(DICTIONARY AND NAMED ENTITIES) ATTESTATION TOOL

Author: Mathieu Fannee, Tom Kenter. INL Internal Review: Katrien Depuydt – December 2013.

TOOL DESCRIPTION

The Attestation Tool is a multi purpose GUI used in the production of computational lexica and gold standard data for NE tagging. It addresses the situation of having a comprehensive dictionary consisting of headwords (lemmata) and corresponding quotations in which occurrences of the headwords were automatically matched. This tool is meant to manually evaluate and correct those occurrences of the headwords, especially in cases where occurrences were faultily or yet partially matched. It can also be used to manually correct automatically executed tagging of Named Entities in texts.

In terms of workflow, working with the Attestation Tool will mean:

[1] Loading either dictionary data or text data data into the tool,
[2] Using the tool to process the data,
[3] Finally exporting the result of your work in the same format as it came in.[1]

Points 1 and 3 are about importing and exporting data, so working with the Attestation Tool is essentially point number 2. The tool has the form of a web application that can be run from any computer in a local network. It allows several users to work and deliver their input from different computers at the same time.
The tool has been built for speed. It presents information to the users in such a way that quick evaluation is possible. When user actions are needed, especially the frequent ones, those can be performed from the keyboard, since this allows for faster responses than clicking the mouse on screen buttons.
This way, when the automatic matching has worked out reasonably well, users can very easily scan through the results, quickly correct some mishaps and hit the spacebar to get the next lemma.

TECHNICAL OVERVIEW

The Attestation Tool is based on a LAMP[2] architecture. It is an AJAX application designed for Mozilla Firefox (other browsers may work as well, but are discouraged). It uses a MySQL

[1] At the moment, the only supported format is ALTO-XML. But both the import and export scripts can be adapted to process other (XML) formats.

[2] The acronym LAMP refers to a solution stack of software, usually free and open source software, used to run dynamic Web sites or servers. The original expansion is as follows:
- Linux, referring to the operating system;
- Apache, the Web server;
- MySQL, the database management system (or database server);
- PHP or others, i.e., Perl, Python, the programming languages.
http://en.wikipedia.org/wiki/LAMP_(software_bundle)

database, and is written in PHP, Perl and Javascript.
The interface consists of just one PHP-page: *attestationTool.php*. It is a so-called rich Internet application which means that it uses AJAX to communicate with the database server and display the results.

Users will need a web browser to access the application. We tested the user interface on all Mozilla Firefox versions from 3.0.5 on, running on MS Windows 7 computers.

HARDWARE REQUIREMENTS

The application consists of an application and a database. It is considered good practise to install the two components on separate servers. The first one (which will house the PHP and Perl code) is the application server, and the second one will act as the (MySql) database server.

The tool was originally developed and tested on MySQL 5.0.27, PHP version 5.1.6, and Apache 2.0 on Red Hat Enterprise Linux 5.

As a reference for hardware requirements, here are the specifications of the servers on which the Attestation Tool is developed and maintained at the moment, with good performance as a result:

Application server:
CPU : Intel Xeon e5 2,3 GHz FIO
Mem : 4Gb
Storage : 8Gb iscsi storage
Application-storage : 30 MB
OS : Debian squeeze
Software : Apache 2.2.16 / PHP 5.3.3 / Perl v5.10.1 (*) built for x86_64-linux-gnu-thread-multi

Database server:
OS : Redhat 5
Software MySql : ver 14.12
CPU : Intel Xeon e5 2,3 GHz FIO
Mem : 4Gb
Storage : 60Gb
Appstorage : 700MB

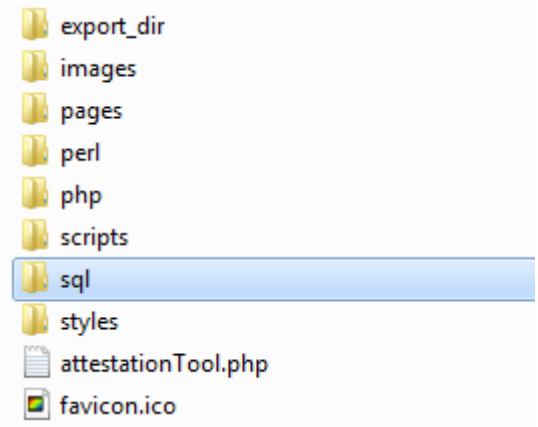
INSTALLATION PACKAGE

The installation package consists of a zip file. The unpacked archive should contains a PHP directory together with some Perl scripts. We will see how to deal with it in the following sections.

MYSQL

CREATE DATABASES

The tool expects a MySQL database to be there with the right table structure.[3] To make sure you have just that, the distribution of the tool comes with some SQL files in the 'sql' directory of the installation:



This directory contains a file called 'emptyAttestationDatabase.sql', which contains the data structure needed for the tool database. This file needs to be loaded into MySQL. Before this can be done, the database needs to be created. This is done in MySQL by running the following query:

```
mysql> CREATE DATABASE myAttestationDatabase;
```

In this queries the red part should be replaced by your own database name.

Now, the SQL file that come with the distribution in the 'sql' directory should be loaded into the database just created. On the command line you can do this by executing the following command:

```
mysql -h hostname -u username -p myAttestationDatabase <
   emptyAttestationDatabase.sql
```

[3] Check the appendix about that.

CREATE A MYSQL USER

When a database has been created, a MySQL user should be added. This can be done by running the following statements:

```
mysql> GRANT ALL ON myAttestationDatabase.* TO 'newUser'@localhost IDENTIFIED BY 'password';

mysql> GRANT ALL ON myAttestationDatabase.* TO 'newUser'@'%' IDENTIFIED BY 'password';

mysql> FLUSH PRIVILEGES;
```

Again, the red parts in the queries should be set to your own desired values.

NOTE that later on you'll have to declare the username and password of the MySQL user in the tool configuration, in such a way that the tool can access the database. We will see that in the 'Tool configuration' section.

FILL THE REVISORS TABLE (REDACTORS)

The entire database can be empty at start up, except the users table which needs at least one row (that is: one revisor entitled to log in). New users can be added by running the following command:

```
mysql> INSERT INTO revisors (name) VALUES ('Amy'), ('Billy'), ('Duffy'), ('Ella');
```

Now Amy, Billy, Duffy and Ella will be able to log in just by typing their names in the Log-in screen of the tool.

FILL THE TYPES TABLE

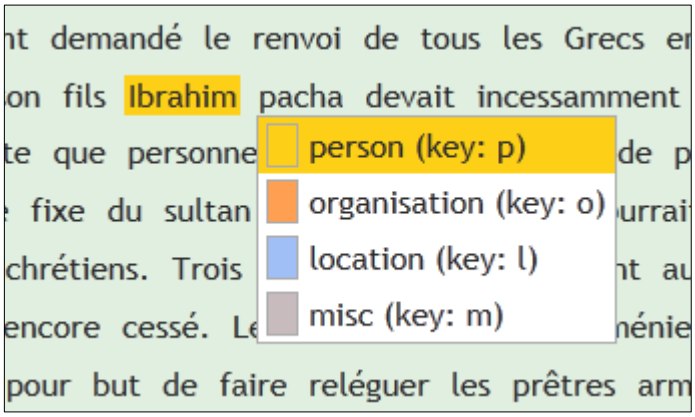
The tool allows the revisors to annotate the attestations in the lemma quotations, by assigning a type chosen out of several available types. For example, in a project dealing with named entities, you might want to annotate the attestation tokens as ‘person’, ‘location’ or ‘organisation’.

To do so, you have to insert the annotation types into the ‘types’ table:

```
mysql> ALTER TABLE types AUTO_INCREMENT = 1;
```
(never forget this line!)

```
mysql> INSERT INTO types (name, color, shortcut)
       VALUES ('Person', '#F5A9A9', 'p'),
              ('Location', '#F3F781', 'l'),
              ('Organisation', '#A9BCF5', 'o');
```

- The first argument (name) is the name of the type as it will be shown to the revisor:



- The second argument (color) is the color the annotated token will be given as it is assigned a given type (which makes it very easily recognisable in the text).

- The third argument (shortcut) is a key which can be pressed on your keyboard to assign a given type to a token, without having to click onto its name. You can also give it a NULL value (without quotes) if you don’t want any shortcut to be available.

If you only want to mark the tokens without assigning them any differentiated type, just insert a single default type into the ‘types’ table:

```
mysql> ALTER TABLE types AUTO_INCREMENT = 1;
```
(never forget this line!)

```
mysql> INSERT INTO types (name, color, shortcut)
    VALUES ('default', 'yellow', NULL);
```

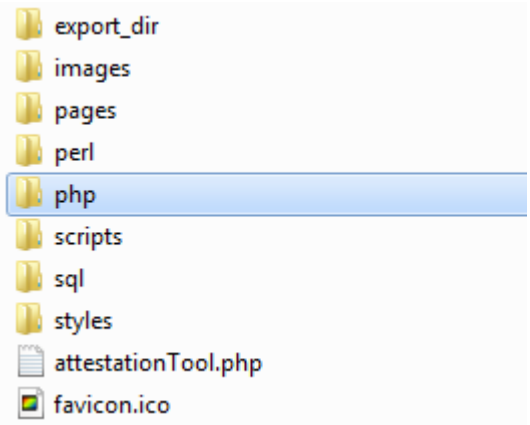
Finally, if you want to be able to add comments to the tokens (which is actually a separate, custom type), you must insert a ‘comment’ type:

```
mysql> INSERT INTO types (name, color, shortcut)
    VALUES ('comment', 'red', NULL);
```

This will enable the user to type in some comments in the same pop-up window as above, and to assign these comments as a type instead of the other standard types.

TOOL CONFIGURATION

The tool needs a bit of configuration before you can use it. The configuration is set in the globals.php file, which can be found in the PHP directory the your installation:



GLOBALS.PHP

[1] The first thing to set in this file is the database host and the username and password needed to access it. Put here the username and password of the MySql user you created before in the 'Create a MySql user' section.

```
$sDbHostName = "yourhost.com";
$sDbUserName = "username_of_your_host";
$sDbPassword = "password_of_your_host";
```

[2] Secondly, you have to declare at least one project to be run in the tool (which is why you actually use the tool for).

Say you have some Dutch dictionary project and its database is called 'NL_dictionary'. This database must have been created before in the 'Create databases' section.

You have to declare each of your project database(s), together with a user friendly project description (this description will be shown to users in the log-in screen).

```
$asProject['NL_dictionary'] = 'My Dutch dictionary project';
```

If more projects are declared, it can be convenient to have the project being mostly used at the moment selected by default in the log-in screen. This can be done by giving the $sChecked variable the name of the relevant project database as a value:

```
$sChecked = 'NL_dictionary';
```

If no project must be preselected, give this variable NULL as a value:

```
$sChecked = NULL;
```

[3] Third, as the tool offers the possibility to quickly look up the lemma you're working on in an external dictionary, you have to declare how the tool should access this external dictionary (if needed).

The small '0' sign to the right side of the lemma 'verstandelijcheit' gives straight access to this lemma in the external dictionary.

This is done by mean of an array associating a project database name to an URL at which the dictionary can be accessed.
This declared URL must contain a substring '<ID>'. This is because when the dictionary button in the user interface is being clicked on, the tool will replace the <ID> string in the declared URL by the id stored in the database for the lemma being worked on,[4] and open this URL in a new window. That way, one can access the right lemma in the external dictionary in just one click.
The declaration must look like this:

```
$sExternalDictionary['database_name'] = "http://some_url?id=<ID>";
```

Examples:

```
$sExternalDictionary['OED_English'] =
                                    "http://oed.com/view/Entry/<ID>";

$sExternalDictionary['MNW_Lexicon'] =
       "http://gtb.inl.nl/iWDB/search?actie=article&wdb=MNW&id=<ID>";
```

[4] Finally, as a project will always end with an export of the data you worked on, you have to declare how the tool should deal with this data export. That is: which path the data export should be send to, and how the export file should be called:

[4] in the 'externalLemmaId' field of the 'lemmata' table.

```
// Location to export the data to
$sFilePathForExport = "../export_dir/";

// Name the export zip file should be given
$sZipFileName = "AttestationToolExport.zip";
```

ALTO:

If the files originally loaded into the tool were ALTO files (so you'll have to export to ALTO too), there is some additional configuration needed. As the tool needs each token in an ALTO file to have an identifier of some sort (so as to be able to match the exported data to the right tokens in your files), you have to declare this identifier explicitly. There are two possibilities.

If the tokens do have some identifier attribute (like in `<String ID="123"…/>`), just declare this attribute name:

```
$AltoUniqueIdentifier = "ID";
```

But if the tokens lack an unique identifier, you can simulate such an identifier by combining the other attributes available. For example, in an ALTO token as `<String WC="0.5199999809" CONTENT="de" HEIGHT="50" WIDTH="63" VPOS="472" HPOS="100"/>`, we can combine the attributes WC, HEIGHT, WIDTH, etc.. Those will together hopefully always give a unique combination of characters, which we can use as an unique identifier, so we can match the exported data to the right tokens in the file. Declare your custom identifier in this case the following way:

```
AltoUniqueIdentifier = array("WC", "CONTENT", "HEIGHT", "WIDTH", "VPOS",
"HPOS");
```

WORKING WITH THE TOOL

IMPORTING DATA INTO THE TOOL

To be able to work with the tool, you'll first have to load your data into it. The tool is provided with an import script, which is written in Perl.

This script is designed to process ALTO-XML. ALTO is an open XML standard to describe OCR text and layout information of printed documents. It consists of three main sections, 'Description', 'Styles' and 'Layout':

```
<?xml version="1.0"?>
<alto>
  <Description>
    <MeasurementUnit/>
    <sourceImageInformation/>
    <Processing/>
  </Description>
  <Styles>
    <TextStyle/>
    <ParagraphStyle/>
  </Styles>
  <Layout>
    <Page>
      <TopMargin/>
      <LeftMargin/>
      <RightMargin/>
      <BottomMargin/>
      <PrintSpace/>
    </Page>
  </Layout>
</alto>
```

The last section 'Layout' is the one that matters for our work, since it contains the actual document content, which we want to load. The information in the 'Description' and 'Styles' sections will be ignored since it's not relevant for our goal.

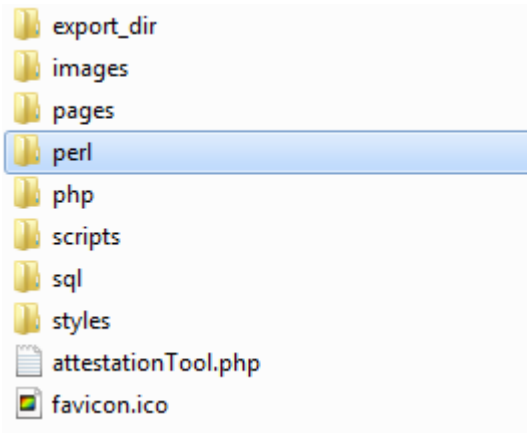
The actual document text should be stored in the subsection 'PrintSpace'. This should consist on its turn of 'TextBlock' sections (say paragraphs), each divided into 'TextLine' sections (self-explanatory). Finally, each text line will contain the distinct tokens stored in 'String' sections.

Here is an example (for readability, some xml attributes are replaced by dots):

```
<TextBlock …>
      <TextLine …>
        <String ID="P1_ST00002" … CONTENT="Woensdag," … />
        <SP ID="P1_SP00001" …/>
        <String ID="P1_ST00003" … CONTENT="30" …/>
        <SP ID="P1_SP00002" …/>
        <String ID="P1_ST00004" … CONTENT="Januarij." …/>
      </TextLine>
</TextBlock>
```

Further information about ALTO can be found at
http://en.wikipedia.org/wiki/ALTO_%28XML%29

The ALTO import script is called 'alto2db.pl' and can be found in the 'perl' directory of the installation:



To run the script, first put the ALTO-XML files you'd like to use into a separate directory. You'll then be able to import those files into the tool by running the script with the following arguments:

```
perl alto2db.pl -h hostname -d db_name -u username -p password
                -i input_directory
```

The first four arguments are the location and credentials of your database, as stated in the globals.php file we discussed above. The last argument is the name of the directory containing the ALTO-XML files to be imported.

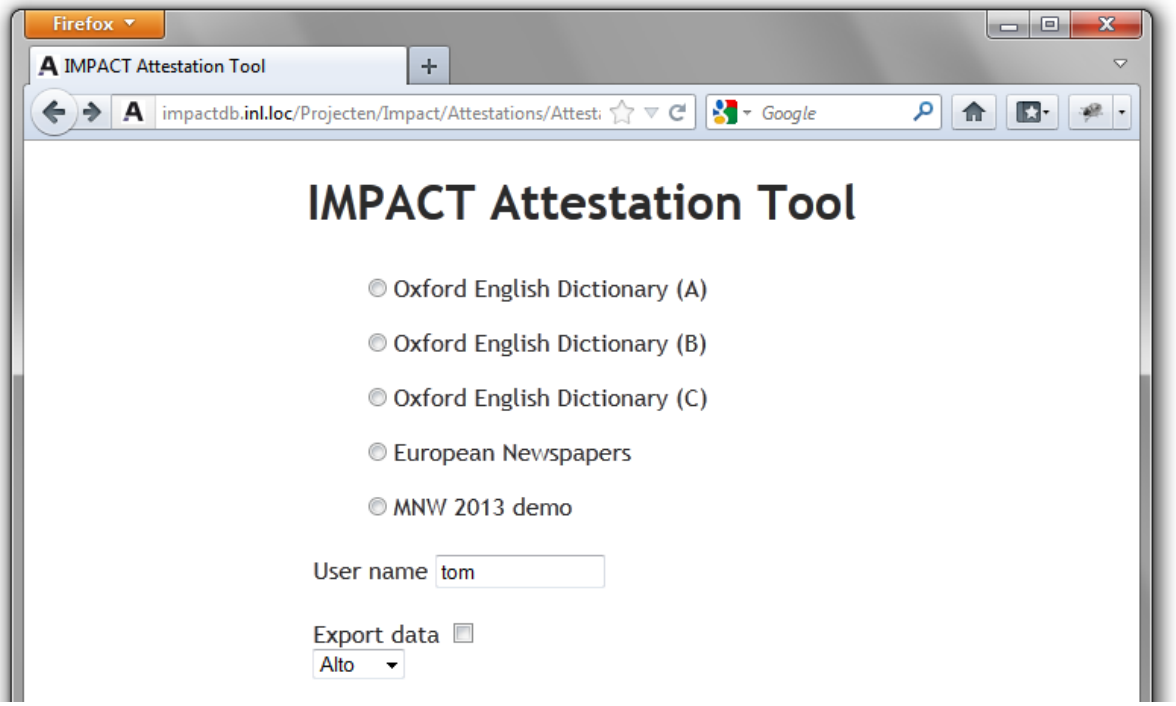
Loading other XML formats

Of course, you are free to adapt the 'alto2db.pl' script in such a way that other XML types can be loaded. If this is what you choose for, you'll have to adapt the 'handleText' function in particular, since this function is the one that parses the XML content.

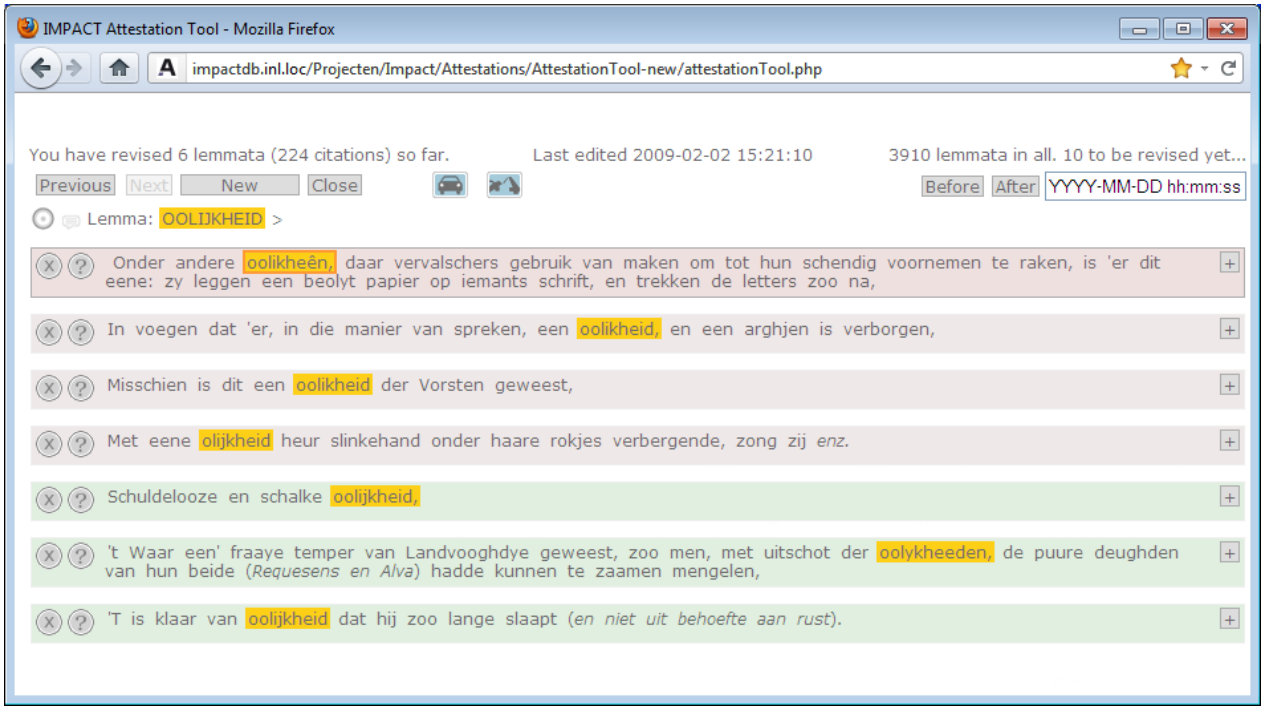
USER INTERFACE

The attestation tool was designed to enable multiple concurrent users to view the data in the database and to make changes to it.

Users first have to log in. This happens in the log in screen. The different projects listed here should have been set in the 'Tool configuration' section. The user logs in by simply typing in his/her name and chooses a project to work on.



When the user has logged in, he/she will access the main screen where all attestation work will be done. In the screenshot below you see an example page with the Dutch lemma "*oolijkheid*" and seven quotations for this word:



The tool always lists quotations per lemma. Quotations are sorted by uncertainty. The most uncertain ones (containing words least similar to the headword) appear at the top and are marked red(dish). Literal matches are at the bottom, marked green.

By using the arrow keys or the mouse, users can select or deselect words or move a selection.

The 'X' button can be used to mark quotations requiring special attention (e.g. because they were extracted in the wrong way). The '?' button can be clicked to mark quotations that are 'unfortunate' (e.g. the headword doesn't appear in the quote as such but only in a compound). The target button left of the lemma head word can be used to mark an analysis. The text balloon icon between the mark button and the lemma headword can be used to add a comment to a lemma.

AUTO ATTESTATION

When a very frequent variant has been missed in automatic matching, auto attestation can come in handy. A user can select a word and, by hitting the auto attestation button, all occurrences of this word form will be highlighted.

AUTO DE-ATTESTATION

When multiple occurrences of the same word form should be de-attested the auto de-attestation button can be used. By clicking on it the selected word form and all identical occurrences will be de-attested.

LEAVING THE TOOL AFTER A WORKING DAY

A very important feature of the tool is to make sure that a lemma is not being worked on by two revisors at the same time. That would cause double work to happen, and could also cause data corruption.

To prevent such a situation, the tool locks a lemma whenever it is accessed by a revisor. As the lemma is locked, no other revisor will be able to see it.
When a revisor leaves the page of a lemma, the lock is released so the lemma gets visible again (at least if it still needs to be processed). However, if a revisor decides to leave the tool and just clicks the window away, the lemma will kept locked and will never be released again. This is why the tool has a 'Close' button. When this button is clicked, the active lemma get unlocked and the tool closes properly.

In case a revisor happens to leave the tool by closing the window by mistake (instead of clicking on the 'Close' button), it is still possible to unlock the locked lemmata by running a simple query on the project database:

```
UPDATE your_project_database.lemmata
SET revisorId = NULL, revisionDate = NULL
WHERE revisorId <0;
```

Don't forget to change the red part into your own project database name.

BEWARE: running this query while some revisor is working in the tool may cause malfunction. So make absolutely sure nobody's working before running this query!

KEYBOARD SHORTCUTS

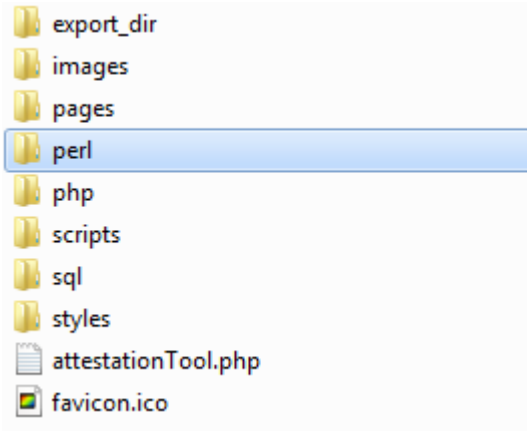
To enhance the usability the interface can be used with the mouse, with the keyboard or both.

Key	Action
F2 or d	Previous revised lemma
F4 or f	Next revised lemma
F8	Add a comment
F9 or x	Toggle quotation as wrongly/correctly parsed
Spacebar	Save current lemma, and display a new unrevised one
a	Auto attestation
m	Mark the lemma
u	Toggle quotation as fortunate/unfortunate
z	Auto de-attestation
INSERT	Insert a new attestation
DELETE	Delete currently selected attestation
CTRL	Walk through attestations of the selected quote

GENERATING STATISTICS ABOUT THE REVISORS

As a project run in the Attestation Tool might need to keep running on schedule, it may be needed to get a statistical report about the revisors activities.

The Attestation Tool provides a Perl script that generates such a report: *generateStatistics.pl*. It can be found in the 'perl' directory of the application:



You need to call this script with a few parameters:

```
perl generateStatistics.pl -h hostname -u username -p password
      -d date -D databasename
```

The 'date' parameter is the close date of the period you'd like statistics about. The parameter must have the format 'YYYY-MM-DD'.
So, let's say you wish to get statistics about the redactional activities till the 2nd of may 2013, your 'date' parameter will have to read:

```
... -d 2013-05-02 ...
```

The 'databasename' parameter must contain the name of the project database(s) you'd like to get a report about. If you'd like to get a report about more than one database, the different database names must be comma separated.
So, say you'd like to get a report about the project databases 'uk_english' and 'old_greek', your 'databasename' parameter will have to read:

```
... -D uk_english,old_greek...
```

Running the script as stated above will only give you screen output. If you'd like to get the report into a file, add a 'greater than' sign followed by the export file name after the script call.

```
perl generateStatistics.pl -h hostname -u username -p password
      -d date -D databasename > filename.txt
```

EXPORTING DATA

When a project has reached its end, you'll most likely need to export the database you've been working on.

The tool has a built-in export function in the log in screen. We already discussed the fact that the tool supports the import of ALTO-XML by default. This is expectedly also the default export format.
Of course, you can export to some other format by modifying the export function of the tool. This function is to be found in the 'export2alto.php' file in the 'php' directory of the application.

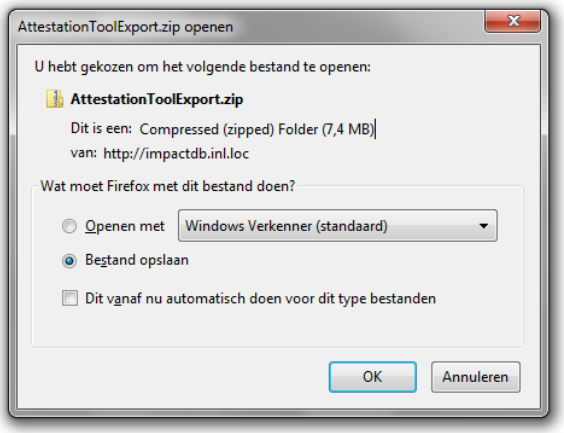
In case you choose for the default ALTO-XML export format, you'll be able to recognize the newly annotated tokens in the export files thanks to a new attribute giving their types:

```
<TextLine ID="P1_TL00005" …>
      <String ALTERNATIVE='B-person' ID="P1_ST00010" … CONTENT="MICHAEL" …/>
      <SP ID="P1_SP00005" …/>
      <String ALTERNATIVE='I-person' ID="P1_ST00011" … CONTENT="JACKSON" …/>
</TextLine>
```

Exporting

This is what you'll need to do to export a project:
- select the project name in the main menu
- check the 'Export data' check box
- type in your user name, and press 'enter'

Then the export will start. This can take quite a while, so don't panic if it looks like the tool is doing nothing. When the export is finished, the tool will ask you where to save the export file, which will have the form of a zip archive (containing a collection of ALTO files):



APPENDIX

ATTESTATION TOOL DATABASE

TABLE LEMMATA

Field	Type	Description
id	number	Internal identifier. Primary key.
lemma	string	Head word (lemma) corresponding to the word form.
partOfSpeech	string	Part of speech of the lemma.
initialVariants	string	Set of variants. This field is used by the scripts in automatic pre-processing.
revisionDate	date	Date of revision.
revisorId	string	Identifier of revisor performing the revision.
externalLemmaId	string	The identifier of the article in the dictionary.
marked	boolean	Indicates whether or not the lemma is marked.
hide **(not in all versions)**	boolean	Indicates whether or not the lemma is shown in the tool.
comment	string	Comment field (this allows general comments at lemma level; for comments at attestation level, see the attestations table)

TABLE QUOTATIONS

Field	Type	Description
id	number	Internal identifier. Primary key.
lemmaId	number	Identifier of the lemma this quotation belongs to.
quotation	string	The actual quotation.
tokenizedQuotation	string	The quotation split into tokens.
quotationSectionId	string	This field is for internal use of the scripts.
dateFrom	int	Year indicating the first occurrence of the word quoted.

dateTo	int	Year indicating the last occurrence of the word quoted.
specialAttention	bool	Can be set when the quotation is somehow out of the ordinary.
unfortunate	bool	Can be set when the headword (lemma) doesn't really occur in the quotation.
updated **(not in all versions)**	bool	Can be set if, after an update of the dictionary data, there is a discrepancy between the quotation in the data and the quotation in this record.

TABLE ATTESTATIONS

Field	Type	Description
id	number	Identifier of the attestation.
quotationId	number	Identifier of the quotation this attestation belongs to.
onset	number	Character position of the start of the word attested.
offset	number	Character position of the end of the word attested.
reliability	float	Indicates how certain the match is (the more different the higher this number).
wordForm	string	The word attested as occurring in the quotation.
typeId	number	Identifier of the attestation type. Only relevant when doing multiple typed attestations (e.g. named entity attestations where there is distinction between NE_LOC, NE_ORG and NE_PER).
error	boolean	Indicates whether or not a word was marked as erroneous.
dubious	boolean	Indicates whether or not a word was marked as dubious.
elliptical	boolean	Indicates whether or not a word was marked as being elliptical. This can be used to mark a word like *North* in *North and South America*.
tokenId **(not in all versions)**	string	Id of a token as stated in the imported ALTO-file.
comment **(new)**	string	Comment field (this allows specific user comments at attestation level, whereas the comment field of the lemmata table only allows general comments at lemma level).

TABLE GROUPATTESTATIONS

Field	Type	Description
id	number	Identifier of the group
attestationId	number	Identifier of an attestation.
pos	number	The position number the attestation has in the quotation.

TABLE TYPES

This table should only exist when typed attestations are needed. The interface of the tool will automatically detect whether or not this table exists. If it does the appropriate interface functions are loaded. For 'simple' attestation (like marking a headword in a citation) just leave out the entire table.

Field	Type	Description
id	number	Identifier of the type. **BEWARE**: ids must be listed in ascending order, or it will cause malfunction.
name	string	The name of the type
color **(new)**	string	The colour an attestation of this type should have in the tool (defined as HTML hexadecimal color encoding, e.g. '#A1BFF7').
shortcut **(new)**	string	The shortcut (key) one can press to assign this type to a given token. If this is NULL, the type will have no shortcut (which is ok, since it's not compulsory).

TABLE REVISORS

This table is the one in which users must be set. Each user has an id, and a name to log in with.

Field	Type	Description
id	number	Identifier of the revisor
name	string	Name of the revisor

LICENSING