The Glozz platform: a corpus annotation and mining tool

Antoine Widlöcher, Yann Mathet

The Glozz Platform: a Corpus Annotation and Mining Tool

Antoine Widlöcher[1,2,3]
antoine.widlocher@unicaen.fr

Yann Mathet[1,2,3]
yann.mathet@unicaen.fr

[1]Université de Caen Basse-Normandie, UMR 6072 GREYC, Caen, France
[2]ENSICAEN, UMR 6072 GREYC, Caen, France
[3]CNRS, UMR 6072 GREYC, Caen, France

ABSTRACT

Corpus linguistics and Natural Language Processing make it necessary to produce and share reference annotations to which linguistic and computational models can be compared. Creating such resources requires a formal framework supporting description of heterogeneous linguistic objects and structures, appropriate representation formats, and adequate manual annotation tools, making it possible to locate, identify and describe linguistic phenomena in textual documents.

The Glozz platform addresses all these needs, and provides a highly versatile corpus annotation tool with advanced visualization, querying and evaluation possibilities.

Keywords

Annotation formats and tools, Corpus Linguistics, Natural Language Processing

1. INTRODUCTION

More and more works in the linguistics, computational linguistics or Natural Language Processing (NLP) fields manifest an increasing interest for corpus studies. Through a wide range of approaches, the need for a systematic confrontation between models and corpora makes it necessary to have – and consequently, to produce – reference annotations to which linguistic models can be compared. Such reference corpora are also necessary for machine learning, to automatically learn models, and for evaluation tasks, to evaluate the results of NLP systems.

The elaboration of such annotations is a complex process which requires adequate formal grounds, encoding standards and dedicated applications. Despite the availability of several annotation tools, different requirements, especially in terms of abstraction, genericity and ergonomics, are overall not satisfied, as we will see in section 2.

The Glozz platform[1] [24], which will be presented in this paper, takes these constraints into account and provides a highly configurable environment, usable for corpus annotation and mining of various linguistic phenomena.

In order to satisfy the requirements of genericity and to support consequently the annotation of heterogeneous linguistic objects (in terms of structure, granularity...), Glozz relies on an abstract metamodel presented in section 3.

Given a specific linguistic model conforming to this metamodel, locating, identifying and describing linguistic objects in texts require adequate annotation tools. The incremental annotation process, Glozz GUI (presented in figure 5), as well as its main annotation features and tools, will be presented in section 4 and 5.

The annotation process, as well as the subsequent use of annotated data, require the ability to access information featured by the corpus. Glozz allows easy access to this information through different "navigation" tools and, in particular, by the mean of a powerfull query language called GlozzQL, which will be presented in section 6.

Building reference corpora makes it also necessary to align annotations and to measure agreement among annotators, in order to test the reliability of the annotated resources. Glozz features for alignment and agreement measure are presented in section 7.

Technologies, interoperability and availability will be discussed in section 8.

Presenting some recent research projets using Glozz, the last section will also mention its main strengths and weaknesses, and announce our roadmap.

2. STATE OF THE ART

2.1 A well-known problem

Several works and studies reveal the need for guidelines and tools to ensure production, long-term availability, inter-

[1]Glozz was initially developed within the Annodis project [20], supported by the french Agence Nationale de la Recherche (ANR). Glozz has also been supported by the french Contrat de Projet Etat-Région (CPER) and the Région Basse-Normandie. The URL address of its website is: `http://www.glozz.org`.

operability and efficient use of annotated corpora. Scientific events focus on these needs and reveal their importance for the NLP community. Let us just mention, for example, the LAW workshops [7] the LRT Standards workshop [3], the XBRAC workshop [25] or the ACL workshop on Discourse Annotation [22].

Among the works on linguistic annotation, three main topics may be identified : 1) annotation encoding, 2) annotation merging, mining and querying and 3) annotation process and tools.

Several works concern formats, standardisation and interoperability and aim to provide standards on which various annotation works could rely. From this point of view, guidelines developed by the Text Encoding Initiative (TEI)[2] are important contributions. Other works intend to provide unified means to represent heterogeneous linguistic annotations, and led to well-established formats. Works on LAF [8], GraF [9], annotation graph [2] and OLiA/PAULA [4] are, from this point of view, particularly important.

Other works mainly focus on annotation mining and querying. Due to dependencies between various linguistic phenomena, at different levels (word, syntagm, phrase, discourse unit, etc.) and from different points of view (morphological, syntactic, semantic, etc.), such mining and querying make it necessary to merge highly heterogeneous data, in order to explore them in a unified way. Frameworks such as CorpusReader [11] or Annis [4] enable integration of annotation data from different annotation tools and tag sets. As a consequence, integrated querying is possible, using unified query languages such as AnnisQL [4].

Other works aim to develop annotation tools and to make the annotation process as simple and reliable as possible. Some of these tools are embedded in more general purpose environments, made to deal with versatile linguistic data. They are consequently rather customizable. Widely used in the NLP community, the Gate platform [5] incorporates such a manual annotation tool, which can be configured by a user-defined annotation model. In the same way, the manual annotation tool available for the popular UIMA framework[3] has to be mentioned. Integrated to the well-known Protégé[4] environment, the annotation plugin Knowtator [18] takes advantage of ontologies to define rich annotation models.

Other standalone tools, on the contrary, are fully devoted to manual annotation. They provide more advanced features and enable annotation of more complex structures. Some of them, such as RSTTool [17], devoted to specific theories, are designed to deal with very specific annotation models, and may not be used for unrelated annotation campaigns, dedicated to other linguistic phenomena. In the same way, available tools for syntactic annotation, may not be used for annotation tasks devoted to other objects, in order to annotate, for example, argumentative structures. We deliberately focus here on more multipurpose tools.

Other works aim to provide more generic annotation tools. Nevertheless, their adaptability is often limited, not to linguistic categories, but to "structural types" : some of them mainly focus on text segmentation (annotation of textual segments); others focus on link-oriented annotation (annotation of relations between textual objects). Focusing on segment annotation, UAM Corpus Tool[5] offers a flexible annotation environment for which complex and hierarchical annotation scheme can be defined. The integration of statistical tools makes it possible to mine available annotations. One of the noticeable features of Wordfreak [14] concerns the multiplicity of views it offers on text and annotations (*in situ* view, concordancers, representation as a tree...). Particularly used for anaphora/coreference annotation, MMAX [15] is nevertheless a customizable tool for creating and visualizing annotations on multiple levels. It pays particular attention to relations between segments, which can be graphically annotated, conforming to a user-defined annotation model. PALinkA [19] aims at providing a general purpose annotation tool. It allows segment or relation annotation, and user-defined annotation models. The importance it attaches to already available annotations, on which the annotation process may rely, has to be noted. The SLAT browser-based annotation tool [16] insists on the necessity of abstract and multipurpose environments, to take variety of annotation tasks into account. It allows segment and link-based annotation, relying on customizable tag sets. An other recent web-based annotation tool called BRAT [21] can be set up to support various collaborative annotation tasks involving segments and relations at quite low granularity levels. Some of its key strengths are its high-quality visualisation features, the embedded system for checking annotations, and the interoperability with NLP or machine learning systems to support manual annotation.

2.2 Yet another tool ?

Despite the availability of several annotation tools, and even if some of them are highly customizable, it must be noted that they do not meet the needs of all varieties of linguistic annotations. In a way, it is necessary to bridge the gap between broad enough formats (LAF, GraF, etc.) and too specialized tools. From this point of view, the following limits have to be emphasized:

1. Priority is globally given to the annotation of objects at quite local granularity levels, making it difficult to represent and then to explore, for example, structures at discourse level.

2. Available tools are often restricted to a particular theory or to a specific class of linguistic structures (segments, relations, chains...). Annotation tasks involving heterogeneous structures are then made difficult.

3. When a class of structure (segment, relation...) is available, ergonomic limits may nonetheless make its annotation process uneasy. For example, it is uneasy to express and visualise relations between textual segments, when annotation only consists in the attribution of a same ID to linked elements. A graphical artefact is necessary.

4. Strong constraints often restrict usage of the available classes of structures. For example, in the case of annotation of textual segments, embedded or overlapped structures are often not allowed or not adequately represented.

[2] `http://www.tei-c.org`.
[3] `http://incubator.apache.org/uima/`
[4] `http://protege.stanford.edu`.
[5] `http://www.wagsoft.com/CorpusTool/`.

5. It may be impossible to represent complex structures using only segments and relations (in particular if relations can only link segments). For example, annotation of enumerative structures [6] or complex discourse units [1] makes it necessary to link sub-structures and not only primary textual data. Moreover, linking non-adjacent elements in sub-structures is often needed, and can not be represented by embedding segments.

6. Annotations (segments or relations) are "labeled" to state the relevant information concerning identified objects. Simple tagsets are not expressive enough to meet requirements of rich annotation models, and it is necessary to implement richer labelling possibilities, using for example feature structure-based models.

Above mentioned limits mainly concern the expressive power of the data model. Other important limits concern the annotation process and the annotation environment:

1. Multiple views (*in situ*, concordancers, trees, graphs, etc.) on the same data are often required. And any of these views should notify any change to other views. However, available tools often give priority to one of these paradigms or feature multiple views which do not "observe" each others.

2. Querying/Mining of annotations should not be considered as a post annotation possibility. Indeed, at any stage of an annotation process, annotators need to rely on some specific existing configurations to which query languages can give access.

Most of these requirements, in terms of abstraction, genericity and ergonomics, have already been implemented in other tools, but not, to our knowledge, in a same tool. The general-purpose Glozz platform takes these constraints into account and provides a graphical and highly configurable annotation environment, usable for corpus annotation and exploration of various linguistic phenomena.

3. UNDERLYING MODEL

Due to the diversity of linguistic phenomena, corpus linguistics and NLP studies lead to a variety of models, theories and formalisms. This diversity often results in heterogenous description formats and annotation tools, each approach developing its own framework.

However, deep interactions between the different kinds of linguistic phenomena and paradigms make it necessary to define common frameworks and standards where most kinds of objects, resulting of heterogenous models or paradigms, can be described, in order to compare or combine various approaches.

3.1 The Unit-Relation-Schema metamodel

Glozz relies on an abstract metamodel, called URS (for Unit-Relation-Schema), originally coming from [23], which provides an adequate framework, unrestricted to a particular theory or to a specific class of objects, allowing description of existing or future linguistic models.

This metamodel, represented by the figure 1, relies on three abstract categories of *elements*: *units*, *relations* and *schemas* which will be described below.

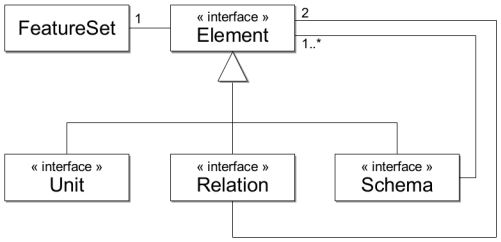

Figure 1: UML class diagram of URS

3.1.1 Metamodel and models

Within the general framework defined by the meta-model, specific models can be expressed, depending on the linguistic theory or approach. Each specific model declares available linguistic object types (identified by the theory) and explicits the way their instances have to be characterized (or labeled). The specialization of URS for a specific campaign will be presented in section 5.

3.1.2 Element

All available linguistic objects, called *elements*, may be *units*, *relations* or *schemas*. All of them are characterized by a *type name*, which explicits their linguistic category, and a *feature set*, representing their properties. Type names, expected features for a given type and possible values for these features depend on the specific user-defined model designed for a campaign.

3.1.3 Unit

Units, illustrated by figure 2, are textual segments, sequences or spans, of any size.

lectus tortor, dignissim sit amet, adipiscing nec, ultricies sed, dolor. Cras elementum ultrices diam. Maecenas ligula massa, varius a, semper congue, euismod non, mi. *Proin porttitor, orci nec nonummy molestie, enim est eleifend mi, non fermentum diam nisl sit amet erat.* Duis semper. Duis arcu massa, scelerisque vitae, consequat in, pretium a, enim. Pellentesque congue. Ut in risus volutpat libero pharetra tempor. Cras vestibulum bibendum augue. Praesent egestas leo in pede. Praesent **blandit odio** eu enim. Pellentesque sed dui ut augue blandit sodales. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Aliquam nibh. Mauris ac mauris sed pede pellentesque fermentum. Maecenas adipiscing ante non diam sodales hendrerit. Ut velit mauris, egestas sed, gravida nec, ornare ut, mi. Aenean ut orci vel massa suscipit pulvinar. Nulla sollicitudin. Fusce varius, ligula non tempus aliquam, nunc turpis ullamcorper nibh, in tempus sapien eros vitae ligula. Pellentesque rhoncus nunc et augue. Integer id felis.

Curabitur aliquet pellentesque diam. Integer quis metus vitae elit lobortis egestas. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi vel erat non mauris convallis vehicula. Nulla et sapien. Integer tortor tellus, aliquam faucibus, convallis id, congue eu, quam. Mauris ullamcorper felis vitae erat.

Figure 2: Units

A part of speech annotation task could for example define a unit type *word*, and two features to represent its morpho-syntactic tag and its lemma (the former having a predefined

set of possible values). An annotator could then annotate all words of a text, each of them, instance of the type *word* (derived from the abstract meta-type *unit*), having its own tag and lemma values. Named entities, propositions, sentences, topical units, argumentative segments, sections or the whole document give other examples of possible units, at higher granularity levels.

3.1.4 Relation

Relations, illustrated by figure 3, designate links (directed or not) between two *elements*.

If relations between units are widely used, it must be noted that the possibility of relations linking whatever elements, including schemas or relations, significantly improves the expressive power of the metamodel.

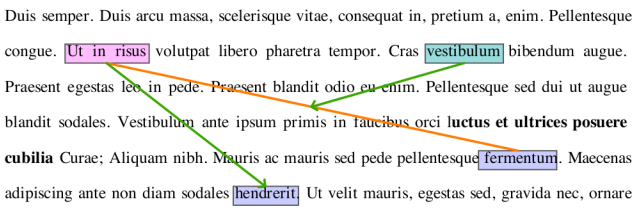

Figure 3: Relations

At a syntactic level, *dependancies* could, for example, be represented by directed relations. At a higher granularity level, a rhetorical annotation task would make use of directed relations to represent *causality* and benefit from symmetric relations to represent *contrast*, between propositions delimited as *units*, or between more complex patterns represented by *schemas*.

3.1.5 Schema

If both previous elements are quite common (even if designated otherwise), the schema category, illustrated by figure 4, is more original. Schemas are used to represent complex configurations or patterns involving any number of elements (units, relations or sub-schemas).

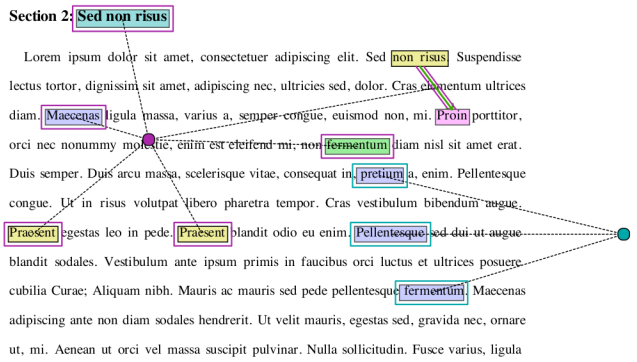

Figure 4: Schemas

Coreference chains could, for example, be represented by a set (or a path) of binary relations, all grouped in a schema, whose features could describe the common reference. *Enumerative structures* provide a more complete example. Composed of a set of consecutive *item* units, the enumeration is usually embedded in a larger structure (an *enumerative structure*), introduced by a *header* which is thereby in an *introduction relation* with items. In addition, *inheritance* relations between header and items, and *similarity* or *contrast relations* between items, often complete this quite frequent textual configuration.

3.2 Metamodel scope

This very abstract metamodel enables the representation of very many (if not all) configurations. If linguistic objects can then often be representend within the URS framework, it must be noted that "non-linguistic" information can also be encoded in this way.

For example, Glozz also uses the URS metamodel to represent document structure (titles, section titles...) and typographical information (ordered lists, emphasis...). Thus, with Glozz, everything but the raw text, is an annotation.

Represented in a unified way, all the available information can be used or mined in a unified way, as we will see in section 6.

3.3 Granularity and topology

The proposed meta-model makes no hypothesis on granularity level of elements or on distance between elements involved in relations or schemas. In particular, it meets the requirements of annotation at discourse level, which are overall not satisfied by multipurpose tools.

Furthermore, this data model accepts embedded and overlapping structures, as illustrated by the figure 2. More difficult to represent, the latter are often not well supported by annotation tools.

3.4 Standoff representation of annotations

Glozz uses standoff annotations. Units are linked to textual data using position offsets. Relations and schemas refers to objects they link or group.

3.5 URS and other data models

The URS metamodel is not "better" than established data models such as LAF/GrAF, and it shares a lot of features with them. Its expressivity is nevertheless slightly different. In particular, complex structure modeling using schemas differs from other representation strategies. They give a good framework to represent, for example, argumentative patterns, enumerative structures or coreference chains or for grouping several elements spread over a text, in particular when binary relations are not appropriate.

However, in most cases, translations between URS and other well-established data models are possible and achieving this interoperablity is a crucial point.

4. MAIN FEATURES

4.1 Polymorphism and heterogeneity of input

Annotated texts involve heterogenous data. In particular, in a same text, annotations may: come from different annotators; belong to various granularity levels (word, sentence, paragraph, text, etc.); be related to various linguistic paradigms (syntax, semantics, discourse, coreference, etc.).

An annotation environment should allow such an heterogeneity, and provide adequate ways to deal with it.

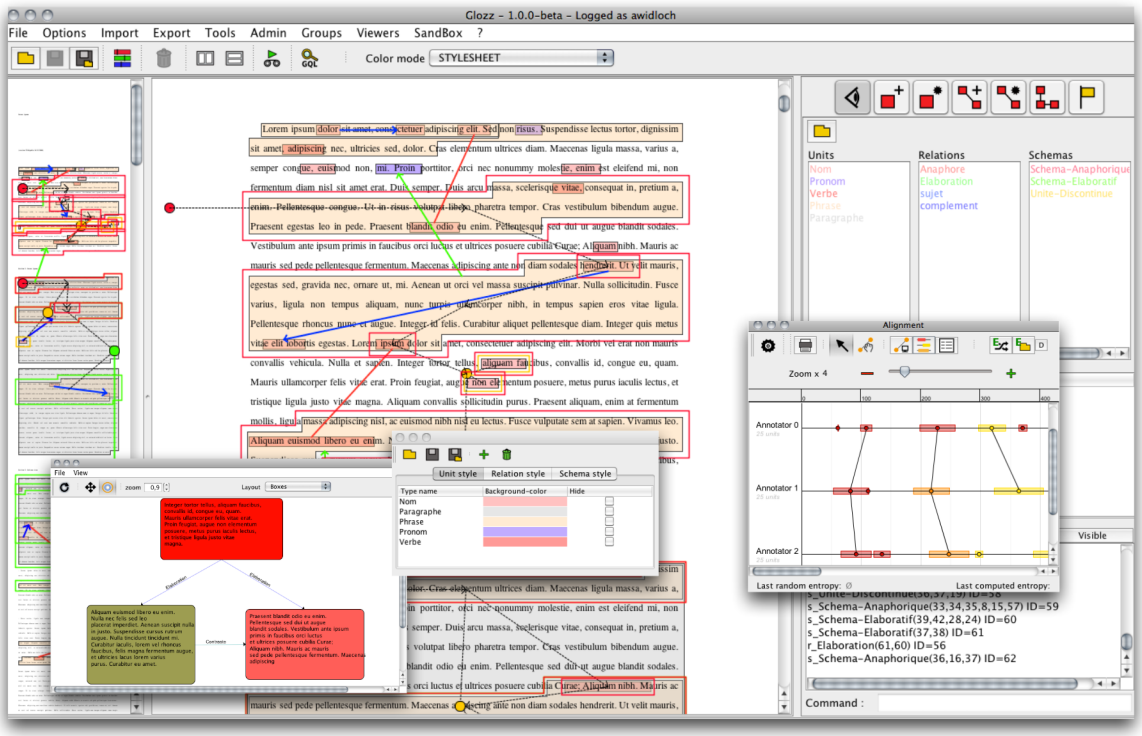

Figure 5: The Glozz main GUI

4.1.1 Several annotators

For a given annotation campaign, several annotators may add annotations to a same document. In Glozz, each annotator is authenticated, and each annotation is stamped with the identifier of its creator. This prevents collisions, and makes it possible to allow or disallow modifications by others, and to filter annotations by authors afterwards.

4.1.2 Granularity

Annotations attached to a same document may concern several levels of granularity. It may be a problem since each granularity level needs a specific modality to work with, particularly in terms of display.

Glozz proposes two simultaneous text views of the annotated document which are respectively set to "macro" and "micro" granularities. Thus, it is possible to have a global view of the document, where macro structures appear, and, at the same time, to have a focused local view, where micro structures can be well represented.

4.1.3 Several linguistic paradigms

It is necessary that several linguistic paradigms (syntax, semantics, etc.) can combine in a same document, because some paradigms may depend on others. However, too many structures at the same time make interpretation uneasy.

With Glozz it is possible to focus on one or several specific paradigms, and to hide annotations that do not belong to them. Indeed, annotation models can define groups of types, and each type can belong to one or several groups (see section 5). Users can hide as many groups as necessary.

4.1.4 Several linguistic types

Annotated items, instances of units, relations or schemas, are grouped in types (each type belonging to one or several paradigms). For example, in order to annotate the argumentative structure of scientific texts (this is an annotation paradigm), we could annotate unit objects having *types*: *introduction*, *background*, *state of the art*, *own work*, *experiment*, *evaluation*, *future works* or *conclusion*.

Annotation display, in Glozz, uses a stylesheet. This stylesheet makes it possible to define visual properties for each type, and, if necessary, to hide all instances of a given type.

4.2 Several representation paradigms

Different annotation paradigms (coreference chains, argumentative structures, rhetorical relations, etc.) often require different representation paradigms, called here *views*.

Nonetheless, we often need different paradigms at a same time, in a same campaign, hence in a same tool.

4.2.1 Annotations over the text, in situ annotation

Of course, the most usual way to annotate texts is to process directly upon them, in order to select and "highlight" identified objects. As shown in figure 6, it is possible to add or modify units, relations and schemas through a so-called WYSIWYG interface.

4.2.2 Annotations as a graph

If relations and schemas are integrated in complex constructions, or for specific annotation paradigms, a graph representation is obviously a good way to reveal what would be

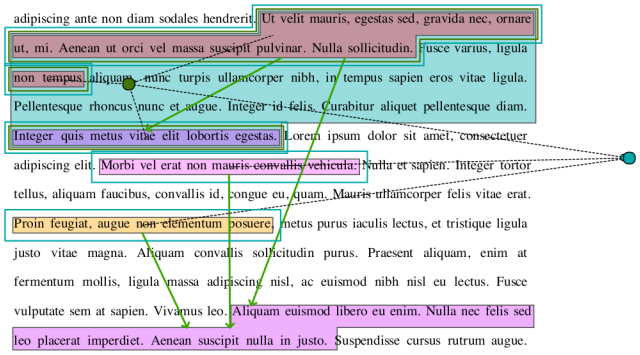

Figure 6: Annotations over the text

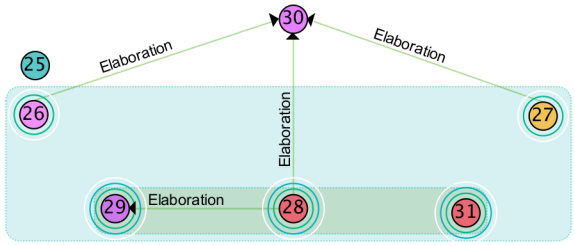

Figure 7: Annotations as a graph

confused on the flat view of the text. Hence, in figure 7, the annotations of figure 6 are represented by a graph, where the relations (of *elaboration*) and their interaction with schemas more clearly appear. In this configuration, units are represented by circled numbers and schemas by boxes.

4.2.3 *Annotations as predicates*

It may be convenient, as well, to read and create annotations directly as predicates, straight expressing, for example, that a relation should exist from annotation 1 to annotation 2, and so on. In Glozz, a module (illustrated by figure 8) permanently shows the list of all existing annotations in this way. A prompt may also be used to create new objects, with the help of auto-completion for element metatypes and type names, as well as for syntax checking.

```
...
u_Thesis(13260,13627) ID=25
u_Antithesis(13686,13731) ID=26
u_Synthesis(13858,13900) ID=27
s_Thesis/Antithesis/Synthesis(26,27,40) ID=39
u_Introduction(13260,13381) ID=28
u_Exemplification(13580,13627) ID=29
u_Exemplification(14157,14265) ID=30
r_Elaboration(28,29) ID=35
r_Elaboration(27,30) ID=36
s_Elaboration(28,29,44) ID=40
r_Elaboration(26,30) ID=41
r_Elaboration(28,30) ID=43
...
```

Figure 8: Annotations as predicates

4.2.4 *Several simultaneous views*

Moreover, a real strength of Glozz is its ability to make all its representation paradigms working at the same time, and together. Indeed, Glozz keeps central control of what is being selected through any view, and transmits the selection to all other views. This interaction is illustrated by figure 9, where the selection of an object in any of the 3 views selects it in the two other views.

This way, Glozz enhances the annotation process in two ways. Indeed, it makes it possible:

- to observe a same annotation from different points of view, in order to consider, for instance, its exact position in the text on the one hand, as well as its hierarchical position among other annotations, on the other hand;
- to select an annotation using a first view (the most adequate one to detect the searched object), and modify it using another one (more adequate to edit its properties).

5. CUSTOMIZING AND USING GLOZZ

For a given annotation task, it may be necessary to configure the annotation environment of Glozz.

5.1 Annotation campaign

An annotation platform should conform to the requirements of collaborative work. The concept of *annotation campaign* refers to an annotation task involving several annotators sharing resources.

Such resources, built by the *campaign managers*, and distributed to each annotator, include:

1. texts to be annotated;
2. one or several annotation models, to work with different paradigms;
3. one or several stylesheets to configure "points of view" on the data;
4. filters (see section 6.1), either to check reliability of the current annotation, or to bootstrap higher order annotations.

5.2 Annotation model

For a given annotation task, a specific annotation model is defined, conforming to the metamodel URS, which declares available types of units, relations and schemas. This *ad hoc* model also specifies the way of describing each instance of these types, by means of a feature set, and expresses constraints on the possible values for each feature. Available element types may also be grouped in categories or levels (see below and section 4). Relations may be declared directed or not.

Conforming to this specific model, annotators can locate instances of these types in corpora, and feed or select adequate feature values.

5.3 Customizing display

There are several ways in Glozz 1) to select annotations to be shown, and 2) to configure the way they are represented:

Filtering. Three options are given: 1) using style setting of types, since one of the style properties is visibility; 2) switching visibility of a whole group declared by the annotation model; 3) (temporary) switching visibility of individual annotation instances.

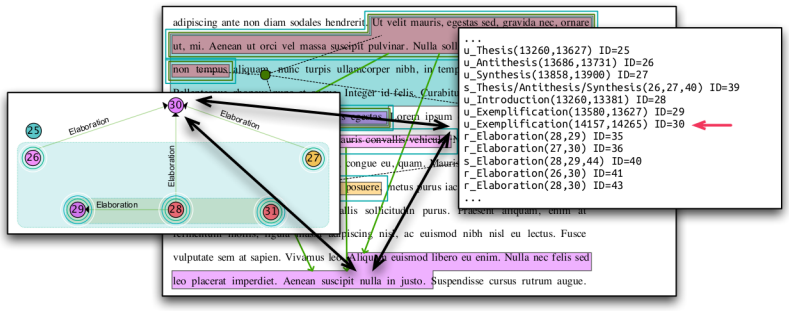

Figure 9: Several views on a same annotation

View settings. Type styling consists in choosing a color (for background or edge), and, for schemas, a shape. Several stylesheets can be defined for a same annotation model, in order to adopt successively different points of view on a same data.

6. ANNOTATION MINING

The annotation process, as well as the subsequent use of annotated data, requires the ability to access information featured by the corpus. This information concerns, of course, raw textual content, but it is also necessary to give a convenient access to linguistic (morphologic, syntactic, semantic, etc.) or infra-linguistic (document structure, typographical data, etc.) information, which may result from preliminary annotation (manual or computational) steps.

Annotating is an incremental process, which requires to take current annotations into account, to produce new ones. At a given stage of an annotation task, annotators need to find some specific configurations of annotations from previous steps (e.g. relations of a given type, units linked to other units by a relation of a given type, and so on).

Besides, it may be very helpful to check that all annotations conform to the annotation directives. Hence, the ability to locate non-valid configurations while annotating is a very convenient way to do so.

Glozz provides such facilities within annotation tasks by the means of some basic tools (not presented here), and a more advanced one named GlozzQL.

6.1 Introducing GlozzQL

6.1.1 Principles

GlozzQL (Glozz Query Language) is a language dedicated to Glozz annotations, and comes with an associated engine.

It is designed to select in a corpus each instance of element (unit, relation or schema) that satisfies expressed constraints. Requests are built piece after piece, in an incremental manner, using two interdependent concepts: *Constraint* and *Constrained-Annotation*.

Constraint: A constraint expresses one condition an annotation must satisfy in order to be selected. They are classified in 4 categories, depending on their domain, i.e. the kind(s) of element they concern (units, relations, schemas, any of them).

Constrained-Annotation: This simple concept refers to a set of annotations (of a given corpus) that all satisfy a given constraint. For a given text, and depending on its associated constraint, a Constrained-Annotation contains 0 to n entities.

6.1.2 Examples

To get an idea of GlozzQL expressivity, let us mention some incremental possible queries:

- getting all Units of a given type (this set is called U1)
- getting all Units from U1, with a given value for a given feature (this set is called U2)
- getting all Relations whose target is an element of U2 (this set is called R1)
- getting all Schemas containing a relation among R1, with a maximum depth of 3 (this set is called S1)
- getting all Schemas belonging to S1 and from a given annotator

6.1.3 How it works

A more complete explanation of this system is provided in [12]. Let us only introduce here one real-world example coming from the Annodis project [20], in order to give an overview of the principles.

We need to find all schemas having *SE* type (acronym for Enumerative Structures, in french) embedding a unit having *amorce* type (french name of the header of an enumerative structure), as well as all the *SE* not embedding one.

To do so, we have first to define a Constrained-Unit which represents all *amorce* units. This is done by `Unit1`, in figure 11, which relies on `C1` Constraint, in figure 10.

ConstraintID	Content	Domain
C1	TypeName = amorce	Any
C2	Contains(Unit1), Level=1	Relation/Schema
C3	Not(C2)	Relation/Schema
C4	TypeName = SE	Any
C5	And(C2,C4)	Relation/Schema
C6	And(C3,C4)	Relation/Schema

Figure 10: Constraints

`C1` is a constraint which concerns the type name, which must be *amorce*. This constraint may be used with any kind of annotation (which is mentioned by its domain `Any`), hence with units. At this stage, having declared `Unit1` already

Annotation	Constraint	Matches
Unit1	C1 → TypeName = amorce	13
Schema1	C5 → And(C2,C4)	12
Schema2	C6 → And(C3,C4)	2

Figure 11: Constrained annotations

implies a mining process : as stated in figure 11, 13 units fitting `C1` were found.

The second step consists in building `Schema1`, which represents all utterances of the first kind of searched schemas, that is to say schemas a) containing an utterance of `Unit1`, and b) having *SE* type. We express two preliminary constraints `C2` and `C4` respectively for a) and b). Then, a logical `And` constraint named `C5` is built over `C2` and `C4`. As a consequence, we get, with `Schema1`, 12 utterances of *SE* containing an *amorce*.

Then, we do the same to find all utterances of *SE* not containg any *amorce*, through `C6` built over `C3=not(C2)` and `C4`. We get 2 utterances.

Hence, we have discovered that in the annotated text, 12 *amorce* units out of 13 are contained in a *SE* schema, and that 2 *SE* schemas out of 14 do not contain any *amorce*.

6.2 Annotating and querying simultaneously

As already mentioned, it is helpful to query what is currently being annotated, and to be able to go from query results to current annotations.

It's the reason why GlozzQL is integrated to Glozz, and interacts with it as shown in figure 12. On the right of the figure, is the list of all queries. We can click on any of them to make the list of its results appear, just below (arrow 1). Then, a click on one result will select it in Glozz main interface (arrow 2). We can immediately see this object in its context and are able to modify or delete it if needed.

7. ALIGNMENT AND AGREEMENT

In order to check the relevance of an annotation campaign, and to get reliable annotations, several annotators usually work on the same documents with the same task. Concurrent annotations may then be compared, and their agreement may be measured.

Glozz provides a dedicated view where each annotator is given a line representating the whole text. His annotations appear at their exact location (proportionaly to the line width), as shown in figure 13.

Moreover, we have developed a new approach to compute alignment and agreement measure, via a unified and holistic method described in [13]. This method is fully implemented in Glozz and we can get a visual representation of auto-alignment, as shown in figure 14, as well as an agreement score (from 0 to 1).

This method is generic enough to cope with complex unit configurations (with overlaps and so on), which are difficult to compare with other known methods.

8. REQUIREMENTS AND AVAILABILITY

8.1 Technologies

Fully implemented in Java, and representing annotation data, annotation models, styles and queries in XML formats,

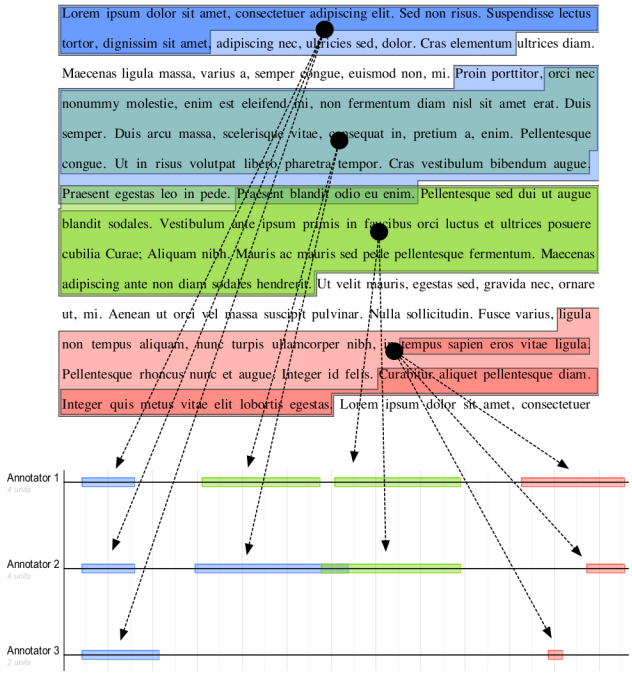

Figure 13: "One line per annotateur" view

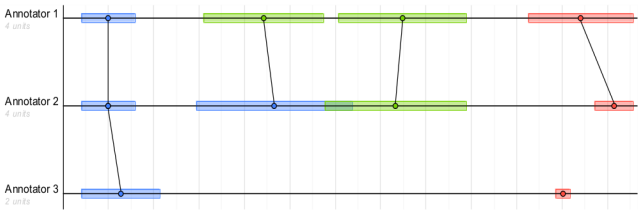

Figure 14: Alignment computation

Glozz may be used on several platforms, including Mac OSX, Linux or Windows on each of which it was tested.

8.2 Interoperability

Interoperability with other data formats (for corpora and annotations) relies on a set of import/export plugins. Raw text, and simple XML formats are already supported, as well as specific XML formats used in our NLP exprimental environment.

The import/export plugin set should grow in the future. In particular, well established formats, such as LAF/GrAF, should be supported soon. A work on a plugin dedicated to XML TEI is in progress.

It is also possible to export annotation data as an SQL file contaning structures and data. Loading this file in a RDBMS makes it possible to query annotations using SQL expressivity. It is also possible to export various annotation data as simple matrices, in order to explore them using for example a spreadsheet or any statistical tool.

8.3 Availability

Concerning availability and licence, this platform is already fully and freely available for academic research pur-

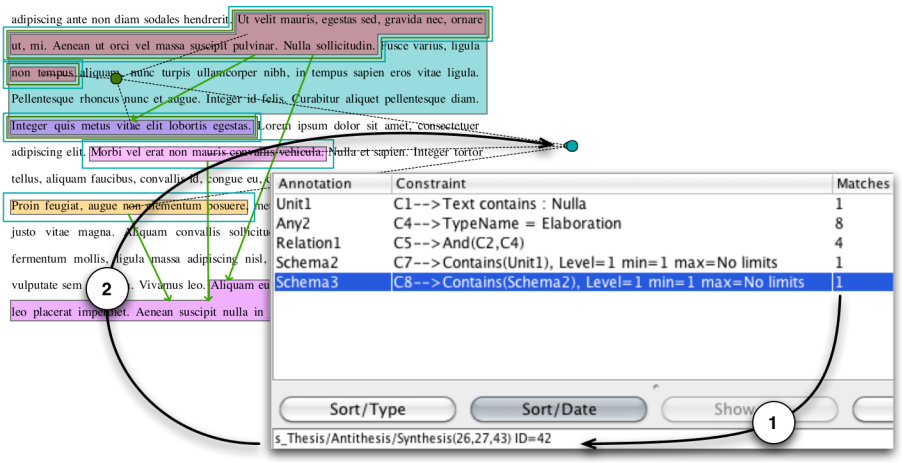

Figure 12: Querying while annotating

poses, and is widely used, even if mainly in France. Besides, its modular architecture relies on a core which is not open-source at this time (to keep control during the first years of development), and a plugin system allowing anyone to build additional tools (with whatever licence he chooses including open-source ones). XML formats are obviously open.

9. CONCLUSION AND FUTURE WORK

9.1 Recent works using Glozz

In conclusion, let us mention some recent research projects using the Glozz platform and proving its genericity and its usefulness for real-world annotation projects.

Within the Annodis project [20], it was used to create a corpus of discourse structures at several granularity levels.

Glozz is also used for the french Ontopitex ANR project, which is dedicated to opinion analysis. At low granularity levels, it allows annotation and fine description of linguistic structures. Post-annotation tools, were also used for alignment and evaluation purposes (agreement measure...).

Another recent work presented in [10], uses Glozz to study the combination of topical and rhetorical structures.

A french workgroup on coreference uses it to annotate complex (possibly imbricated) coreference chains, using expressive power of schemas.

9.2 Strengths and weaknesses

As far as we know, no other software provides such a range of capabilities for editing, viewing and mining annotations, in a same tool, at the same time. When an annotation campaign has to cope with various paradigms and scales, and has to edit and to explore data, it can rely on this comprehensive tool rather than using several tools at different stages and for different tasks. The versatility of Glozz is one of its main strengths.

But even when a campaign focuses on one phenomenon (e.g. coreference), and when dedicated tools do exist, some annotation campaign managers (well knowing other tools) switched to this one because of ergonomic and expressivity capabilities such as: direct view and "drag and drop" editing of relations, versatility of the meta-model (e.g. possibility to represent coreference chains as schemas or as pathes), multiple views (coreference chains *in situ* or as graphs).

Furthermore, its user-friendly interface makes it easy to use and learn without advanced computer skills. Users from linguistics domain successfully use it for annotation and annotation mining tasks.

The main limitation of Glozz is certainly the following: even if very versatile, its graphical interface is much more adequate for annotation tasks on discourse structure than for high density annotation at token level, for example for part-of-speech tagging or syntactic analysis. Indeed, in such a situation, the current visual environment may become somehow overloaded. Even if a workaround consists in setting the views to filter displayed annotations, a far better solution will be to provide a new dedicated visualization paradigm.

It must also be noted that the main interface can not be used to edit the content of the annotated text. However, an external tool makes it possible to correct a text, and then update annotations consequently.

9.3 Future works

At the moment, Glozz considers each text as an isolated entity, and annotators have to load individually each text they have to work with. A next step in the annotation campaign management will provide a "corpus" entity, composed of texts, which will make annotation and mining tasks on several texts easier, and will enable actions on the whole.

Currently, Glozz annotations are limited to the text scope. This scope will be extended to any text set, so that relations and schemas can apply to objects from different texts. Hence, it would be possible, for instance, to represent "alignment" relations between a text and its translations.

At last, it will be important to ensure interoperability of data produced (and imported) by Glozz, using standard such as LAF/GraF and TEI specifications for representing annotations and feature structures.

10. ACKNOWLEDGMENTS

The authors would like to warmly thank Jérôme Chauveau, who recently joined Glozz team and strongly contributes to its development.

11. REFERENCES

[1] N. Asher, A. Venant, P. Muller, and S. Afantenos. Complex discourse units and their semantics. In *Proceedings of Constraints in Discourse*, 2011.

[2] S. Bird and M. Liberman. A formal framework for linguistic annotation. *Speech Communication*, 33:23–60, 2000.

[3] G. Budin, L. Romary, T. Declerck, and P. Wittenburg, editors. *Workshop on Language Resource and Language Technology Standards – state of the art, emerging needs, and future developments*, La Valetta, Malta, 2010. Language Resources and Evaluation Conference (LREC 2010).

[4] C. Chiarcos, S. Dipper, M. Götze, U. Leser, A. Lüdeling, J. Ritz, and M. Stede. A Flexible Framework for Integrating Annotations from Different Tools and Tag Sets. *Revue Traitement Automatique des Langues (TAL)*, 49(2):189–215, 2008.

[5] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics*, pages 168–175, Philadelphia, USA, 2002.

[6] L.-M. Ho-Dac, M.-P. Péry-Woodley, and L. Tanguy. Anatomie des Structures Énumératives. In *Actes de la 17e Conférence Traitement Automatique des Langues Naturelles (TALN 2010)*, Montréal, Canada, 2010.

[7] N. Ide and A. Meyers, editors. *Linguistic Annotation Workshop*, Portland, Oregon, USA, 2011. Conference ACL/HLT 2011.

[8] N. Ide and L. Romary. Representing linguistic corpora and their annotations. In *Proceedings of the Fifth Language Resources and Evaluation Conference (LREC)*, 2006.

[9] N. Ide and K. Suderman. GrAF: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, pages 1–8, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[10] A. Labadié, P. Enjalbert, Y. Mathet, and A. Widlöcher. Discourse structure annotation : Creating reference corpora. In Budin et al. [3].

[11] S. Loiseau. CorpusReader : construction et interrogation de corpus multiannotés. *Revue Traitement Automatique des Langues (TAL)*, 49(2):189–215, 2008.

[12] Y. Mathet and A. Widlöcher. Stratégie d'exploration de corpus multi-annotés avec GlozzQL. In M. Lafourcade and V. Prince, editors, *Actes de la 18e Conférence Traitement Automatique des Langues Naturelles (TALN'11), volume 2, papiers courts*, pages 143–148, Montpellier, France, juin 2011. LIRMM.

[13] Y. Mathet and A. Widlöcher. Une approche holiste et unifiée de l'alignement et de la mesure d'accord inter-annotateurs. In M. Lafourcade and V. Prince, editors, *Actes de la 18e Conférence Traitement Automatique des Langues Naturelles (TALN'11)*, pages 247–258, Montpellier, France, juin 2011. LIRMM.

[14] T. Morton and J. LaCivita. WordFreak: An Open Tool for Linguistic Annotation. In *Proceedings of Human Language Technology (HLT) and North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 17–18, Edmonton, Canada, 2003.

[15] C. Müller and M. Strube. Multi-level annotation of linguistic data with MMAX2. In S. Braun, K. Kohn, and J. Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany, 2006.

[16] M. Noguchi, K. Miyoshi, T. Tokunaga, R. Iida, M. Komachi, and K. Inui. Multiple Purpose Annotation using SLAT - Segment and Link-based Annotation Tool. In *Proceedings of the 2nd Linguistic Annotation Workshop*, pages 61–64, may 2008.

[17] M. O'Donnell. RSTTool 2.4 – A Markup Tool for Rhetorical Structure Theory. In *Proceedings of the International Natural Language Generation Conference (INLG'2000)*, pages 253 – 256, Mitzpe Ramon, Israel, 13-16 June 2000.

[18] P. V. Ogren. Knowtator: A Protégé plug-in for annotated corpus construction. In *Proceedings of Human Language Technology (HLT) and North American Chapter of the Association for Computational Linguistics (NAACL)*, New-York, USA, 2006.

[19] C. Orăsan. PALinkA: a highly customizable tool for discourse annotation. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialog*, pages 39–43, Sapporo, Japan, July, 5-6 2003.

[20] M.-P. Péry-Woodley, N. Asher, P. Enjalbert, F. Benamara, M. Bras, C. Fabre, S. Ferrari, L.-M. Ho-Dac, A. Le Draoulec, Y. Mathet, P. Muller, L. Prévot, J. Rebeyrolle, L. Tanguy, M. Vergez-Couret, L. Vieu, and A. Widlöcher. ANNODIS: une approche outillée de l'annotation de structures discursives. In *Actes de la 16e Conférence Traitement Automatique des Langues Naturelles (TALN'09), session poster*, Senlis, France, 2009.

[21] P. Stenetorp, S. Pyysalo, G. Topic, T. Ohta1, S. Ananiadou, and J. Tsujii. BRAT: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceeding of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, 2012.

[22] B. Webber and D. Bryon, editors. *Proc. of the ACL 2004 Workshop on Discourse Annotation.*, Barcelone, Espagne, 2004.

[23] A. Widlöcher. *Analyse macro-sémantique des structures rhétoriques du discours - Cadre théorique et modèle opératoire*. PhD thesis, Université de Caen Basse-Normandie, 17 octobre 2008.

[24] A. Widlöcher and Y. Mathet. La plate-forme Glozz: environnement d'annotation et d'exploration de corpus. In *Actes de la 16e Conférence Traitement Automatique des Langues Naturelles (TALN'09), session posters*, Senlis, France, juin 2009.

[25] A. Witt, U. Heid, H. S. Thompson, J. Carletta, and P. Wittenburg, editors. *Workshop on XML-based richly annotated corpora (XBRAC)*, Lisbonne, Portugal, 29 mai 2004. Language Resources and Evaluation Conference (LREC 2004).