

MMAX2 Annotation Tool

Quick Start Guide

© Christoph Müller
EML Research gGmbH
<http://mmax.eml-research.de>

27th October 2004



Contents

1	About this Document	3
2	Installing MMAX2	4
3	MMAX2 Display Basics	6
3.1	The Main Window	6
3.2	Markable Selection	7
3.3	Attributes and the Attribute Window	7
3.4	Relations	9
3.5	Saving the Annotation	10
4	Modifying the Annotation	11
4.1	Creating a Markable	11
4.2	Deleting a Markable	12
4.3	Resizing a Markable	12
4.4	Modifying Markable Relations	13
4.4.1	Adding to and Removing from a <i>MARKABLE_SET</i> -type Relation	13
4.4.2	Adding to and Removing from a <i>MARKABLE_POINTER</i> -type Relation	16
5	Modifying Base Data	18
5.1	Modify an Element	18
5.2	Delete an Element	19
5.3	Insert an Element	19

6	Tools	20
6.1	Markable Browser	20
6.2	Project Wizard	22
6.2.1	Input File Selection	23
6.2.2	Tokenization	23
6.2.3	Markable Level Creation	24
6.2.4	Project Creation	25

1 About this Document

This document is intended as a quick hands-on introduction to the MMAX2 annotation tool (version beta 3). It provides information on how to install and run the tool, and how to create, load, browse and manipulate annotations and base data. If you want to get an idea about MMAX2 and what you can do with it, this document is for you.

Important note: The examples used throughout this document are from the Heidelberg Text Corpus (HTC), and they deal solely with coreference and bridging annotation. This does *not* mean that MMAX2 is in any way restricted to or especially tailored for this type of annotation! Rather, we are convinced that the tool is sufficiently general and powerful to be useful for the annotation of a wide range of diverse phenomena. Please note the other sample files supplied with this distribution, in particular the samples taken from the Trainline corpus and from the Korean Treebank.

This document does **not** describe everything you can do with MMAX2, and even the things that are described are not necessarily complete in every detail. In particular, it does not explain how to design and implement user-definable annotation schemes. A detailed document describing this central aspect of MMAX2 will be added to the tool's documentation later. For information about other topics not covered in the current document, please refer to the following:

- MMAX2 Annotation Tool Style Sheet Guide
- MMAX2 Query Language Reference Manual

2 Installing MMAX2

MMAX2 requires a current Java version to be properly installed on your system. MMAX2 has been tested and is in use on both Windows, LINUX and Macintosh (MAX OS X) platforms and with various versions of Java (incl. 1.3., 1.4 and 1.5). If you encounter problems during installation or later, please contact the developers via the MMAX2 web site (<http://mmax.eml-research.de>).

Installing the MMAX2 annotation tool is very simple. You should have received the MMAX2 distribution as a compressed archive file.

Just uncompress the archive. This will create a single top level directory `MMAX2_version` plus some subdirectories:

```
MMAX2_version
  Libs
  Samples
    HTC
      Basedata
      Customizations
      Markables
      Schemes
      Styles
    KoreanTreeBank
      Basedata
      Customizations
      Markables
      Schemes
      Styles
    ...
```

The root directory will contain a file named `MMAX2Key.jar`. This file is required for MMAX2 to run. If this file is missing, MMAX2 will not start and a message will be displayed. If this happens, please contact us via the MMAX2 web site (<http://mmax.eml-research.de>) to obtain a valid key. The `Libs` directory contains a number of libraries for XML and XSL support¹ and the MMAX2 library proper. The `Samples` directory contains (among others) the `HTC` directory which in turn contains the actual annotation files, including a file named `common_paths.xml`. This file is used by MMAX2 to locate the different files pertaining to a complete MMAX2 project. **Note:** In contrast to earlier versions of MMAX2 (beta 2 and earlier), the `common_paths.xml` is not platform-dependent any more.

The installation is complete. In order to test it, run the script `startmmx2.htc.win.bat` (Windows) resp. `startmmx2.htc.linux.sh` (LINUX/MAC)². It is recommended that you call either script initially by typing its name on the console (as opposed to double-clicking it in your file explorer). This way, you will be able to read the screen output and error messages even if the tool crashes immediately after startup. While this does not normally happen, there *can* be cases, e.g. if Java is not correctly installed. Af-

¹Note that MMAX2 uses for XML and XSL support versions of the *Apache* Xalan and Xerces libraries which are **not** the most recent ones (but may very well be compatible with these). Therefore, the required libraries are supplied with the MMAX2 distribution and reside in the `Libs` directory. If you already have more recent versions of the required libraries on your system, you are encouraged to try them out with MMAX2, but note that the tool has only been tested with the supplied versions!

²Under LINUX and MAC, you first have to make the script executable using the command `chmod u+x filename`

terwards, when you know that everything does work, you can start the tool by double-click. After executing the correct `startmmax2_htc_.*` file, you will see the screen described in the next section.

3 MMAX2 Display Basics

3.1 The Main Window

After opening the supplied sample HTC file, you will be asked whether you wish to validate the current annotation. Validation will check the annotation for consistency, and supply default values for those markable attributes that do not yet have a value. Validation is only necessary once, i.e. after the annotation has been created or modified outside the tool. Therefore, the default option is 'Do not validate'. After the validation dialog, several windows appear on the screen. For the purposes of this introduction, a particular style sheet has to be used. In order to select it, please open the *Settings* menu in the *Markable level control panel* (small window initially in the upper right corner). From this menu, please select the *StyleSheet* menu, and from this menu the entry *htc_3.xml*. After a few moments, the MMAX2 main window should look as in Figure 1. Initially, MMAX2 uses a default (ASCII-based) display font. Both the name and the



Figure 1: MMAX2 Main Window

size of the used display font can be changed via the **Font** submenu in the **Display** menu. Note that the list of fonts that are offered here depend on which fonts are installed on your system! By default, changes to the font name will affect only the display, and not the popup menus used for markable selection. If you want to use the display font also in the popup menus, you can check the corresponding menu item in the **Font** submenu. This is necessary when using e.g. Korean fonts (as for the KoreanTreeBank sample), which cannot be displayed otherwise. In the HTC sample display you see, some parts of the text are black, some are blue, and some are blue and in italics. Black is the default text color; it is used for all portions of the text for which no special rendering style has been defined. For markables on the coreference level, on the other hand, a customized rendering style has been defined.³ This style specifies that every markable on this level should be rendered in blue, and that every markable in a coreference relation should *in addition* be rendered in italics.

The brackets appearing in the text are so-called *markable handles*. In the sample annotations, each markable on the coreference level is associated with a black, bracket-style markable handle. Markable handles serve several purposes. Apart from simply visualizing the extent of the markable they are associated with, they are also sensitive to mouse moves: Moving the mouse on a markable handle causes the matching handle to be highlighted (Figure 2). Markable handles are also sensitive to mouse clicks and allow direct

³The XML files responsible for these so-called *markable customizations* can be found in the HTC/Customizations directory.



Figure 2: Matching Markable Handles

selection of a markable even in cases of multiple embedding. Once a markable has been selected, it is highlighted (i.e. rendered with a yellow background).

3.2 Markable Selection

Before a markable can be modified or its attributes can be displayed, it has to be *selected*. Generally, the **left mouse button** will always select a markable for the purpose of displaying its attributes. If there is only one markable at a left-clicked position, this markable will be immediately selected and its attributes will be displayed (cf. below). The same is true if a markable handle is present at the left-clicked position: since each markable handle is associated with exactly markable, this markable can be immediately selected. If more than one markable (from one or more levels) is present at a left-clicked position, the click is ambiguous and a popup menu containing all markables will be displayed. From this menu, select the desired markable by left-clicking its respective entry. The **right mouse button**⁴, on the other hand, is used for selecting markables for other purposes than display of their attributes, such as deletion, etc. (cf. Section 4). Markable handles and disambiguation popup menus, however, work the same as for left clicks.

3.3 Attributes and the Attribute Window

Markables are not just sequences of words that have been grouped together; rather, they are the carriers of the actual annotation. Annotations consist of markable *attributes* and *relations* (cf. Section 3.4). The attributes associated with the currently selected markable are displayed in the MMAX2 attribute window (Figure 3). The attribute window contains one tab panel for each markable level in the annotation project (here: *sentences* and *coref*). Since the selected markable is from the coreference level, the *coref* panel is active in Figure 3.

Attributes consist of a name (e.g. **np_form** or **grammatical_role**) and a set of possible values, one of which is always selected. Possible values are displayed as either a number of radio buttons (e.g. **np_form**) or as a drop-down list (e.g. **agreement**)⁵. Changing the value of a given attribute is done by simply clicking

⁴On MAC, hold the **Alt** or **Ctrl** key while pressing the mouse button

⁵The appearance of an attribute in the attribute window is controlled by the type it is assigned in the scheme file. Radio buttons are created by using the type *NOMINAL_BUTTON*, and lists by using the type *NOMINAL_LIST*.

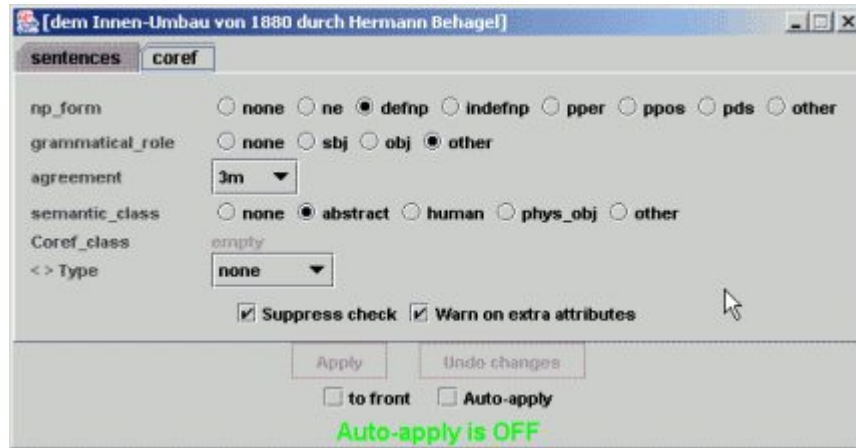


Figure 3: MMAX2 Attribute Window

the corresponding radio button, or selecting the corresponding entry in the list. This causes the attribute window to display the newly selected value. At the same time, the *Apply* and *Undo changes* buttons in the bottom of the attribute window will be enabled. This will happen unless the *Auto-apply* option is activated: if *Auto-apply* is activated, all changes in the attribute window will *immediately* (and, at least for the moment, irrevocably) be applied to the current markable, possibly overwriting earlier values. Therefore, this option should be used with caution, particularly in connection with branching attributes (cf. below).

Branching attributes are those whose current value influences the availability of other attributes who are said to be *dependent* on the former. Branching attributes can easily be identified in the attribute window because they have the characters < > prepended to their names. In the sample file, the attribute **type** is one of several branching attributes. Changing its value from *none* to *anaphoric* causes the dependent attribute **ante_sub_anaphoric** to appear (Figure 4). If the changes are confirmed by clicking the *Apply*

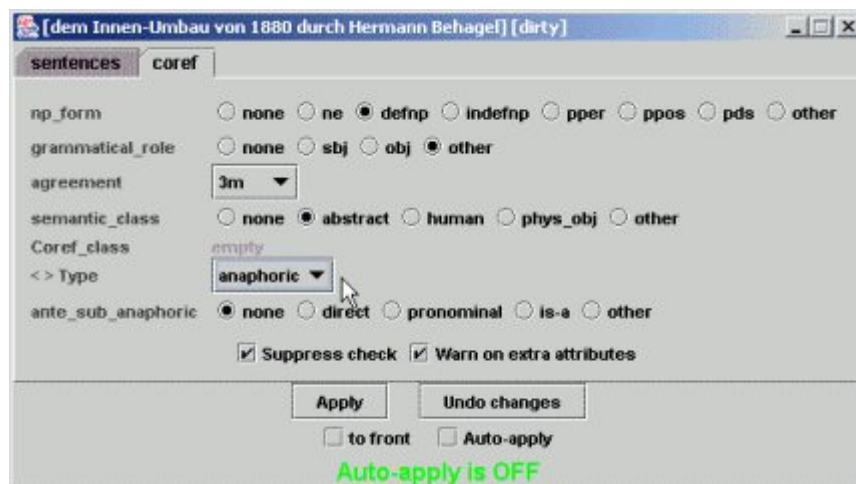


Figure 4: Attribute Window Change as Result of Branching Attribute Modification

button⁶, the dependent attribute and its current value will become part of the currently selected markable's annotation. Likewise, switching the value back to *none* and clicking the *Apply* button will remove the attribute from both the attribute window and the markable. Clicking the *Undo changes* button restores the currently selected markable's attributes to the state directly after the last changes were applied, which is the state directly after the markable was selected, if no changes were applied since then. It is important to understand that at all times the attribute window shows *all* and *only* the valid attributes of the currently selected markable: If no confirmation is pending⁷ and some attribute-value combination is displayed in the attribute window, it is part of the currently selected markable's annotation, and if something is not displayed, it does not currently exist!

3.4 Relations

Apart from carrying annotations in the form of attribute-value pairs, a markable can also be associated with (one or more) other markables to form markable *relations*. While markable relations are also attributes in the sense that their availability is controlled by the attribute window, they are visualized graphically in the main window. Selecting a markable that participates in one (or more) relations causes these relation(s) to be graphically rendered by means of colored lines. Two types of markable relations are currently supported: Relations of type *MARKABLE_SET* associate arbitrarily many markables with each other in a transitive, undirected relation. In the HTC corpus, the *MARKABLE_SET*-type **coref.class** relation is used to represent coreference. Figure 5 shows how a coreference set can be visualized. Note that although all markables in

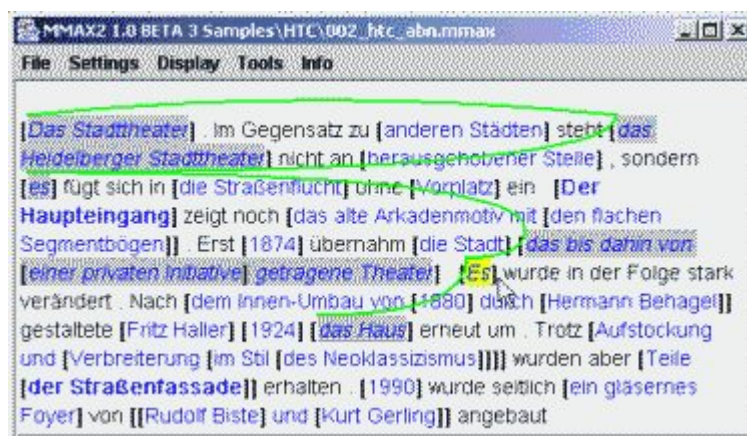


Figure 5: Selected Coreference Set

a *MARKABLE_SET*-type relation are unordered, the markables' document order is used to linearize the set for visualization.

Another type of markable relation currently supported is *MARKABLE_POINTER*. Relations of this type associate with one markable (the *source*) one or more *target* markables in an intransitive, directed fashion. In the HTC corpus, bridging relations are represented by means of the many-to-one⁸ *MARKABLE_POINTER*-type **bridging antecedent** relation. Figure 6 shows how a pair of bridging expression and bridging an-

⁶Or if *Auto-apply* is active.

⁷I.e. if the *Apply* button is disabled.

⁸The relation is many-to-one because although each markable in the HTC corpus annotation can have at most *one* bridging antecedent, one markable can be bridging antecedent to several *source* markables.

tecedent is visualized.



Figure 6: Selected Bridging Expression with Associated Antecedent

The appearance of a relation is controlled by a number of attributes in the scheme file in which the relation itself is defined. The `color` and `width` attributes specify the color and width of the line that is drawn, while the `style` attribute specifies its outline. Possible values for the `style` attribute are `lcurve`, `rcurve`, and `straight` (default).

3.5 Saving the Annotation

If the annotation has been changed, i.e. after markables or their attributes or relations have been modified, they can be saved by means of the **Save** menu item in the **File** menu. Via the **Levels** submenu, each level can be saved separately. The **All** menu item can be used if more than one level has been modified. Only those markable levels will be saved that actually were modified. A backup copy of the previous version will always be created, but *no more than one backup copy* will be maintained!

4 Modifying the Annotation

The main purpose of an annotation tool is to create and delete markables and to set, modify and remove their attributes and relations. This section describes how to do this in MMAX2.

4.1 Creating a Markable

If you want to create a new markable, left-click somewhere in the first word, hold the mouse button down and drag the mouse until it is somewhere in the last word. Note that this will only work if no other markable is currently selected! If some other markable should be currently selected, first unselect it by *left*-clicking on some non-markable or empty space in the display. Upon releasing the mouse button, the selection will automatically be expanded from the beginning of the first to the end of the last word. Also, a popup menu will appear with one *Create Markable* menu item for each currently active markable level. Select *Create Markable on level 'coref'* by left-clicking it, and a new markable will be created. The new markable will immediately be rendered in blue, according to the rendering style defined for markables on this level⁹. Markable handles, however, will *not* appear automatically! This is because markable handles are inserted by means of the XSL style sheet responsible for creating the display. Making the markable handles appear on the newly created markable requires to reapply the current style sheet. In order to do this, open the **Display** menu in the main window and select **Reapply current style sheet**. The entire display will be rebuilt and the markable handles become visible on the new markable. Note that markable handles are not *required* to select a markable,¹⁰ but that they simply offer a convenient alternative. Thus, reapplying the style sheet for *every single* new markable is not necessary. Instead, it might be good practice (and more convenient) to rebuild the display only once after several new markables have been created.

A newly created markable will only have the *default* attributes and values defined for its respective markable level. Default attributes are those attributes that are visible in the markable level's tab panel when no markable is currently selected (Figure 7). What counts as a default attribute is determined by the annotation scheme.

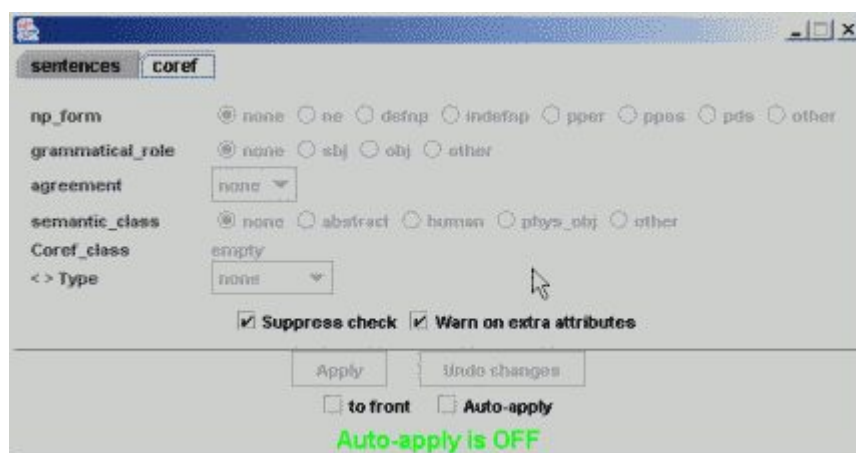


Figure 7: Default Attributes and Values for the *coref* Level

⁹Cf. Section 3.1

¹⁰If no markable handles are available, a left-clicked markable is either selected directly, or, in case of embedding, the desired markable can be selected from a popup menu.

4.2 Deleting a Markable

If you want to delete an existing markable, select it with the *right* mouse button. This will cause a popup menu to appear. From this menu, select the *Delete this Markable* menu item by left-clicking it. Any markable-related rendering styles will immediately be removed from the markable, causing it to be rendered in plain black text.¹¹ Also, the deleted markable's handles will be rendered *disabled* (i.e. in a light gray, strike-through font). With the next style sheet reapplication, they will be removed from the display. Note that deleting a markable will only work if no other markable is currently selected! In particular, this means that you cannot delete the currently selected markable itself. If some markable should be currently selected, first unselect it by *left*-clicking on some non-markable or empty space in the display (cf. Section 4.1).

It is important to realize that deleting a markable can have an effect on other markables as well: If a markable is the second but last in a coreference set (or in some other *MARKABLE_SET*-type relation), deleting it will also remove the **Coref_class** attribute of the remaining markable, thus removing the entire set. Similarly, if a markable is a *target* markable of some *MARKABLE_POINTER*-type relation (like **bridging_antecedent**), deleting it will also remove its reference from the relation's *source* markable.

4.3 Resizing a Markable

Resizing a markable means changing the span of words that the markable covers. Words can be added to or removed from the beginning or the end of a markable. In order to resize a markable, it has to be the currently selected one!

To **add** words to the currently selected markable, left-click somewhere in the first word you want to add, hold the mouse button down and drag the mouse until it is somewhere in the last word. Upon releasing the mouse button, the selection will automatically be expanded from the beginning of the first to the end of the last word. Also, depending on the relative position of the selected words, a popup menu with either a *Prepend to this Markable* or an *Append to this Markable* menu item will appear. Select this menu item by left-clicking it, and the selected words will immediately be added to the markable. As a result, the added word(s) will be rendered according to the rendering style defined for markables on this level (i.e. in case of the *coref* level, in blue). The markable handle, however, will not be moved to reflect the new markable size until the next style sheet reapplication. Note that discontinuous markables are not supported yet. This means that only *adjacent* (sequences of) words can be added to a markable!

To **remove** words from the currently selected markable, left-click somewhere in the first word you want to remove, hold the mouse button down and drag the mouse until it is somewhere in the last word. Upon releasing the mouse button, the selection will automatically be expanded from the beginning of the first to the end of the last word. Also, a popup menu will appear with a *Remove from this Markable* menu item. Select this menu item by left-clicking it, and the selected word(s) will immediately be removed from the markable. As a result, any rendering styles related to the currently selected markable will disappear from the words that have been removed. The markable handle, however, will not be moved to reflect the new markable size until the next style sheet reapplication. Note that since discontinuous markables are not yet supported, words can only be removed from the beginning or the end of a markable. Also, removing *all* words from a markable is not possible.

¹¹Unless the deleted markable was embedded in some other markable, in which case the remaining markable's font attributes may prevail.

4.4 Modifying Markable Relations

While markable attributes are modified via the attribute window (cf. Section 3.3), markable relations are added and removed by means of mouse actions in the main display, using the *right* mouse button. Markable relations are always seen with respect to the currently selected markable: if no markable is currently selected, clicking the right mouse button on a markable will not give access to any relation-specific menu items.

4.4.1 Adding to and Removing from a *MARKABLE_SET*-type Relation

If you want to **add** a markable to a markable set, select it with the *right* mouse button. If the markable is not yet participating in a markable set (cf. below) and if the annotation scheme allows for this markable and the currently selected one to be in a markable set, a popup menu with the corresponding menu item will appear. The text of this menu item can be customized (in the annotation scheme) to reflect the semantics associated with this action. In the sample HTC annotation, the menu item will be *Mark as coreferent*. In addition to the text, the menu item will also contain a small box in the same color as used for graphically rendering relations of the respective type (Figure 8). Select this menu item by left-clicking it, and the markable

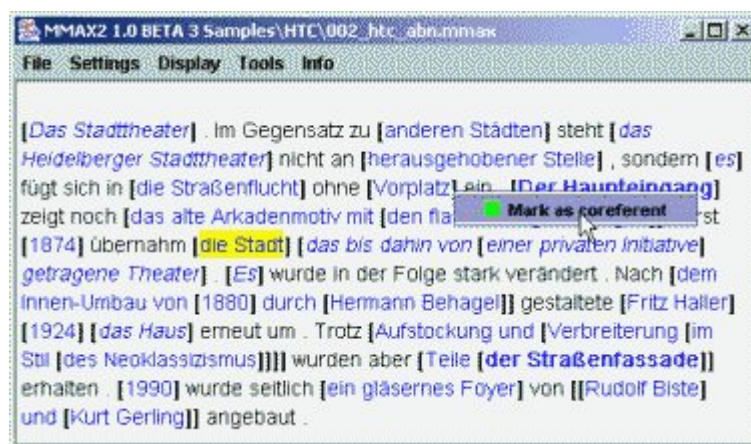


Figure 8: Relation Modification: Adding a Markable to a Markable Set

will be added to the markable set. As a result, the newly added markable will be graphically linked to its most recent document-order predecessor and successor (unless it is the first or the last markable in its set, respectively). Also, the newly added markable will be rendered according to its new status with respect to the relation: since in the HTC sample annotation, italic font is defined for coreference-level markables participating in coreference sets, the markable will be rendered in italics (Figure 9). Finally, the attribute window is also affected by the operation: Since adding a markable to a markable set basically means setting the markable's corresponding attribute value to the ID of the markable set, the value for this attribute (here: **Coref.class**) changes . If the currently selected markable was not previously in a markable set and it was only through the addition of a markable that a set came into existence (as in Figure 9), the currently selected markable's rendering style and **Coref.class** attribute will also be changed.

The procedure described above is the most simple case of adding a markable to a markable set. Things are more complex if the markable to be added is *already participating in a another markable set from the same relation*. If this is the case, you can decide to *adopt* the selected markable, or to *merge* both sets.

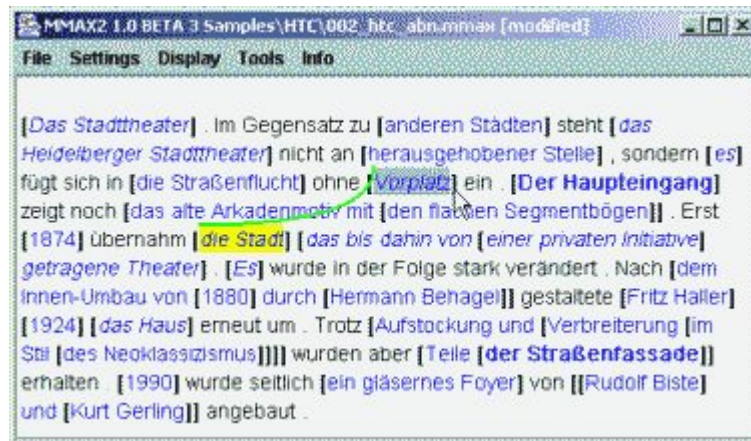


Figure 9: Relation Modification: Newly Created (Two-Element) Markable Set

The set of which the selected markable is part is displayed in a dashed line style. In addition, a popup menu with (at least) two menu items will appear (Figure 10). The texts of both items can be customized (in

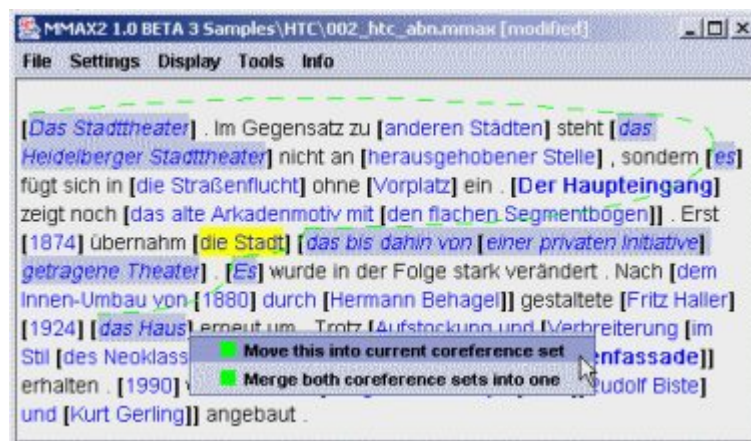


Figure 10: Relation Modification: Adding a Set-Member to another Set

the annotation scheme) to reflect the semantics associated with the respective action. In the sample HTC annotation, the first menu item will be *Move this into current coreference set*. Selecting this item will *adopt* the selected markable, i.e. it will be removed from its current and added to the new markable set (Figure 11). If the markable set that the 'adoptee' markable was part of ceases to exist as a result of the markable's removal, all set-related attributes and rendering styles will be removed from the left-over markable as well. The second menu item will be *Merge both coreference sets into one*. Selecting this item will *merge* the sets of the markable to be added and the set the currently selected markable participates in (if any) (Figure 12). You can **remove** any markable except the currently selected one from a markable set. If you want to remove a markable from a markable set, select it with the *right* mouse button. A popup menu with the corresponding menu item will appear. The text of this menu item can be customized (in the annotation scheme) to reflect the semantics associated with this action. In the sample HTC annotation, the menu item will be *Unmark as coreferent*. In addition to the text, the menu item will also contain a small box

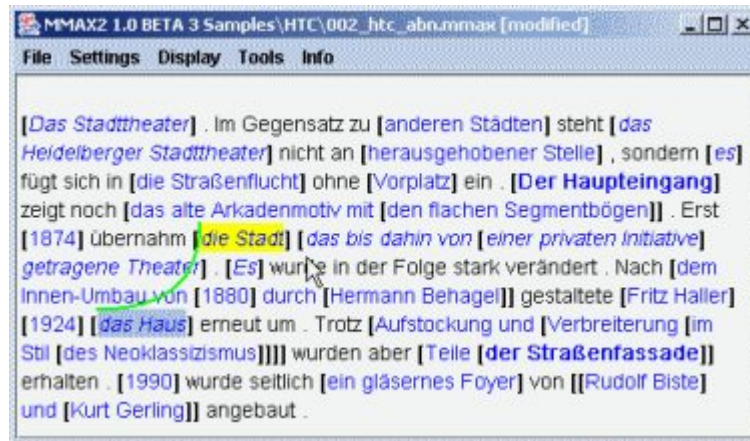


Figure 11: Relation Modification: Result of Markable Adoption

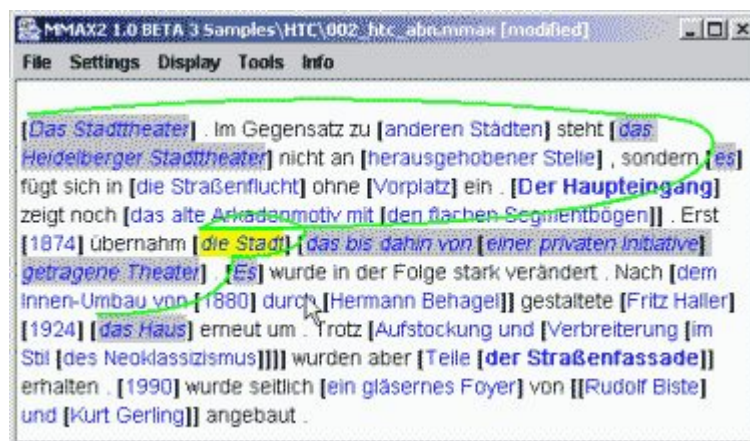


Figure 12: Relation Modification: Result of Markable Set Merge

in the same color as used for graphically rendering relations of the respective type. Select this menu item by left-clicking it, and the markable will be removed from the markable set. Its set-related attributes and rendering styles will be removed, and the graphical links to other members of the left set will disappear. If the removed markable was the second but last in the set, the set ceases to exist, and set-related attributes and rendering styles will be removed from the left-over currently selected markable as well.

4.4.2 Adding to and Removing from a *MARKABLE_POINTER*-type Relation

Due to their non-transitivity, *MARKABLE_POINTER*-type relations are quite simple to manage. To **add** a pointer relation from the currently selected markable (the *source* markable) to some other markable (a *target*), select the latter with the *right* mouse button. Whether the currently selected markable can be the source of a pointer relation, and whether the selected target markable is allowed depends on whether the markables meet certain conditions defined in the annotation scheme. In order to be able to point to a bridging antecedent, a markable has to have the value *bridging* in its **Type** attribute. If **Type** has some other value, the **Bridging_antecedent** attribute will not be visible in the attribute window, and the relation thus not allowed for the markable (Figure 13). Changing the value of **Type** from *none* to *bridging* will cause

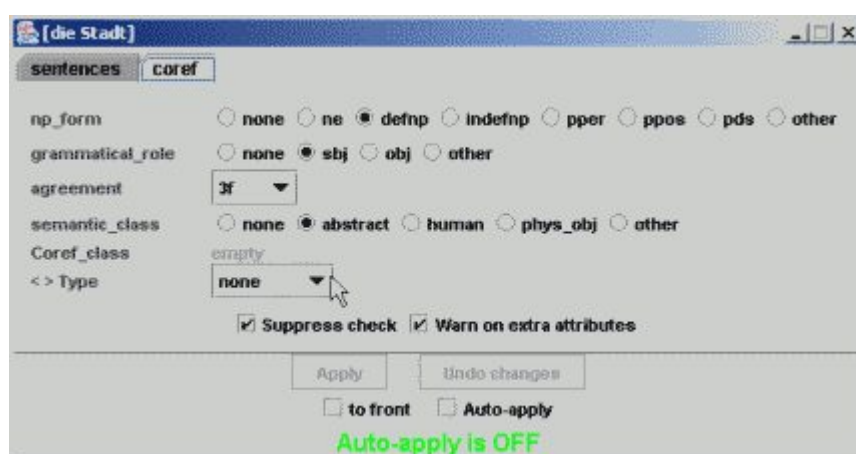


Figure 13: Relation Modification: **Bridging_antecedent** not accessible

an additional attributes to appear in the attribute window (Figure 14). After clicking the *Apply* button, the currently selected markable will be annotated as *bridging*, and will be a valid source markable for the **Bridging_antecedent** pointer relation. For potential target markables, the annotation scheme only defines that they have to be markables from the *coref* level.¹² If source and satellite markable fulfill the respective conditions, a popup menu with the corresponding menu item will appear. The text of this menu item can be customized (in the annotation scheme) to reflect the semantics associated with this action. In the sample HTC annotation, the menu item will be *Mark as bridging antecedent*. In addition to the text, the menu item will also contain a small box in the same color as used for graphically rendering relations of the respective type. In the sample HTC annotation, blue is used for pointers to bridging antecedents. Select the menu item by left-clicking it, and a pointer relation from source to satellite markable will be added. As result, a blue line from the source to the satellite markable will be drawn. Also, the ID of the satellite markable will be added to the **Bridging_antecedent** attribute of the source markable.¹³

¹²The annotation scheme can also define that markables from *other* levels can be target markables.

¹³Note that the attribute window will only show the number part of the ID, and not the *markable* name space.

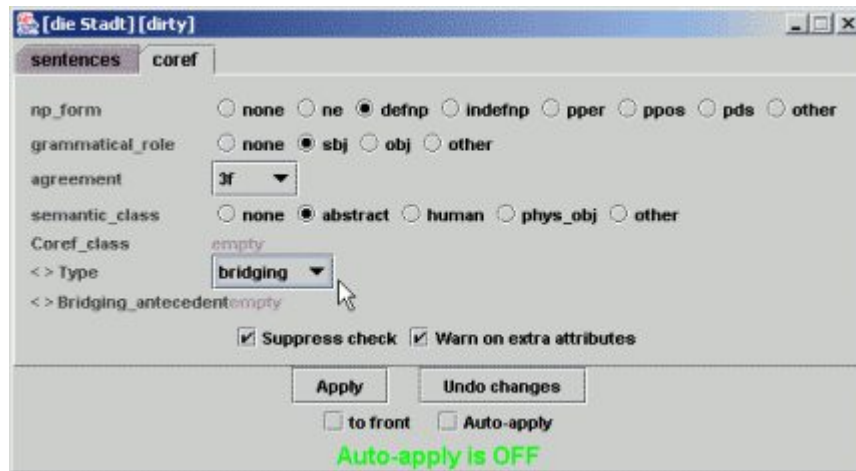


Figure 14: Relation Modification: **Bridging_antecedent** accessible after Attribute Modification

The following things have to be noted: For the HTC sample annotation, the annotation scheme defines that a markable can have at most *one* bridging antecedent. If a markable already has a bridging antecedent, adding another one is thus not possible. The size limitation, however, is optional for the annotation scheme, so that in principle pointer relations with arbitrarily many target markables are allowed. Also, since *MARK-ABLE_POINTER*-type relations are directed, targets themselves do not have any clue that they are being pointed at; consequently, adding a pointer from some markable to a target markable will *not* alter the latter's attributes.

To **remove** a pointer to a certain target markable, select it with the *right* mouse button. A popup menu with the corresponding menu item will appear. The text of this menu item can be customized (in the annotation scheme) to reflect the semantics associated with this action. In the sample HTC annotation, the menu item will be *Unmark as bridging antecedent*. In addition to the text, the menu item will also contain a small box in the same color as used for graphically rendering relations of the respective type. Select this menu item by left-clicking it, and the pointing relation will be removed. As a result, the satellite markable's ID is removed from the source markable's **Bridging_antecedent** attribute, and the blue line disappears.

5 Modifying Base Data

The term *base data* refers to that part of a MMAX2 project which underlies the actual annotation. i.e. the data *to which* annotations in the form of markables are added. In most cases, the base data for a single project will consist of one file containing the individual words in a simple XML format. Although *in principle* base data should be read-only, there are situations in which it becomes necessary to modify the base data. These situations include

- correction of spelling errors in transcribed text
- correction of tokenization errors
- insertion of specially marked *empty* elements as reference elements for markables
- splitting of incorporated clitics
- ...

In order to allow for these modifications, MMAX2 (from version 1.0 beta 3 on) has been improved to support *basic actions* for addition, deletion and modification of base data elements via the GUI. By default, base data editing is *disabled*. In order to enable it, check the item **Enable base data editing** in the **Settings** menu in the main MMAX2 window. Once base data editing is enabled, elements can be modified by holding the **Control** key and at the same time right-clicking on a base data element in the display. The mouse pointer will change to a *cursor* type as long as the *Control* key is pressed.¹⁴ After right-clicking a base data element, a popup menu with a series of options will appear. The individual options will be described below. After each modification, the currently selected style sheet is automatically reapplied in order to make the modifications visible. Also, the base data is marked as 'modified', and the **Base data** menu item in the **Save** menu (under **File**) is enabled. Use this menu item to save the modified base data. A backup copy of the current file will be created before it is overwritten, but note that only *one* backup copy is kept!

5.1 Modify an Element

This option allows to modify the text and (word-level!) attributes of a base data element. Upon selecting it, a small *Edit base data* window appears. This window contains in its upper part a text field with the selected base data element's *text*, and in its lower part a text field with the selected base data element's *attributes*¹⁵. The upper text field can be edited to reflect the modifications that you want to make to the element's text. **Note:** It is not possible to split an element by putting a space character into a word. Even though the element will appear as 'two' words in the display, neither part can be selected separately! Splitting an element into two (or more) must be done by modifying the existing element and adding new ones (cf. below). In the lower text field, space-separated attribute-value pairs can be added or removed. Both fields have an associated list box that contains text that has recently been entered in the respective fields. Selecting an entry from one box will set the text of the associated text field accordingly. If you are satisfied with your modifications, click the **OK** button to apply the modifications. Otherwise, click **Cancel** to leave the base data unaltered.

¹⁴If this does not happen, make sure that the display can recognize key events. Do this by left-clicking once somewhere in the display! This is necessary only once for each MMAX2 session.

¹⁵Note that these attributes have nothing to do with *markable* attributes! Base data element attributes are only used to store information about individual base data elements, e.g. that an element is a separator or an empty element.

5.2 Delete an Element

This option allows to completely delete a base data element from the project. In contrast to mere base data element *modification* (cf. above), element deletion can affect the integrity of existing annotations: If the element to be deleted is the first, last or only element in a markable, deleting it requires to adapt or delete the respective markable(s) as well. Therefore, a check is performed prior to deletion in which potentially affected markables are determined. If at least one such markable is found, the user is prompted to either explicitly acknowledge the modification of the affected markables, or to cancel the deletion process. If no affected markables are found, or if the user acknowledges the markable modification, the element is deleted. Note that base data element deletion does not affect the annotation if the base data element in question is somewhere *in the middle* of one or more markables, since markable spans are defined by their first and last elements only, and the are resolved by simply retrieving all intermediate base data elements.

5.3 Insert an Element

This option, finally, allows to insert a new base data element to the left or right of the selected element. Upon selecting this option, the small *Edit base data* window appears. This window contains in its upper part a empty text field into which the new base data element's *text*, can be entered, and in its lower part a text field for the new base data element's *attributes*. Both fields have an associated list box that contains text that has recently been entered in the respective fields. Selecting an entry from one box will set the text of the associated text field accordingly. After entering the new element's text and / or attributes, click the **OK** button to insert the new element. Otherwise, click **Cancel** to not insert anything. It is worth noting how IDs for manually added base data elements are created: First, the base data elements to the immediate left and right of the insertion position are determined. Then, the numerical part of the left neighbour's ID is subtracted from the numerical part of the right neighbour's ID. This difference is then divided by two and added to the left neighbour's numerical part. The resulting number is then used as the numerical part of the new ID to be created. In effect, this will produce base data element IDs whose numerical parts are not integers but doubles.

6 Tools

6.1 Markable Browser

The *markable browser* is a tool for visualizing and browsing markables on a certain level. A new instance of a markable browser can be opened by means of the respective menu item in the **Tools** menu. The browser will then be opened with default setting, displaying the markables from the top markable level¹⁶ in alphabetic order. It can also be opened by right-clicking a markable in the display and choosing the initial order.¹⁷ A markable browser consists of a scrollable list of markables. If it was opened by means of a right-click on a markable, this markable will initially be selected (Figure 15). The markable that is currently

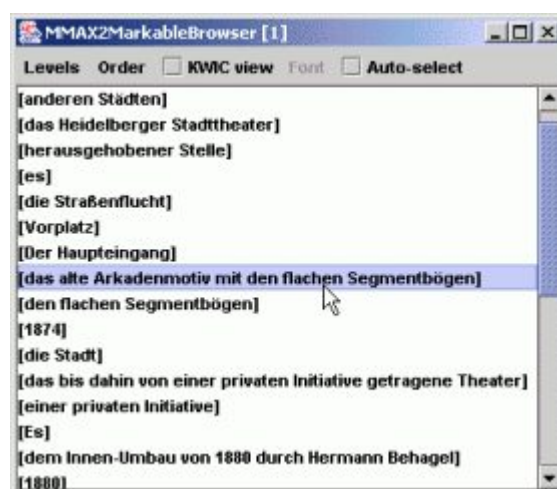


Figure 15: Normal Markable Browser View

selected in the list appears shaded in the MMAX2 display. Moving the selection in the list (by means of the cursor keys or the mouse) moves the shaded markable selection in the display. Double-clicking a markable in the markable browser has the same effect as selecting it in the display: its attributes are displayed in the attribute window, and any relations it participates in are rendered graphically. If the *Auto-select* box in the markable browser's menu bar is checked, a single mouse click or list cursor movement suffices to select the markable. This is particularly useful for stepping through the entire annotation without having to select individual markables with the mouse. The displayed markable level and the order of display can be changed by means of the **Levels** and **Order** menu, respectively. Note that several markable browsers (with different levels and/or display orders) can be open simultaneously! A special feature is the *KWIC* view: By checking the *KWIC view* box in the markable browser's menu bar, each markable will be displayed as a keyword within context (Figure 16). This view is available for both document and alphabetic order (Figure 17). If the *KWIC* view is selected, the **Font** menu becomes available. This menu allows to select the font used for the *KWIC* view. The default font is a fixed-size courier font, which will suffice in most cases. However, if e.g. Korean data is displayed, this menu item must be used to select some (at least more or less) fixed-size font. Without a fixed-size (i.e. non-proportional) font, the *KWIC* view will not work correctly. Each markable browser monitors the markables of the level it is currently displaying. Relevant changes (i.e. creation and deletion of markables) cause the markable browser to automatically refresh itself.

¹⁶As defined in the markable level control panel

¹⁷Note that this will only work if no other markable is currently selected.

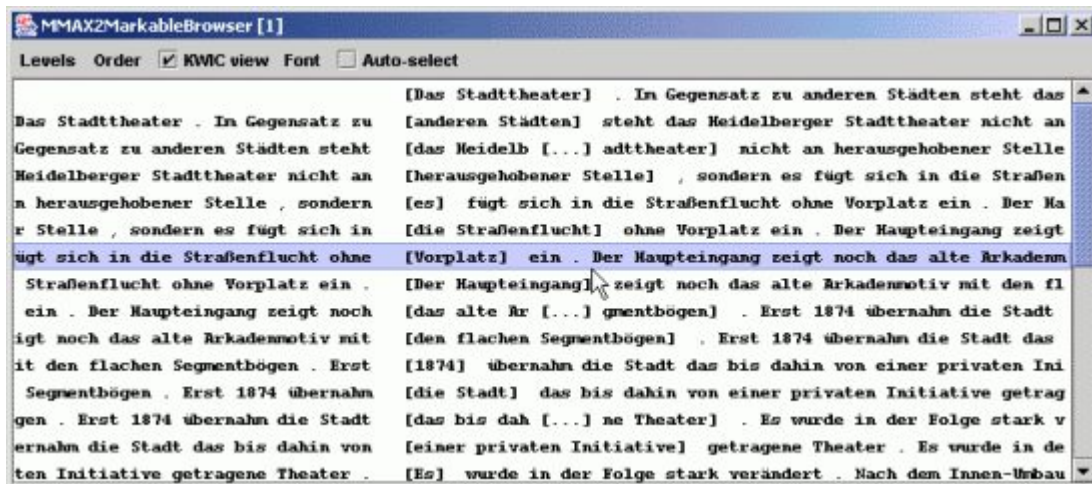


Figure 16: Markable Browser KWIC View: Document Order

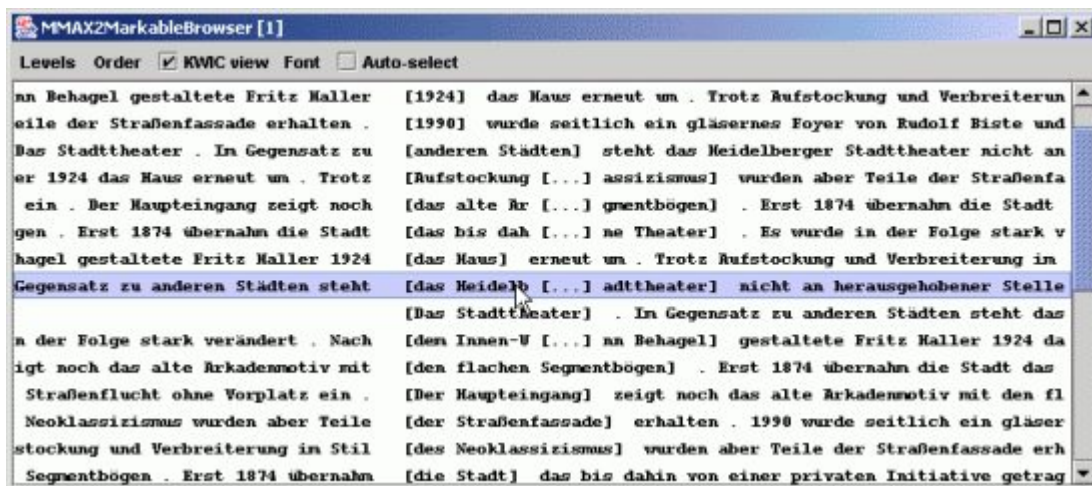


Figure 17: Markable Browser KWIC View: Alphabetic Order

6.2 Project Wizard

The MMAX2 Project Wizard (included in MMAX2 1.0 beta3 and later) can be used to create MMAX2 annotation projects from raw text files. Support for importing other formats (incl. XML) is not available yet, but will be added later. The wizard can be opened by selecting the *Project Wizard* item in the *Tools* menu in the main application window.

Overview Each MMAX2 project consists of several files:

- one *base data* file containing the individual tokens (mostly words) in a simple XML format
- for each markable level defined in the project
 - one *markable* file containing references to elements in the base data file and the associated attributes in a simple XML format
 - one *scheme* file containing the specification which attributes and which possible values can be applied to markables on the respective level
 - one *customization* file containing the specification of how markables are to be displayed depending on their attributes
- at least one XSL *style* file specifying the layout of tokens and markables in the MMAX2 display (see document `mmax2stylesheets.pdf` in the Doc directory).
- a file named `common_paths.xml` containing references to the directories where the above files can be found, and a list of all defined style sheets
- a `.mmax` project file containing references to all files comprising the project

The purpose of the MMAX2 Project Wizard is to assist you in the creation of a MMAX2 project, given that you have a raw text file that you want to annotate. In its current (early) version, the wizard does

- create the *base data* file by applying a simple tokenizer to your raw text input file
- create one or more markable levels, optionally also creating markables on the basis of structural information in the input file
- create customization files, default scheme files, and a (customizable) style file
- create the `common_paths.xml` and the `.mmax` project file

In a nut shell, the whole process consists of the following steps:

1. Open the MMAX2 Project Wizard.
2. Select the raw text input file and its character encoding.
3. Specify tokenization parameters (or leave the default), and tokenize the input text.
4. Create *at least* one markable level by specifying its name and basic display properties.
5. Specify the paths to which the project is to be written.
6. Create the project.

Using the Wizard After opening the wizard (by selecting the *Project Wizard* item in the *Tools* menu in the main application window), you will see a window containing four different sections. Each section contains the controls required for a particular step in the project creation process. These steps will be described in more detail in the following.

6.2.1 Input File Selection

The top window section (**Text Input File**) lets you select the raw text input file that you want to base your annotation project on. To select a file, click the small *Pick* button. A dialog window will open, and you can click your way to the desired file. Note that although files of all types are shown in the dialog window, only raw text files (in various character encodings, though), are supported in the current version. You can then specify the character encoding that you know (or assume) the input file to be in. For plain text files, the default setting US-ASCII is correct. For other (e.g. UNICODE formats), another encoding has to be selected. You can use the button labeled *Analyse File* to read a portion of the text in the encoding specified. Clicking this button will show a window with some input file statistics and the beginnings of the first 15 lines. Empty lines in the input are replaced by the special <EMPTY> tag. If the text is rendered correctly, the selected encoding can be assumed to be correct. Note that, if unsure, you can try several encodings by using the *Analyse File* button more than once. Each change to either the input file or the encoding requires that you re-analyze the file. Note also that you *must* click the *Analyse File* button at least once in order to be able to perform the next processing step.

6.2.2 Tokenization

The next window section (**Tokenization**) contains controls for setting the tokenization parameters for the input file. *Tokenization* refers to the process of splitting the input text into individual words and interpunction. Each token created in this process is assigned a unique ID and is then written to the project's *base data* file. The tokenization process is important because it defines the *granularity* of the annotation, i.e. which base data elements will be individually accessible for annotation.

By default, the input text is first roughly pre-tokenized by splitting everything separated by white space. The following, more fine-grained tokenization is controlled by the parameters specified in this Project Wizard window section. The text field labeled *Split leading* contains individual characters (separated by space) that, when found at the beginning of a token, you want to be separated to constitute a token themselves. You can edit the tokens to be cut off by adding or deleting individual characters in the text field. As long as the text field contains the " character, the option *Convert " to _oq_* is available. Checking this option (default) will cause a " character that is found at the beginning of a word (i.e. an *opening quote*) to be converted into _oq_. The text field labeled *Split trailing* works similarly for characters that, when found at the end of a token, you want to be treated as individual tokens. The period character (.), however, has a special meaning in this text field. As long as the *Split trailing* text field contains the period character, the two options *Use abbrev. list* and *Use abbrev. heuristics* will be available. Checking the first option causes the tokenizer to consult the file *abbrevs.txt* (in the MMAX2 root directory) before cutting off a trailing period character: If the current token is found in this (user-editable) abbreviation list, the period is *not* cut off. Checking the second option will apply a very simple heuristic for abbreviation recognition. The heuristic causes a trailing period *not* to be cut off when the current token contains another period character in non-final position. This is able to recognize abbreviations like *U.S.A* or *I.T.*

Important Note Obviously, the tokenizer in its current form is pretty simple, as it does only know some simple rules and cannot handle exceptions or more complex rules. In addition, it is inherently language-dependent. However, adding a more powerful *generic* or at least highly configurable tokenizer is definitely beyond the scope the project wizard, since that would result in an independent non-trivial application. Therefore, the project wizard has an additional option, which allows to *read a pre-tokenized file as input*. This file can be produced by whatever external tokenizer is considered the best for the current text, as long as it produces a file with one token per line. The built-in tokenizer can be skipped by simply checking the *Input file is one token per line* option. In effect, this will let the input file pass through unaltered. Note that using this option will prevent *any* modifications of the input file, including the pre-tokenization where lines are split at white space.

Click the *Tokenize* button to tokenize the selected input text file using the specified parameters (or none if the input is already tokenized). This will tokenize the entire text, and create a temporary `.tokens` file. In addition, a *Token preview* window containing a list of the first 5000 tokens will be displayed. If applicable, this list will contain opening and closing `<EMPTY>` and `<EOL>` meta-tokens that are automatically inserted by the tokenizer. These will *not* be included in the base data, but can be useful for automatic markable creation (cf. below). You can experiment with different settings by changing parameters and inspecting the tokenization result in the *Token preview* window. Note that the following steps (cf. below) will always use the result of the *last* tokenization process. Note also that changing any tokenization parameters requires a re-application of the tokenizer.

6.2.3 Markable Level Creation

Markable levels and the markables existing on them are of major importance in a MMAX2 project because the markables are the carriers of the actual annotation information. The third window section (**Markable Levels**) contains the means to create markable levels and to (at least partly) populate them with markables based on structural information in the input file (cf. below).

By default, only one (initially empty) markable level is displayed. Arbitrarily many levels can be added by clicking the *Add level* button. New levels will be added at the bottom of the list. The position of a level can be modified by using the arrow buttons associated with it. Clicking the *X* button on the right will delete a level.

A markable level is defined by specifying the following (partly optional) parameters.

- **Name:** A unique name must be specified for each level. The name must not contain any white space or other special characters, and should describe the type of annotation that the level is to contain.
- **Source:**¹⁸ This (optional) parameter can be used to automatically create markables on the respective level, based on structural information in the input file. At present, the following values are supported:
 - **NONE** (default): No markables will be automatically created.
 - **WORD:** One markable will be created for each word, i.e. for each token.
 - **EOL:** One markable will be created for each token sequence ending with a new-line character. If your input data contains e.g. one *sentence* per line, use this value to create a markable level with one markable per sentence. Note that this value is equivalent to the *WORD* value (cf. above) if you use a pre-tokenised input file.

¹⁸The range of values for this attribute will be improved in later versions, including support for *user-defined* meta-tags and ways to directly include external pre-processing components.

- **EMPTY:** One markable will be created for each token sequence ending with an empty line. Similar to the *EOL* value, this value can be used to create a markable level with e.g. one markable per *paragraph*. Alternatively, if you use a pre-tokenized input file containing sentence breaks, empty lines can be used to represent these sentence breaks.
- **Handles:** This (optional) parameter can be used to specify the type of *markable handles* used for markables on this level, if any. By default, no handles will be supplied.
- **Color:** This (optional) parameter can be used to specify the color in which markables on this level are to be displayed. Select a color by clicking in the white field and choosing a color. The default color is black.
- **Style:** This (optional) parameter can be used to specify the font attributes in which markables on this level are to be displayed. The default style is *plain*.
- **CR:** This (optional) parameter can be used to insert a new-line *after* each markable on this level. If markable handles are also specified for this level, the new-line will be inserted after the closing handle. If no handles are specified, the new-line will be inserted after the last token of the markable.

Several things are to be noted here: As mentioned above, the order of markable levels can be modified by means of the arrow buttons. The order of levels has a non-trivial effect on the MMAX2 main display, and on the way markable handles are visualized. As a rule of thumb, levels should be ordered in such a way that those with *higher-level*, i.e. *longer* markables are *below* those with shorter markables. Thus, e.g. a *sentence* level could be at the bottom, a *phrase* level on top of it and finally a *pos* level on top of that. Even without markable handles, this is the only reasonable ordering because level-dependent markable customizations are applied from bottom to top as well: Having, e.g. the *sentence* level as the top-most level would mean that this level's customizations override the customizations of the two lower levels, rendering them potentially invisible.

In addition, it is strongly recommended that for at least one level the *CR* feature (cf. above) be used. Adding occasional new-lines has a positive effect on the MMAX2 main display speed. Put differently, having the entire text of a project in one large paragraph slows down the display *dramatically*. The most natural level to be used for inserting new-lines is the level *sentence/turn* or *paragraph*.

6.2.4 Project Creation

The last window section (**.MMAX Project**) contains mainly controls for specifying output file names and paths for the project to be created. The text field labeled *Project File Name* contains the name suggested by the wizard for the `.mmax` project file. It is created on the basis of the input file name, and can be modified by the user. The same is true for the text field labeled *WORDS File Name*, which contains the name for the base data file to be created. The two options *Create customizations* and *Create default scheme files* control if additional project files are created. Checking the first one (default) will create a customization file for those levels that have a non-default value in their *Color* and *Style* attributes. Note that these two attributes are not available if *Create customizations* is not checked. Checking the *Create default scheme files* option (default) will create an (empty) default scheme file for each level. Uncheck this option only if appropriate scheme files (named `level_scheme.xml`) are available at the scheme path location (cf. below).

The text field labeled *Project Path* contains the main project path, i.e. the directory to which the `.mmax` project file will be written. Specify a project path by clicking the small *Pick* button and selecting the desired path from your file system. Apart from the project path, five other paths have to be set that point to special

directories containing base data, scheme, style, customization and markable files. Each of these paths can be set to a different location in your file system by clicking the associated *Pick* button. It is recommended that these paths point to different directories, so that the files are separated from each other, and are not all in the same directory as the project file. However, if you want to use the same path for each of the five subdirectory paths, you can click the *Use for all* button. When all paths have been set, click the *Create project* button to create the MMAX2 project. After creation, you are prompted to load the newly created project. Note that the *Create project* button will not be accessible if, e.g. an unnamed markable level exists in the *Markable Levels* window section. If you do not like the current appearance, you can simply modify some wizard settings and create and load the project anew. Check the *Overwrite without asking* option to prevent the wizard from asking before an existing file is overwritten.