

CCL format

CCL is a simple format derived from XCES that allows to store:

- division into paragraphs and sentences
- division into tokens and no-space information (XCES-like)
- morphosyntactic annotations (XCES-like)
- chunk-style annotations with possible discontinuities
- syntactic heads of annotations
- properties of tokens and, implicitly, properties of annotations

CCL may be read by corpus2 reader, use ccl format description string when calling `TokenReader.create...` (or using utils like `corpus-get`, `maca-convert`).

CCL/rel is a simple extension that describes inter-annotation relations. The annotations may cross sentence boundaries, and hence, they are read by `corpus2-whole` readers (`corpus2-whole` is a separate but bundled C++ library that loads whole corpora into memory instead of sentence-by-sentence processing; in Python both libraries are wrapped as `corpus2`).

Here we describe the base CCL format, CCL/rel is described at the end of this document.

A DTD which is applicable for both CCL and CCL/rel (but not separate rel file) is available [on the repository](#).

Note that **DTD is quite a weak mechanism** and allows only for superficial validation (e.g. no datatype checks are possible).

Assumptions

We extend the **KIPI/IPIC** dialect of the XCES format. The idea was to get something close, while keeping it also close to the internal data representation. We did not try hard to keep it XCES-compatible as XCES seems no longer supported. Simplicity was more important.

Definitions

Sentence is a sequence of morphosyntactically annotated tokens. Besides that, sentences may be annotated with chunk-style annotations, organised in independent **channels**.

Channel is physically a sequence of numbers (0 or positive integers). The subsequent numbers correspond to tokens in a sentence. Channels are named. Names are unique. One channel is a representation of annotations of the given name/type (e.g. NP). An annotation is represented as a subsequence of the channel consisting of the same non-zero number. E.g. if channel data is 11002200, it corresponds to phrase number 1 (first and second token) and phrase number two (fifth and sixth token). Zeros mean that no annotation of the given name/type crosses this token.

Annotations may be discontinuous, e.g. 0011002211 — here we've got a discontinuous annotation numbered 1. Note that the `corpus2.AnnotatedSentence` contains API that is able to iterate over annotations in channels. The default iteration mode extracts whole annotations, but there is also a mode that extract continuous parts of annotations as separate annotations.

Head is one of annotation's tokens. The intention is that the token should represent whole phrase. Each phrase should have exactly one head. If no head is given, the API assumes the first phrase token is one. If multiple tokens are set as head, the behaviour is undefined.

Token property is a key-value pair, where both key and value are strings. Each token may be assigned arbitrary number of such pairs. The pairs are not ordered.

Annotation property is similar key-value pair, but attached to an annotation.

Header and footer of CCL files

The format is simplified XCES:

```
<?xml version="1.0" encoding="UTF-8"?>
<chunkList>
...
</chunkList>
```

Structure

In XCES the division into sentences and paragraphs was annotated as `<chunk>` elements. We explicitly assume that there is division into paragraphs and then sentences.

```
chunkList -> chunk
chunk -> sentence
```

Sentences (`sentence`) contain tokens described as in XCES — besides allowing extra information that is described in the next section.

Sentences may have ids, e.g. `<sentence id="sent1">`. This is recommended and required for relation description. The value of the id attribute must conform to XML [requirements for xml:id](#), that is the name must start with a letter or underscore and consist of letters, digits, underscores or dot characters.

No nested paragraphs are supported. `chunk` elements may also be assigned ids, e.g. `<chunk id="ch1">` (again this is recommended; also, the `xml:id` restrictions apply).

Additionally, paragraphs may have a type specified, e.g. `<chunk id="ch1", type="p">` may be used to indicate that this is an ordinary paragraph (the values of type attribute are not defined and may be invented and interpreted at user's discretion).

```
<chunk ...>
<sentence>
</sentence>
...
<sentence ...>
</sentence>
```

Annotations in channels

Each token may be added channel description

```
<tok>
<orth>sekta</orth>
<lex disamb="1"><base>sekta</base><ctag>subst:sg:inst:f</ctag></lex>
<ann chan="NP" head="1">1</ann>
<ann chan="AdjP">0</ann>
</tok>
<tok>
<orth>wynaturzona</orth>
```

```

<lex><base>wynaturzyć</base><ctag>ppas:sg:acc:f:perf:aff</ctag></lex>
<lex disamb="1"><base>wynaturzyć</base><ctag>ppas:sg:inst:f:perf:aff</ctag></lex>
<ann chan="NP">1</ann>
<ann chan="AdjP">0</ann>
</tok>
<tok>
<lex disamb="1"><base>seksualnie</base><ctag>adv:pos</ctag></lex>
<ann chan="NP">1</ann>
<ann chan="AdjP">0</ann>
</tok>

```

A token may have a number of `ann` elements. One token may not contain multiple `ann` elements with the same name. The `ann` element describes the state of annotation of the given channel. The state is described by a number for the given token. Number 0 means that the annotation of the given name does not cross the token. Positive number denotes that the token belongs to the annotation of the given number. The numbers in different channels are independently assigned, there is no relation between, say `<ann chan="NP">2</ann>` and `<ann chan="VP">2</ann>`. If two subsequent tokens are assigned different numbers in the same channel, it means they belong to different annotations. E.g. first token with `<ann chan="NP">1</ann>` and `<ann chan="NP">2</ann>`.

The numbers may also be used to express discontinuities, as described in channel definition.

NOTE: each token in one sentence should be assigned the `ann` elements with all the channels employed. That is, if any annotation of name/type X appears in a sentence, all the sentence tokens should have `<ann chan="X">...</ann>` entries.

Annotation heads are marked by `head="1"`.

Token properties

`<tok>` elements may be assigned any number of properties described as below:

```

<tok>
...
<prop key="name1">value1</prop>
<prop key="name2">value2</prop>
</tok>

```

There is a convention to express annotation properties as token properties:

```

<tok>
<orth>Jedz</orth>
<lex disamb="1"><base>jeść</base><ctag>impt:sg:sec:imperf</ctag></lex>
<ann chan="NP">0</ann>
<ann chan="VP" head="1">1</ann>
<ann chan="city_nam">0</ann>
<prop key="VP:type">impt</prop>
<prop key="irrelevant">dummy</prop>
</tok>

```

The idea is to attach token-level properties to tokens being heads of annotations; the keys should be prefixed with `ANN_NAME:.`. In the example above, a `VP` annotation is effectively attached a property `type` of value `impt`.

CCL / rel

Relations may be stored in the same file as CCL data or in a separate file. Usually CCL files are named `ccl-NAME.xml`, `NAME.ccl.xml` or just `NAME.xml`, while relation files are named `rel-NAME.xml` or `NAME.rel.xml`.

In case of relations and CCL data being in the same file, the same structure is kept with additional `relations` element:

```

<?xml version="1.0" encoding="UTF-8"?>
<chunkList>
  <chunk ...>
  </chunk>
  ...
  <relations>
  ...
  </relations>
</chunkList>

```

In case of stand-off relation annotation, the relation files contain rudimentary header and `relations` element:

```

<?xml version="1.0" encoding="UTF-8"?>
<relations>
...
</relations>

```

The content of `relations` is a sequence of `rel` elements, e.g.:

```

<rel name="subj">
  <from chan="chunk_vp" sent="sentence2">1</from>
  <to chan="chunk_np" sent="sentence2">1</to>
</rel>
<rel name="obj">
  <from chan="chunk_vp" sent="sentence2">1</from>
  <to chan="chunk_np" sent="sentence2">2</to>
</rel>

```

Each `rel` element specified a directed relation from the given annotation number in the given channel of the given named sentence into a similarly specified target.

The first example contains two relation instance; first one is a relation of name/type `subj` and holds between annotation of number 1 in a `chunk_vp` channel of `sentence2` and between a `chunk_np` annotation numbered 1 in the same sentence.

Note that the relation annotation requires sentences with ids.