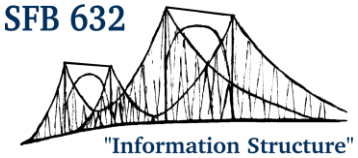


HTML Visualization Guide – Version 1.2.0

(For the latest documentation see also:
http://korpling.github.io/ANNIS/)

title:	ANNIS HTML Visualization Guide
version:	1.2.0
date:	15 September 2016
author:	Amir Zeldes
organization:	Georgetown University
e-mail:	annis@corpus-tools.org
homepage:	http://corpus-tools.org/annis

Contents

1. Introduction

ANNIS is an open source, browser-based search and visualization architecture for multi-layer corpora. The system offers a variety of visualizations tailored to specific types of corpus data, such as constituent trees, dependency trees, coreference, rhetorical structure theory and more.

However, in many cases corpora have unique annotations that cannot be visualized optimally with a generic visualization module. Starting in ANNIS3, we therefore offer the HTML visualizer module, which generates annotation-conditioned HTML elements styled with CSS classes.

2. Visualization Variants and Configuration

This section details how the two variants of the HTML visualizer are constructed and triggered. Each corpus can have an unlimited amount of HTML modules, which consist of two files each and can visualize either entire corpus documents or the immediate search result in context.

Document and Hit Context Visualizers

The two HTML visualizers are called *html* and *htmldoc*. Both visualizers, like all ANNIS visualization modules, are configured in the relANNIS file *resolver_vis_map.tab* to be either available by default (namespace and triggering element are left as NULL) or are triggered by the presence of a specific annotation namespace. See the ANNIS User Guide for more details on configuring *resolver_vis_map.tab* and triggering visualizations.

The *html* visualizer, once triggered, is fed the tokens from the search result and its immediate context (e.g. ±*n* tokens), including all annotations above these tokens. The *htmldoc* visualizer, by contrast, is given access to all tokens and annotations in the document that the search result comes from. Typically, *htmldoc* is used for visualization meant for close reading of text content, whereas *html* is better suited to highlighting or marking smaller annotation structures, where the reading of the running text is less important.

Adding a Visualization to a Corpus

To make a visualization available for a corpus, two files are required, which should be placed in the subfolder *ExtData/* within the relANNIS folder for the corpus. Both files should have the same name (case sensitive), but with different extensions. The configuration file, which specifies which HTML elements are triggered by which annotations, should have the extension *.config*. The Cascading Stylesheet file defining the styles referred to in the configuration file should have the extension *.css*. The common file name (without extensions) should be listed in *resolver_vis_map.tab* in the mappings column (last column) using the syntax: *config: myfile*. The configuration file in this example will be called *myfile.config* and the CSS file will be called *myfile.css*.

3. Building the Visualization

An HTML visualizer is defined by a pair of files: the configuration file (.config) and the stylesheet (.*css*). The same files may be used for *html* and *htmldoc* visualizers.

The .*config* File – Triggering HTML Elements using Annotations

The config file is a simple, tab delimited table with three columns and no headings. Comments may be added after the # sign within a line, or a line may contain only a comment if it begins with #.

The three columns of the configuration file represent the **triggering condition**, the **generated element** and the **content value** (optional). The first column specified an annotation name, value or name and value combination which should be used as a trigger for generating HTML elements, whereas the second column specifies which element should be generated and with what attributes, as soon as the condition is met. The third column specifies what literal content should be written within the element, and can be either a double quoted string literal, the value of the triggering annotation (the reserved string *value*) or the name of that annotation (*anno*). If the value of the annotation is expected to contain characters that could be interpreted as markup (e.g. it contains angle brackets '<' which would be rendered as HTML tags), it is possible to specify **escaped_value** instead of **value**. This will escape HTML entities in annotation values as appropriate.

As a simple example, consider the following configuration file:

```
title     b                            value        #prints the title in bold
chapter   p                                         #surrounds each chapter with a p
chapter   i                            "Kapitel: "  #prints "Kapitel: " in italics
chapter   i                            value        #value of 'chapter' in italics
="God"    span; style="color: red"                  #red where anno value is "God"
pb        span; style="color: grey"    "page "      #grey span with text "page "
pb_n      span; style="color: grey"    value        #grey span with value of 'pb_n'
trans     span:title; style="transl"   value        #value of 'trans' in title attr.
tok       span; style="tokStyle"                    #prints each token
```

The first instruction tell the visualizer to surround any annotation called 'title' with … elements and also to output the value of that title annotation between those elements.

The second instruction surrounds chapter elements with tags, and the third adds the text "Kapitel: " within elements. The fourth instruction adds the value of the 'chapter' annotation. As a result, something like the following might be created:

Kapitel: 1

Note that the double tags are redundant, but have no effect on the visualization in practice.

The instruction ="God" does not specify an annotation name. Any annotation having that value will cause a styled span to surround the specified area. It is equally possible to write lemma="God" to limit this behavior to annotations called *lemma*.

It is also possible to generate attributes inside HTML elements. For example, the trans instruction creates a span with a title attribute using the syntax *span:title*. When an attribute is declared in this way, the third column is interpreted as filling that attribute, rather than the element content, i.e. the value of 'trans' is put into the title element (this is useful for creating a hovering translation tooltip, since title attributes are interpreted as tooltips by most browsers).

As a final instruction, note the declaration of the tok instruction. Tokens are **not** outputted by default. If you wish for the visualization to print out each token, use the tok instruction. In this case each token will be surrounded by a span element, which is given the style tokStyle. This style, and all others, are defined in the accompanying CSS file, which is described in the next section.

Styling the Elements with CSS

The style file is a simple Cascading Style Sheet (CSS file), which may contain any valid CSS instructions. The range of possible CSS instructions depends primarily on browser support and not on ANNIS. Current examples implement features of CSS3, such as selectors and pseudo-elements. For more information see the CSS documentation at http://www.w3.org/Style/CSS/.

Some ANNIS specific points to keep in mind when writing CSS for the HTML visualizer are the following. The HTML visualization is wrapped within a element of the class *htmlvis*. To set things like padding, height or width for the visualizer, add instructions for this element, as shown below. Note that ANNIS does not wrap text in visualizers by default, so *white-space: normal* must be specified and flagged as important to override the inherited setting from the rest of the ANNIS interface.

A second point to keep in mind is that tokens and other textual elements printed out by the visualizer are not separated by spaces by default. This is done, among other things, in order to allow for continuous script visualizations of historical corpora, where spaces between tokens are undesirable. In order to introduce spaces, you may have them printed by special instructions in the *.config* file which produce a string literal space (" "), or you can produce them using CSS pseudo elements, as in the example below. The class *.tokStyle* has an 'after' pseudo element, which is given a single space as 'content'. This results in a space being inserted after each token.

Finally note that CSS allows for powerful interactive features, such as styling behavior while hovering over elements. The *.transl* class below colors the area enclosed by its elements blue when the text is hovered over. Combined with the *title* attribute from the *.config* file in the previous section, this produces a tooltip containing e.g. a translation, and corresponding coloring for the section that is being translated. For some more elaborate examples, see the following section.

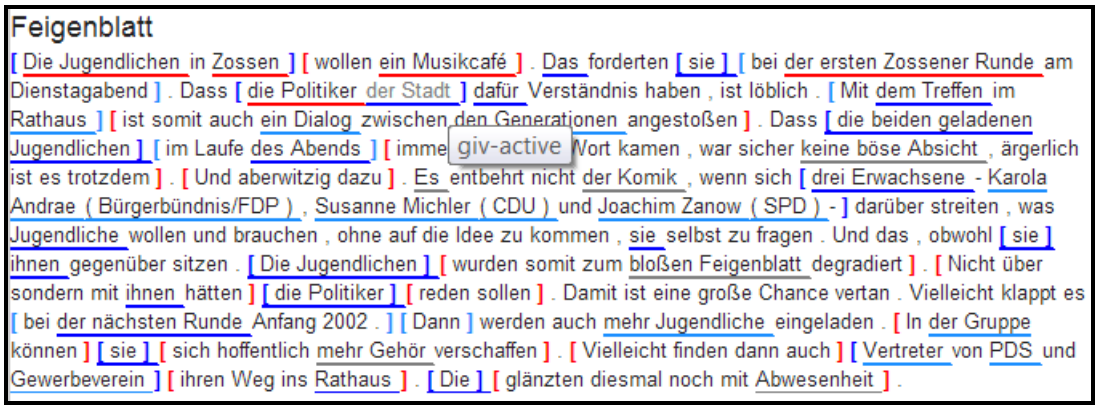
```
div.htmlvis {
width: 500px; white-space: normal !important; margin-top:10px; margin-bottom:10px
}
.tokStyle:after{content: " "}
.transl:hover{color: blue}
```

4. Some Examples

The following examples may be used as full document or hit context visualizations, though the subtitle and manuscript examples are probably better suited to the full document *htmldoc* visualizer. All of the examples are downloadable along with the entire corpus from http://annis-tools.org/corpora.html.

Information Structure Visualization

This visualization was designed by SFB632/D1 (http://www.sfb632.uni-potsdam.de/en/cprojects/d1.html) for the pcc2 corpus (Stede 2004), which can be freely downloaded from the ANNIS website. As seen in the image below, the visualization renders a large title for the text, underlines expressions in shades of red and blue depending on their information status (discourse new, given etc.), and surrounds topic and focus spans with blue and red brackets. A hovering tool tip gives the exact value of the information status annotation (e.g. *giv-active*) and the hovered area is colored in grey.



The *.config* file giving with the triggering instructions for the visualization is given below. Each token is printed using *tok* as an instruction and *value* as its content. The style *tok* is the applied to a surrounding each token. Headings are surrounded by a with the style *heading*. The three information structure annotations Topic, Focus_newInf and Inf-Stat each create a span with an attribute called *topic* or *focus* in the first two cases. The Inf-Stat annotation, by contrast, generates a *title* attribute, which will be rendered by most browsers as a hovering tool tip. The value for all of these attributes is the value of the corresponding annotation, as indicated by the instruction *value*.

```
tok             span; style="tok"               value
Focus_newInf    span:focus; style="focus"       value
Topic           span:topic; style="topic"       value
Inf-Stat        span:title; style="infstat"     value
heading         div; style="heading"
```

The CSS file below is used to render the styles triggered by the *.config* file:

```
div.htmlvis {
width: 500px; white-space: normal !important; margin-top:10px; margin-bottom:10px
}
.heading{display: block; white-space:nowrap; font-size: large;}
.tok:after{content: " "}
.infstat[title*=new]{border-bottom: 2px solid red; border-right: 2px solid white}
.infstat[title*=acc]{border-bottom: 2px solid DodgerBlue; border-right: 2px solid white}
.infstat[title*=giv]{border-bottom: 2px solid blue; border-right: 2px solid white}
.infstat[title*=idiom]{border-bottom: 2px solid grey; border-right: 2px solid white}
.infstat:hover{color: grey}
.focus:before{content: "[ "; color: red; font-weight: bold}
.focus:after{content: "] "; color: red; font-weight: bold}
.topic[topic*=fs]:before{content: "[ "; color: DodgerBlue; font-weight: bold}
.topic[topic*=fs]:after{content: "] "; color: DodgerBlue; font-weight: bold}
.topic[topic*=ab]:before{content: "[ "; color: blue; font-weight: bold}
.topic[topic*=ab]:after{content: "] "; color: blue; font-weight: bold}
```

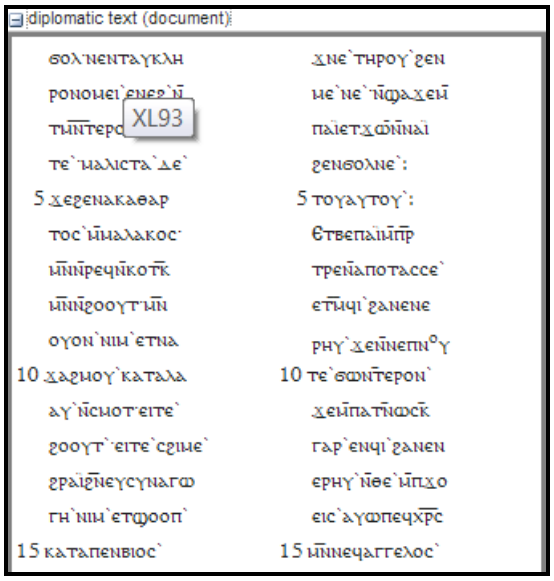
The *div.htmlvis* style gives the settings for the entire visualizer, such as width and margin, and sets the text white-space to wrap normally, e.g. when the window is resized. The *.heading* style makes the heading display as a block in large font (cf. the heading *Feigenblatt* in the image above). The *tok:after* instruction inserts a space after each token, as described in Section 3.

The next block of instructions styles the information status annotations. The title attribute, which contains the value of the annotation Inf-Stat is checked using a CSS selector to see whether the span corresponds to some type of *new*, *acc(essible)*, *giv(en)* or *idiom(atic)* information. Note that the CSS * selector matches substrings, so that *giv-active* matches the selector *=*giv*. Depending on the information status, a different color is chosen for a bottom border for that span, which serves as an underline. For all *.infstat* spans, the hover behavior is defined as changing the color of encompassed text to grey.

A similar set of instructions is applied to focus annotations, except that instead of underlines, pseudo elements 'before' and 'after' place red brackets before and after the annotation span. Finally, the set of topic instructions combines the two strategies above and generates brackets before and after topic annotations, but makes the color depend on a CSS selector which distinguishes framesetter topics (fs) from aboutness topics (ab).

Diplomatic Manuscript View

The manuscript view is suitable for diplomatic editions of historical corpora. It displays text in a two column, page by page view, gives line numbers in multiples of five at the margin of each column, and allows for words broken up across lines or pages. It also renders superscript characters as superscripts. In the example below, the visualization has been applied to a Sahidic Coptic corpus, containing a sermon by the Coptic archimandrite Shenoute (4th-5th century). A tooltip gives the number of the page when hovered over. The visualization and the corpus were developed by the project Coptic SCRIPTORIUM (http://copticscriptorium.org).



The *.config* file is given below. Tokens are simply printed in unstyled spans. This will lead to tokens being serialized without intervening spaces, which is desirable for a diplomatic edition of the manuscript. Only spaces actually present in the corpus will be rendered. Any annotation with the value "superscript" is rendered in a <hi_rend> element with an attribute *rend*. Note that though these names do not correspond to actual HTML element names, they are rendered and styled by the browser nonetheless.

Line annotations using the annotation name *lb* (for line-break) generate elements with the style *line*. The annotation *pb_xml_id*, which gives the page break and page number in the manusacript, creates two elements: a <table> with the style *pb* and a title containing the page number annotation, and a <tr> (table row) element for the page which creates a table row to house the two columns of the manuscript. The table title attribute creates the tooltip in the image above. Finally, each column within a page receives its own table cell, a <td> element, which has the style *cb* (column break).

```
tok              span                      value
="superscript"   hi_rend:rend              value
lb               div; style="line"
pb_xml_id        table:title; style="pb"   value
pb_xml_id        tr
cb               td; style="cb"
```

The CSS file for the visualization is the following:

```
.htmlvis {
font-family: Antinoou, sans-serif; counter-reset: line 0;
}
div.line:nth-of-type(5){text-indent:10px;}
div.line:nth-of-type(5n):before{content:counter(line)" "}
div.line{display: block; white-space:nowrap; counter-increment: line ; height: 22px}
div.line:not(:nth-of-type(5n)){text-indent:20px;}
.pb{border-style:solid;}
.cb{vertical-align: top; counter-reset: line 0; width: 160px;}
hi_rend[rend~=superscript] {vertical-align: super; font-size: 80%}
```

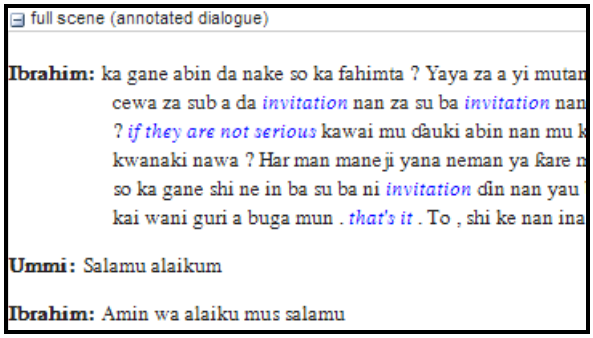
The *.htmlvis* container has been given the font Antinoou, which displays Coptic characters encoded in UTF-8. The font may be embedded in ANNIS for users who do not have the font (see ANNIS User Guide on font embedding). The container is also given a CSS counter called *line*, which will keep track of line numbers and is set to 0 at the beginning of the visualization with the reset instruction.

The next four instructions style the individual line elements. For line 5, which is the only single digit line number, an indent of 10px is added to align all line numbers neatly. All lines in multiples of 5, using the *nth-of-type* CSS selector, receive the line number from the *line* counter. The value of the counter is inserted into a :before pseudo element and followed by a single space " ". The third line instruction set lines to display as blocks, without wrapping, at 22px height each. Whenever a line occurs, the counter *line* is incremented using the CSS *counter-increment* instruction. For lines that are not multiples of 5, a text indent of 20px is added to occupy the space that the line numbering takes up in lines that are multiples of 5.

The two following instructions are responsible for the page and column styling. Pages (*.pb*) are given a solid border, and columns are aligned to the top (in case a column has not been transcribed fully or is empty at the bottom) and given a width of 160px. Whenever a column begins, the line counter is reset to zero, so that lines will be numbered anew in the following column. Finally, the instruction for the 'made up' element <hi_rend> which was generated by the *.config* file checks for a rendering instruction containing the word superscript using the word matching CSS selector ~=. If this value is found in the *rend* attribute, vertical-align is set to superscript and the letter is scaled down to 80% of its size. An example of this rendering can be seen in the image above in line 9 of the second column, just before the end of the line (a superscript 'o').

Script View for Subtitle Corpora

This visualization is meant to convey sequential dialogue from a film, in which speakers do not overlap, i.e. one character speaks after the other. The corpus is a transcription of a Nigerian film in the Hausa language, called Umarnin Uwa ("Mother's Decree"). Code switching and words of foreign origin are annotated in the corpus and highlighted in blue in the visualization. The name of each character is given before their indented dialogue. The corpus was prepared within SFB632 by projects D1 and A5 (see http://www.sfb632.uni-potsdam.de/en/cprojects/).



The *.config* file for the corpus is given below. There are only four instructions generating the visualization:

```
tok     span; style="word"       value
speaker div:speaker; style="spk" value
lang    span:lang; style="lang"  value
info    t:title; style="info"    value
```

Each token is placed in a span with the style *word*. The spans containing the tokens are always annotated with the speaker name, so for every speaker annotation a element is created with an attribute *speaker* containing the speaker name value and the style "spk".[1] If a *lang(uage)* annotation is present, then a span with the attribute *lang* is generated giving the value of the language annotation in the attribute. Finally, some information from *info* annotations, which in this corpus give information about relevant extralinguistic events in the film, are placed in the title attribute of a <t> element, so as to be available as hovering tooltips.

The CSS style sheet for the visualization is given below.

[1] Note: the class name *speaker* as a style should be avoided since the ANNIS style sheet uses this class name itself to mark A/V data alignment in spoken corpora.

```
div.htmlvis {
width: 500px; white-space: normal !important;
}
.word:after{content: " ";}
div.spk{display: block; padding-top: 6px; padding-bottom: 6px; text-indent: -65px;
padding-left: 65px}
div.spk:before{content: attr(speaker) ": "; font-weight:bold}
.lang{color: blue; font-style: italic}
.info:hover{color: red}
```

As in the information structure visualization, white-space has been set to normal wrapping and the width has been fixed for the visualizer. The *.word* style creates the spaces between the tokens with an :after pseudo element, again as in the information structure visualization.

The speaker elements are displayed as indented blocks with a large left padding to make space for the speaker name. Before each speaker , the name is inserted into a :before pseudo element, which is given as content the value of the attribute speaker, followed by ": ", and rendered in bold.

Finally the *.lang* style gives foreign elements of any language blue, italic rendering and the *.info* style colors info spans in red, while the information tooltip is generated by the title attribute created in the *.config* file.

5. Advanced features

Including metadata or constants at the top or bottom

It is possible to include metadata or arbitrary strings in the HTML visualizer. Since metadata doesn't 'happen' at any point in the document, typically you will want to trigger this kind of information to render at the beginning or end of the visualizer frame. To do so, you can use the special instructions *annis:BEGIN* or *annis:END* in the first column of the config file.

The second column of such instructions behaves as usual and can generate any HTML element you like. You can also include multiple BEGIN and END instructions, which are generated in sequence. For the third column of such lines, you may want to incorporate metadata. To do so, use the meta:: prefix, and the name of a metadatum in your document. The example below illustrates this mechanism.

```
annis:BEGIN     span    meta::title
chapter         span    meta::chapter_num
annis:END       span    meta::author_name
```

Using HTML templates

Sometimes you may want to include an annotation value in the visualizer's output, but you'll want to wrap it in a special way. For example, you may want to generate a link containing the value of your annotation as some sub-part of that link. You can do this by declaring constant strings in the third column of the config file, but including the reserved terms %%value%% and %%name%%, which embed the annotation's name and value respectively. For example, you could create a lemma lookup in an external site like this:

```
lemma   div   <a href='http://my_dictionary/%%value%%'>%%value%%</a>
```

This instruction generates a div showing the lemma, but also wrapping the lemma value in a link which contains the lemma itself.

6. Further Information

The possibilities for creating annotation triggered HTML visualizations are highly varied. If you would like to create a new kind of visualization but have trouble getting the module to do what you want, don't hesitate to contact the ANNIS team (see ANNIS website for contact information: http://corpus-tools.org/annis).

In many cases, difficulties are caused by conflicting hierarchies in HTML elements, which arise because of the fact that ANNIS supports SGML style conflicting nesting in annotation spans that are not foreseen in the HTML standard. Other problems are caused by the HTML data model, which does not accept the nesting of certain elements within other elements. Often experimentation can lead to a solution, though some things remain impossible with the current technology. We are constantly working to improve and expand ANNIS – if you have a request for features not covered in the current release or have found a bug, please use our issue tracker and submit your feature request or bug report at: https://github.com/korpling/ANNIS/issues.

References

Stede, M. 2004. The Potsdam Commentary Corpus. In Webber, B./Byron, D. K. (eds.) *Proceeding of the ACL-04 Workshop on Discourse Annotation*. Barcelona, Spain, 96–102.