Representing Multimodal Linguistics Annotated data

Brigitte Bigi, Tatsuya Watanabe, Laurent Prévot

Laboratoire Parole et Langage, CNRS, Aix-Marseille Université, Ortolang
5, avenue Pasteur, 13100 Aix-en-Provence France
brigitte.bigi@lpl-aix.fr, tatsuya.watanabe@atilf.fr, laurent.prevot@lpl-aix.fr

Abstract

The question of interoperability for linguistic annotated resources requires to cover different aspects. First, it requires a representation framework making it possible to compare, and potentially merge, different annotation schema. In this paper, a general description level representing the multimodal linguistic annotations is proposed. It focuses on time and data content representation: This paper reconsiders and enhances the current and generalized representation of annotations. An XML schema of such annotations is proposed. A Python API is also proposed. This framework is implemented in a multi-platform software and distributed under the terms of the GNU Public License.

Keywords: multimodality; annotation; framework

1. Introduction

Multimodal text analysis has become a crucial part of research, teaching and practice for a wide range of academic and practical disciplines. Multimodal annotation is facing the necessity of encoding many different information types, from different domains, with different levels of granularity (Jewitt, 2009). In the Linguistics field, multimodal annotations contain information ranging from general linguistic to domain specific information. Some are annotated with automatic tools, and some are manually annotated. This implies technical and methodological levels to produce high quality multimodal annotations, as proposed in (Blache et al., 2010).

Linguistics annotation, especially when dealing with multiple domains, makes use of different tools within a given project. The annotation itself should be done with the most ergonomic and convenient tool for the given annotation task. Then it is up to the person to develop the exploitation tool to create "views" in the very rich and layered object that corresponds to the whole annotation set. This is to make sure that analysts, annotators and end-users never face the intrinsic complexity of a rich multi-level annotation framework. Acoustic analysis, phonetics and prosody annotation are very often done using Praat (Boersma and Weenink, 2011). Gesture annotations, and more generally multimodal studies, now rely on higher level systems such as Anvil (Kipp., 2011) or Elan (Nijmegen: Max Planck Institute for Psycholinguistics, 2011) (see (Rohlfing et al., 2006) for a comparison of multimodal annotation tools). As for orthographic transcriptions, a great number of tools can be used. None of the annotations tools are directly *interoperable*, each using a native format, some of them on top of XML, some others developing an ad hoc markup language. The heterogeneity of such annotations has been recognized as a key problem limiting the interoperability and re-usability of NLP tools and linguistic data collections.

Manipulating linguistic annotated resources requires to be independent from the coding format. This implies two kinds of pre-requisites:

- to specify and organize the information to be encoded independently from the constraints or restriction of the format (or the annotation tool);
- to encode the information into an exchange format, readable whatever the edition or annotation system.

But schemas and tools multiplication renders the data manipulation very problematic. On the other hand, despite standardization efforts it is unlikely that a unique generic schema or tool emerge.

This paper addresses the problem of *representing these multimodal annotated speech corpora*. Metadata (speaker information, recording condition, etc) are not addressed. This paper focuses on the annotation representation itself: time representation, content representation and data organization.

2. Related works

Many tools and frameworks are available for handling rich media data. The practice of taking advantage of such a rich tool offer will not change. What can change however is the habit of developing a new language for each new project. Interchange format such as AIF (Bird et al., 2000) constitute an important step for answering this need of common frameworks. Although necessary for bringing some tool interoperability, such interchange formats remained relatively poor with regard to their descriptive power. Indeed the development of these formats was done according to a "tool perspective" as illustrated by the importance of the notions of tiers or tracks.

Nowadays, most of linguistic annotation projects require a rich framework in order to deal with all the information levels involved. For example, dialogue act annotation involve multi-dimensional communicative functions (Bunt, 2009) that are organized in a complex taxonomy. Another example concerns discourse units that can be discontinuous which prevent them from treating them a simple sequence of units on a tier. Instead of being answered by richer model, extensibility and flexibility are gained by tricks and ad-hoc solutions to overcome the basic nature of the original model.

The other way toward interoperability is the standardization of annotation frameworks. The ISO TC37/4[1] initia-

[1]http://www.tc37sc4.org/

tive is crucial for making converge annotation frameworks. However, ISO standards usually provide precise guidelines for developing interoperable systems but do not resolve the technical issues of this interoperability.
Web semantics languages such as RDF (Resource Description Framework) and OWL (Ontology Web Language) also constitute an important path to explore for practical interoperability. *Unique Resource Identifier* allows to uniquely identified and facilitate sharing schema and resources. Finally, OWL has been developed for providing a standard semantics to the ingredients present in all these resources. These features should allow a real re-usability and interoperability. Moreover the mapping between those well-defined schema should be easier to perform thanks to existing tools and methodologies in the semantic community.
In spite of their benefits, those semantic web languages do not impose themselves easily as standard languages in language annotation projects. This is mainly due to their verbosity and their complexity as illustrated in the short example below. This has two important consequences: (i) they need sophisticated tools to be easily handled while annotation model development early stages is usually not performed with these tools; (ii) it drastically increase data size (although this should not be an issue for other corpora than massive web corpora and some more compact syntax[2] has been proposed).

```
<owl:Class rdf:about="#syllable">
<rdfs:subClassOf rdf:resource="#object"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasPart"/>
<owl:someValuesFrom
       rdf:resource="#constituent"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>

<owl:DatatypeProperty
      rdf:about="#proeminent">
<rdfs:domain rdf:resource="#syllable"/>
<rdfs:range rdf:resource="&xsd;boolean"/>
</owl:DatatypeProperty>
```

One of the more complete usage case of these languages for the description of multi-modal interaction is illustrated by Nomos (anNOtation of Media with Ontological Structure) (Gruenstein et al., 2008) developed within the CALO project. Nomos takes advantage of multi-modal discourse ontology (MMDO) that includes discourse-related concepts, an ontology of linguistics (GOLD, General Ontology for Linguistic Description) (Farrar and Langendoen, 2003), as well as separate ontological modules describing the other physical aspects of interaction. Unfortunately, the richness of the underlying model resulted in a very complex framework.
More globally, it seems like simple frameworks are too basic for describing the range of phenomena we are dealing with. And complete "ontological" frameworks result in tools and annotated data too complex to be handled conveniently.

3. Linguistics: Annotated data

The annotation of audio-video corpora is concerned by many Linguistics sub-fields as Phonetics, Prosody, Gestures or Discourse. Consequently, organizing such annotations in a hierarchy or simply performing an inventory of all possible annotations is impractical. It ranges from utterances, orthographic transcription, prosodic cues, phonetic segmentation, to gestures and nowadays motion-capture or more physiological data.
In annotation tools, annotated data are mainly represented in the form of "tiers" or "tracks" of annotations. Of course, this is not just an artifact of such tools, and clearly, many annotators are using such representation during the annotation procedure. In most tools, tiers are series of intervals defined by:

- a time point to represent the beginning of the interval;
- a time point to represent the end of the interval;
- a label to represent the annotation itself.

In Praat, tiers can be represented by a time point and a label (such tiers are named PointTiers and IntervalTiers).
Of course, depending on the annotation tool, the internal data representation and the file formats are different. For example, in Elan, unlabeled intervals are not represented nor saved. On the contrary, in Praat, tiers are made of a succession of consecutive intervals (labeled or un-labeled). Another difference among tools concerns relations between tiers. In Praat for example, no relation can be defined. On the contrary, in Anvil, hierarchies can be established between tiers, by mean of an XML schema describing all annotations and their structures. This XML schema is performed by annotators themselves, before the annotation process. It is then assumed that the user is familiar with XML. Elan proposes an intermediate solution with possibility to connect tiers by a relation named "subdivision".
Our proposal of data representation is based on the common set of information all tool are currently sharing: Tiers. Each tier includes a set of annotations. As usual, an annotation is represented by an interval (or a point) and a label. Our proposal focus on the way to represent such information: time and label. Therefore, this paper reconsiders and enhances the current and generalized representation of annotations.

4. Representation of annotations

4.1. Overview

Time in the various file formats is described in different ways. In the TEI format, all temporal points are grouped in a timeline object and an annotated element can refer to a point using his id. Some other data formats do not use a specific timeline object and directly put the time value inside an element using 'start' and 'end' attributes.
Tools are using real numbers to represent time. They are described thank to a timeline: each number corresponds to one point on the line, and each point determines one real number. From the "computer science" point of view, *real*

[2]See http://www.w3.org/DesignIssues/Notation3.html for this last point on RDF more economic syntax.

numbers are approximated and depend on the number of coding digit. Moreover, the time value is expressed with various time units: nanoseconds is the unit used by some tools, milliseconds by some others, and seconds are mainly used (seconds are represented from 4 to 10 digits). These approximation can cause troubles in some cases, as for example by comparing numbers:

```
>>> a = 4.4 - 4.0 - 0.4
>>> b = 4.4 - (4.0 + 0.4)
>>> print a, b
3.33066907388e-16 0.0
>>> print (a==b)
False
>>> print  (round(a,4)==round(b,4))
True
```

In this case $a == b$ should be true. Consequently, a the framework representing data, as time in our example, and using *real* numbers must take into account such phenomena.
Furthermore, annotated data are often created or modified manually. Then, the precision of the time value depends on various factors, as:

- the human decision (aims of the work, time dedicated to annotate, etc);
- with an audio or a video media, the annotation itself suffers of various precision: it can be very precise (about 1ms) while annotating Tones in the prosody domain; and for the gestures annotation, the precision of a time value can't be less than the duration of one picture in a video (often 40ms).
- the graphical user interface of the tool: in the best case, the precision of the "time value" corresponds to 1 pixel. This means that the time value is not a real but a duration, which depends on the zoom level.

4.2. Time points

Our proposal is to define a *TimePoint X as an imprecise value*. It is possible to characterize a point in a space immediately allowing its vagueness by choosing the representation space Midpoint / Radius. A TimePoint is modeled as follows:

$$X \in \mathbb{R}_+, X = (M_X, R_X) \tag{1}$$

where M_X is the midpoint (center) of the value and R_X is the radius (as proposed in (D'Urso and De Giovanni, 2011)) representing the uncertainty of X (see Figure 1). It is to be noted that $M_X - R_X \geq 0$.

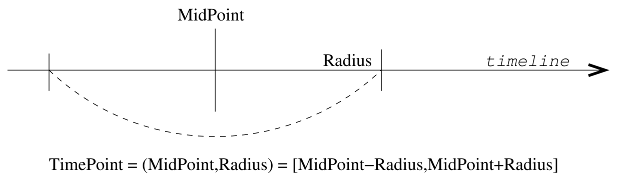
MidPoint
Radius
timeline
TimePoint = (MidPoint,Radius) = [MidPoint−Radius,MidPoint+Radius]

Figure 1: TimePoint representation

A TimePoint X can be represented in two distinct ways, either in common space $X = [x^-, x^+]$ or in the Midpoint/Radius space as $X = (M_X, R_X)$. In this case, the Midpoint represents the middle of the interval and is given by:

$$M_X = (x^- + x^+)/2$$

and the Radius or half-width is the vagueness of the interval a, which becomes directly accessible:

$$R_X = (x^+ - x^-)/2$$

The transition from one representation to the other is achieved thanks to the previous equations, or by using the following: $x^- = M_X - R_X$ and $x^+ = M_X + R_X$.
The equality and inequalities between 2 TimePoint instances X and Y are then defined as:

$$\begin{aligned} X = Y &\Leftrightarrow |M_X - M_Y| \leq R_X + R_Y \\ X < Y &\Leftrightarrow \neg(X = Y) \wedge (M_X < M_Y) \\ X > Y &\Leftrightarrow \neg(X = Y) \wedge (M_X > M_Y) \end{aligned} \tag{2}$$

4.3. Time intervals

A TimeInterval is defined as a temporal interval-valued datum:

$$X = [X^-, X^+] \tag{3}$$

where X^- and X^+ denote, respectively, the lower and upper TimePoint bounds of the interval as we proposed in equation 1.

4.4. Labels

Each annotation holds at least one label, mainly represented in the form of a string, freely written by the annotator or selected from a list of categories. A general framework, aiming at representing any kind of linguistic annotations, have to allow to assign a score to each label, and to allow a multiple label selection.

5. XML representation

XML has become a *de facto* standard for the representation of annotated corpus resources (Ide et al., 2000). It is also a widely established practice to produce a particular type of annotation with a special-purpose tool that outputs XML or maps its output onto XML. Moreover, any markup or encoding schema will have to be robust enough to handle existing data. As the framework proposed in this paper calls into consideration the common representation of time and labels, none of the existing annotation schema allows to encode such information; and unfortunately it was not relevant to adapt one. It was then appropriate to develop a new schema, that we name XRA.
The physical level of representation of XRA obviously makes use of XML, XML-related standards and stand-off annotation. The format specification as well as the texts encoded in XRA are "reusable, i.e., potentially usable in more than one research project and by more than one research team, and extensible, i.e., capable of further enhancement" (Ide and Brew, 2000).

5.1. Main structure

An XRA document is linked to a set of recordings of a corpus. Each document holds a set of tiers, which are directly linked to the primary media. Tiers are used to distribute the

annotations of a recording. Each tier consists of a set of independent annotations, that all pertain to one or more Time instances. The definition of a tier proposed in (Brugman, 2003) is then largely recovered: “a tier is a group of annotations that all describe the same type of phenomenon, that all share the same metadata attribute values and that are all subject to the same constraints on annotation structures, on annotation content and on time alignment characteristics.”. Then, an annotation is made of: a time reference, and (optionally) a label.
As the whole description of the XRA format is very long, only the annotation-related part is detailed.

```
<xsd:complexType name="annotationType">
    <xsd:sequence>
        <xsd:element name="Time" type="
            annotationTime" minOccurs="1"
            maxOccurs="1" />
        <xsd:element name="Label" type="
            annotationText" minOccurs="0"
            maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
```

The proposed Annotation element contains a Time element, which can be of different types: TimePoint or TimeInterval.

```
<xsd:complexType name="annotationTime">
    <xsd:choice>
        <xsd:element name="Point" type="
            timePoint"   />
        <xsd:element name="Interval" type="
            timeInterval"/>
    </xsd:choice>
</xsd:complexType>


<xsd:complexType name="timePoint">
    <xsd:attribute name="midpoint" type="
        decType" use="required"/>
    <xsd:attribute name="radius"   type="
        decType" use="required"/>
</xsd:complexType>

<xsd:complexType name="timeInterval">
    <xsd:all>
        <xsd:element name="Begin" type="
            timePoint" minOccurs="1"
            maxOccurs="1" />
        <xsd:element name="End"   type="
            timePoint" minOccurs="1"
            maxOccurs="1" />
    </xsd:all>
</xsd:complexType>
```

Annotation also contains a Label element which is a complex object, made of a list of couples text/score. Within the XRA-annotated XML files, the annotations are ordered linearly, but they may overlap in time.

```
<xsd:complexType name="annotationText">
    <xsd:sequence>
        <xsd:element name="Text" type="
            annotationTextValue" minOccurs=
            "0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="scoretype" type="
        stringScoreType" use="optional"
        default="proba" />
    <xsd:attribute name="scoremode" type="
        stringSort"      use="optional"
        default="max" />
</xsd:complexType>

<xsd:complexType name="annotationTextValue"
    >
    <xsd:simpleContent>
    <xsd:extension base="xsd:string">
        <xsd:attribute name="score" type="
            xsd:double" use="required"/>
    </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
```

Due to an intuitive naming convention XRA documents are human readable as far as possible within the limits of XML.

5.2. Inter-tier relations

As we already said, it is not possible to decide over an exhaustive list of annotation levels for all annotation projects. Each project and each annotator as its own habits and needs. It is not possible to fix a set of ”normative” tags such as $< u >$ is an utterance and $< w >$ is a word, and $< t >$ is a tone, and $< p >$ is a phoneme and $< gt >$ is a gesture type, and so on... and then to create a static normative hierarchy between them, at least with the purpose to create a generic framework.
The genericity and flexibility of “tiers” is appropriate to represent any multimodal annotated data because it simply maps the annotations on the timeline. Then, dependencies between such tiers must be represented and we propose to extend the sub-division concept, proposed in Elan.
These inter-tier relations are managed by establishing alignment or constituency links between 2 tiers:

- alignment: annotations of a tier A have only Time instances included in those of annotations of tier B;
- constituency: annotations of a tier A have only Time instances included in those of annotations of tier B and labels of B are made of those of A.

Example of alignment: Phonemes is the reference tier and Tokens are alignment tier:

```
    Phonemes:    | l  | S | a | e | l | a |
    Tokens:      | le | chat  |est|  la   |
```

Example of constituency: Phonemes is the reference tier and Syllables are made of Phonemes:

```
    Phonemes:    | l | S | a | e | l | a |
    Syllabes:    |           |   |       |
```

There are several advantages to represent data with this solution. Firstly, the relations can easily be obtained by an automatic analysis of time values, even while importing flat-data. Secondly, an existing tree-structure can be mapped on it and vice-versa.
The corresponding XML representation is as follow:

```
<xsd:complexType name="constraintType">
    <xsd:sequence>
        <xsd:element name="SubDivision"
            minOccurs="0" maxOccurs="
            unbounded" />
    </xsd:sequence>
    <xsd:attribute name="link"      type="
        stringTierRelation" />
    <xsd:attribute name="reftier"   type="
        xsd:IDREF" use="required" />
    <xsd:attribute name="subtier"   type="
        xsd:IDREF" use="required" />
</xsd:complexType>

<xsd:simpleType name="stringTierRelation">
  <xsd:restriction base="xsd:string">
      <xsd:pattern value="alignment|
          constituency"/>
  </xsd:restriction>
</xsd:simpleType>
```

6. Implementation in an API

6.1. Description

Given the current use of large corpora in NLP work, automatic conversion into the corpus representation is desired, as well as automatic annotation and exploitation. Several tools were created in the scope of this work, to support resource conversion to .xra and to enable the integration of annotation/analysis tools.
An Application Programming Interface (API) is proposed in the programming language Python. We implement behaviors in abstract classes (Figure 2) and each class is documented using docstrings. The current version allows to import data from Praat, Elan, Transcriber or from CSV files. Moreover, the XRA-schema is freely available in the package.
This API is already used in the automatic annotation tool SPPAS (Bigi and Hirst, 2012; Bigi, 2012) and is distributed under the terms of the GNU Public License. Among other functions, it proposes an automatic speech segmentation at the phone and token levels for French, English, Spanish, Italian, Chinese, Taiwanese and Japanese.

6.2. Main usage

Open/Read a file of any format (TextGrid, Elan, Transcriber, CSV) and load in a Transcription() instance:

```
trs = annotationdata.io.read(filename)
```

Save/Write a Transcription instance in a file of any format:

```
annotationdata.io.write(filename, trs)
```

Print all tiers of a Transcription() on the standard output:

```
for tier in trs:
    print tier
```

Print all annotations (intervals and/or points) of a tier on the standard output:

```
for ann in tier:
    print ann
```

There are 2 solutions to get a tier in a Transcription() instance. The user can either get it from its index by using `trs[index]` or find it from its name as:

```
tier = trs.Find(name, case_sensitive=True)
```

Create an annotation made of an interval and a label, and add it into a tier:

```
p1 = TimePoint(1.5, radius=0.01)
p2 = TimePoint(1.6, radius=0.01)
t = TimeInterval(p1,p2)
l = Label("foo")
ann = Annotation(t,l)
tier.Add(ann)
```

Removing an annotation of a tier:

```
tier.Pop(index)
```

Get an annotation of a tier from its index:

```
ann = tier[index]
```

Getting all annotations between 2 time values:

```
annlist = tier.Find(from_time, to_time, \
                    overlaps=True)
```

Get the first occurrence of a pattern in a tier:

```
annfirstindex = tier.Search([pattern], \
  function='exact', pos=0, forward=True, \
  reverse=False)
```

Then, the next occurrence of such pattern can be retrieved by setting the `pos` argument to annfirstindex.
Get the annotation contents:

```
b = ann.BeginValue
e = ann.EndValue
t = ann.TextValue
```

Change the radius value of all points of a tier:

```
tier.SetRadius(0.005)
```

Change the value of a point:

```
ann.BeginValue = 0.423
```

Add a sub-division link between two tiers:

```
trs.AddSubdivision(reftier,\
          alignedtier,type="alignment")
trs.AddSubdivision(reftier, \
          alignedtier,type="constituency")
```

Remove a sub-division link between two tiers:

```
trs.RemoveSubdivision(reftier,alignedtier)
```

7. Conclusion

The development of an expressive and generic framework for linguistic annotations is very beneficial. This paper particularly focused on the problem of representing annotations, and an annotation representation was proposed. The paper reconsidered and enhanced the current and generalized representation of speech annotations. We proposed a framework that can capture the indeterminacy of annotations.
We are currently working on extending this work to the construction of a software for the creation, manipulation, and analysis of annotated corpora (Figure 3).

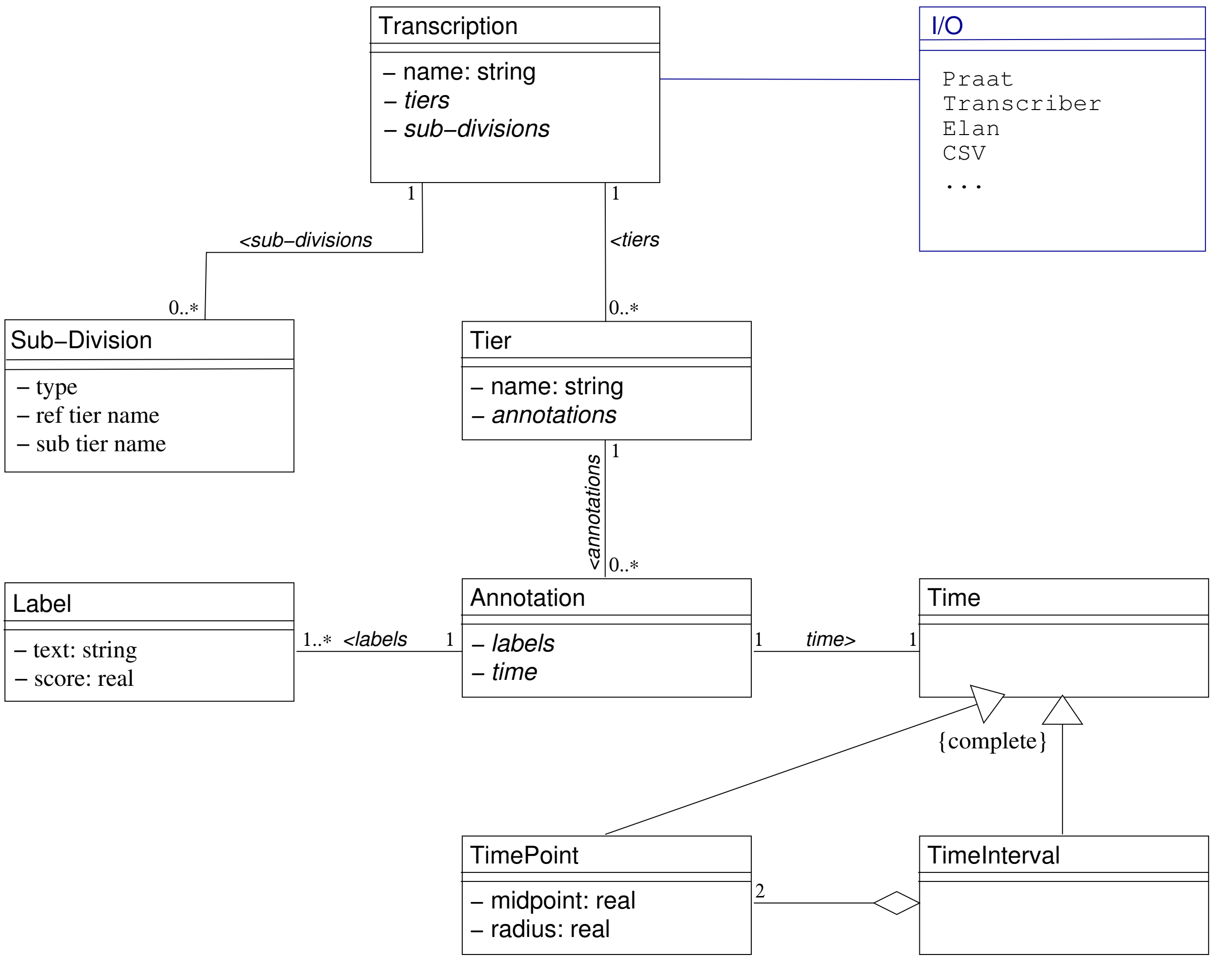

Figure 2: API class diagram

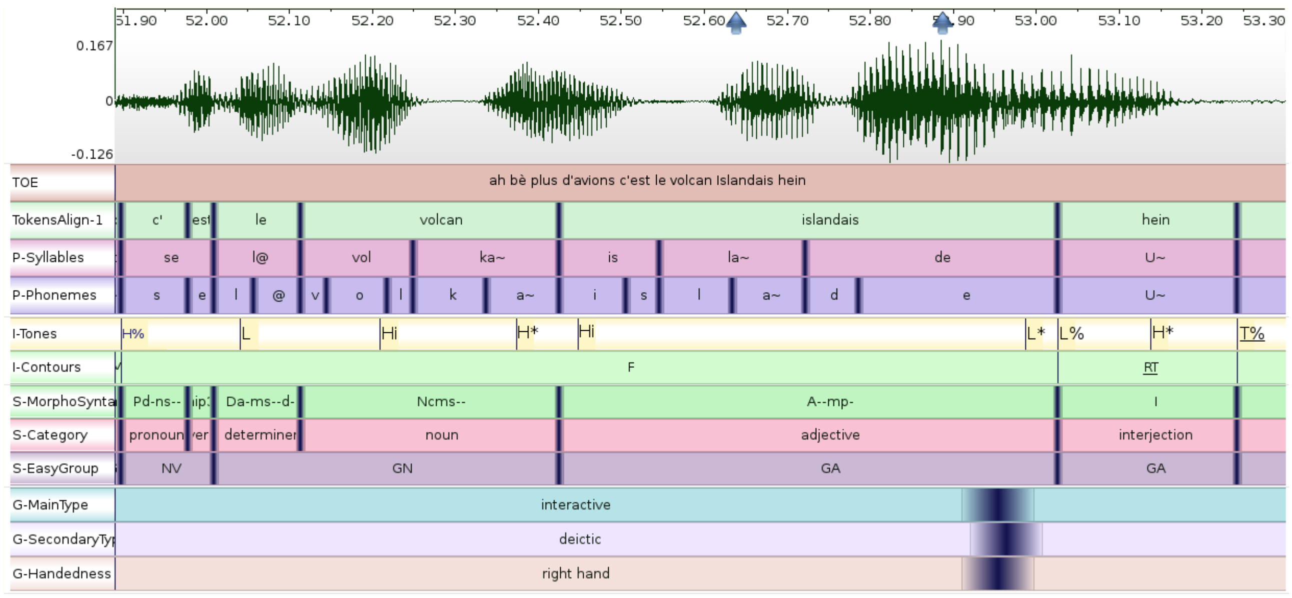

Figure 3: Example of multimodal annotated data, imported from 3 different annotation tools. Each *TimePoint* is represented by a vertical black line with a gradient color to refer to the radius value (0ms for prosody, 5ms for phonetics, discourse and syntax and 40ms for gestures in this screenshot). Uncertain labels are underlined and the related information is highlighted when it is clicked on.

Acknowledgments

This work has been carried out thanks to the support of the French state program ORTOLANG (Ref. Nr. ANR-11-EQPX-0032) funded by the " Investissements d'Avenir " French Government program, managed by the French National Research Agency (ANR).

8. References

B. Bigi and D. Hirst. 2012. SPeech Phonetization Alignment and Syllabification (SPPAS): a tool for the automatic analysis of speech prosody. In ISBN 978-7-5608-4869-3 Tongji University Press, editor, *Proc. of Speech Prosody*, Shanghai (China).

B. Bigi. 2012. SPPAS: a tool for the phonetic segmentations of Speech. In *The eighth international conference on Language Resources and Evaluation, ISBN 978-2-9517408-7-7*, pages 1748–1755, Istanbul, Turkey.

S. Bird, D. Day, J. Garofolo, J. Henderson, C. Laprun, and M. Liberman. 2000. ATLAS: A flexible and extensible architecture for linguistic annotation. In *Language Resources and Evaluation Conference*, Athens (Greece).

P. Blache, R. Bertrand, B. Bigi, E. Bruno, E. Cela, R. Espesser, G. Ferré, M. Guardiola, D. Hirst, E.-P. Magro, J-C. Martin, C. Meunier, M-A. Morel, E. Murisasco, I. Nesterenko, P. Nocera, B. Pallaud, L. Prévot, B. Priego-Valverde, J. Seinturier, N. Tan, M. Tellier, and S. Rauzy. 2010. Multimodal annotation of conversational data. In *The Fourth Linguistic Annotation Workshop*, pages 186–191, Uppsala (Sueden).

P. Boersma and D. Weenink. 2011. Praat: Doing phonetics by computer. [Computer Software] Amsterdam: Department of Language and Literature, University of Amsterdam. . http://www.praat.org/.

H. Brugman. 2003. Annotated recordings and texts in the dobes project. In *Proceedings of the EMELD Language Digitization Project Conference 2003: Workshop on Digitizing and Annotating Texts and Field Recordings. LSA Institute, University of Michigan, Lansing MI USA*.

H. Bunt. 2009. Multifunctionality and multidimensional dialogue semantics. *Proceedings of DiaHolmia*, 9.

P. D'Urso and L. De Giovanni. 2011. Midpoint radius self-organizing maps for interval-valued data with telecommunications application. *Applied Soft Computing*, 11:3877–3886.

S. Farrar and D-T. Langendoen. 2003. A linguistic ontology for the semantic web. *Glot International*, 7(3):97–100.

A. Gruenstein, J. Niekrasz, and M. Purver. 2008. Meeting structure annotation. In *Recent Trends in Discourse and Dialogue*, pages 247–274. Springer.

N. Ide and C. Brew. 2000. Requirements, tools, and architectures for annotated corpora. In *Proceedings of data architectures and software support for large corpora*, pages 1–5. Citeseer.

N. Ide, P. Bonhomme, and L. Romary. 2000. An xml-based encoding standard for linguistic corpora. In *Proceedings of the Second International Conference on Language Resources and Evaluation*, pages 825–830.

C. Jewitt. 2009. *Handbook of Multimodal Analysis*. London: Routledge.

M. Kipp. 2011. Anvil, DFKI, German Research Center for Artificial Intelligence. http://www.anvil-software.de/.

Nijmegen: Max Planck Institute for Psycholinguistics. 2011. ELAN - linguistic annotator. language archiving technology portal [computer software]. http://www.lat-mpi.eu/tools/elan/.

K. Rohlfing, D. Loehr, S. Duncan, A. Brown, A. Franklin, and I. Kimbarra. 2006. Comparison of multimodal annotation tools - workshop report. In *Online-Zeitschrift zur Verbalen Interaktion, Ausgabe 7*, pages 99–123.