

Wizard for Information Extraction from Tables

Nikola Milosevic, Rob Hernandez, Goran Nenadic

Overview

Wizard for information extraction from tables (WIET) is a GUI tool for generating information extraction rules (including structural, linguistic, syntactic and semantic) from tables in biomedical literature. The tool facilitates rule creation with the Graphical User Interface (GUI), performs information extraction and stores the data into the selected database. The motivation behind the project is to create an easy to use environment for people that do not necessarily need to be professionals in software engineering for developing table information extraction rules. The tool primarily should enable information extraction from tables in biomedical domain and is aimed to help professionals from biology, pharmacology and medicine domain to extract information easier from tables.

The methodology for information extraction and the flow of the wizard depends on the type of information that should be extracted. We identified three main data types: numeric, categorical and string. Numeric data is presented as one or more numeral divided with optional formatting symbols, such as plus-minus sign, brackets, dashes, etc. They can semantically present either single measured value or cumulative statistical data for certain population, such as mean values, standard deviations, ranges, percentages, etc. Categorical values are textual; however they are defined and limited to certain set of strings. String values are unlimited and can, but don't need to comply with certain syntactical rules.

Tables are more complex structures than text. Text is also necessary to analyse from the multiple perspectives and define rules for all of them (lexical, syntactic, semantic). We analyse tables from functional, structural, syntactic, lexical, pragmatic and semantic perspective. Information extraction rules are not defined only over one space, but in many cases they span over several of these perspectives. For example, if we are looking for certain word in the header of the table, we need to define the cues and syntax of the word/phrase we are looking for (lexical, syntactical) and location of the cell (functional and structural). Table extraction rules require the overlaps, but it is possible to structure the rule creation. In the beginning rule metadata are defined, such as Information class, pragmatic class of the table, possible and default unit for numerical values and possible categories for categorical. In the second step, user defines lexical cues and functional location where the cue should be searched for. At the end syntactic rules are defined.

Processing iterates through rules and applies them on the table data, extracting information and storing the extracted data to the database. Again, extraction methodology depends on the extraction value type and how the rules were defined. For numerical values, syntactic rules are applied. In the case of categorical data, possible categories are consulted. In the case of the string value, the content of the cell is either extracted as a whole or in case syntactic rules are defined, the part that matches at least one of them is extracted.

Requirements and pre-requisites

Wizard for information extraction from tables is written in Python 2.7. It uses Tkinter and Tix package. It also retrieves table data and stores extracted data to MySQL database. The database need to be filled with the data from TableAnnotator program, that transforms tables from papers in XML format to the relational data and annotates functions of the cells and structural relationships between the cells. In the case data is from biomedical domain, it is advisable to annotate the content of the cells with UMLS using MetaMap. These annotations can be later used by the Wizard for Information Extraction from Tables. System should contain the following for extracting information:

- Python 2.7
- MySql package for Python (recommended installation through pip)
- MySql server or XAMPP/LAMPP
- Java Runtime engine and Java Development Kit
- TableAnnotator
- MetaMap
- WIET

The program pipeline should include the following:

1. Collecting documents in one folder
2. Create database using TableAnnotator/DataBaseFileDrugs.sql
3. Running TableAnnotator over the files
4. Running Pragmatic classifier using TableAnnotator/src/classifiers/SpecPragmatic.java
5. Running UMLS annotation using
TableAnnotator/src/Annotation/MetaMapAnnotationMain.java
6. Developing extraction rules using WIET
7. Running WIET

The pipeline facilitates information extraction process from PMC or DailyMed XML documents to the final stage when the information is extracted and stored in the database in the structured format that can be easily queried.

Wizard flow

The flow of the wizard views is presented in the Figure 1. The wizard allows multiple projects to be defined. Each project should define one information extraction processing task. User is able to define multiple rules per project. Every rule consists of (1) metadata, (2) lexical cues with functional targets and (3) syntactic rules. As the rules are made, they are store on the file system as a set of files that describe user selections and crafted rules. The rules follow the recipe of necessary information for information extraction. The recipe can be seen on Figure 2. In the wizard controlled and uncontrolled string type of data were merged into the single data extraction data type.

We will explain in more details mechanism of creating and storing the rules by walk through the views of the wizard.

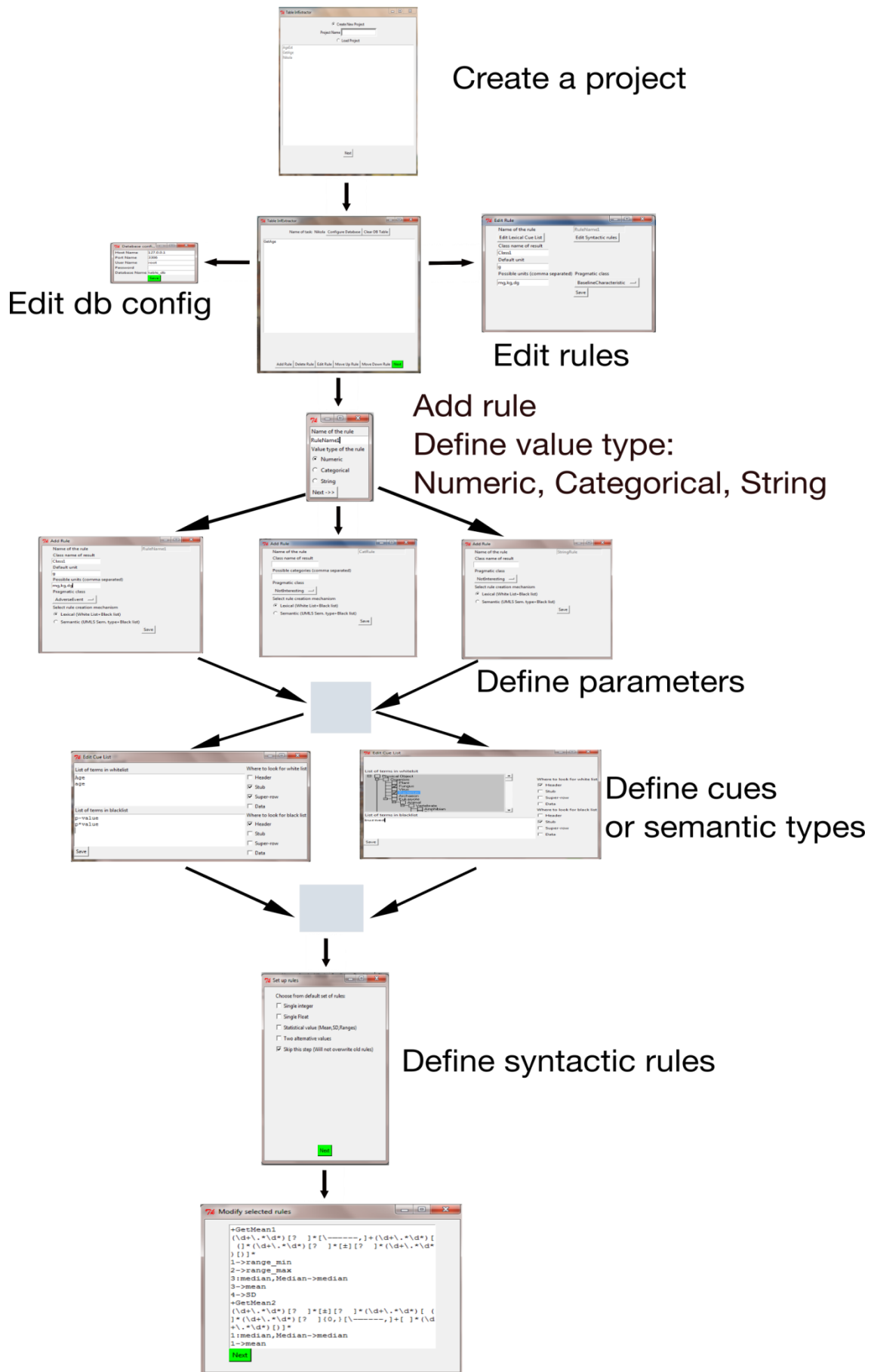


Figure 1: Wizard storyboard

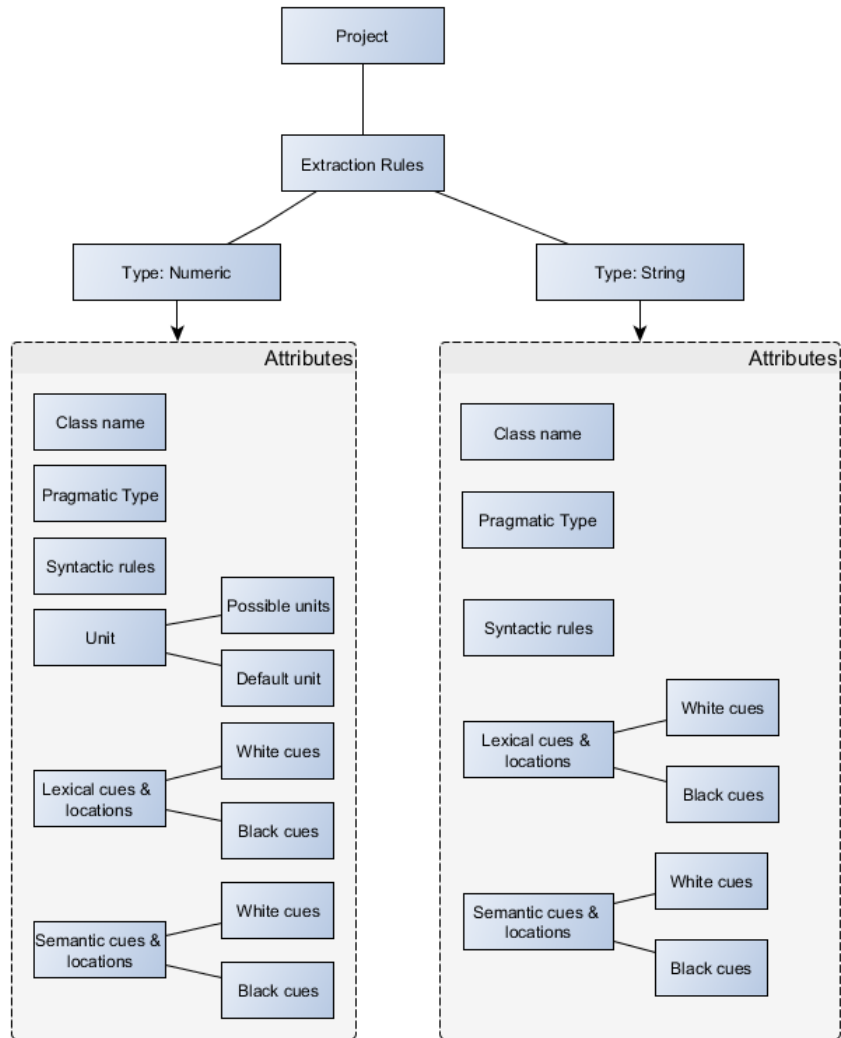


Figure 2: Information that need to be defined for information extraction task for each value type

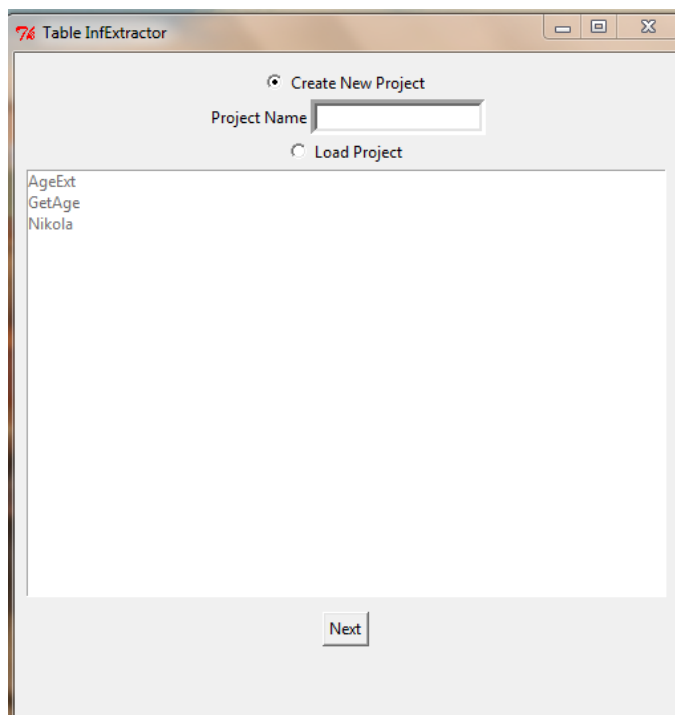


Figure 3: Project view in wizard

In the Figure 3 is presented the project view, which is the first view of the wizard. The purpose of this screen is for user to choose either he/she wants to define a new project or load and continue developing rules for some previously defined project.

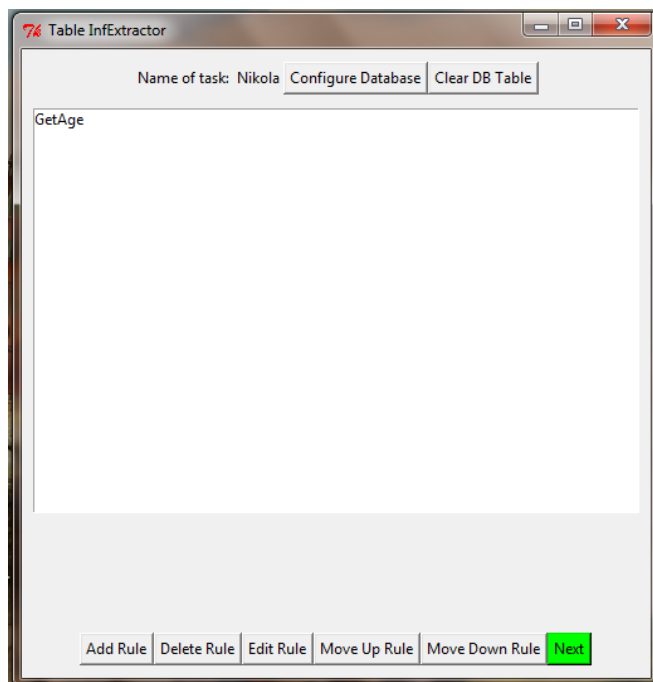


Figure 4: Main rule screen

In the Figure 4 is presented the main rule screen. On this screen user can add, edit, delete or rearrange the rules for information extraction. Rules are applied in the order they are ordered in this view. Therefore, rearranging may be in certain situations necessary. Also in this screen user can clear up the table there the extracted information is stored (Clear DB Table button) or configure database which will be used (Configure Database button), which will open the screen presented on Figure 5. On database configuration screen user should define host name, port number, username, password of the database and the name of the database.

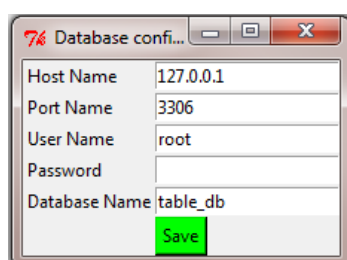
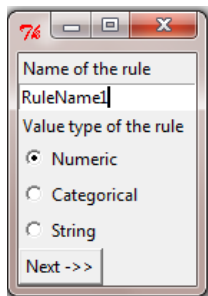


Figure 5: Database settings screen

When adding new rule, user will be presented with view presented on Figure 6. In this screen user need to define the name of the rule and the type of the data he/she will be looking for. The rule cannot be later renamed neither value type can be changed. All other characteristics of the rule can be edited. However, these two characteristics define the fundamentals of the rule and cannot be edited. If user made a mistake, he/she will need to remove and add rule again. Rules can be looking for numeric, categorical or string values. Depending on the user choice, the diagram flow will differ based on the recipe presented on Figure 2. For some data types certain information are required,

while for the others are not. The example can be unit which is required for numerical values, but not for string and categorical values.



7% Add Rule

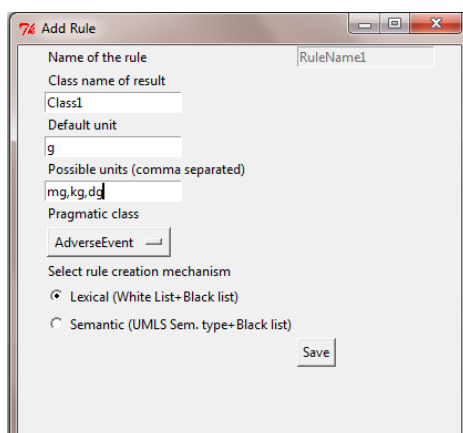
Name of the rule
RuleName1

Value type of the rule

☒ Numeric
☐ Categorical
☐ String

Next ->>

Figure 6: Adding new rule - 1st step



7% Add Rule

Name of the rule
RuleName1

Class name of result
Class1

Default unit
g

Possible units (comma separated)
mg,kg,dg

Pragmatic class
AdverseEvent

Select rule creation mechanism

☒ Lexical (White List+Black list)
☐ Semantic (UMLS Sem. type+Black list)

Save

Figure 7: Adding numerical rule

When adding numerical rule it is necessary to define information class of the result, default unit, possible units, pragmatic type of the targeted tables in which the information can be found and rule creation mechanism. Rule creation mechanism determines how the cues will be created. There are two types of supported cue creation mechanisms: (1) lexical and (2) semantic. In lexical cue creation mechanism user defines white and black list of lexical cues with target location of them (header, stub, data, super-row). On the other hand, semantic cue creation mechanism uses semantic resource, in this case UMLS and MetaMap annotations of the content of the table to generate white list.

Figure 8: Creation of categorical rule

The views for defining metadata for categorical and string data are presented on Figures 8 and 9. None of them contains unit definition. However, categorical data require possible categories to be defined.

Figure 9: Creation of string rule

On the Figure 10 is presented a screen for defining lexical cues. User need to define white cues – cues that would trigger extraction of the cell's content; and black cues – cues that even if the white cue is triggered would discard the extraction of the cell's content. For both white and black list of cues user need to define the target where the cues will be looked for.

Figure 10: View for defining lexical cues

On Figure 11 is presented view for defining lexical cues using semantic resource (UMLS). It differs from the default view that white list contains semantic types from UMLS and user can select semantic types that he/she wants to look for. Instead of using content of the cell, in this case will be used annotations of the content of the cells.

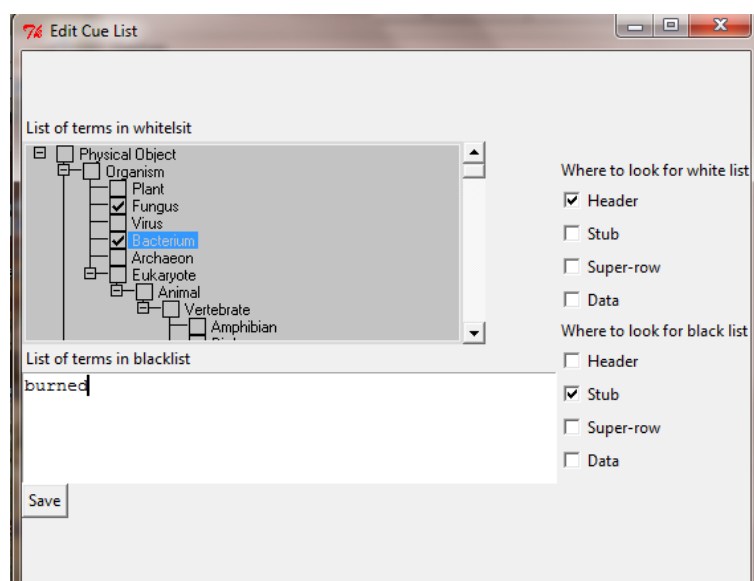


Figure 11: View for defining lexical cues using semantic resource

The final step is to define syntactic rules. On the Figure 12 is presented view for choosing one of the default sets of syntactic rules. Once selected, they can be modified in a view presented on Figure 13.

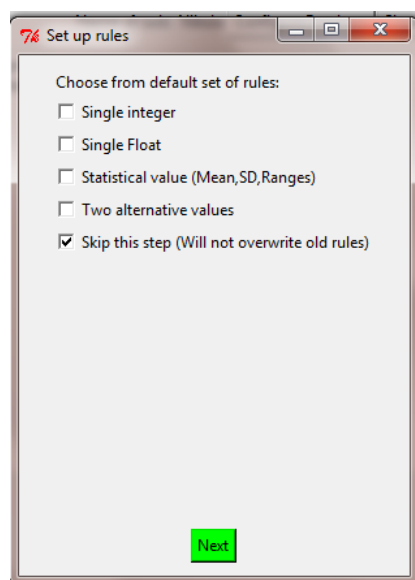


Figure 12: View for choosing syntactic rules starting point

Syntactic rules have a purpose to extract exact value that may be just a part of the cell's content selected by lexical cues. For example author of the table may present means, standard deviation and ranges in the same cell, however, user wants to break them up and store them in the database structured separately.

We developed meta-language that uses regular expressions to define syntactic rules. Definition of syntactic rule starts with the line that contains plus (+) symbol in the beginning and the name of the

rule. Second line is regular expression of the rule. It is advisable to group in brackets the parts of the selections that one want to store. The rest of the lines define semantics of the regular expression groups, starting with index 1. Syntax starts with the group number, arrow symbol (->) and the name of the semantics of the value that will be stored in the database with the value, as option (look output format later on). Index can be followed by semi-colon (;) and list of coma separated values, for which algorithm will be looking in headers and stubs. This serves for defining different semantic for some part of the value in case it is explicitly stated in the headers or stubs. Otherwise, it would use index of the group, followed by the arrow as default. For example, imagine rule states the following two lines:

```
3:median,Median->median
```

```
3->mean
```

The rule will store third group from the regular expression with median in case it contains words *median* or *Median* in stub or header. Otherwise it will store it as mean value.

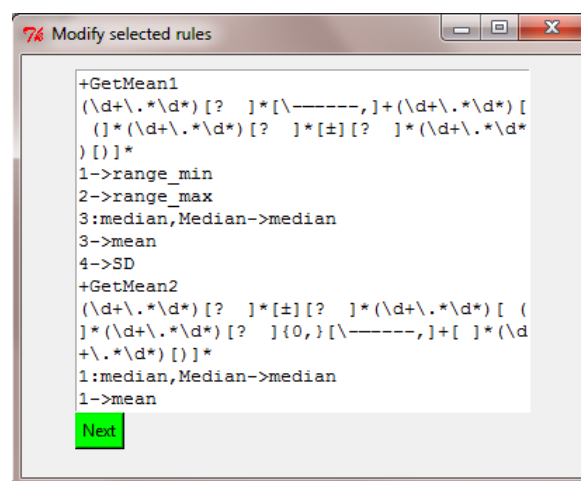


Figure 13: View for modifying syntactic rules

Next button will close the window, store rule files in appropriate folders and return to the main rule creation view. From there user can initiate processing.

Processing methodology

The processing methodology depends on whether or not semantic resource was used in order to create a cue list.

In case semantic resource was not used and lexical cues were provided by the user, our algorithm iterates over the defined rules. For each table in the database it checks metadata, such as pragmatic class of the table. If the pragmatic class is different than expected, it disregards whole table, not analysing any of its cells. Lexical cues have defined white list and blacklist with possible locations of these cues. Firstly white list is checked. Since database table generated by table annotator already contains table headers, stubs and super-rows in columns, the algorithm only needs to check these columns for analysed table cell. In case multiple headers, or stubs, they are concatenated in the database. If the lexical cue is found on the target location, the algorithm checks location and cues from the black list. In case cue from the black list exist, the cell is disregarded and no further

processing is done on that cell. In case cue from black list does not exist, algorithm moves to syntactic analysis. Algorithm iterates over the defined syntactic rules. The rules should be ordered so the most complex ones should be on top and processed first. Once the algorithm finds the best matching rule, it uses it, processes data according to it and disregards the following ones. The rule is splitting the data into the groups using regular expression and to each group is associated meaning. The association with the meaning is called semantic explanation for certain group (the one enumerated). In case the some rule's semantic explanation contains cues in square brackets ([or]), the cues are looked for in header and stubs. If the cue is found, algorithm will apply that semantic explanation to the extracted group. Otherwise, it will use explanation without cues. The groups that were matched with syntactic rules and its explanations are extracted. In case extracted information are numerical, cell, headers and stub is searched for mentions of possible units. In case some of the possible units are matched that unit is extracted. Otherwise, default unit is stored in the database.

In case semantic resource is used for defining cues, the processing is executed differently, since semantic annotations are stored in stand-off manner in the database. The stand-off annotations are only related to the cell's content, so headers, stubs and super-rows annotation cannot be accessed directly. The processing starts with selecting cells that are in appropriate location (selected by user) that contains defined annotation. In case the header cell is annotated, the column that is defined by that header is selected and the data cells are analysed. In case stub is annotated, row that is defined by that stub is selected. When the super-row is annotated, rows between the selected and the next super-row are selected and analysed. Each selected data cell is checked against black cue list. Cues are looked at places that were defined by the rule author. In case cue from the black list is not on the selected places, the algorithm processes content of data cells by using syntactic rules. Syntactic processing is the same as previous case, when the semantic resource was not used.

Output format

The output of the Wizard is database table in the same database that was used for input. The table contains PMC number of document and Table order, so the table can be unambiguously identified. Further, table contains name of the information class. Value option is retrieved from the explanations in syntactic rules. Source is information that further defines the value. In case information class was defined in stub or super-row, header is stored in source attribute; otherwise stub value is stored in this attribute. Value attribute contains extracted value. Unit is used only with numerical information for storing unit. Cue rule and syntactic rule are field manly used for debugging storing the name of the rules that extracted given information.

Extracted information
PMC_id
Table_order
InformationClass
ValueOption
Source
Value
Unit
CueRule
SyntacticRule

Figure 14: Database table of the output