Comparing TCF and LIF

Marc Verhagen

February 12, 2018

1 Introduction

This document has some comparisons of the Text Corpus Format (TCF) used by WebLicht and the LAPPS Interchange Format (LIF) used by the LAPPS Grid. The descriptions here are based on the information in:

- The TCF Format explained.
  http://weblicht.sfs.uni-tuebingen.de/weblichtwiki/index.php/The_TCF_Format
- The TCF schema
  https://github.com/weblicht/tcf-spec/blob/master/src/main/rnc-schema/textcorpus_0_4.rnc
- The specifications of the LAPPS Interchange Format: http://wiki.lappsgrid.org/interchange/
- The LAPPS vocabulary website: http://vocab.lappsgrid.org/

The TCF and LIF formats are relatively similar: they both aim to promote interoperability of NLP web services, they both claim to be compatible to the Linguistic Annotation Format and they both use annotation layers. In this document, the emphasis is on the differences.

2 Top-level representation

The TCF top-level XML is a `D-Spin` tag with in it a `MetaData` and a `TextCorpus` tag. The former contains all metadata including the tool chain used and the latter has all the linguistic data, including the text source and the annotation layers.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="..." type="application/relax-ng-compact-syntax"?>
<D-Spin xmlns="http://www.dspin.de/data" version="0.4">
  <MetaData xmlns="http://www.dspin.de/data/metadata">
    ...
  </MetaData>
  <TextCorpus xmlns="http://www.dspin.de/data/textcorpus" lang="de">
    ...
  </TextCorpus>
</D-Spin>
```

The LIF format is based on JSON-LD and it is embedded inside the JSON-LD that is handed over between web services. The input to a web service has three elements: a discriminator telling the service what kind of payload to expect, an optional dictionary of processing parameters (like what model to use) and the payload. In the example below, the payload is a LIF structure, which at its top level has three attributes: `text`, `metadata` and `views` (the LAPPS name for annotation layers, see below for differences between TCF annotation layers and views).

```
{
   "discriminator": "http://vocab.lappsgrid.org/ns/media/jsonld#lif",
   "parameters": { }
   "payload": {
      "text": { ... },
      "metadata": { }
      "views": [ { ... }, { ... }, ... ]
   }
}
```

It should be noted that it is possible for LAPPS Services to produce or accept a payload that is not LIF and therefor it is possible that the payload is a TCF representation. This would however make it the responsibility of the service to understand TCF, and most services do not do that. It is an option though to have a converter service that provides automatic mappings from TCF to LIF, similar to the converter services that map the GATE format to LIF and vice versa.

3 Meta data

In TCF, the `MetaData` section is used by WebLicht web applications to store metadata about resources and tools that were used to generate and process the document.

```
<MetaData xmlns="http://www.dspin.de/data/metadata">
  <source/>
  <Services>
    <CMD xmlns:xsi="..." xmlns="..." xsi:schemaLocation="...">
      <Header/>
      <Resources>
        <ResourceProxyList/>
        <JournalFileProxyList/>
        <ResourceRelationList/>
      </Resources>
      <Components>...</Components>
    </CMD>
  </Services>
</MetaData>
```

The `Components` tag has some general information followed by a list of `ToolInChain` tags where each tool has an identifier and an optional parameter:

```
<Components>
  <WebServiceToolChain>
    <GeneralInfo>
      <ResourceName>Custom chain</ResourceName>
      <ResourceClass>Toolchain</ResourceClass>
      <Descriptions><Description/></Descriptions>
    </GeneralInfo>
    <Toolchain>
      <ToolInChain>
         <PID>11858/00-1778-0000-0004-BA56-7</PID>
         <Parameter name="version" value="0.4"/>
      </ToolInChain>
      <ToolInChain>
         <PID>11858/00-1778-0000-0004-BA7B-4</PID>
      </ToolInChain>
    </Toolchain>
  </WebServiceToolChain>
</Components>
```

LIF has currently no metadata at the top level. Instead, it has metadata information within each view about what tool created the content of the view. The views are interpreted as a list and each new component adds to the end or inserts information into an existing view. As with TCF, the processing tools add the metadata. The metadata in a view look as follows:

```
"views": [
  { "metadata": {
      "contains":
        "http://vocab.lappsgrid.org/DependencyStructure": {
          "producer": "lappsgrid.stanford.corenlp.DependencyParser",
          "type":"dependency-parser:stanford" },
        "http://vocab.lappsgrid.org/Dependency" {
          "producer": "lappsgrid.stanford.corenlp.DependencyParser",
          "type":"dependency-parser:stanford"},
        "http://vocab.lappsgrid.org/Token": {
          "producer":"lappsgrid.stanford.corenlp.DependencyParser",
          "type":"tokenizer:stanford" }}},
    "annotations": [ ... ]
  }
]
```

The metadata spell out what the layer contains, for example the above LIF fragment says that the view contains annotations of types `DependencyStructure`, `Dependency` and `Token`, and all elements of the `annotations` list have to be of those types. And for each type it says what created the annotation (dependency parser) and some type information.

There are several differences compared to the TCF tool metadata:

1. In LIF, as opposed to TCF, there is no tool identifier. The closest is the value of the producer attribute, which in a real-life LIF structure would be longer, referring to a Java class and for example including a version number. However, even this long version is not an identifier. In fact, tools do not know their own identity and simply add to the metadata what they know

about themselves.

2. In LIF, a LAPPS service says exactly what it creates and the tool metadata and the annotation the tool produced are in the same sub structure while in TCF this is not the case. In fact, in TCF it is impossible to see what tool created what data[1]. I do not know this for sure, but I presume that the identifier allows you to look up some information about the tool, which could be used if a TCF file is translated into a LIF file, without that we would need to generate the view metadata from what is found in the TCF layer itself.

None of these issues seem serious. If TCF does require unique identifiers then we may have to make sure (semi-automatically) that the value of producer is a unique identifier.

Question: does TCF require unique identifiers for tools?

4 Representing the text and anchoring to primary data

In TCF, the `TextCorpus` section contains the text, a specification of the language of the data (a two-letter ISO 639-1 code) and the linguistic data itself. The source text can be in two fields, the `text` field and the `textSource` field. The latter is optional and is used for embedding the original source, which can for example be a TEI source.

TCF annotations are not grounded to character offsets in the primary source text, a departure from the LAF standard. However, TCF regards the `tokens` layer as the central layer to which all other annotations refer (similar to the base segmentation in LAF, except that the token layer does not refer to character offsets). In the words of the TCF document:

> *TCF stores linguistic annotations inside annotation layers which are organized in a stand-off manner. From an organizational point of view, the token layer can be seen as the central, atomic element to which other annotation layers refer. For example, the part-of-speech annotations refer to the token IDs in the token annotation layer via the attribute tokenIDs.*

Most annotations do indeed refer to token identifiers, the only exceptions I see are in the `synonomy` and `parsing` layers. The former refers to lemma identifiers (which themselves refer to token identifiers), the latter uses embedding to encode phrase structure (but leaf nodes do refer to tokens).

Below is a fragment of a TCF representation that illustrates how text is represented and that shows the primacy of the `tokens` layer:

[1]Technically, this is the case for LIF as well since you could have a layer with two tools producing the same type of annotations, in practice this does not occur however.

```
<TextCorpus xmlns="http://www.dspin.de/data/textcorpus" lang="de">
  <text>Karin fliegt nach New York. Sie will dort Urlaub machen.</text>
  <tokens>
    <token ID="t_0">Karin</token>
    <token ID="t_1">fliegt</token>
  </tokens>
  <POStags tagset="stts">
    <tag ID="pt_0" tokenIDs="t_0">NE</tag>
    <tag ID="pt_1" tokenIDs="t_1">VVFIN</tag>
  </POStags>
  <textSource type="tei/tokenized"> ... </textSource>
</TextCorpus>
```

In LIF, all annotations refer to either character offsets or to other annotations. If an annotation refers to an other annotation, then there is presumed to be a chain of annotations eventually leading to character offsets. So annotations are anchored in character offsets in the primary source, which is always included in the `text` tag.

```
{
   "metadata": { },
   "text": {
      "@value": "Karen flies to Berlin. She needs a break.\n",
      "@language": "en"
    },
   "views": [
     {
        "metadata": {
           "contains": { "http://vocab.lappsgrid.org/Token": ... }},
        "annotations": [
           { "id": "tk_0_0", "start": 0, "end": 5, "@type": "Token",
             "features": { "pos": "NN", "word": "Karen" } },
           { "id": "tk_0_1", "start": 6, "end": 11, "@type": "Token",
             "features": { "pos": ":", "word": "flies" } }
      }
   ]
}
```

Given the absence of an explicit link from annotations to character offsets, the mapping from TCF to LIF can be a bit tricky and will involve some kind of alignment. In the worst case, the LIF text will be computed directly from the TCF tokens layer. There is no problem in the opposite direction.

5 Annotation layers and annotation categories

TCF has a fixed number of annotation layers, currently there are 19 layers (this is including the `text` and `textSource` layers which in TCF are still called layers) as illustrated in Table 1.

Each layer has a fixed structure and this structure is different depending on what layer you look at. The `tokens` and `POStags` layers shown earlier are flat lists of simple tags, as are many other layers

text	parsing	synonymy	textstructure
tokens	depparsing	matches	orthography
sentences	morphology	WordSplittings	wsd
lemmas	namedEntities	geo	textSource
POStags	references	Phonetics	

Table 1: The TCF Layers

(`lemmas`, `sentences`, `namedEntities`, `synonymy`, `Phonetics`, `text structure`, `orthography` and `wsd`). Other layers (`morphology`, `depparsing`, `references`, `matches`, `WordSplittings` and `geo`) have slightly more structure, for example:

```
<morphology>
  <analysis tokenIDs="t_0">
    <tag>
      <fs>
        <f name="cat">proper name</f>
        <f name="gender">feminine</f>
        <f name="case">nominative</f>
        <f name="number">singular</f>
      </fs>
    </tag>
    <segmentation>
      <segment cat="proper name" type="stem">Karin</segment>
    </segmentation>
  </analysis>
</morphology>
```

The `parsing` layer is special in that it uses embedding to encode parenthood relations in a tree:

```
<parse>
  <constituent cat="EN" ID="c_10">
    <constituent cat="NX" ID="c_9">
      <constituent cat="NE" ID="c_7" tokenIDs="t_3"/>
        <constituent cat="NE" ID="c_8" tokenIDs="t_4"/>
      </constituent>
    </constituent>
  </constituent>
</parse>
```

In LIF, the number of layers, called views, is not limited. Each service can add as many views as it wants and can put anything it likes in the view, as long as it tells us what it is doing in the metadata. A service can also add to an existing view, again as long as it updates the metadata appropriately. As a result, information from a LIF view will possibly need to be distributed over several TCF annotation layers.

All layers have the same structure in that the `annotations` list is a list of elements that all follow the same template, which is an annotation object with a type, an identifier, beginning and end character offsets (or a reference to other annotations) and a feature dictionary, for example:

```
{
  "@type": "http://vocab.lappsgrid.org/Token",
  "id":" tk_0_0",
  "start": 3,
  "end": 13,
  "features": {
    "pos": "NN",
    "word": "Message-ID" }
}
```

Those annotations that refer to other annotations will have `targets` instead of `start` and `end`. The `@type` attribute refers to a category in WSEV, the LAPPS vocabulary. The part of the WSEV hierarchy that contains the annotation categories (plus attributes) that can occur in the annotations list of a view are printed below, refer to http://vocab.lappsgrid.org/ for the full vocabulary. Note that the stop-level attributes, with the exception of `features` and many of the attributes in the features dictionary occur in the vocabulary. The `features` dictionary has an unspecified number of key-value pairs. The values are typically strings of lists of strings, further structure is possible not is not recommended. Attributes in this dictionary are not required to be in the vocabulary, but it is assumed that they are defined somewhere.

```
Annotation {id}
    Region {targets, start, end}
        Paragraph
        Sentence {sentenceType}
        NounChunk
        NamedEntity {category, dateType. locType, orgType, gender}
        Token {pos, lemma, tokenType, orth, length, word}
        Markable
    Relation {label}
        GenericRelation {relation, arguments}
        SemanticRole {head, argument}
        Constituent {parent, children}
        Dependency {governor, dependent}
    Coreference {mentions, representative}
    PhraseStructure {constituents, root}
    DependencyStructure {dependencyType, dependencies}
```

To recap, these are the two main differences between TCF and LIF annotation layers:

1. TCF has a hard-coded set of annotation layers, and each layer can be structured differently. In LIF the set of views is determined by the tools and what categories you have in the vocabulary. All views are structured in the same way.

2. LIF has flat annotation layers so constituency is not expressed by embeddings (instead it uses a `children` attribute in the `features` dictionary on the parent, with as the value a list of identifiers).

Many of the differences are due to the kind of processing steps that TCF and LIF have till now supported.

6 TCF layers and LIF views

In general, both formats appear expressive enough to represent layers or views from the other side. TCF will need to define and add extra layers in its metadata to account for all possible LIF views and on the LAPPS end WSEV will need to define new categories (note that while LIF is the vehicle for the information, the changes needed are all in the vocabulary). But the following caveats apply:

1. TCF has a mechanism to deal with character offsets, but in general it is not used. On the other hand, LIF requires character offsets in layers so they will need to be generated. We are ignoring the issue of character offsets in this section.
2. WebLicht and LAPPS services use different metadata and TCF and LIF need extra structure to accommodate the metadata from the other side. The question is to what extent this is needed. For example, LAPPS tools may care about the typing information in the view's metadata but do not, at this point, care at all about identifiers of tools.
3. LIF has no natural way to represent the `textSource` layer. Note that later we are proposing to use (or abuse) this TCF layer to store LIF.
4. TCF has no natural way to represent duplicate information. LIF can host many views with token information, in TCF this is not possible without structural or ad hoc changes to the TCF schema.