Distributed System Architecture for Assisted Annotation of Video Corpora

Christophe COLLET, Matilde GONZALEZ, Fabien MILACHON

IRIT (UPS - CNRS UMR 5505)
Université Paul Sabatier, 118 Route de Narbonne, F-31062 TOULOUSE CEDEX 9
{collet, gonzalez, milachon} AT irit DOT fr

Abstract

This paper present one component of Dicta-Sign, a three-year FP7 ICT project that aims to improve the state of web-based communication for Deaf people. A part of this project is the annotation of sign language corpora. To improve the annotation task in terms of reproducibility and time consuming, several plug-ins for sign language video processing are developed. The component presented in this paper aims to link several plug-ins to annotation software through the network. These plug-ins can be coded in different languages, operating systems and computers. For that, it uses the SOAP Web-service and a specific data-format in XML for the data exchange.

1. Introduction

Nowadays many researches focus on the analysis and recognition of sign language to understand, reproduce and translate to any other communication language (Ong and Ranganath, 2005). In computer science those researches concern the development of automatic treatments applied to sign language videos (Lefebvre-Albaret and Dalle, 2009; Theodorakis et al., 2009). The evaluation of their performances uses annotated corpora which is, in general, manually performed by linguists and computer scientists. Several Annotation Tools (AT) have been developped to achieve this task, e.g. Elan (Wittenburg et al., 2006), Anvil (Kipp, 2001), Ilex (Hanke, 2002; Hanke and Storz, 2008), Ancolin (Braffort et al., 2004), etc. For long video sequences, manual annotation becomes error prone, unreproducible and time-consuming. Moreover the quality of the results mainly depends on the annotator's knowledge. Automatic video processing together with the annotator's knowledge facilitate the task and considerably reduce the annotation time. That is why we propose a way to integrate those automatic treatments, here called Automatic Annotation Assistant (A^3), to the available AT.

From the annotator's point of view, adding automatic treatments must be easy to use, without adding complex A^3 calling or extra working. The annotator should be able to extract a part of a video and to use a previously defined annotation as input parameter of the A^3. For example, the annotator is working in the Annotation tool window (fig. 1.a), any modification done is saved on the two tiers: AG1 and AG2. When the annotator calls an A^3, e.g. movement pose detection, which needs two input parameters, then two additional tiers appear in the window (fig. 1.b). Filling in the two tiers could be done manually or using AG1 and/or AG2. Once the treatment has finished the result is displayed as a new tiers that the annotator can easily save or modify (fig. 1.c). This example shows how using automatic processing in this way can be easily performed.

The complexity of integrating the A^3 to the AT is not just about programming an efficient user friendly interface but also about making A^3s and ATs to communicate with each other knowing that the programming environment used to developed them is not generally compatible. So, in this paper we propose a system architecture to allow the communication between the A^3s and the AT. We mainly focus on specifying communication protocols, data exchange and format.

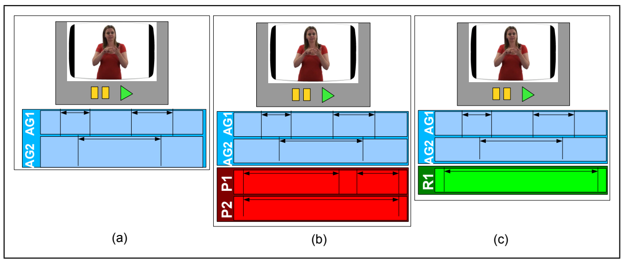

Figure 1: Annotation Tool Example: (a) Normal environment, (b) A^3 call and (c) A^3 result.

This document presents the specification for this distributed system for assisted annotation of video corpus. First, we present a global view of the system. It consist of an overview of the architecture of the system proposed. Second, we illustrate the different communication between each sub-part of the system and the data format used to make them communicate. Finally, we present our choice about development software to use and about the security of the system.

2. Global view

The main problem about the introduction of A^3s to existing ATs is the incompatibility of programming language, operative system and platform of development. Nevertheless it is not possible to restrict unique development conditions to easily use an A^3 to assist the annotation. Moreover treatments can be very complex and it would be preferable to develop them in a specific programming language or, even to execute them in adapted computers. That is why we proposed to overcome this problem by a Distributed System Architecture (DSA) where the A^3s are hosted in different computers.

The communication and the data exchange are, then, done trough the network using a protocol and an exchange data format understandable by all the parts of the system. The data format has to be standardized so that the ATs and the A^3s are able to process the data regardless where it

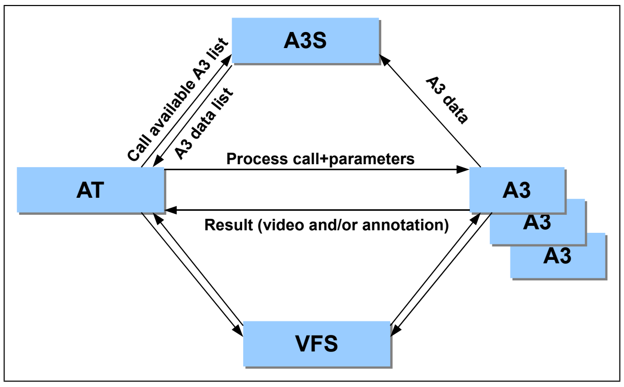

Figure 2: Distributed system architecture for assisted annotation of video corpus

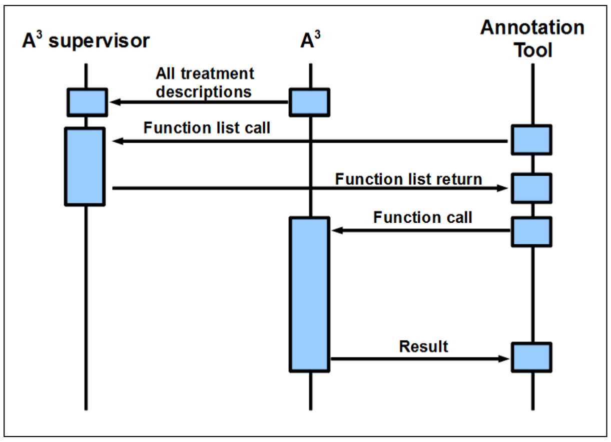

Figure 3: Simple Query Schedule

comes from. Thus the A^3's Application Programming Interface (API) has to use compatible parameters with the AT data structure. The data description standards proposed are XML and Annotation Graph (AG) structure (Bird and Liberman, 2001; Schmidt et al., 2008). The AG is a structure similar to the one used in the ATs, i.e. the hierarchy of named tiers (or levels or tracks...) with a list of possible values associated to each frame sequence. In this way any input parameter needed by the A^3, can be filled in by the annotator with the help of the AT. In addition ATs are, generally, able to easily import/export AG structures. The AG is stored in a XML file which is extended to add the metadata concerning the desired A^3 processing and the video file.

The proposed DSA is illustrated in Figure 2. The principle is to consider the AT as a client and the A^{3}s as remote servers to allow queries exchange. Since the number of available A^{3}s and ATs can vary on time depending on new developments, another server called Automatic Annotation Assistant Supervisor (A^{3}S) is added to manage the information of the A^{3}s at our disposal and to maintain an updated list of them. Thus at each time an A^3 is added it registers itself to the A^{3}S. Then when the AT requires an updated list of A^{3}s it requests the A^{3}S server. Now the AT can directly communicate with the A^3 as long as the A^3 descriptor is known. In addition the need of exchanging video files between ATs and A^{3}s leads to introduce a Video File Server (VFS) to share videos in a simple and fast way.

3. DSA data exchange

The AT allows annotators to easily define and execute various queries in a controlled manner. It interacts with all the parts of the system. Firstly, for the initialization process it queries the A^{3}S. Secondly, to process video it communicates to the respective A^3. Finally, to add or to retrieve processed video files, it interacts with the VFS.

The A^3 communicates with the A^{3}S to register itself when it is added. All those interactions are illustrated in Figure 3

3.1. A^3 Registration

Each A^3 is considered as a unit implementing various processing functions for the annotation. To reference these functions, each time an A^3 is added, it transmits its descriptor to the A^{3}S. The descriptor is a XML code containing API which has, among other information, a unique identifier (ID), the address (@), the port number (P) and the help text.

3.2. AT Initialization

The first query is automatically performed by the AT when it is loaded. This query is sent to the A^{3}S to ask for the list of available A^3 descriptors. The list can also be manually requested for updating at any moment. It does not require any parameter. In return, the A^{3}S sends the list of A^{3}s and their descriptor. The AT can therefore decode the list and show to the annotator the available A^3's functions descriptions.

3.3. A^3 calling

When the annotator selects a function, the parameters of this function are set up by filling in the AG provided by the A^3 descriptor. The minimum functionality that is expected from the AT, is an interactive editor for this AG. Two indispensable data elements are a list of videos and the ID of the process. The list of videos could correspond to different views for a same corpus. Once processing has been performed, it encapsulates the results, again in the form of an AG structure and sends them as a reply to the requesting AT.

4. XML Format

Previously, we defined that the API of each A^3 process uses a data format similar to the one used in AT, the AG. Due to the diversity of Annotation systems used, we need a simple global and compatible annotation graph system. That is why we decided to use an annotation graph format based on the one defined in Schmidt et al. (2008). The one we define is simple, easy to use and open. Thanks to this format, client can easily define frames and parameters for each frames, to use. In the server side, it has to read this AG to get the parameters needed to its process, execute it and finally put the result in the same AG. In order to simplify the processing of the parameters of the API and to get coherence, the video names, the process to call and every parameter are defined in this AG.

Finally, we have all annotation data, input and output, temporally described in an AG and encapsulated in one XML file. In addition this file contains the whole informations like API, option parameters, process descriptor and video location.
This AG encoded in XML format is described here step by step, through a simple API example.

```
<Metadata>
  <Parameter_in position="1">
    <Name value="Threshold_1"/>
    <Type value="Integer"/>
    <Default_value value="0,5"/>
    <Source value="A3_UPS"/>
  </Parameter_in>
  <Parameter_out position="2">
    <Name value="Coord_y_1"/>
    <Type value="Double"/>
    <Source value="A3_UPS"/>
  </Parameter_out>
```

Figure 4: XML data exchanged: Input and output parameters metadata

The figure 4 describes an input parameter and an output parameter in the API. Thanks to XML, this format can be easily parsed to get the different information about the input parameter, like its type and its default value. Each different parameter use its own `Parameters_in` or `Parameters_out` tag with a different position number.

```
<Process_Metadata>
  <Name value="Process1"/>
  <IP_adress value="127.0.0.1"/>
  <Port value="8080"/>
  <Source value="A3_UPS"/>
  <Help>
    Here the HELP of the process.
  </Help>
</Process_Metadata>
```

Figure 5: XML data exchanged: Process metadata

```
  <Video_Metadata>
    <Name value=""/>
    <IP_adress value="127.0.0.1"/>
    <Port value="8008"/>
    <Source value="SFTP"/>
    <Login login="" pwd="" />
  </Video_Metadata>
</Metadata>
```

Figure 6: XML data exchanged: Video metadata

Figure 5 and figure 6 describe respectively metadata about the process described by the API and inform about treated video location. Most of the process metadata are about location of the process too. Moreover, it encapsulates the identification parameters (login and password) for the video server. If identification is needed to call process too, it can be easily added in the process metadadta one the same way.

```
<Timeline id="A3_Timeline1">
  <Signal id="A3_Timeline1_Signal1"
          unit="frames" mimeClass=""
          mimeType="" encoding=""
          xlink:href=""/>
</Timeline>

<AG timeline="A3_Timeline1" id="A3_AG1">
   <Anchor id="T0" offset="0" unit="frames"/>
   <Anchor id="T1" offset="1" unit="frames"/>
   <Annotation id="Annotation_1_T0"
               type="Parameter_in_1"
               start="T0" end="T0">
       <Feature name="Threshold_1"
            value="0,5"/>
       <Feature name="coord_x" value=""/>
   </Annotation>
</AG>
```

Figure 7: XML data exchanged: Annotation Graph (Time-Data)

Finally, figure 7 is a classical use of AGlib (Annotation Graph library) with definition of two time anchor and one AG in between. This AG contains, at the beginning, the minimal data: input parameters with default values and empty output results.
So, this format enables to represent all needed metadata. The user of the Annotation Tool has just to fill some parameters, like the video to use, and add each anchor and each annotation he needs. Afterward he will fill in those annotations with the desired input parameters and their values. When it is done, he sends this XML and the A^3 will decode what it has to do. To send the result, it will automatically create result Annotation (and anchor if it need an anchor couple for each frame), fill their value, and send them.

5. Software development

We need a software library for network programming, which allows to develop this fairly simple architecture. Development constraints are that this system must be multi-platform and multi-language - including for the annotation software : RealBasic, C/C++ and Java - therefore we exclude proprietary libraries such as Java RMI or Twisted. Most of the time, the data to be transmitted are already in XML format, so a string can suffice. To achieve this kind of system two types of library are distinguished: the middleware - ICE (ZeroC, URL), CORBA (ObjectManagementGroup, URL) - and the Webservices - SOAP (W3C, URL), XML-RPC (XML-RPC, URL). The main difference between these two categories is that the first one, the middleware, is based on the use of objects and method calls on these objects, while the second one, web-services, is based on the use of messages sent to URLs. It should be noted that each of these technologies meet our expectations, with different degrees of difficulty and complexity. We decided to use SOAP for the first specification of this architecture because it meets our needs in a simple way and it is totally open-source.

6. Security

In this system, we need security measures especially concerning the video corpus database. Indeed, those video files are not necessarily publicly accessible. To implement a sufficient security level, we propose two components: a secure transfer protocol, HTTPS, to transfer data by SOAP ; and a SFTP protocol for the transfer of video between the video file host and A^3 or Annotation Tool.

7. Conclusion

In conclusion we propose a communication system architecture to easily add and call automatic treatments supporting annotation task in existing annotation tools.
Thanks to our specifications and the use of SOAP for software development, this architecture is multi-platform and multi-language (including RealBasic, C/C++ and Java) and the model used for data exchange is adaptable to many annotation formats. Furthermore, this model contains every needed information like location of the process to call, video to use, input and output parameters. The system is composed of four parts : the Annotation Tool (AT), the Automatic Annotation Assistants (A^3), the A^3 Supervisor (A^{3}S) and a Video File Server (VFS). All communications between those entities are made through SOAP and are secured.
The programming of this system is underway during the project Dicta-Sign, and will enable to do evaluations of the contribution of automatic annotation process during annotation tasks
For future work we intend to enable asynchronous communication between AT and A^3 in order to avoid waiting for the end of a long process (more than few seconds) and enable a deferred query for results. We also would like to enable time synchronized communication between the AT and interactive applications like the Signing avatar synthesizer (Kennaway et al., 2007) or Signing space annotation tool (Lenseigne and Dalle, 2005).

8. Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^o 231135.

9. References

S. Bird and M. Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(Issues 1-2):23–60, January.

A. Braffort, A. Choisier, C. Collet, P. Dalle, F. Gianni, B. Lenseigne, and J. Segouat. 2004. Toward an annotation software for video of sign language, including image processing tools and signing space modelling. In *Proc. of 4th International Conference on Language Resources and Evaluation - LREC 2004*, volume 1, pages 201–203, Lisbon, Portugal, May.

T. Hanke and J. Storz. 2008. ilex - a database tool for integrating sign language corpus linguistics and sign language lexicography. In *Proc. of 6th International Conference on Language Resources and Evaluation, LREC 2008*, pages W25–64–W25–67, Marrakesh, May.

T. Hanke. 2002. ilex - a tool for sign language lexicography and corpus analysis. In *Proc. of 3rd International Conference on Language Resources and Evaluation, LREC 2002*, pages 923–926, Las Palmas de Gran Canaria, Spain.

J.R. Kennaway, J.R.W. Glauert, and Zwitserlood I. 2007. Providing signed content on the internet by synthesized animation. *ACM Transactions on Computer-Human Interaction*, 14(3):15/1–19, September.

M. Kipp. 2001. Anvil - a generic annotation tool for multimodal dialogue. In *Proc. of 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1367–1370.

F. Lefebvre-Albaret and P. Dalle. 2009. Body posture estimation in a sign language video. *In Proc of The 8th International Gesture Workshop*, Feb.

B. Lenseigne and P. Dalle. 2005. Using signing space as a representation for sign language processing. In *Proc. of 6th International Gesture Workshop - GW 2005*, pages 25–36, Berder Island, France, 18-20 May. Springer-Verlag.

ObjectManagementGroup. URL. Corba documentation. *http://www.omg.org/technology/documents*.

S.C.W. Ong and S. Ranganath. 2005. Automatic sign language analysis: A survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 873–891.

T. Schmidt, S. Duncan, O. Ehmer, J. Hoyt, M. Kipp, D. Loehr, M. Magnusson, T. Rose, and H. Sloetjes. 2008. An exchange format for multimodal annotations. In *Proceedings of the 6th International Language Resources and Evaluation (LREC'08)*, pages 207–221, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

S. Theodorakis, A. Katsamanis, and P. Maragos. 2009. Product-hmms for automatic sign language recognition. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 00, pages 1601–1604. IEEE Computer Society.

W3C. URL. Soap documentation. *http://www.w3.org/TR/soap/*.

P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes. 2006. Elan: a professional framework for multimodality research. In *Proc. of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, pages 1556–1559.

XML-RPC. URL. Xml-rpc documentation. *http://www.xmlrpc.com/spec*.

ZeroC. URL. Ice documentation. *http://www.zeroc.com/download/Ice/3.4/Ice-3.4.0.pdf*.