The VIW project

(Visuals into Words)

Technical documentation

September 2016

Contributors:

Anna Matamala, Marta Villegas.

Description:

VIW (Visuals Into Words) is a project funded by the Spanish "Ministerio de Economía y Competitividad" (FFI2015-62522- ERC) that aims to research audio description (AD) from a multilingual and multimodal corpus perspective.

This is a Technical Documentation document that describes the VIW data and web application. It provides guidance for learning about the eventual implementation of the project and intends to assist VIW users and future developers. The document also reports the problems encountered when using external tools and provides some recommendations.

Transmedia Catalonia

http://transmediacatalonia.uab.cat/

Universitat Autònoma de Barcelona (UAB)

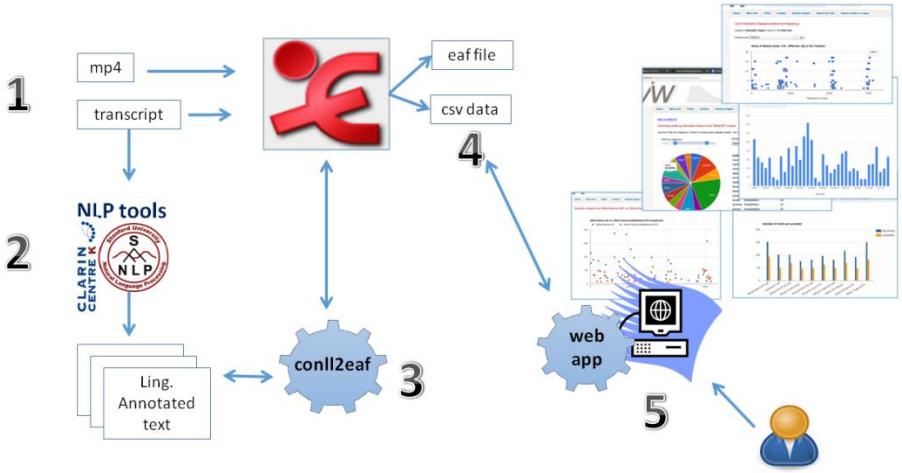
Index

1 Preface

This is a Technical Documentation document that describes the VIW data and web application. It provides guidance for learning about the eventual implementation of the project and aims to assist VIW users and future developers.

The figure below offers a general picture of the project which can be summarized as follows:

1. Input data (a set of movies plus their corresponding audio description transcripts) are loaded into ELAN tool.
2. Transcripts are sent to NLP tools which produce linguistic annotations.
3. These linguistic annotations are loaded into ELAN.
4. Once all annotations are in ELAN, the tool is used to query and export data.
5. The Web app allows browsing these data and provides different visualizations.



The VIW project overview: data and processes

This is both a descriptive document and a user manual and focuses on the steps highlighted above. Sometimes it deals with 'small' technical details that may surprise a general user. These technical details are labeled as **Warnings** and report the problems we found when running tools (and /or dealing with data) and provide some recommendations.

The document is structured as follows:

Section 2 The VIW corpus describes the corpus of the project and how it is organized. For further details on data organization see section 5.1 The data directory

Section 3 Working with ELAN reports on ELAN usage in the VIW project. It describes the annotation schema, offers guidance to users/developers to create and add new data, and explains how to use ELAN to analyze the resulting data.

Section 4 Adding linguistic annotations into .eaf files explains how to use NLP tools to linguistically annotate transcriptions and aims to assist future users/developers when importing new annotations into the project.

Section 5 The web app describes the VIW web application. This is a 'user manual' section that aims to guide eventual users/developers when developing new data.

Section 6 Using the linguistic data with CQPweb explains how to use VIW data in CQPweb.

2 The VIW corpus

The VIW corpus is made up of audio descriptions of the same short film in different languages. It currently includes 10 professional audio descriptions in English, 10 professional audio descriptions in Spanish and 10 professional audio descriptions in Catalan. All 30 audio descriptions were commissioned to professionals, who were remunerated. It also includes 10 audio descriptions in Spanish and 7 audio descriptions in Catalan made by students (text only, not recorded audio/video files).

All these audio descriptions are grouped into five different corpora: three 'professional' corpora (one per language: English, Spanish and Catalan) and two 'student' corpora (Catalan and Spanish).

From now on the word **'corpus'** will be used to refer to one of the five corpora described above and **'provider'** will be used when referring to a specific audio description that belongs to a particular corpus.

All data files are grouped in a directory structure that reflects the corpus/provider schema. For each provider a video file and a set of data files are available. The video files can be found in the project's web page and are stored in the UAB digital repository (http://pagines.uab.cat/viw/content/corpus and https://ddd.uab.cat/record/147267). All data files are available in the GitHub repository (https://github.com/TransmediaCatalonia/viw-project/tree/master/data).

Essentially, data files include the text files with the audio description transcripts, the ELAN files containing the corresponding annotations and a number of extra files. In section 5.1 The data directory more details about data files and data structure are to be found.

3 Working with ELAN

The ELAN tool[1] is used to create complex annotations on audio descriptions and to analyze the resulting data. ELAN allows annotations to be linked to their corresponding video files and saves these links in the annotation file. The annotation file is an XML file conforming with the .eaf format[2]. ELAN also provides a powerful set of tools to assist video encoding and to perform eventual analysis.

3.1 The .eaf files in the VIW project (tiers and annotations)

As defined in the ELAN manual

> an **annotation** is: *"any type of text (e.g. a transcription, a translation, coding, etc.) that you enter on a tier. It is assigned to a selected time interval of the video/audio file (e.g., to the utterance of a speaker) or to an annotation on another tier (e.g., a translation is assigned to an orthographic transcription)."*
>
> and a **tier** is: *"a set of annotations that share the same characteristics, e.g., one tier containing the orthographic transcription, or another tier containing the free translation."*

The VIW project distinguishes between two kinds of tier according to the type of annotations they contain: linguistic tiers and filmic tiers.

The **linguistic tiers** (see 3.1.1 Linguistic tiers) consist of a parent tier containing the transcription of the audio description (AD unit) plus a set of dependent or referring tiers used to linguistically annotate the AD units. The AD tier is linked to a time interval of the media file whereas the referring tiers are symbolically linked to their parent annotation (they are tied to a specific annotation in the AD itself).

The **filmic tiers** (3.1.2 Filmic tiers) encode relevant elements in film construction.

---

[1] http://tla.mpi.nl/tools/tla-tools/elan/ developed by the Max Planck Institute for Psycholinguistics, The Language Archive, Nijmegen, The Netherlands.

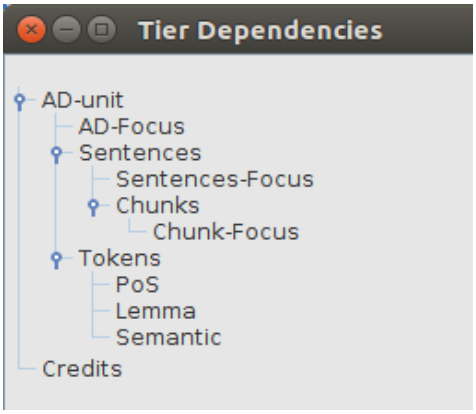
[2] The ELAN Annotation Format, also known as the EUDICO Annotation Format.

3.1.1 Linguistic tiers

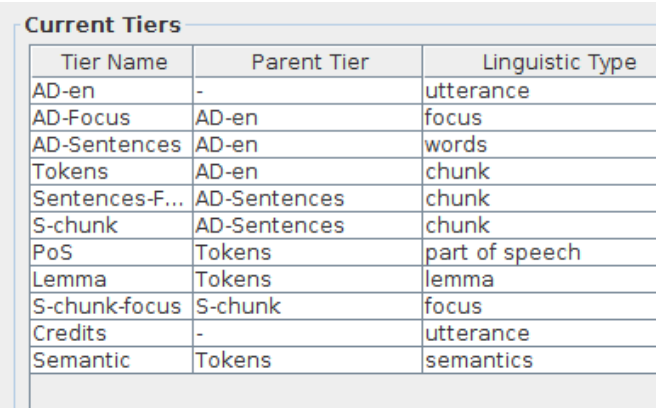
Linguistic tiers include two top tiers: the AD-unit tier (containing the AD units) and the Credits tier (with the credits). Both top tiers are time-aligned.

The AD-unit tier has three dependent tiers: AD-Focus, Sentences and Tokens. AD-Focus was intended to encode the focus of ADs. The Sentences and Tokens tiers are used to further split the AD units into smaller pieces: sentences and tokens respectively. The Sentences tier has two dependent tiers: the Sentences-Focus and the Chunks, which in turn has a focus dependent tier. Focus tiers were all intended to encode the focus of their parent tier. Unfortunately, due to time constraints in the project, no focus annotations are encoded. The Chunks tier further splits sentences into chunks. Finally, the Tokens tier has three dependent tiers: PoS, Lemma and Semantic. They are all symbolically associated to Tokens.

The Credits tier contains the credits of the movie normally at the end of the films and it does not have any dependent tier.



Tier dependencies



Tier Name	Parent Tier	Linguistic Type
AD-en	-	utterance
AD-Focus	AD-en	focus
AD-Sentences	AD-en	words
Tokens	AD-en	chunk
Sentences-F...	AD-Sentences	chunk
S-chunk	AD-Sentences	chunk
PoS	Tokens	part of speech
Lemma	Tokens	lemma
S-chunk-focus	S-chunk	focus
Credits	-	utterance
Semantic	Tokens	semantics

Tier attributes

See section 4.1 AD units or sentences? to find out why we use this AD unit/Tokens and AD unit/Sentences hierarchy instead of the AD unit/Sentences/Tokens one.

In VIW we distinguish between ‘segmentation tiers' (sentences, chunks and tokens) and ‘annotation tiers' ('focus', Lemma, PoS and Semantic).

See section 3.2.2 Segmentation: tokens, sentences and chunks to find out how to add segmentation annotations into *eaf* files.

See section 4 Adding linguistic annotations into .eaf files for more details about importing linguistic annotations into .eaf files.

3.1.2 Filmic tiers

Visual tagging took into account relevant elements in film construction. The film director was consulted as well as literature on filmmaking, and the following tiers were selected.

Visual tagging was carried out in a unified way for all three versions except for sound, which can present differences and was tagged independently for each language.

3.1.2.1 The SCENE tier

The SCENE tier is based mainly on the setting where the action takes place, but it also includes two black screens and the final credits where there is no action. In the case of a scene which is cut by the title, it is coded as one single scene indicating that it has been cut.

The coding used is: letter S followed by the number and a description of the location.

This is the whole list of codes:

S1- Promenade (with text cut)
S2- Beach
S3-Street
S4-Mountain
S5-Park
S6-Flat
S7-Street
S8-Mountain
S9-Clearing
S10-BlackScreen
S11-Roof
S12-BlackScreen
S13-Credits

3.1.2.2 The SHOT tier

The SHOT tier includes a code which refers to the type of shot which, using the human body as a measure, indicates what the camera is showing. Bordwell and Thompson's (2008:191) monograph *Film Art: an Introduction* has been used as a reference to define shot types. The coding used is as follows:

ELS: extreme long shot
LS: long shot
MLS: medium long shot
MS: medium shot
MCU: medium close-up
CU: close-up
ECU: extreme close-up
DS: detail shot (including a description of the object being shot).

When text appears, be it a title, credits or other types of text, the word "Text" is used.

When a black screen appears, the words "BlackScreen" are used.

A combination of codes has also been used: for instance "MLS-to-MS" means that a medium long shot evolves towards a medium shot. Or "CU-DS(desk)-CU" means that a close-up changes into a detail shot which shows a desk and then moves to a close-up again. This is a way of indicating camera movements which have not been coded explicitly.

3.1.2.3 The SOUND tier

The SOUND tier includes reference to sound elements, which can overlap. The following codes have been used:

- **ND** sound effect: non-diegetic sound effects such as sound shot transitions, dramatic sounds. Only a selection of the most significant ones has been tagged.
- **D** sound effect: diegetic sound effect. Only the most significant ones have been tagged, such as a mobile phone ringing.
- **Speech** refers to the verbal language used by characters.
- **Paralinguistic** refers to non-verbal sounds made by characters, such as coughs, sneezes, etc.
- **Music**
- **Sound motif** refers to a meaningful cricket-like sound which is highly present in the film and gives meaning to it.

The overlapping of various sounds is tagged using a plus sign, such as "ND sound effect + sound motif" or "Speech + sound motif".

When no sound is present, the code "No sound" has been used.

3.1.2.4 The CHARACTER tier

The CHARACTER tier indicates the character or characters shown on screen, if any, using the following codes.

Extra/ Extras, James, Rick, Jess and Zoe

When no characters are on screen, the code “Null” is used. When more than one character is on screen, it is represented as follows: “James + Rick + Jess”.

3.1.2.5 The TEXT tier

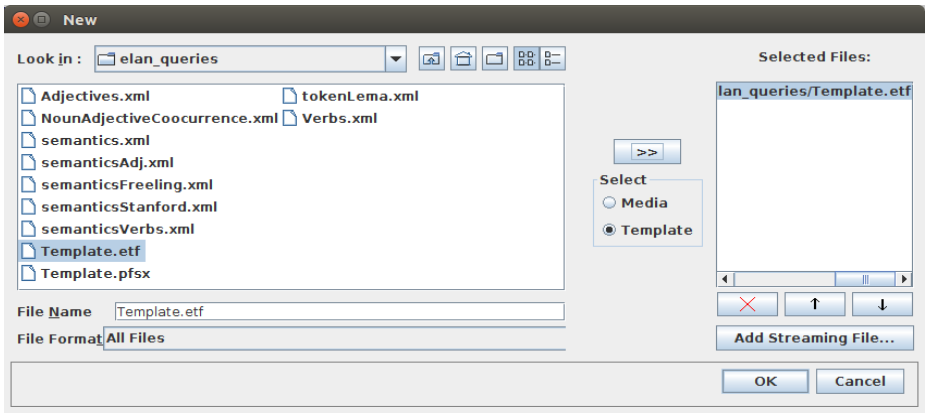
The tier TEXT refers to text on screen and includes the following possible elements:

- **Title** it refers to the title of the short film.
- **Chyron** refers to text on-screen which is not part of the action and that provides additional information, such as location, time, character identification, etc.
- **Mobile text** refers to text which is part of the action and shown on a mobile phone.
- **Subtitle** refers to text on screen which translates the audio in another language.
- **Credits** refers to the cast members listed in this short film at the end.

Sometimes an additional indication can be added when a title cuts a scene. For instance, “Title (cutting S1)”.

3.2 Creating a new VIW .eaf file

To create a new .eaf file, go to the New option in the File tab, select the media files involved and choose the template as indicated below (the template file can be found in the elan_queries directory in the GitHub repository):



Creating a new .eaf file (select Template)

This will create a new .eaf file with empty tiers. Now you have to import the AD transcript into this new.eaf file and start annotating it.

3.2.1 Importing AD transcripts

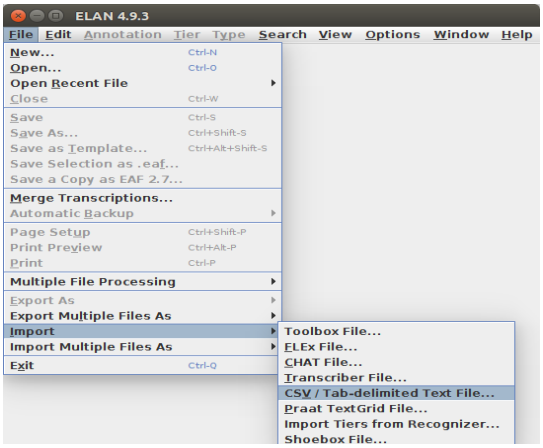
AD transcripts must be collected in tabular files where each line contains an AD unit and each AD unit is assigned a begin and end time as follows[3]:

```
00:00:04.848    00:00:11.848    De día, en el paseo marítimo de …
00:00:12.448    00:00:14.448    El cielo está algo nublado y gris.
00:00:17.167    00:00:18.849    Funde a blanco con la frase:
00:00:21.403    00:00:25.083    Un coche de alta gama aparca en ...
```

The process of loading the ADs from the tabular file into the new.eaf file you created involves two steps: first, you have to import the AD transcripts into an additional .eaf file and, then, merge both files.

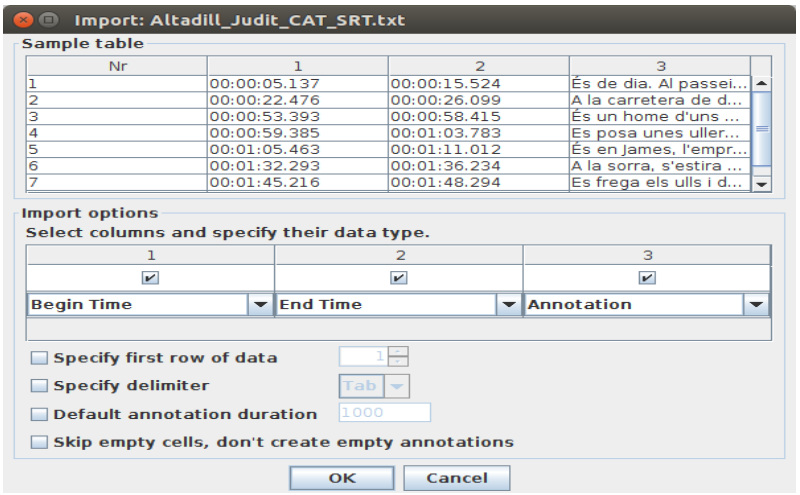
First, select the import option as illustrated below:

[3] Make sure you follow the correct time format.



Importing audio descriptions from a tabular file

In the new popup window, select the tabular file containing the ADs and fill in the form as illustrated:



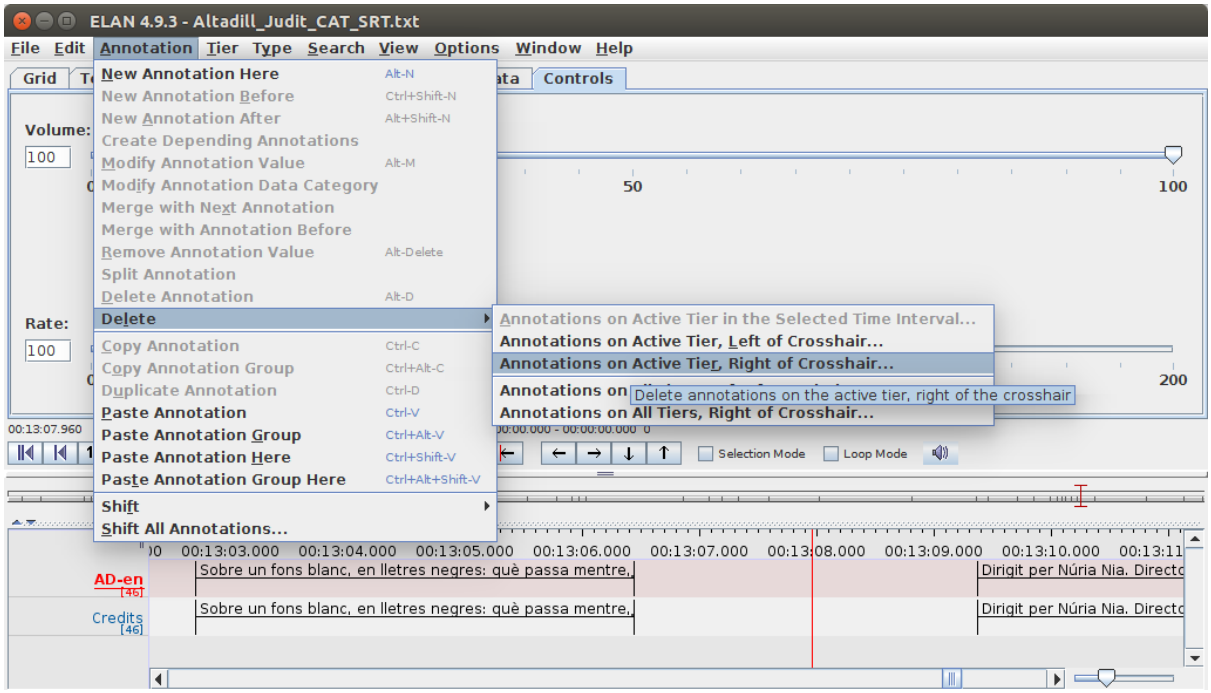
Specifying import options

This will load AD units into a default tier (Tier-0). You have to rename this default tier and create the Credits tier as follows:

1. Rename the default tier (Tier-0) as AD-unit using the “Change Tier Attributes” in the Tier tab.
2. Copy the AD-unit tier using the “Copy Tier” option in the Tier tab. This will create a new parent tier called AD unit-cp.
3. Rename the newly created tier as Credits.

This will produce two tiers (AD unit and Credits) with the same content. You only have to remove the AD units containing the credits from the AD-unit tier and delete the ‘descriptive’ units from the Credits tier. To do this, proceed as follows:

Select the AD-unit tier (or Credits tier) and use the mouse to place the crosshair where the credits start. Select the Delete option in the Annotations tab and choose the “... right to crosshair” or “...left to crosshair” option depending on the tier you are working with. Finally save the file.

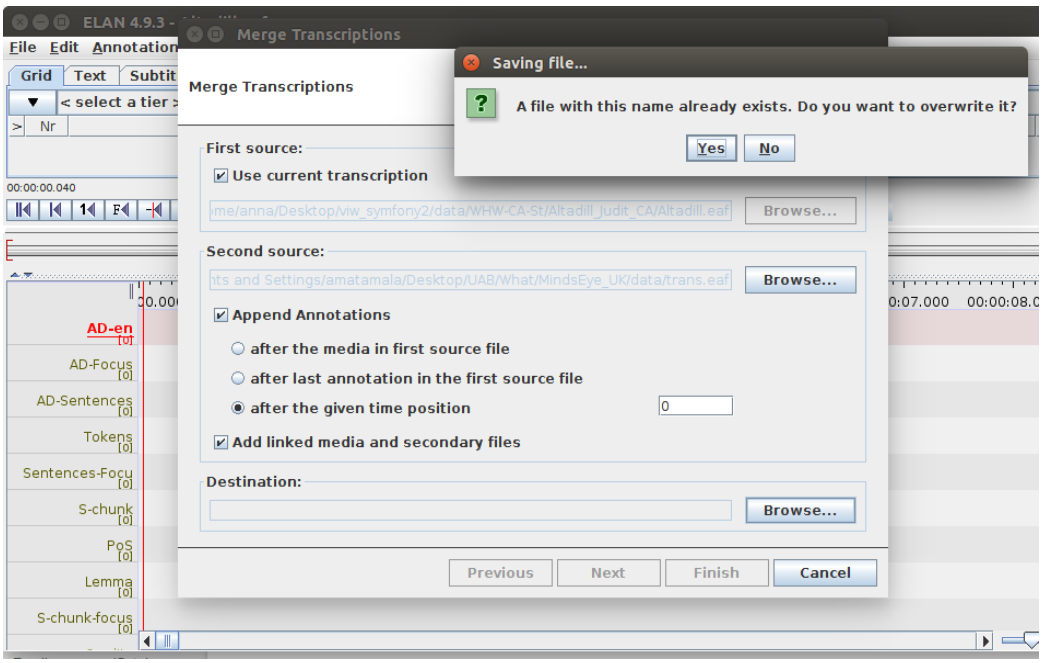


Editing the AD-unit and Credits tiers

This will create an .eaf file with the transcriptions. We will call this file *trans.eaf*.

Finally, you need to merge the *new.eaf* file (the one with empty tiers) with this *trans.eaf* file. Once merged, the data in the second file will be loaded into the first one.

To merge the files, select “Merge Transcriptions” in the File tab. In the new window fill in the fields as indicated. If you have the *new.eaf* file open, just check the “Use current transcription” option. In the “Second source” option you have to select the file you created with the ADs (*trans.eaf*). Finally, in the “Destination” field, select the *new.eaf* file (the one that is open). This will produce a warning message. Click OK and proceed.



Merging transcriptions

Once the merging is done, the AD-unit and the Credit tiers in the *new.eaf* file will contain the corresponding data.

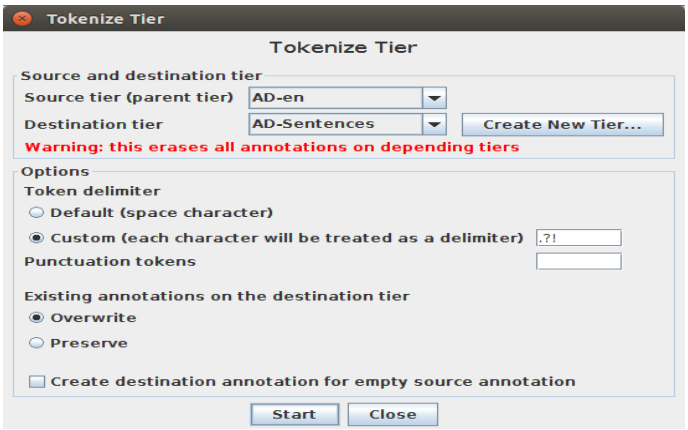
Warning. At this point ELAN will have two open windows: one with the new merged version and one with the old version. Be careful to close the old one and save the new merged one.

3.2.2 Segmentation: tokens, sentences and chunks

The ELAN tool includes a “Tokenize tier” service that splits the content of an annotation unit on a parent tier into smaller parts on another tier. We can use this tool to split AD units in the AD-unit tier into sentences and chunks.

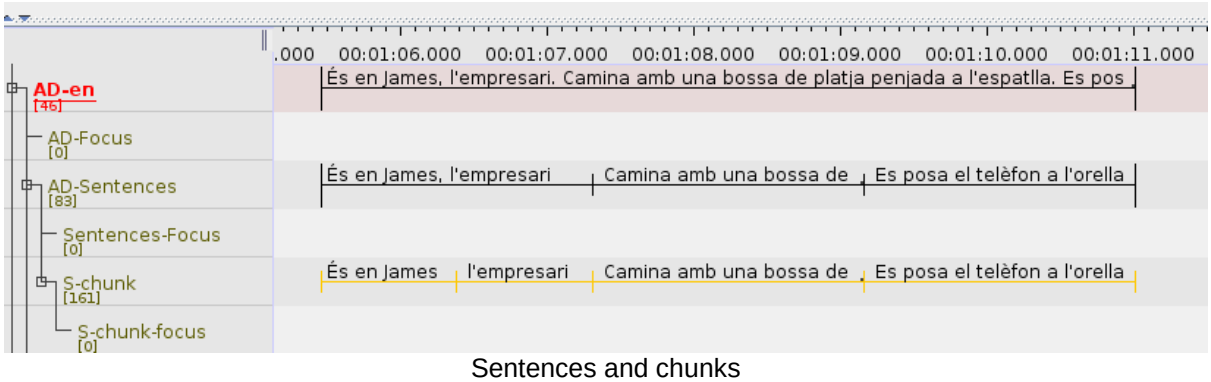
Warning: Note that for Tokens and its derived tiers we did not used this service but rather use external NLP tools (see section 4.2 Annotating ADs) as the tokenisation produced by ELAN service and the tokenisation produced by NLP tools will not match.

To split AD units into sentences (and/or chunks), select the “tokenize tier” option in the Tier tab and fill in the form as follows (the idea is to use sentence delimiter characters):



Splitting AD units into Sentences

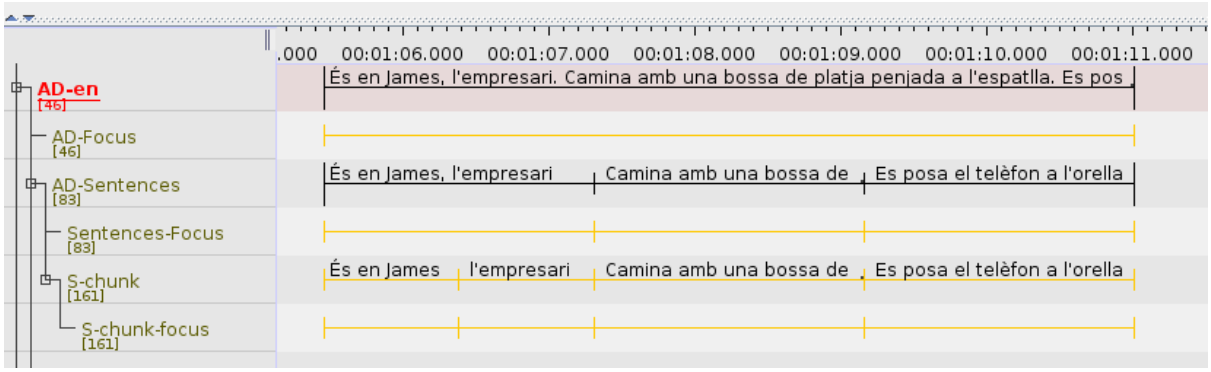
Similarly, you can ‘tokenize’ the Sentences tier into chunks (using delimiters such as: .;:.!?). Once the ‘tokenization’ is done, the dependent tiers (sentences and chunks) are split into the corresponding segments. In the following example we have an AD unit which is split into three different sentences. Note, in addition, that the first sentence is further split into two chunks:



Sentences and chunks

If you want to use the focus tiers to annotate AD units, sentences and chunks you need to create the corresponding dependent annotations. These annotations will be symbolically linked to their corresponding parent annotation. Select the “Create annotations on dependent tiers” option in the Tier tab. Check the AD, Sentence and Chunk tiers and press the Next button. In the new window, select the corresponding focus tiers and hit “Finish”.

This will create empty dependent annotations in the focus tiers. Thus, for each annotation in the AD unit, Sentences and Chunks tiers, a corresponding annotation is created in their dependent tiers:



AD units, sentences and chunks + their dependent focus annotations

Now, you can edit the focus tiers to add the annotations you want.

3.2.3 Comments and recommendations about input data

When we commissioned the professional audio descriptions, we did not give strict instructions about the format to follow for the textual transcriptions. Consequently, we received the data in a variety of formats which made text manipulation a rather time-consuming task.

It was surprising to see that only very few providers used the ‘standard’ transcript format and that they used different file types, different format conventions and even different time codes (including wrong ones).

Students providers were given much clear instructions and they supplied better formatted texts.

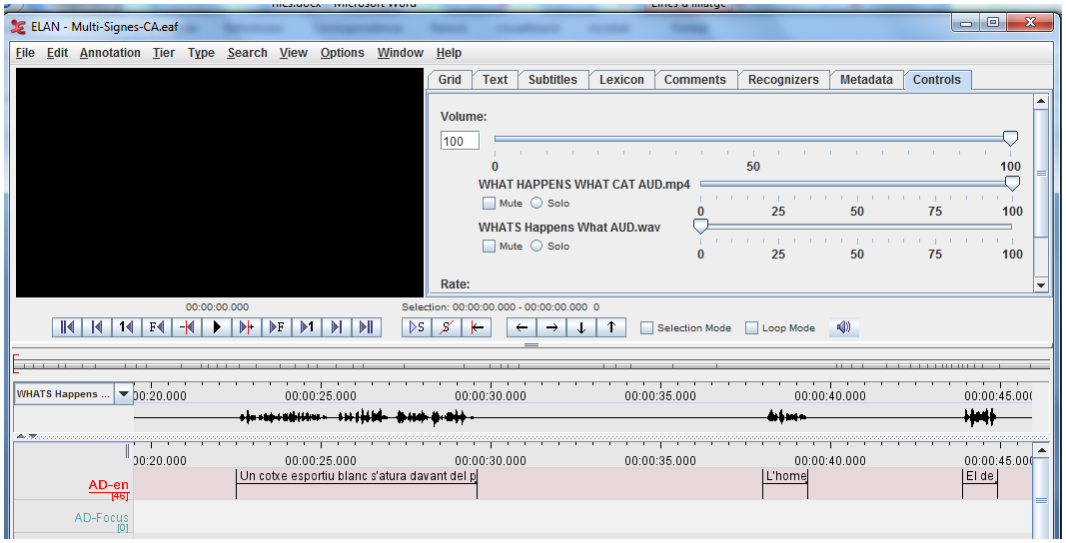
In any case, very often time assignment was not accurate and, in many cases, the time annotated and the real time were different. This forced us to undertake a manual checking task.

When checking time annotations, the *wav* files proved very useful. The ELAN tool allows different media files (audio+video) to be worked with. We used the audio files (*wav* files) and the waveform viewer for greater precision when assigning time intervals to AD units. You can see a screenshot some lines below.

It is also interesting to note the different criteria when defining AD units. Typically, an AD is made of AD units separated by ‘long’ pauses whereas AD units are made of sentences separated by very ‘short’ pauses (a second or less). Sometimes, however, this long/short distinction is not so obvious. Although, in general, we observed the segmentations carried out by providers, occasionally we had to join different AD units into a single unit because there was no pause between them (the pause was too short). Similarly, we split AD units when they include long pauses. In general, ‘long pauses’ are longer than one second.

Only in a very few cases an AD unit may include a long pause. This occurs when the speaker produces a 'long' pause in the middle of a sentence to accommodate the speech with the action on the screen. In such cases we decided not to split the sentence.

Warning: We found that some ELAN functionalities do not behave correctly when time slots have no gaps (when the end time of an annotation is the same as the beginning time of the following annotation). This situation generates erroneous results when running a “Multiple Layer Search”.



Using the waveform to edit time intervals

3.3 Using ELAN Domains as corpora

Normally, when using the ELAN tool we work with only a single file at a time. Sometimes, however, it is useful to work with multiple files at once. ELAN allows editing, querying and exporting a group of files. This feature allows corpus files to be grouped in ELAN domains.

In section 5.1.2.2 Sem.csv we explain how to define and use Domains when querying multiple files and exporting data.

3.4 Analyzing data in ELAN

ELAN includes a huge variety of functionalities that can be used to analyze the data we produce. Especially useful are the search functionalities among which we distinguish the Multilayer Search facility we exemplify in section 5.1.2.2 Sem.csv.

A description of ELAN possibilities is out of the scope of this document and we just describe some of the analyses we carried out and the way these are used by the web application (see section 5 The web app for more details on the web application).

In the VIW project, transcripts are annotated with linguistic and filmic information and everything is collected in .eaf files and managed using ELAN. ELAN is not a lexical analysis tool or a corpus linguistics tool. If you want to run lexical analyses you can use other tools. What makes ELAN special here is that our data is **time aligned** and enriched with **filmic annotations.** This means that our experiments deal with linguistic material but also with time and filmic information. In the following section we give some examples.

3.4.1 Linguistic annotations in the timeline

We used the “Export Multiple Files As” option to get a tabular file with relevant information about AD units (see section 5.1.2.1 Hits-All.txt for more details). This allows us to provide different views concerning AD unit distributions in time:

- comparing distributions from different providers as in:
  http://transmediacatalonia.uab.cat/web/hits/timeline/WHW-EN-Pr
  http://transmediacatalonia.uab.cat/web/hits/timelinejs/WHW-EN-Pr
  http://transmediacatalonia.uab.cat/web/density/graph/Mennell-CA/Plurals-CA

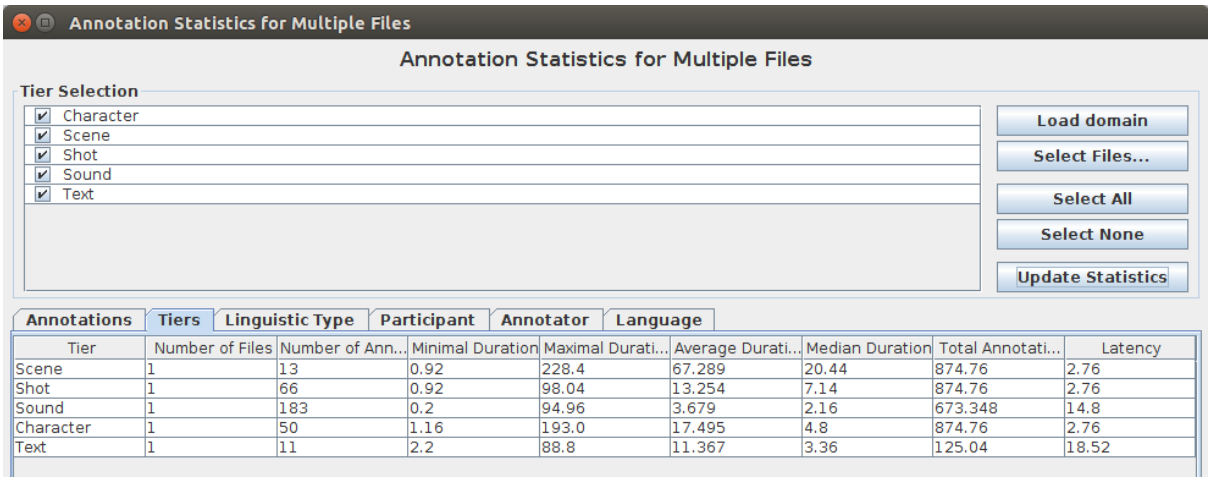
- seeing the duration, length and density of AD units of a specific provider as in:
  http://transmediacatalonia.uab.cat/web/hits/time/WHW-EN-Pr/Bridge-US
  http://transmediacatalonia.uab.cat/web/hits/words/WHW-EN-Pr/Bridge-US
  http://transmediacatalonia.uab.cat/web/hits/timewords/WHW-EN-Pr/Bridge-US

For verbal semantic classes, the "Multilayer Search" facility allows us to get information about words and semantic classes in a single *csv* file (see 5.1.2.2 Sem.csv). We use this file to see how verbal semantic classes are distributed in the timeline. You can see an example here: http://transmediacatalonia.uab.cat/web/vocabulary/verbssem/WHW-EN-Pr

3.4.2 Interaction between filmic and linguistic annotations

In an AD corpus, it is reasonable to think that there must be some kind of (co)-relation between filmic and linguistic annotations. For example, we expected that scene transitions would trigger audio descriptions introducing the new scene. We initiated a rather preliminary study concerning filmic and linguistic interactions as exemplified in: http://transmediacatalonia.uab.cat/web/hits/timevisual/WHW-EN-Pr/BTI-UK

Note, in addition, that the different corpus .eaf files can be used to export relevant data or to perform some relevant queries. For example, we can use the "Multiple File Processing/Statistics for Multiple Files" tool (applied to a single .eaf corpus file[4]) to get information about annotations on filmic tiers. In the figure below, we get the number of annotations and different time information for each filmic tier. This allows us to see (and compare) the weight (presence) of the filmic tiers.

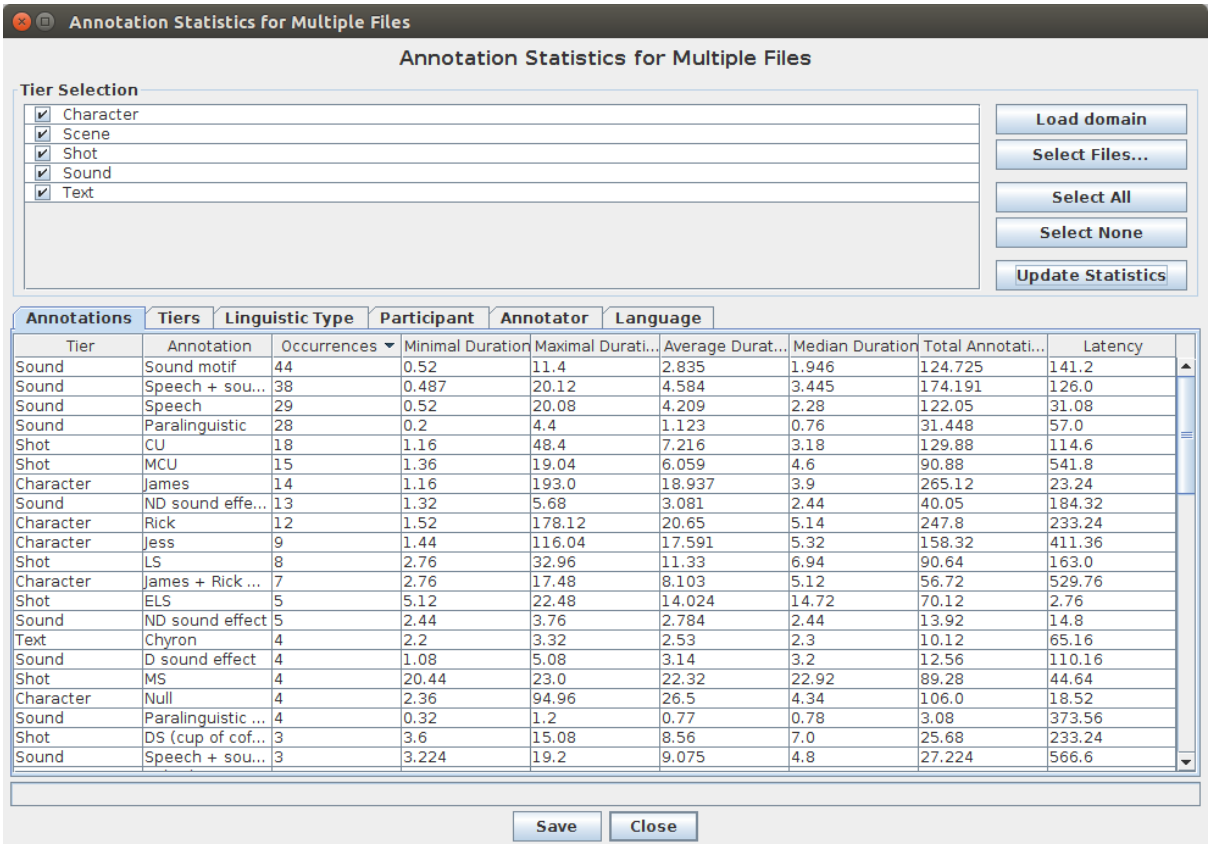


Tier	Number of Files	Number of Ann...	Minimal Duration	Maximal Durati...	Average Durati...	Median Duration	Total Annotati...	Latency
Scene	1	13	0.92	228.4	67.289	20.44	874.76	2.76
Shot	1	66	0.92	98.04	13.254	7.14	874.76	2.76
Sound	1	183	0.2	94.96	3.679	2.16	673.348	14.8
Character	1	50	1.16	193.0	17.495	4.8	874.76	2.76
Text	1	11	2.2	88.8	11.367	3.36	125.04	18.52

Using "Multiple File Processing" on EN.eaf file (Tiers view)

If we switch to the "Annotations" view (select the Annotations tab) we get the same information for each annotation:

[4] Use the "Load domain" or "Select Files..." button to define the scope of your query. In this example, we selected the EN.eaf file in the WHW-EN-Pr corpus.



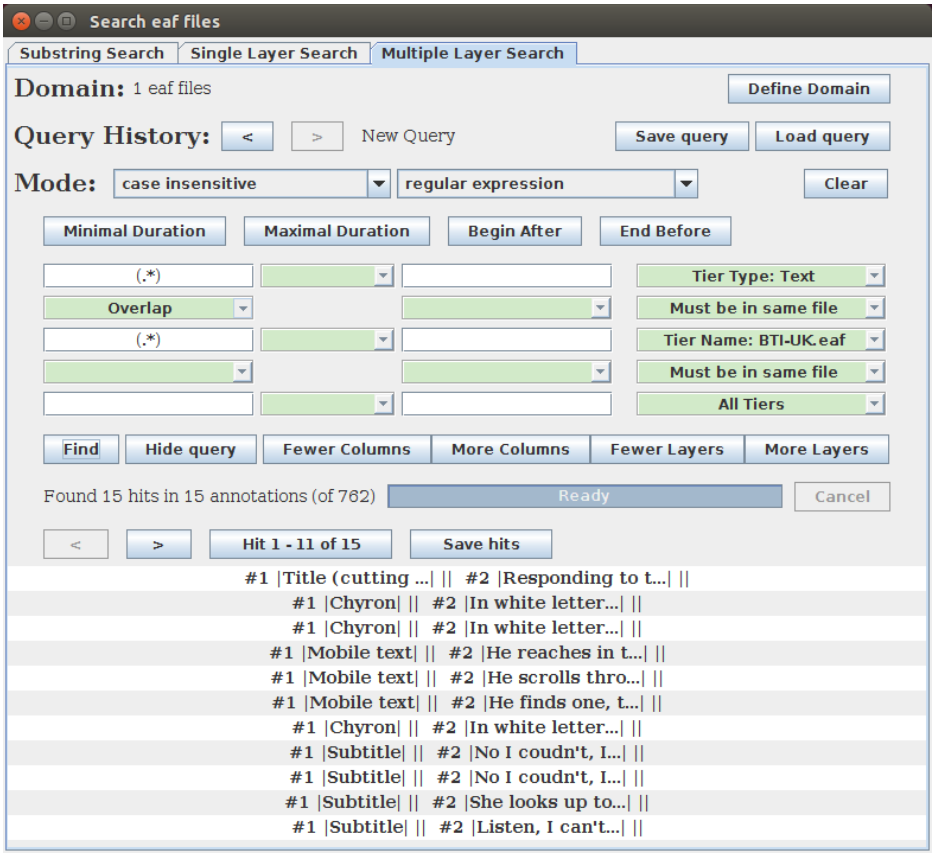
Tier	Annotation	Occurrences	Minimal Duration	Maximal Durati...	Average Durat...	Median Duration	Total Annotati...	Latency
Sound	Sound motif	44	0.52	11.4	2.835	1.946	124.725	141.2
Sound	Speech + sou...	38	0.487	20.12	4.584	3.445	174.191	126.0
Sound	Speech	29	0.52	20.08	4.209	2.28	122.05	31.08
Sound	Paralinguistic	28	0.2	4.4	1.123	0.76	31.448	57.0
Shot	CU	18	1.16	48.4	7.216	3.18	129.88	114.6
Shot	MCU	15	1.36	19.04	6.059	4.6	90.88	541.8
Character	James	14	1.16	193.0	18.937	3.9	265.12	23.24
Sound	ND sound effe...	13	1.32	5.68	3.081	2.44	40.05	184.32
Character	Rick	12	1.52	178.12	20.65	5.14	247.8	233.24
Character	Jess	9	1.44	116.04	17.591	5.32	158.32	411.36
Shot	LS	8	2.76	32.96	11.33	6.94	90.64	163.0
Character	James + Rick ...	7	2.76	17.48	8.103	5.12	56.72	529.76
Shot	ELS	5	5.12	22.48	14.024	14.72	70.12	2.76
Sound	ND sound effect	5	2.44	3.76	2.784	2.44	13.92	14.8
Text	Chyron	4	2.2	3.32	2.53	2.3	10.12	65.16
Sound	D sound effect	4	1.08	5.08	3.14	3.2	12.56	110.16
Shot	MS	4	20.44	23.0	22.32	22.92	89.28	44.64
Character	Null	4	2.36	94.96	26.5	4.34	106.0	18.52
Sound	Paralinguistic ...	4	0.32	1.2	0.77	0.78	3.08	373.56
Shot	DS (cup of cof...	3	3.6	15.08	8.56	7.0	25.68	233.24
Sound	Speech + sou...	3	3.224	19.2	9.075	4.8	27.224	566.6

Using "Multiple File Processing" on EN.eaf file (Annotations view)

We can use the Search tools to get relevant relations between linguistic and filmic tiers. In the following section we list some example queries we may use.

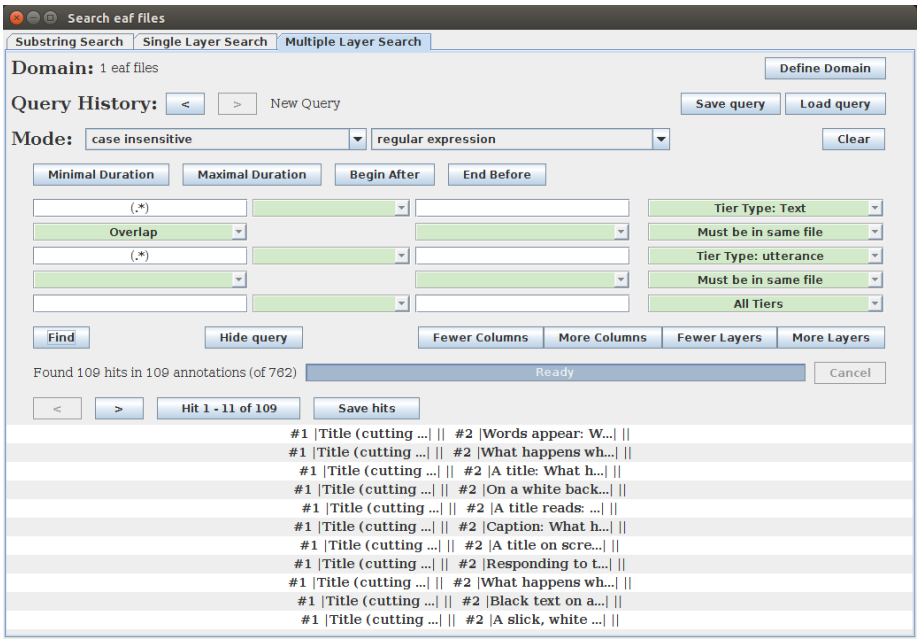
3.4.2.1 Analysing ADs and 'text on screen' simultaneity

We can easily get all AD units overlapping Text annotations in order to build a kind of 'text on screen' sub corpus: select the domain you want (we used the EN.eaf file) and choose "Tier Type: Text" and "Tier Name: BTI-UK" (or whatever provider you want) on the dropdown menus on the right. Be sure to select "regular expression mode" and "Overlap" relation:



Using "Multiple Layer Search" to build a 'text on screen' sub corpus

Note that we can extend the query to all providers by selecting just the "Tier Type: utterance" option as shown below:

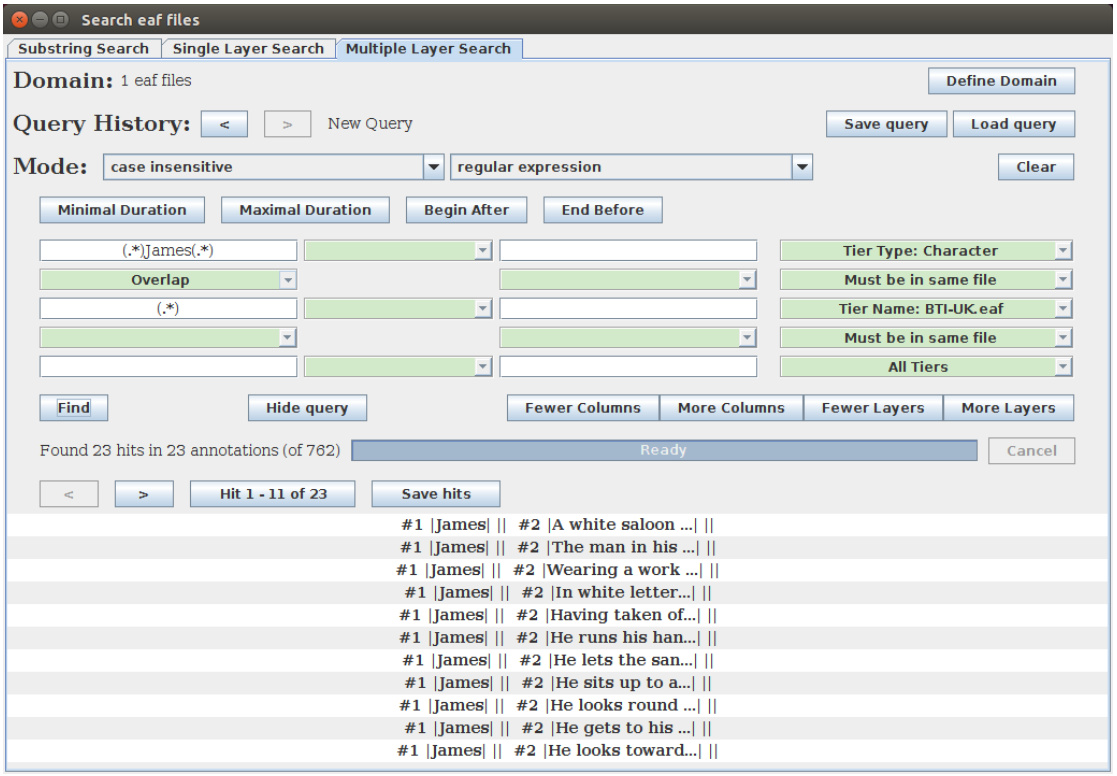


Running the query on all providers

We can make more complex queries by defining a larger domain that includes some provider .eaf files and adding an extra layer such as Lemma or Semantic.

3.4.2.2 Looking for ADs when a given character is on the screen

In the following example we show how to get the ADs produced when a given character is on the screen. In this case we look for occurrences of 'James' in the Character tier that overlap some annotations in the BTI-UK provider. We could run a second query looking for 'Jess' to eventually compare the way the male and female characters are described.



Looking for 'AD / James' overlapping

4 Adding linguistic annotations into .eaf files

Linguistic annotation is performed outside the ELAN tool. Different tools and/or services are used in this process and the resulting annotations are eventually included in the .eaf files. In this section we will see how to linguistically annotate the AD transcriptions and how to load these annotations into the .eaf files.

4.1 AD units or sentences?

Most AD units are paragraphs with two or more sentences. Initially we considered sentences as the 'core unit' for linguistic annotation rather than AD units[5] and, thus, we designed a hierarchy that followed the paragraph/sentence/token structure where the Tokens tier was the child tier of Sentences.

Typically, in ELAN, top tiers are time-aligned (they are assigned a time slot) and dependant tiers may be (i) time-aligned, when the annotation on the parent tier can be subdivided into smaller units, which, in turn, are linked to time intervals or (ii) symbolically linked, when the dependent annotations are not

[5] Note that most NLP tools work at sentence level in the sense that they start with some kind of sentence detection algorithm and then initiate their process using a "for each sentence" loop.

linked to a time interval. In our case, only AD units were assigned a time interval and, consequently, Sentences were defined as a child tier, symbolically linked to AD units[6].

As we will see in section 4.3 Merging linguistic annotations with .eaf files, we used the Pympi library (http://dopefishh.github.io/pympi/Elan.html) to load linguistic data into ELAN files. Crucially, Pympi only allows reference annotations to be added if these have a parent tier with a time slot. The functions are as follows:

```
add_ref_annotation(id_tier, tier2, time, value='', prev=None, svg=None)
```

where ***tier2*** is the tier of the referenced annotation and ***time*** is the time of the referenced annotation. Since, our Sentence tier is symbolically linked to AD unit and has no time intervals, we cannot use Pympi to work with tokens if these are children of sentences:

```
add_ref_annotation(Tokens, Sentences, ???, value='a token', prev=None, svg=None)
```

This led us to use AD-unit (the top, time aligned tier) as the parent of Tokens and explains why we did not follow the 'paragraph/sentence/token' structure but rather the 'paragraph/sentence' + 'paragraph/tokens' approach.

4.2 Annotating ADs with linguistic information

We used the Stanford[7] and the Freeling[8] parsers to add PoS tags to the audio descriptions. In both cases, the parser runs on the sentences.txt file in the corresponding data subdirectory and produces a coNLL file -a tabular file, where the first columns correspond to the word form in the AD and the rest of the columns contain the lemma and part of speech for that word form. Additionally, we use the RDF version of the Multilingual Central Repository[9] to semantically annotate verbs, adjectives and some nouns and adverbs.

4.2.1 Using the Freeling parser (Catalan and Spanish texts)

Catalan and Spanish texts were annotated using the Freeling parser [10]. For this task, we used the web service available at the "Competence Centre IULA-UPF-CC CLARIN"[11]. The web service can be found and executed here:

http://ws04.iula.upf.edu/soaplab2-axis/#morphosintactic_tagging.freeling3_morpho_row

The input files containing the ADs[12] are UTF-8 text files with one AD unit per line. The parser returns a tabular file (following the coNLL format) with one word/token per line together with the morphosyntactic information. This coNLL file will eventually be merged with the .eaf file so that each AD unit in the .eaf file is tokenized and annotated as follows:

- The Tokens tier is split according to the tokenisation performed by the parser.
- The Lemmas and PoS tiers are filled in with the corresponding annotations from the coNLL file.

When merging the .eaf and coNLL files, we need to identify and align the AD units in both files. The .eaf file is an XML file where AD units are clearly marked. Note, however, that the coNLL file has no trace of AD units. The Freeling service performs sentence detection and ignores paragraphs: the service splits input texts into sentences, processes them and, finally, returns the results where sentences are separated by blank lines. This eliminates the initial paragraph organisation[13]. To be able to recover 'paragraph' structure, we decided to manually mark boundaries for the AD units in the sentence.txt file as follows:

---

[6] We rejected the idea of splitting AD units into sentences and manually assigning them the corresponding time intervals.
[7] http://nlp.stanford.edu
[8] http://nlp.lsi.upc.edu/freeling/node/1
[9] http://adimen.si.ehu.es/web/MCR
[10] http://nlp.lsi.upc.edu/freeling/node/1
[11] http://lod.iula.upf.edu/index-en.html
[12] The data/sentences.txt files
[13] Where each paragraph is an AD unit.

- Before the parser is executed we edit the sentences.txt files and add a 'boundary marker': a line with: ###.
    - In vim use: s/^$/\r###.\r/
    - In text editors use: find \n\n and substitute it with \n\n###.\n\n

Once the sentence.txt file is parsed, the corresponding coNLL file includes the 'boundary markers'' which need to be removed before running the merging process. When the manual boundaries are removed, AD units are separated by blank lines. Therefore blank lines can be used as AD unit separators.

To remove 'manual markers' from coNLL files, use the following find/replace commands:

Find	substitute
\n\n	\n
^#(.*)\n	\n

Information about the Spanish and Catalan Freeling tag set can be found here:

https://talp-upc.gitbooks.io/freeling-user-manual/content/tagsets/tagset-es.html

https://talp-upc.gitbooks.io/freeling-user-manual/content/tagsets/tagset-ca.html

4.2.2 Using the Stanford parser (English texts):

Version used: Stanford Lexicalized Parser v3.5.2 - 2015-04-20

Command: $ ./lexparser.sh sentences.txt > stanford.txt

The lexparser.sh script used to run the Stanford parser can be found at the GitHub repository: https://github.com/TransmediaCatalonia/viw-scripts

As in the case of the Freeling parser, input files (sentences.txt files) need to be manually edited before running the parser. Thus the AD unit 'boundary markers' are added as explained before (s/^$/\r###.\r/).

Again, the output coNLL file needs to be edited to remove blank lines and, afterwards, substitute 'manual markers' by blank lines. Once edited, blank lines are the eventual boundary markers.

The tag set used by the Stanford parser includes:

1. CC Coordinating conjunction
2. CD Cardinal number
3. DT Determiner
4. EX Existential there
5. FW Foreign word
6. IN Preposition or subordinating conjunction
7. JJ Adjective
8. JJR Adjective, comparative
9. JJS Adjective, superlative
10. LS List item marker
11. MD Modal
12. NN Noun, singular or mass
13. NNS Noun, plural
14. NNP Proper noun, singular
15. NNPS Proper noun, plural
16. PDT Predeterminer
17. POS Possessive ending
18. PRP Personal pronoun
19. PRP$ Possessive pronoun
20. RB Adverb
21. RBR Adverb, comparative
22. RBS Adverb, superlative
23. RP Particle
24. SYM Symbol
25. TO to
26. UH Interjection
27. VB Verb, base form
28. VBD Verb, past tense
29. VBG Verb, gerund or present participle
30. VBN Verb, past participle
31. VBP Verb, non-3rd person singular present
32. VBZ Verb, 3rd person singular present
33. WDT Wh-determiner
34. WP Wh-pronoun
35. WP$ Possessive wh-pronoun
36. WRB Wh-adverb

4.2.3 Character encoding problems and error checking

Some input files (sentences.txt files) may contain wrong characters. This is due to the fact that the original data (the AD transcripts) were delivered as MS word (or even PDF) documents and the conversion to text files is not always good enough. Especially troublesome are odd apostrophes and quotation marks. Most (NLP) tools need UTF-8 files and do not behave correctly when texts have character-encoding problems.

In Linux, you can use the following command to look for all non-ASCII characters:

```
$ grep --color='auto' -P -n "[\x80-\xFF]" file.txt
```

This will give you the line number, and will highlight non-ascii chars in red.

Note also that, the Stanford and Freeling coNLL files need a careful manual error check before merging them with the .eaf files. We had to allocate more resources than expected to this error checking task.

4.2.4 Multiword expressions

Semantic tagging (see section 4.2.5 Semantic annotation) led us to manually identify multiword units and, consequently, re-encode tokenization. Multiword units are collapsed in a single line where the different units are separated by underscores and assigned a semantic tag as follows:

Form	Lemma	Postag	SemanticClass
alza_la_vista	*alzar_la_vista*	*VMIP3S0*	*Seeing*

Typically, multiword units are those involving English phrasal verbs like the following:

amble_back	*Walking*	gaze_up	*Seeing*
catch_up	*Motion*	get_out	*Motion*
climb_up	*Walking*	get_up	*BodyMotion*
continue_on	*Process*	give_up	*Process*
die_out	*Process*	go_back	*Motion*
drink_up	*Drinking*	hang_out	*BodyMotion*
fish_about	*IntentionalPsychologicalProcess*	hang_out	*Process*
flop_out	*BodyMotion*	hang_up	*Process*
gaze_around	*Seeing*	head_back	*Motion*

Some Catalan multiword expressions:

aixecar_les_espatlles	apartar_la_vista
a_la_llunyania	a_poc_a_poc
a_la_vora_de	a_prop
alçar_els_ulls	apuntar_un_somriure
alçar_la_mirada	arquejar_les_celles
alçar_la_vista	arran_de
amb_quatre_gambades	

Spanish multiword examples:

al_frente	*Location*	alzar_la_vista	*Seeing*
a_lo_lejos	*Location*	arquear_las_cejas	*Gesture*
alrededor_de	*Location*	a_través_de	*Location*
al_tiempo_que	*Time*	bajar_la_mirada	*Seeing*
alzar_la_mirada	*Seeing*	boca_arriba	*PositionalAttribute*

4.2.5 Semantic annotation

Semantic tagging, understood as semantic class assignment, was a challenging task. The lack of a specific tool and the shortage of resources and time led us to take a rather preliminary and pragmatic approach.

Semantic tags were taken from the **Suggested Upper Merged Ontology** (SUMO[14]). Not all words were annotated and, when annotated, not all were encoded using the same fine-grained criteria. Thus, many words were encoded with top level tags in the ontology (such as Process for verbs or Object for nouns).

Many **verbs** were assigned a top node in the ontology (Process) as we focused on a rather small set of verbal semantic classes. We were interested in (i) verbs used to describe the 'spatial' aspects of the scene such as the distribution and movement of things; (ii) communication and 'sensorial' verbs and (iii) verbs used to describe the characters on the screen (both, physically and psychologically):

Semantic class	examples
BodyMotion	(sit, clap, rub, …)
Motion	(leave, go, stop, move)
PositionalAttribute	(stand)
Putting	(leave, put, remove, ...)
Walking	(walk, run)
located	(be)
Communication	(talk, answer, ...)
Gesture	(shake, smile)
Hearing	(listen)
Seeing	(look, look around, glance, peer, stare, …)
Smelling	(smell)
Touching	(tap, hold, scroll, touch)
Dressing	(wear, pot on)
IntentionalPsychologicalProcess	(search, think, scan, locate)
possesses	(have)
StateOfMind	(worry)
Process	(happen, make, finish)

Most **Adjectives** are assigned the generic A-SubjectiveAsssessmentAttribute tag. Here, we focused on identifying 'psychological' adjectives used to describe the mood of the characters and adjectives dealing with hearing and sight. Following SUMO conventions, all adjectival semantic tags use the "A-" prefix.

Note that past participle verbs were semantically annotated as adjectives. This explains why we find verbs (the lemma of the corresponding verb) when listing adjectival semantic classes:

Lemma	SemClass	Frequency
extrañar	A-StateOfMind	24
blanco	A-ColorAttribute	20
negro	A-ColorAttribute	17
sentar	A-PositionalAttribute	14
mismo	A-SubjectiveAssessmentAttribute	9
sobreimprimir	A-FilmLanguage	9
próximo	A-PositionalAttribute	8
rizar	A-SubjectiveAssessmentAttribute	8

Adjectives in
http://transmediacatalonia.uab.cat/web/vocabulary/adjsdash/WHW-ES-Pr

[14] http://www.ontologyportal.org/

For **nouns** we just focused on a rather small subset:

Semantic class	examples
Location	(playa, camino, cielo, izquierda)
BodyPart	(mano, pie, dedo)
StateOfMind	(desencanto, asombro, desconcierto)
Time	(amanecer, minuto)
Object	(vaso, maletín)
Human	(jubilado, estudiante, chica)
Clothing	(camisa, corbata, gafas, bolso)

Finally, for **adverbs** we distinguished Time and Location.

4.2.5.1 WordNet.py

The WordNet.py script reads a coNLL file and sends verbs, nouns and adjectives to http://lodserver.iula.upf.edu/sparql to get the SUMO semantic tag. This SPARQL endpoint[15] contains the RDF version of the Multilingual Central Repository (MCR) EuroWordNet lexicons (version 3.0)[16] which can be consulted here: http://adimen.si.ehu.es/cgi-bin/wei/public/wei.consult.perl?

The script sends a SPARQL query like this:

```
prefix lemon:   <http://lemon-model.net/lemon#>
prefix lexinfo: <http://www.lexinfo.net/ontology/2.0/lexinfo#>

SELECT ?sense ?class FROM <http://ewnES.edu>
                         WHERE {
                         ?entry lemon:canonicalForm ?form ; lexinfo:partOfSpeech lexinfo:verb;
lemon:sense ?sense .
                         ?form lemon:writtenRepresentation "llevar".
                         ?sense <http://lodserver.iula.upf.edu/euroWordNetMCR/sumo_plus> ?class.
                         }
```

Warning: The semantic annotation process was just a first approach and did not include any semantic disambiguation at all. This means it required a careful manual post-editing process for error checking.

4.3 Merging linguistic annotations with .eaf files

The ELAN tool allows the importation of annotations contained in tabular files (see, for example, section 3.2.1 Importing AD) and we initially used this facility to load the linguistic annotations we produced using the NLP tools.

Often, linguistic annotation is an incremental process: new annotations and/or layers are added, old ones are deleted or modified, new criteria is adopted, new tags are used, errors are corrected, etc. Whenever we modify the linguistic annotations we need to update the .eaf files[17]. This means removing old annotations and loading the new ones.

Using the ELAN import facility means a 'one by one' process that in our case involves modifying nearly 50 .eaf files performing a rather long chain of operations, namely: opening the .eaf file; removing all annotations from Tokens Tier; importing *csv* files with new tokens; creating empty annotations in dependent Tiers and, eventually, importing lemmas, PoS and semantic tags.

To facilitate this editing (and/or updating) task, we created some scripts that modify .eaf files outside the ELAN tool. This allows (all) .eaf files to be modified at once and in a single step.

---

[15] http://lodserver.iula.upf.edu/#EuroWordNet
[16] http://adimen.si.ehu.es/web/MCR
[17] Note, we assume that linguistic annotations are modified outside the ELAN tool.

4.3.1 conll2eaf.sh

conll2eaf.sh is a bash script used to pipe tree operations[18]:

tokenise.py | annotate.py | xmllint --format

The script runs the pipe on the set of files matching a given pattern. The pattern needs to be enclosed between quotes. The following command runs the pipe on all .eaf files in any subdirectory of the WHW-CA-Pr directory.

Usage: $ ./conll2eaf.sh '../data/WHW-CA-Pr/*/*.eaf' 'freeling-2.txt'

The script modifies the input .eaf files and makes a backup copy in “originalFile-Source.BAK” files.

Conll2eaf.sh scripts works with UTF-8 files and requires that input .eaf files are non-tokenized files. If the user needs to remove previous annotations she can use the **deleteAnnotationsGlob.py** script.

In the following sections we describe the scripts involved in the pipe.

4.3.2 Tokenize.py

Tokenize.py is a Python script that takes an .eaf file and fills in the Tokens tier with tokens from a coNLL file (Stanford or Freeling files).

We use Pympi (http://dopefishh.github.io/pympi/Elan.html) to load tokens to .eaf files. As we saw in section 4.1 AD units or sentences?, Pympi only allows the addition of annotations that are time-aligned (the AD-unit tier) or reference annotations whose parent tier are time-aligned (the Tokens tier). These excludes dependent tiers that are symbolically linked to the Tokens tier (Lemma, PoS and Semantic tiers) as these have no time slot. To add annotations to these dependent tiers we use the ElementTree XML API in the annotate.py script (see section 4.3.3 Annotate.py).

The script checks that the .eaf file and the coNLL file have the same number of AD units and that the Tokens tier is empty. If Tokens tier is not empty, the system gives a warning and quits. In this case, linguistic annotations can be removed using the **deleteAnnotationsGlob.sh** script.

deleteAnnotationsGlob.py removes annotations Tokens + dependent Tiers of .eaf files matching a given path pattern. The original .eaf file is modified and a .bak file is created.

Example command:

$ python deleteAnnotationsGlob.py "/data/What-CA/*/*.eaf"

Broadly speaking, the tokenise.py script works as follows:

- The script parses a coNLL file and gets the tokens grouped in paragraphs. The coNLL file is split into paragraphs and paragraphs into lines. (Remember, paragraphs are AD units and lines are tokens).
- It gets AD units from the .eaf file.
- Paragraphs and AD units are aligned according to their order of occurrence.
- for each AD unit in the .eaf file:
    - get the corresponding paragraph from the linguistic data and
    - for each line in paragraph create the Token annotation:
        - calculates ANNOTATION_ID
        - calculates ANNOTATION_REF ID
        - calculates PREVIOUS_ANNOTATION
        - prints ANNOTATION_VALUE

[18] The xmllint --format command is used to get a pretty-print version.

Warning: Paragraphs in coNLL files and AD units are aligned according to their order of occurrence. Thus the first paragraph in the coNLL file corresponds to the first AD unit, the second paragraph to the second AD unit, and so on. When reading AD unit using the the Pympi *get_annotation_data_for_tier* command[19], the units come unordered. This is obviously problematic as order is crucial for alignment. The script needs to sort AD units according to their ID before aligning.

4.3.3 Annotate.py

The Annotate.py script adds annotations (lemma, PoS tag and semantic class) to an .eaf file containing tokens. Annotations are read from a conLL file and placed into the corresponding dependent tier (PoS, lemma and semantics Tiers).

usage: $ annotate.py elanFile.eaf conllFile.txt

The output is written in a new .eaf file: *elanFile-Annotated.eaf*.

The script assumes the conLL file is a tabular file with three initial columns with form, lemma, and PoS tag and an additional column with the semantic tag (the semantic tag column needs to be the last one, no matter what the number). If you have any other format, look for "CHECK!!" in the script and configure as necessary.

Warning: The ElementTree XML API used by the script has character encoding problems when returning data as pretty-printed XML. To avoid these problems, we use the "xmllint --format" command on the output file to get an indented version.

The script works as follows:

1. (l*emas, PoS, semclass*) = getAnnotations(*conLL_file*)
2. (*tokens_Id*) = getTokensIds(*Tokens_Tier*)
3. *lastId* = getLastId
4. Foreach *tokens_Id* -> create_dependent_annotations(l*emas, PoStags, semclass*)

Warning: we cannot trust the HEADER/PROPERTY[@NAME='lastUsedAnnotationId in .eaf file as this is not updated by Pympi. We rely on the Tokens tier: we annotate after tokenizing, this means that the last ID in Tokens is the last ID in the .eaf file.

Warning: When editing .eaf files using the ElementTree XML API the original order of elements is not preserved. The resulting files can be correctly loaded into the ELAN tool and they seem to work correctly except when running some complex queries. To avoid problems, open and save the resulting .eaf file with the ELAN tool (you don't need to edit anything). This will correctly reorder the elements (when saving files, ELAN orders the elements following the ID numbering).

4.3.4 eaf2conll.py

The .eaf2conll.py script allows exportation of the linguistic annotations in the .eaf files into a tabular file (following coNLL format: form, lemma, PoStag, semantics).

This file can be used in other applications (CQP) and can be modified (for example when doing manual corrections or when adding new semantic annotations). Once modified, we can update the source .eaf file by using the conll2eaf.sh chain we saw before.

4.3.4.1 Working with coNLL files

Even though ELAN includes nice search and editing functionalities, we made extensive use of Linux commands (grep, awk and sed) to analyse, check and edit coNLL files. Here we list some example commands we used:

[19] units = eaf.get_annotation_data_for_tier('AD unit')

- List and count all adjectival semantic classes in the English corpus:

$ grep 'JJ' data/WHW-EN-Pr/*/data/stanford.txt | awk '{print $4;}' | sort | uniq -c | sort -n

```
1 -
1 A-Clothing
3 A-Human
3 A-Vegetable
6 A-Dressing
8 A-BodyPart
8 A-RadiatingSound
12 A-RadiatingLight
21 A-StateOfMind
43 A-PositionalAttribute
121 A-ColorAttribute
262 A-SubjectiveAssessmentAttribute
```

- List and count all verb lemmas together with their semantic class in the English corpus:

$ grep 'VB' data/WHW-EN-Pr/*/data/stanford.txt | awk '{print $2,$4;}' | sort | uniq -c | sort -nr

```
67 be Aux
47 look Seeing
30 wear Dressing
28 talk Communication
26 walk Walking
23 look_around Seeing
18 sit BodyMotion
16 shake Gesture
16 happen Process
16 frown Gesture
15 have possesses
14 walk_along Walking
...
```

- Find and count all multiword units in the English corpus:

$ grep '_' data/WHW-EN-Pr/*/data/stanford.txt | awk '{print $2;}' | sort | uniq -c | sort -nr

```
23 look_around
14 walk_along
14 put_on
11 look_up
8 get_up
8 get_out
8 craig_anderson
7 sit_up
7 look_about
7 lie_back
6 pull_up
…
```

- Add Human semantic class to all occurrences of “james”

$ sed -i 's/\tjames\t-/\tjames\tHuman/' data/WHW-EN-Pr/*/data/stanford.txt

This will substitute all occurrences of 'james -' with 'james Human'

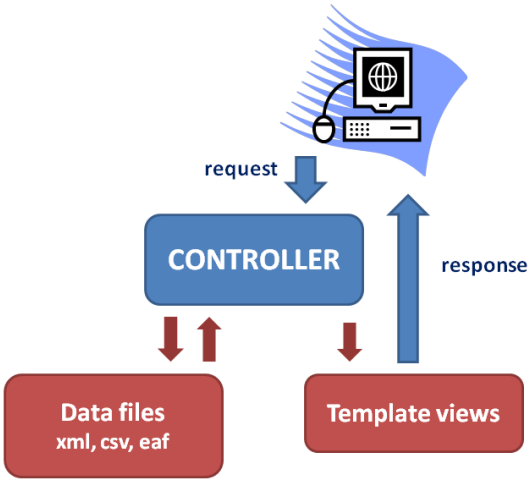
5 The web app

One of the goals of the VIW project was the deployment of a web application that allow easy access to the data generated by the project (http://transmediacatalonia.uab.cat/web/)

The web application was deployed using Symfony (http://symfony.com/) and uses the hosting services offered by the UAB. Due to the *php* version restrictions of the hosting service, the web app was deployed using Symfony version 2.8.

All code and data can be found at GitHub https://github.com/TransmediaCatalonia/viw-project.

The web app has no database and it was designed as a data browser and visualisation service. All relevant data are located in the root 'data' directory and were exported from the ELAN tool. The application includes different controllers which take the data files and display them using chart APIs (mostly Google chart tools). In most cases the system allows downloading of the data source as *csv* files:

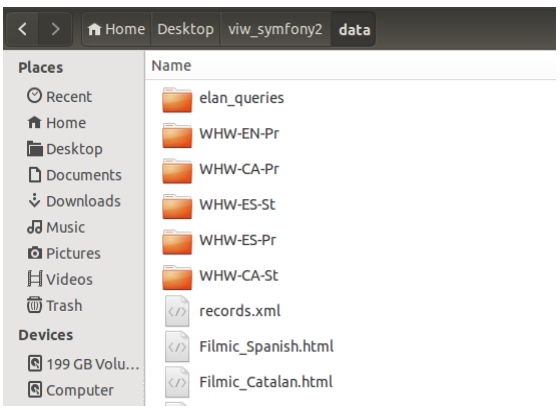


The web app

In the remainder of this section we describe the 'data directory' and the controllers.

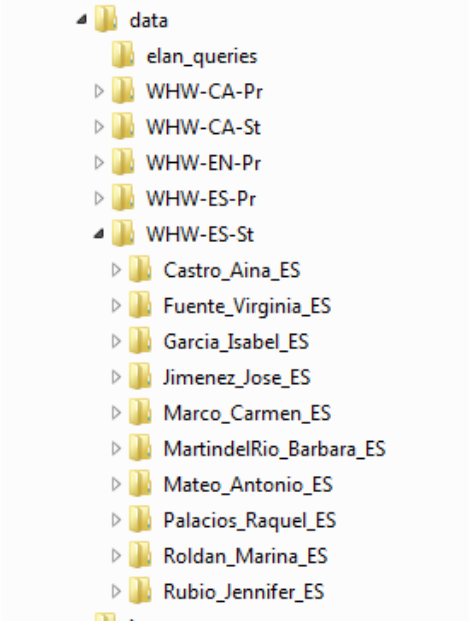
5.1 The data directory

The 'data' directory is a root directory that contains (i) the records.xml file which works as an index file (ii) one subdirectory for each corpus (the project currently includes five different corpora), (iii) a directory with some sample ELAN queries and (iv) a number of .eaf/html files with the filmic annotations (see section 3.1.2 Filmic tiers). Here you can see the partial screenshot we get when listing the current 'data' directory:



The data directory

Each corpus subdirectory collects the data relevant to a particular corpus. These include a small set of data files and a number of subdirectories containing the data for each provider.



The data/corpus/provider hierarchy

5.1.1 The records.xml file

The records.xml file is used to list the contents of the corpus and to provide associated metadata. The file substitutes the database: it contains and describes all relevant elements in the web application.

Records.xml includes two different elements, ‘corpus’ and ‘record’, which are used to describe the corpora and providers respectively.

Corpus example:

```
<corpus id="WHW-EN-Pr">                  = corpus directory name
  <title>WHW-EN-Pr</title>
  <description>...</description>
  <language>EN</language>
  <identifier>https://ddd.uab.cat/record/147267</identifier>
</corpus>
```

Record example:

```
<record id="BTI-UK">                     = provider directory name
  <title>BTI-UK version of "What Happens While"</title>
  <creator>BTI</creator>
  <language>EN-UK</language>
  <language>EN</language>
  <source>WHW-EN-Pr/BTI-UK</source>      = path to provider directory
  <movie>"What Happens While"</movie>
  <corpus>WHW-EN-Pr</corpus>             = some corpus name
  <identifier>...</identifier>
  <expertise>professional</expertise>
</record>
```

New corpora and records can be added or modified as required. The web application will read the records.xml file and include the new data in the system. When adding new data (corpora or providers) some rules must be followed as data is organised as follows:

- **Corpora** are distributed in directories and described by *corpus* elements in the records.xml file as follows:
    - ID: the *corpus/@id* attribute in the records.xml file must be the same as the corpus directory name.
    - Title: any title (can be modified)
    - Description: any description (can be modified)
    - Language: any language (can be modified)
    - Identifier: link to the UAB Digital Repository where all data is stored.
- Corpus directories contain subdirectories with data from providers
- Each **provider** is described in a *record* element in the records.xml file. *Record* elements work as follows:
    - ID: the *record@id* attribute must be the same as the name of the provider's subdirectory (data/corpus/provider)
    - Title: any title (can be modified)
    - Language: any language (can be modified)
    - Source: the exact path to the provider's directory
    - Movie: any string (can be modified)
    - Corpus: the corpus the record belongs to (needs to be the same as some *corpus@id*)
    - Identifier: link to the UAB service where the corresponding movie is hosted
    - Expertise: any string (can be modified)

5.1.2 Corpus data files

Each corpus directory contains few data files which are used by the Controllers. Typically, Controllers read data files, perform some calculus and return data in some graphical shape. The following table summarises the information about corpus files which are further described in the following sections.

File	Controller	Description/Main task
Hits-all.txt	Hits controller Kwic controller	Statistics on AD units (usually linked to the timeline)
sem.csv	Vocabulary controller	Statistics on words and their semantic class
Filmic-Hits.txt	Hits controller	
countWords.txt	Hits controller	Statistics on AD units
sentences.txt		(Used to compute countWords)
similarity.csv	Similarity	A file with text similarity measures among files in the corpus.
.eaf .pfsx		The .eaf (and associated files) of the corpus.

Files in corpus directories

5.1.2.1 Hits-All.txt

Hits-All.txt file is a tabular file that contains all ADs in a corpus together with the begin/end time, duration and provider information as shown below:

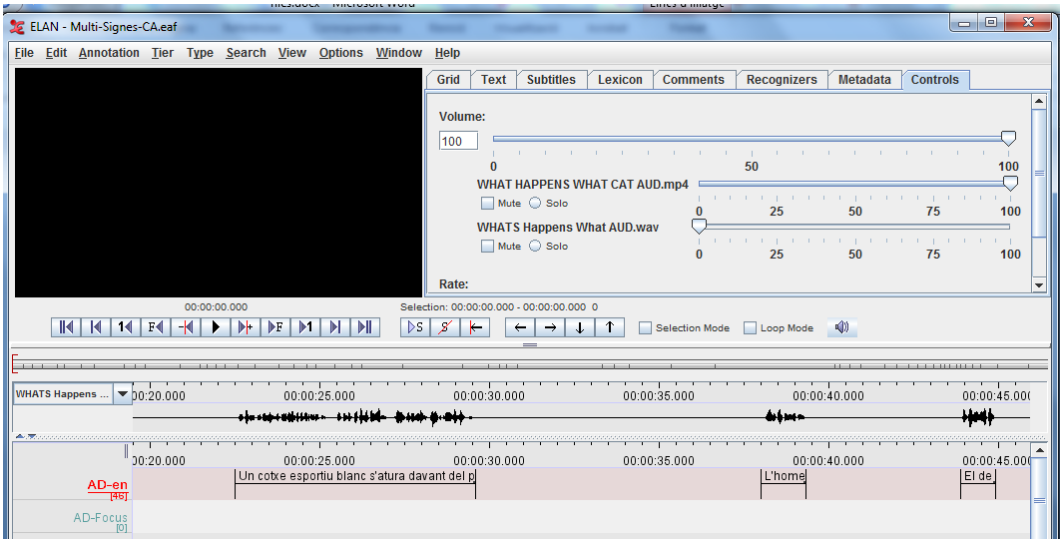
```
4010    8029    4019    En una playa un hombre …        Castro_Aina_ES.eaf
29210   31040   1830    Un coche aparca delante …       Castro_Aina_ES.eaf
…
```

This file is used by the Kwic and Hits Controllers as follows:

- Kiwc Controller: the script searches for a string in the file and displays matching AD units together with the link to the source file. If the user clicks the link, the search focuses on that particular provider.

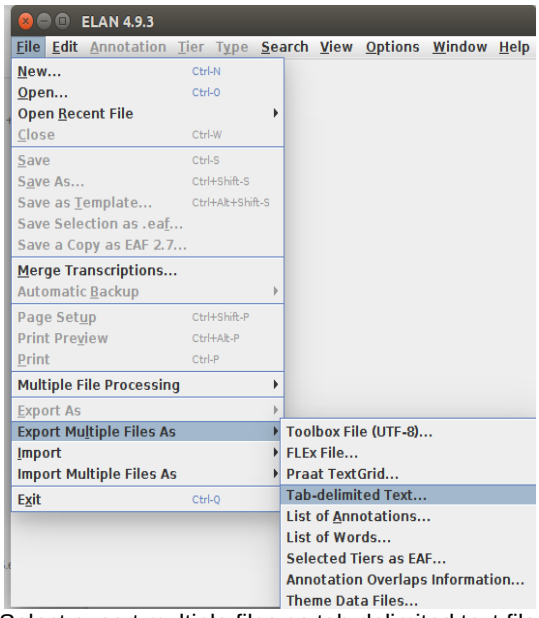
- Hits Controller: The Controllers read the Hits-All.txt file, perform different statistical calculations and display the results in a graph.

To produce the Hits-All.txt file you need to export data as a tab delimited text file using the “Export Multiple Files As” option. Then you have to select the Domain (corpus) you want to export. You should have one Domain for each corpus. See section 0

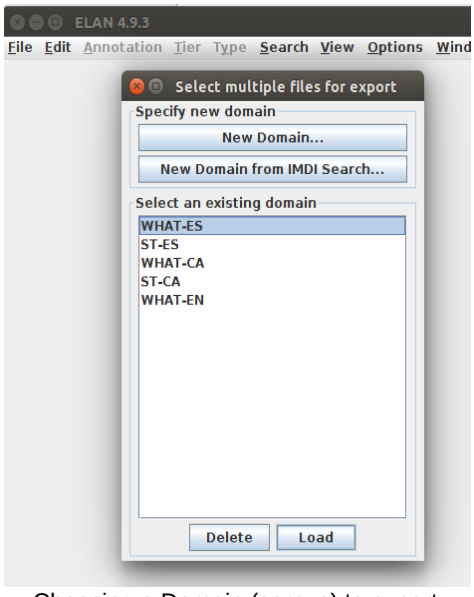


Using the waveform to edit time intervals

Using ELAN Domains as corpora to find out about ELAN Domains.

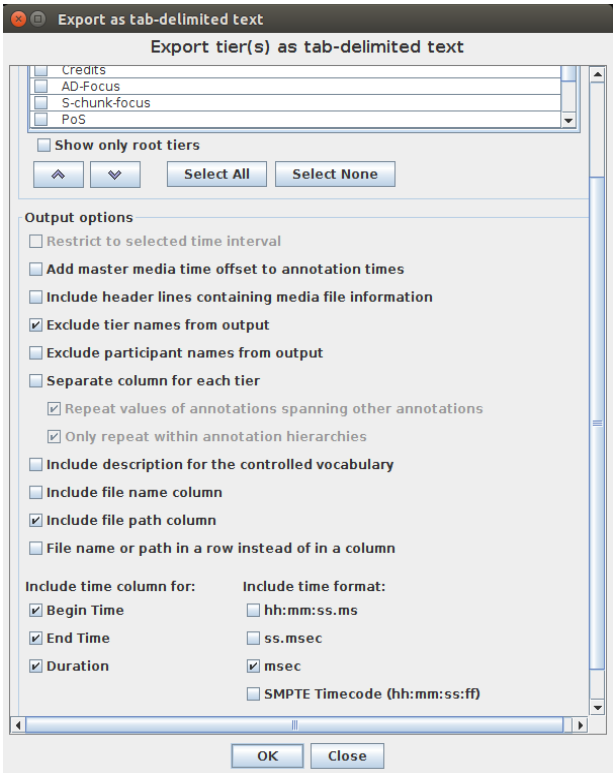


Select export multiple files as tab-delimited text file



Choosing a Domain (corpus) to export

In the new window, select the AD-unit tier and mark the fields as shown in the following image:



Export data as Tab-delimited text

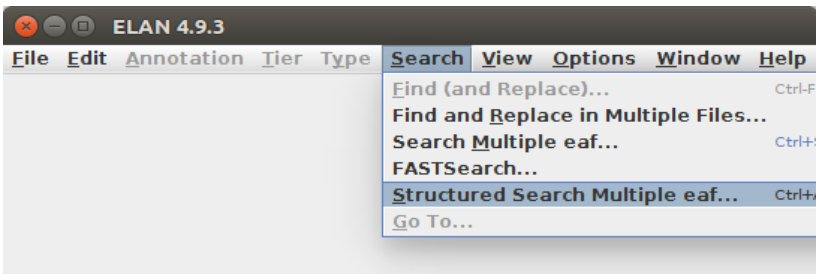
Finally, save the results as a Hits-All.txt file.

Warning: Before using the Hits-All.txt file you have to edit it as follows:

- Remove the empty column at the beginning of each line. For some reason, the ELAN tool adds an extra empty column that needs to be removed (s/^\t//).
- Add a blank like at the beginning of the file.

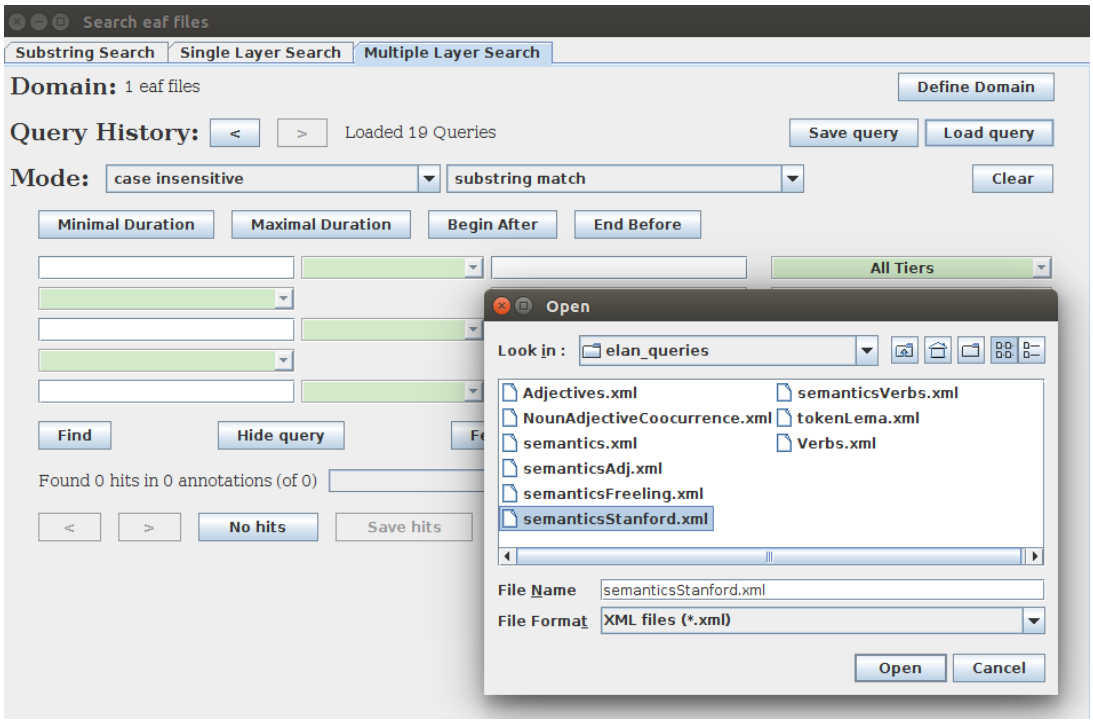
5.1.2.2 Sem.csv

Sem.csv is a comma-separated value file that contains all nouns, verbs, adjectives and adverbs in the corpus together with their semantic class, time information and the file (provider) they occur in. The file is the result of executing a 'structured multiple search' in the ELAN tool as follows:



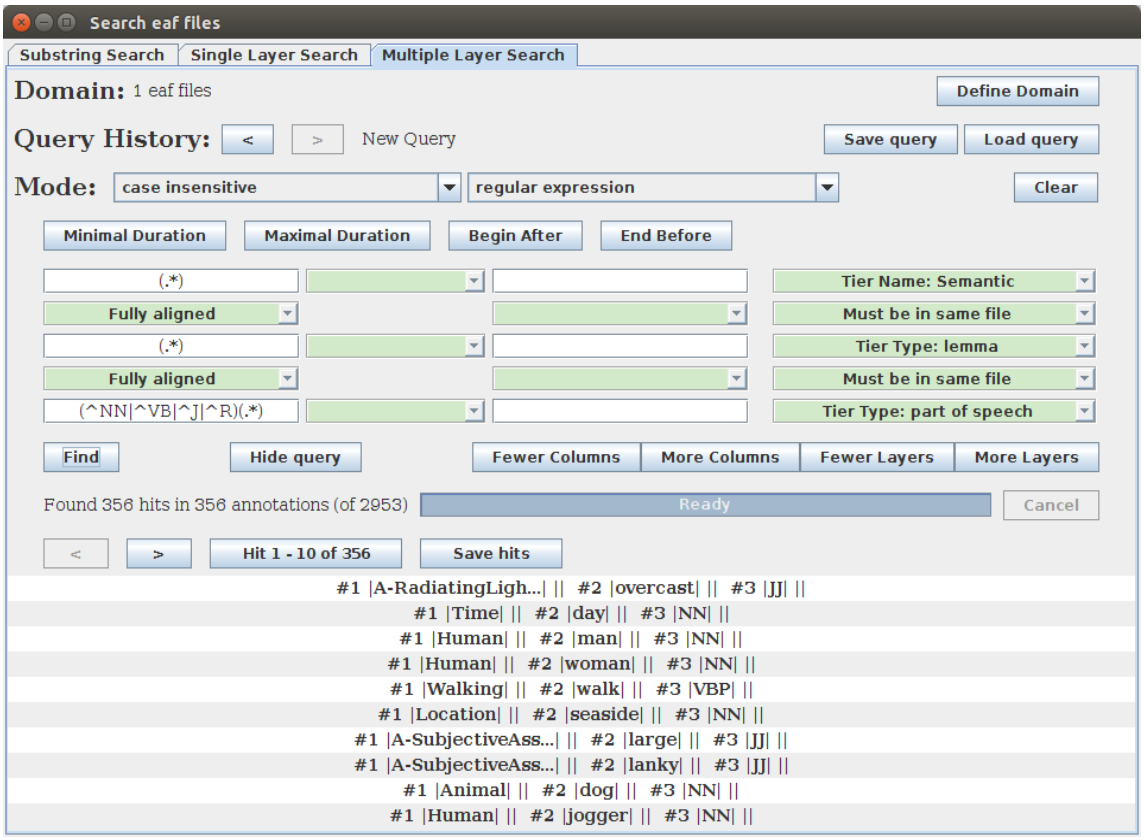
Select Structured Search Multiple .eaf files

A new pop-up window will open. Select the "Multiple Layer Search" tab and press the 'Define Domain' button to select the Domain (i.e. corpus) you want to work with. Once the Domain/corpus is selected, you can load the query by pressing the "Load Query" button and selecting the query. You can find the queries in the data/elan-queries directory of the web application. Use the semanticsFreeling.xml file for Spanish and Catalan corpora and the semanticsStanford.xml file for English corpora.



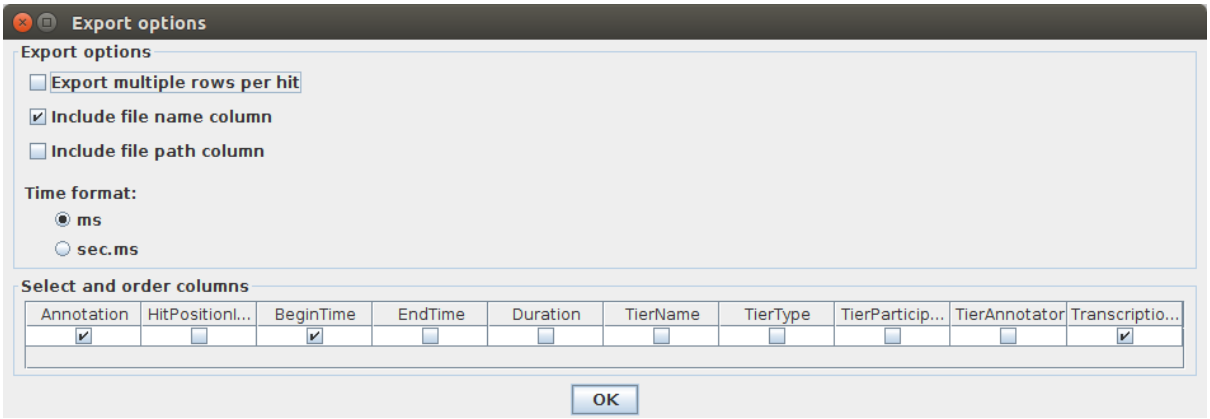
Select and load the query

When the query is loaded hit the 'Find' button. You will get something like this:



Running the semanticsStanford.xml query

To save the results, click the “Save hits” button and, in the new pop-up window, check the export options as follows:



Saving the results, export options

A new file will be created. It includes an initial header line and set of lines with the results. Although the ELAN tool assigns the *csv* extension to the new generated file, this is a tabular file. You have to edit the file and substitute all tabs for commas.

Warning: Finally, the sem.csv file is a UTF-8 Unicode with BOM text[20]. This BOM feature causes many problems to *php* scripts using the *str_getcsv* function. Before using any sem.csv file check that the files have no BOM (you may run the Linux file command to check this). You can remove BOM by running the following *awk* command in Linux:

```
$ awk '{if(NR==1)sub(/^\xef\xbb\xbf/,"");print}' old.csv  > new.csv
```

sem.csv sample:

```
"Annotation1-1","BeginTime","Annotation2-1","BeginTime","Annotation3-1","BeginTime","TranscriptionName"
A-RadiatingLight,4350,partly-cloudy,4350,JJ,4350,"ADAssociates-US.eaf"
Location,4645,sky,4645,NN,4645,"ADAssociates-US.eaf"
A-SubjectiveAssessmentAttribute,5530,wide,5530,JJ,5530,"ADAssociates-US.eaf"
A-SubjectiveAssessmentAttribute,6120,sandy,6120,JJ,6120,"ADAssociates-US.eaf"
Location,6415,beachfront,6415,NN,6415,"ADAssociates-US.eaf"
Location,7005,parking,7005,NN,7005,"ADAssociates-US.eaf"
Location,7300,area,7300,NN,7300,"ADAssociates-US.eaf"
```

5.1.2.3 countWords.txt

All corpus and provider directories include a countWords.txt file. This contains statistical information about ADs and it is used by the Hits controller as in

Counting corpus: http://transmediacatalonia.uab.cat/web/words/corpus/WHW-ES-Pr/countWords.txt

Counting provider: http://transmediacatalonia.uab.cat/web/words/WHW-EN-Pr/BTI-UK/countWords.txt

To generate a new countWords.txt file you need the CountWords.py script (https://github.com/TransmediaCatalonia/viw-scripts) and to run the following command:

```
$ CountWords.py sentences.txt > countWords.txt
```

[20] When running the Linux file command we get: “sem.csv: UTF-8 Unicode (with BOM) text”

5.1.2.4 Sentences.txt

Sentences,txt files in corpus directories include all AD units in a particular corpus without any other information. The file is only used to generate the countWords.txt file described before and to compute text similarity.

5.1.2.5 Similarity.csv

The similarity.csv file contains text similarity scores for all AD files in one language. Text similarity was computed using **Ted Pedersen's Text-Similarity** module[21]. It measures the similarity of two documents based on the number of shared words scaled by the lengths of the files.

Text similarity computes the F-Measure, the Dice Coefficient, the Cosine, and the Lesk measure.

The similarity.csv file includes the results we get when running the command (Dice Coefficient):

$ perl ./bin/text_similarity.pl --type Text::Similarity::Overlaps FILE1 FILE2

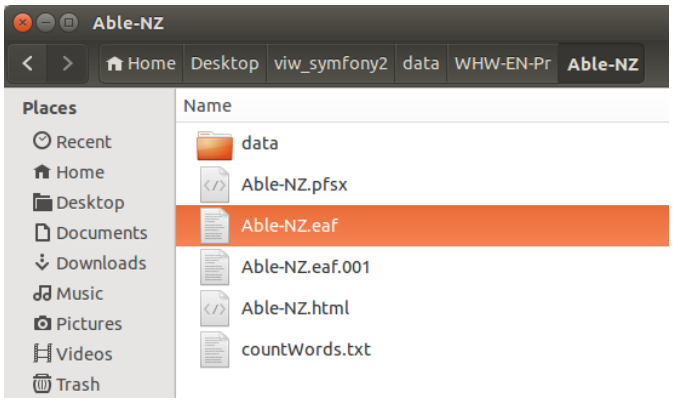
We used the **Ted Pedersen's Text-Similarity** service available at the "Competence Centre IULA-UPF-CC CLARIN":

http://ws04.iula.upf.edu/soaplab2-axis/#statistics_analysis.text_similarity_row

Results are collected in a table where column headers and rows headers are the same. An example can be found here: http://transmediacatalonia.uab.cat/web/similarity/WHW-EN-Pr

5.1.3 Provider data files

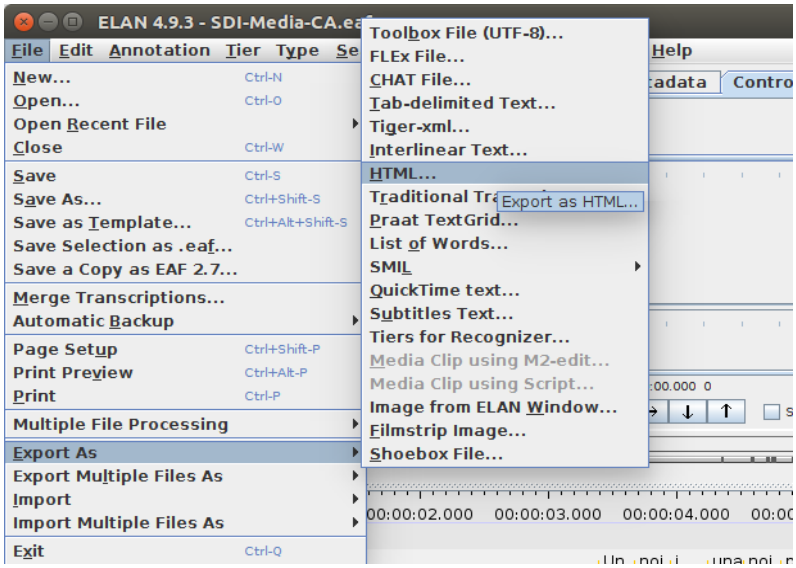
Each provider directory contains the data relevant to a specific provider. A screenshot of a sample provider directory listing is shown below:



5.1.3.1 provider.html

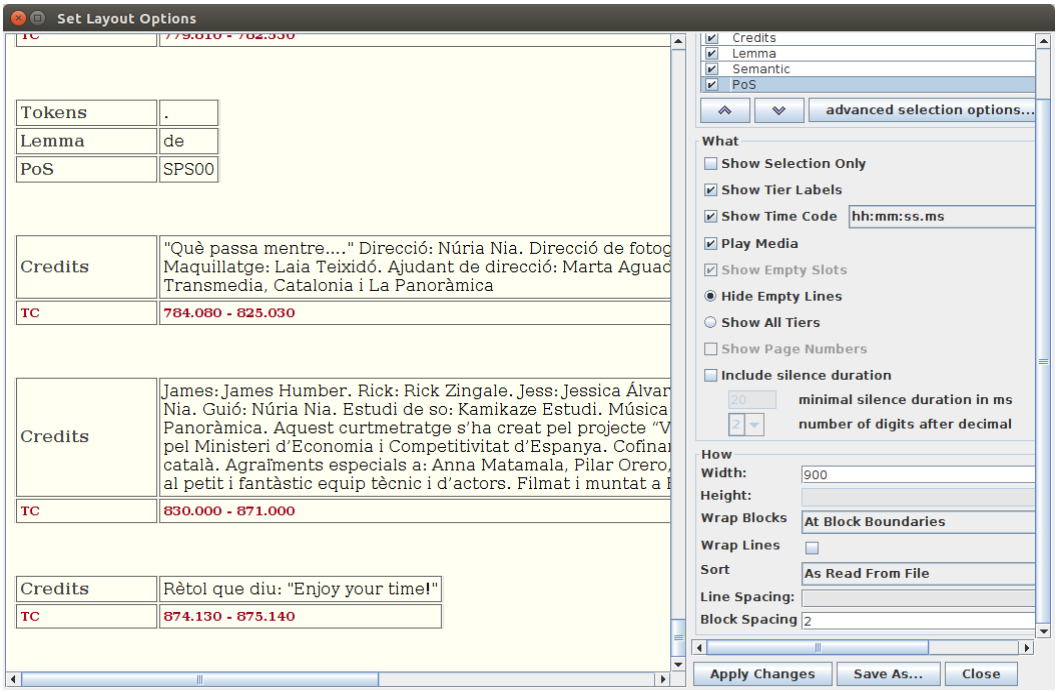
Each provider directory contains the html version of the provider .eaf file. To produce an html version file you have to load the .eaf file into the ELAN tool and choose the "Export as HTML" option as shown below:

[21] http://www.d.umn.edu/~tpederse/text-similarity.html



Exporting .eaf files as HTML files

In the new window select all tiers available and check the fields as follows:



This will generate an HTML version of the current .eaf file. The new file needs to be edited as follows:

- Look for a line containing “ <audio”
- Replace the line with <video”
- Edit the video element as required. (you can check other ‘html’ files).

Warning: To save space and traffic in the web application hosting server, we use the video files hosted in the digital repository of the UAB.

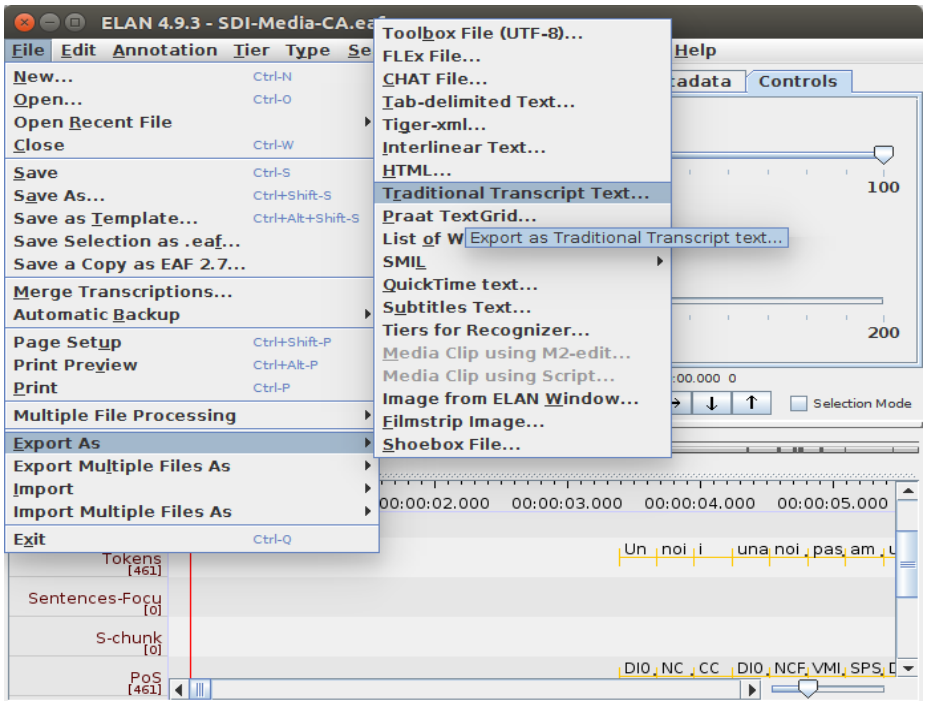
Any other HTML version/format can be produced if desired.

5.1.3.2 data/sentences.txt

Each provider directory contains a ‘data’ subdirectory with a sentence.txt file which is used by:

- the NLP tools (see section 4.2 Annotating ADs with linguistic information)
- the Kwic Controller when searching for words in a specific provider
- The countWords.py script to generate the countWords,txt file

To generate a sentence.txt file, you need to load an .eaf file and follow select the “Export and traditional transcript text” option:



Exporting ADs as sentences.txt

In the new window, mark the AD-unit tier and uncheck all fields.

5.2 Routes and controllers

In the following table we list all routes in the application. For each route we give:

- The Controller involved.
- The file, if any, used by the controller (all data files are somewhere in the data directory as explained in section 5.1 The data directory)
- The scope: indicates whether the route applies on a corpus or on a provider
- The additional output: indicates whether the route allows for data downloading (as *csv* file) or produces an image view of the current page (png)

To get a list of all routes in Symfony 2 run the following command in the root directory:

$ ./app/console *router:debug*

Route	Controller	data file	scope	out
.../	Default	records.xml	corpus	
.../corpus/WHW-EN-Pr/	Default	records.xml	corpus	
.../metadata/WHW-EN-Pr/Able-NZ	Default	records.xml	file	
.../density	Density	records.xml	corpus	
.../density/graph/Able-NZ/Bridge-US	Density	Hits-All.txt	corpus	png
.../hits/time/WHW-EN-Pr/Able-NZ	Hits	Hits-All.txt	file	png
.../hits/timevisual/WHW-EN-Pr/Able-NZ	Hits	Hits-All.txt	file	png
.../hits/words/WHW-EN-Pr/Able-NZ	Hits	Hits-All.txt	file	png
.../hits/wordsvisual/WHW-EN-Pr/Able-NZ	Hits	Hits-All.txt	file	png
.../hits/timewords/WHW-EN-Pr/Able-NZ	Hits	Hits-All.txt	file	png
.../hits/timewordsvisual/WHW-EN-Pr/Able-NZ	Hits	Hits-All.txt	file	png
.../words/corpus/WHW-EN-Pr/countWords.txt	Hits	countWords.txt	corpus	-
.../words/WHW-EN-Pr/Able-NZ/countWords.txt	Hits	countWords.txt	file	-
.../hits/timeline/WHW-EN-Pr	Hits	Hits-All.txt	corpus	-
.../hits/timelinejs/WHW-EN-Pr	Hits	Hits-All.txt	corpus	-
.../kwic	Kwic	records.xml	corpus	-
.../kwic/WHW-EN-Pr	Kwic	Hits-All.txt	corpus	-
.../kwic/corpus/WHW-EN-Pr/Able-NZ	Kwic	sentences.txt	file	-
.../search	Search	records.xml	corpus	-
.../search/language/EN	Search	records.xml	corpus	-
.../show/WHW-EN-Pr/BTI-UK/BTI-UK.html	Search	*eaf*/html	both	-
.../similarity/WHW-EN-Pr	Similarity	similarity.csv	corpus	-
.../vocabulary/pos/WHW-EN-Pr	Vocabulary	sem.csv	corpus	csv
.../vocabulary/verbs/WHW-EN-Pr/Able-NZ.eaf	Vocabulary	sem.csv	file	-
.../vocabulary/verbssem/WHW-EN-Pr/	Vocabulary	sem.csv	corpus	png
.../vocabulary/verbsdash/WHW-EN-Pr	Vocabulary	sem.csv	corpus	csv
.../vocabulary/nounsdash/WHW-EN-Pr	Vocabulary	sem.csv	corpus	csv
.../vocabulary/adjsdash/WHW-EN-Pr	Vocabulary	sem.csv	corpus	csv
.../vocabulary/advsdash/WHW-EN-Pr	Vocabulary	sem.csv	corpus	csv
.../vocabulary/verbsdash/WHW-EN-Pr/Able-NZ.eaf	Vocabulary	sem.csv	file	csv
.../vocabulary/nounsdash/WHW-EN-Pr/Able-NZ.eaf	Vocabulary	sem.csv	file	csv
.../vocabulary/adjsdash/WHW-EN-Pr/Able-NZ.eaf	Vocabulary	sem.csv	file	csv
.../vocabulary/advsdash/WHW-EN-Pr/Able-NZ.eaf	Vocabulary	sem.csv	file	csv
.../vocabulary/posdash/WHW-EN-Pr	Vocabulary	sem.csv	corpus	csv
.../vocabulary/posdash/WHW-EN-Pr/Able-NZ.eaf	Vocabulary	sem.csv	file	csv

5.3 Modifying and deploying the web app

The code of the web application can be found in the GitHub repository at:
https://github.com/TransmediaCatalonia/viw-project

To get a local copy of the repository use:

```
$ git clone https://github.com/TransmediaCatalonia/viw-project.git
```

You can edit the repository (to add new providers and/or corpora, for example) and push the new code. Remember:

```
$ git add --all.
$ git commit -m “some comment documenting your commit”
$ git push -u origin master
```

The repository is prepared to be auto-deployed for every commit in master pushed to the GitHub repository using Travis. This means that for every push, Travis code sends (via ftp) the new code to the UAB hosting server[22].

Warning: The UAB hosting server does not allow SSH connections and data can only be published via ftp. GitHub/Travis allows automatic deployment via ftp.

6 Using the linguistic data with CQPweb

Although ELAN is a powerful tool that includes a complete query system, the linguistic data can be better exploited using a corpus workbench such as the CQPweb (http://cwb.sourceforge.net/cqpweb.php).

We supply all linguistic annotations as CQP files ready to be loaded into CQPweb. These can be found in the GitHub repository at https://github.com/TransmediaCatalonia/viw-scripts.

Additionally, the system provides a script that joins a set of Freeling/Stanford files into one single file ready to be indexed in CQP. This allows to build a CQP corpus out of a set of coNLL files.

6.1 freeling2CQP.sh

freeling2CQP.sh is a shell script that joins coNLL files in one single file ready to be indexed in CQP. Each file is included between 'text' tags where text/@id = file_name and sentences in the coNLL file are enclosed between 's' tags.

CQP file sample:

```
<corpus>
        <text id="filename">
                <s>
                Es    ser    VSIP3S0    Process
                un    uno    DI0MS0
                día    día    NCMS000    TimeInterval
                …
                </s>
        </text>
…
</corpus>
```

Warning: Note that CQPweb does not allow 'special' characters in text/@ids. Check that your texts/@ids do not contain things like _,-,/ etc.

CQPweb is not very easy to install. Fortunately, the developers supply CQPwebInABox: a Virtual Machine PC with CQPweb pre-installed. You can find more info here: http://cwb.sourceforge.net/cqpweb.php#inabox

[22] The code performs some additional actions which we omit here.