
      <ptr type="nonhead" target="s1-seg2.1.2" />
      <!-- czerwony -->
      <ptr type="head" target="s1-seg2.1.3" />
    </seg>
    <!-- ... -->
  </p>
</body>
```

4.7 Shallow Parsing

Syntactic groups represented in the annotation model contain pointers (`<ptr>`s) to immediate constituents of the group — syntactic words specified in the preceding layer or other syntactic groups of the same layer, irrespective of their continuity. `<ptr>` elements may specify the type of the constituency relation (`head` or `nonhead`).

```
<body>
  <p xml:id="g1-p1" corresp="s1-p1">
    <!-- ... -->
    <!-- biało-czerwona flaga (EN: white-and-red flag) -->
    <seg xml:id="g1-seg1" type="sgroup">
      <ptr type="nonhead" target="w1-seg2.1"/>
      <ptr type="head" target="w1-seg2.2"/>
    </seg>
    <!-- ... -->
  </p>
</body>
```

4.8 Deep Parsing

Currently the only available tool responsible for deep parsing of Polish is Marcin Woliński's Świgra [7], a parser using Marek Świdziński's Formal Grammar of Polish [6]. One of its output formats is a shared parse forest, i.e. a collection of parse subtrees stored in a packed graph format, with each unique subtree stored only once. This proprietary representation has been adopted and included into the packaged format by means of standard `<seg>`ments carrying feature structure descriptions:

```
<body>
  <p xml:id="p1-p1" corresp="s1-p1">
    <seg type="forest" xml:id="p1-seg1.1" corresp="s1-seg1.1">
      <seg xml:id="p1-seg1.1.1" type="nonterminal"
           subtype="wypowiedzenie">
        <fs>
          <f name="from"><symbol>0</symbol></f>
          <f name="to">  <symbol>4</symbol></f>
        </fs>
        <seg type="children" subtype="w">
          <ptr type="head" target="p1-seg1.1.2"/>
          <ptr type="nonhead" target="p1-seg1.1.21"/>
        </seg>
      </seg>
      <!-- ... -->
      <seg xml:id="p1-seg1.1.11" type="terminal">
        <fs>
          <f name="from"><symbol>2</symbol></f>
          <f name="to">  <symbol>3</symbol></f>
          <f name="msd-ref" fVal="m1-symbol1.1.2.1.1"/>
        </fs>
      </seg>
      <!-- ... -->
    </seg>
  </p>
</body>
```

5 Final Notes

The model presented above is a small, but useful extension of the representation prepared for NKJP. The idea of merging separate annotation levels into a single file facilitates usage of one the key features of stand-off annotation: interdependence of layers.

In its generality, the format is ready to store output of different variants of the same annotation type produced by competitive tools which fosters linguistic comparisons. Similarly, it is open to enlargement of scope of description (for Polish it is soon planned to be adopted by higher-level annotation tools such as coreference resolvers, named entities recognizers or word-sense disambiguators) and adoption to other languages.

The format has been tested in practice both in the process of representation of linguistic information (by NKJP) and its interchange – by the linguistic Web service for Polish.

References

1. Bański, P., Przepiórkowski, A.: The TEI and the NCP: the model and its application. In: LREC 2010 Workshop on Language Resources: From Storyboard to Sustainability and LR Lifecycle Management. ELRA, Valletta, Malta (2010)
2. Burnard, L., Bauman, S. (eds.): Guidelines for Electronic Text Encoding and Interchange (TEI P5). The TEI Consortium (2007), `http://www.tei-c.org/Guidelines/P5/`
3. ISO:24610-1: Language resource management – feature structures – part 1: Feature structure representation. ISO/DIS 24610-1, 2005-10-20 (2005)
4. Przepiórkowski, A.: TEI P5 as an XML Standard for Treebank Encoding. In: Passarotti, M., Przepiórkowski, A., Raynaud, S., Van Eynde, F. (eds.) Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT 8). pp. 149–160. Milan, Italy (2009)
5. Przepiórkowski, A., Bański, P.: XML Text Interchange Format in the National Corpus of Polish. In: Goźdź-Roszkowski, S. (ed.) The proceedings of Practical Applications in Language and Computers PALC 2009. Peter Lang, Frankfurt am Main (2009), forthcoming
6. Świdziński, M.: Formal grammar of Polish. [In Polish]. Warsaw University Dissertations, Warsaw (1992)
7. Woliński, M.: Computer-aided verification of Świdziński's grammar. PhD. diss., Warsaw (2004), `http://www.ipipan.waw.pl/~wolinski/publ/mw-phd.pdf`, [In Polish]. PhD dissertation. Institute of Computer Science, Polish Academy of Sciences