A collaborative infrastructure for handling syntactic annotations

Éric Villemonte de La Clergerie

A collaborative infrastructure for handling syntactic annotations

Éric Villemonte de la Clergerie
INRIA Paris-Rocquencourt
Eric.De_La_Clergerie@inria.fr

Abstract

We believe that collaborative annotating is needed to build and improve large syntactically annotated corpora such as tree banks or dependency banks. We present such a collaborative infrastructure, implemented as a WEB service in the context of the French project **Passage** whose primary objective is the automatic syntactic annotating of a large corpus over 100 millions words.

1 Introduction

The work presented in this paper takes place in the context of **Passage** (*Produire des annotations syntaxiques à grande échelle* – Large Scale Production of Syntactic Annotations), a 3-years French action[1] (2007–2009) that includes the following tasks:

- automatically annotating a 100 million words French corpus using 10 parsers;
- merging the resulting annotations using a rover, in order to get better quality annotations;
- manually building a reference annotated subcorpus (around 500 000 words), starting from the rover annotations. It should allow us to assess the quality of the rover and to prepare freely available data for the community;
- running linguistic knowledge acquisition experiments on the rover annotations;
- running two parsing evaluation campaigns on the model of the EASy French evaluation campaign (Paroubek et al., 2006). The first campaign, to be run at the end of 2007 should provide data to calibrate the rover. The second campaign, at the end of **Passage** (2009), should provide information about the evolutions of the parsers during the project.

Passage requires a solid infrastructure to handle very large syntactically annotated corpora. Given the various tasks to be accomplished, the infrastructure has to provide many functionalities for handling annotations, such as storing, viewing, querying, comparing, editing, revising, downloading, uploading, ... Furthermore, **Passage** involves around 10 distinct teams, which suggests using a collaborative infrastructure providing sufficient computational power. This power may not be locally available, given the size of the corpus.

Others have already promoted the notion of collaborative annotating (Balasubramanya et al., 2006; Artola et al., 2004; Ma et al., 2002) but we believe this notion is also important for the following reason. We are well aware that syntactically annotating a corpus is a difficult and tedious task, best accomplished by experts following a precise protocol. However, these experts are rare while a larger set of people may prepare (possibly automatically) annotated data for a majority of simple cases, the experts being mainly consulted for validation and for handling the difficult cases. A collaborative infrastructure may help to involve more actors, specially if they have nothing to install.

A WEB service on the model of WIKIPEDIA seems a good option. The actors would start browsing annotations and, in a natural way, should get the possibility to signal errors and, even better, to correct them. Another pertinent model is provided by bug reporting systems such as BUGZILLA, allowing people to report and discuss bugs, and *experts*

[1]The PASSAGE project is supported by the French National Research Agency (ANR-06-MDCA-013)

to handle them. A last pertinent model is versioning systems such as SUBVERSION allowing people to commit revisions to some central place.

Of course, one may argue that understanding and editing syntactic annotations is a much more complex task than reading and editing a wikipedia page. The difficulties arise from the linguistic subtleties but also from the richness of syntactic annotations. They are complex to visualize, potentially involving long distance phenomena. Good annotation guides and rigorous protocols are needed to reduce the amount of errors. However, many errors may also be avoided by designing a system that presents information in an adapted way and that cleverly guides/constrains the annotators in their tasks.

We started developing EASYREF, a prototype of collaborative annotation management system to correct the set of reference annotations manually built for the original **EASy** parsing evaluation campaign, because of a relatively high rate of remaining errors. However, we now see its larger potential for the other tasks in **Passage**.

Section 2 presents the **EASy** annotation format. Section 3 lists some design principles which have guided the development of EASYREF. Section 4 explains how these principles have been instantiated through the existing functionalities of EASYREF. Section 5 sketches the possible evolution of EASYREF in the context of **Passage** and beyond, in particular related to the scaling issues.

2 The EASy annotation format

The **EASy** XML annotation format was designed to provide information about:

- the segmentation of corpora into **sentences**;
- the segmentation of sentences into **forms**;
- non-recursive typed **chunks**, embedding forms. Table 1(a) lists the possible types;
- labeled **dependencies** that are anchored by either forms or chunks. All dependencies have a binary arity but for the ternary COORD dependencies. The 14 kinds of dependencies are listed in Table 1(b).

The XML format is use to derive the standard EASy HTML view of the annotations, as shown in Figure 1 for the sentence *Meantime, the tragedy of Liberians continues*. This view cleverly uses a linear representation of the sentence and chunks with color codes to identify quickly the chunk types. On the other hand, the dependencies are listed in 14 separate tables with their anchors provided through their identifiers. To decode these tables, the reader has to retrieve the content of an anchor through an indirection given an identifier.

Type	Explanation
GN	Nominal Chunk
NV	Verbal Kernel
GA	Adjectival Chunk
GR	Adverbial Chunk
GP	Prepositional Chunk
PV	Prepositional non-tensed Verbal Kernel

(a) Chunks

Type	Anchors	Explanation
SUJ-V	suject,verb	Subject-verb dep.
AUX-V	auxiliary, verb	Aux-verb dep.
COD-V	object, verb	direct objects
CPL-V	complement, verb	other verb arguments/complements
MOD-V	modifier, verb	verb modifiers (such as adverbs)
COMP	complementizer, verb	subordinate sentences
ATB-SO	attribute, verb	verb attribute
MOD-N	modifier, noun	noun modifier
MOD-A	modifier, adjective	adjective modifier
MOD-R	modifier, adverb	adverb modifier
MOD-P	modifier, preposition	prep. modifier
COORD	coord., left, right	coordination
APPOS	first, second	apposition
JUXT	first, second	juxtaposition

(b) Dependencies

Table 1: EASy format

It should be stressed that an annotated sentence may represent a lot of information to display: several tens of forms, almost as many chunks and dependencies with their type, each dependency involving two or three anchors that may stand relatively far away in the sentence.

The standard HTML view was also used as the base to annotate, completed by HTML forms to fill. The method was tedious and rather error prone, because the annotators had to explicitly fill the fields with identifiers.

Although there now exists a good and rather complete **EASy** annotation guide, it may be noted that the original annotation of around 4000 sentences was performed by several teams while stabilizing

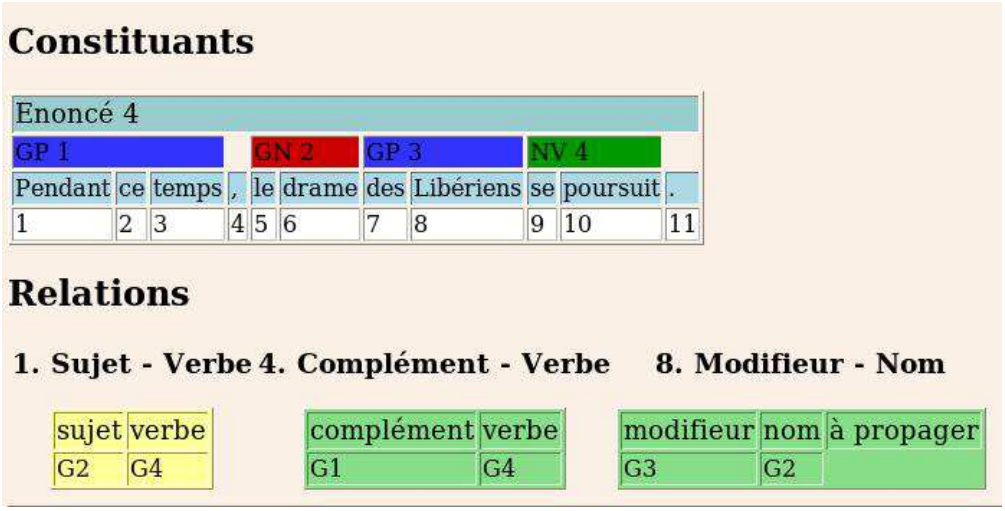

Figure 1: Standard HTML view of **EASy** annotations

the guide. The resulting reference annotation set is therefore not homogeneous, with some errors related to evolutions in the guide and divergences of interpretation among the various teams. Since then, many errors have been detected by the participants to the **EASy** campaign and have been submitted by mail or as notes. However, these notes were not integrated with a view of the erroneous sentences, making difficult their analysis and corrections.

3 Guiding principles

The development of EASYREF was guided by a few design principles. First, as already advocated, we opted for a **collaborative** infrastructure. The idea is that many people should have a way to access annotated data, to exploit them but also to progressively improve them. It also facilitates the management for a coordinator that has to follow the work of several annotators.

Secondly, the system should be able to **present rich and complex information** in a graspable way for human annotators. The solution seems to use abstract synthetic views completed by mechanisms to zoom on more precise information, for instance when moving over the visual representation of an entity (including the annotations). Furthermore, combining abstract global views and more local precise ones should involve minimal navigational efforts. Another issue concerns the aggregation of pieces of information that come from several sources (annotations, reports, corrections, alternate versions, ...).

Thirdly, the system should do whatever possible to **reduce the risks of errors**, for instance by using *clever* and *constraining* interfaces. In particular, direct interactions with the visual representation of entities (such as forms, chunks, dependencies) should be preferred over manually typing identifiers. The use of contextual menus that list only those actions that are allowed in the current context is also helpful. And checking the validity of each action is of course a strong requirement!

Last but not least, the system should **keep traces** by using logs, databases, versioning, ... with precise information about actions, dates and authors. These traces are important to follow the evolution of the data, possibly providing ways to undo some actions. The system should clearly also provide ways to exploit these traces.

Inspired by these principles, EASYREF has been implemented as a WEB service with AJAX-based server-client communications.

The server side is implemented in the Perl CATALYST framework based on the *Model-View-Controller* [MVC] paradigm. Persistent data at the model level are organized following schema (Database schema or XML-like schema), these schemata guiding the allowed actions and specifying the integrity conditions to be satisfied (such as

unique identifiers for instance). The controllers are Perl modules, used to process URL requests and generate views from the data through the use of templates. Standard CATALYST modules were used to handle users, user rights, and user sessions.

On the client side, information is presented through HTML pages that may be displayed by most WEB browsers (such as FIREFOX). Cascaded Style Sheets [CSS] are used to separate structuring from rendering. To get a smarter client, various actions have been implemented with Javascript on the client side. In particular the Javascript library PROTOTYPE.JS has been heavily used to establish server-client communications through AJAX requests. The answers to these requests may be used to locally update the content of a page, avoiding the need for a full refresh of the page. The advantages are of course a reduced bandwidth but also fluid interfaces that can transparently use the power of the server side to present contextual information on the client side.

4 Existing fonctionalities

4.1 Visualizing the annotations

Figure 2 shows a screenshot for EASYREF. The leftwards sidebar acts as a global menu for accessing the various views provided by EASYREF. The main panel concerns the annotations with the topmost panel being a form used to query sentences and the bottom one to display the retrieved annotated sentences.

We have kept the linear representation of the sentences with color-coded chunks above the forms. The idea was extended for dependencies, represented on several lines below the forms, using color codes related to their type. The span of a dependency is given by its anchors. A simple algorithm is used to reduce the number of lines and to sort the dependencies according to their span and position (left-to-right, shortest first). One may select which kinds of dependencies are to be displayed. At first glance, it is not possible to precisely identify the anchors of a dependency and their role. However, when moving the mouse over a dependency, its anchors are highlighted and a tooltip box is displayed, locally providing more detailed information.

Information is also integrated in the sense that, for a given sentence, it is possible to show (or hide) various pieces of information, such as the list of its bug reports (see section 4.3), the history of its revisions (see section 4.2) and a list of potential errors automatically detected by EASYREF. The potential annotation errors are raised when some basic constraints are violated, for instance for

- dependencies whose anchors refer to non existing forms or chunks;
- wrong types on dependency anchors (when they are chunks);
- dependencies that multiply refer a same chunk or form.

Sentences may be searched using combined administrative and linguistic criteria. For instance, one may search all the sentences with potential errors but no bug reports, or sentences with reports but no corrections. A more linguistic query such as "`\bévaluer@NV \bles@GN`" would return all the sentences where the word "évaluer" (*evaluate*) in a NV chunk is followed by "les" (*the*) in a GN chunk. Linguistic queries are applied as regular expressions on a linear representation of both text and chunk annotations.

4.2 Editing annotations

Following our guiding principles, editing entities such as chunks or dependencies is done by directly interacting with their visual representations. Practically, as illustrated by Figure 3, double-clicking on a chunk opens, within the chunk, a contextual menu (provided by the server) listing the possible operations such as deletion, type change, merging with the leftwards or rightwards component (chunk or form) and shrinking (removing the leftmost or rightmost form in the chunk). Creating a new chunk is done by double-clicking a form of the future chunk (generally its *head*) and then (possibly) extending it.

Because editing a chunk may have an impact on the dependencies it anchors, EASYREF takes sometimes the responsibility to update them. However, in a more conservative way, a revision mark is added to all potentially impacted dependencies. Visually displayed (by a $*$), the mark clearly alerts the annotator that these dependencies should be checked.

Modifying an existing dependency also starts by double-clicking its visual representation, opening a

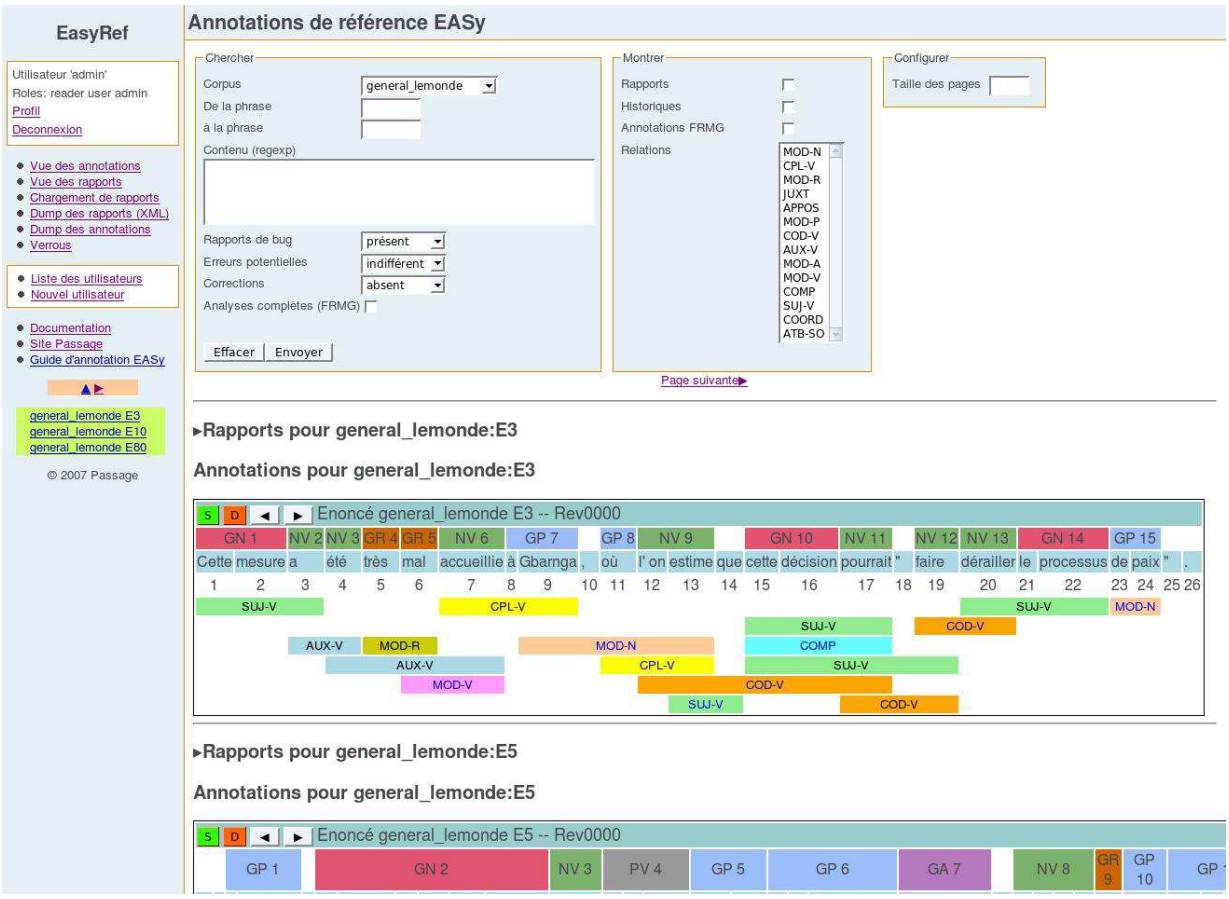

Figure 2: Visualizing annotations with EASYREF

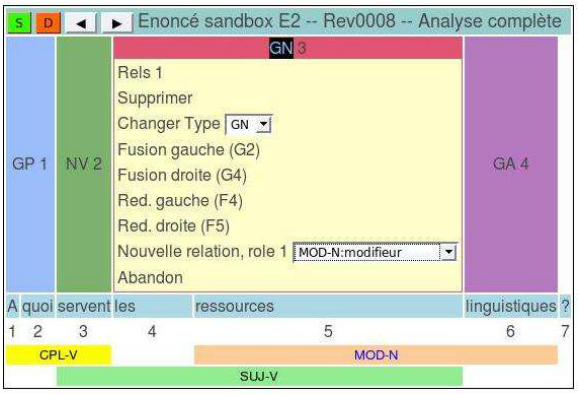

Figure 3: Modifying a chunk

contextual menu (Figure 4). Because it was found to be a frequent error, one possible editing actions concerns the swap of the anchors for the binary dependencies (and more general permutations for the ternary COORD dependencies). While the menu also allows to manually modify the value of a given anchor (or of an extra parameter), the preferred and less error-prone alternative consists of double-clicking onto some chunk or form and to select it as an anchor for some role in the selected dependency.

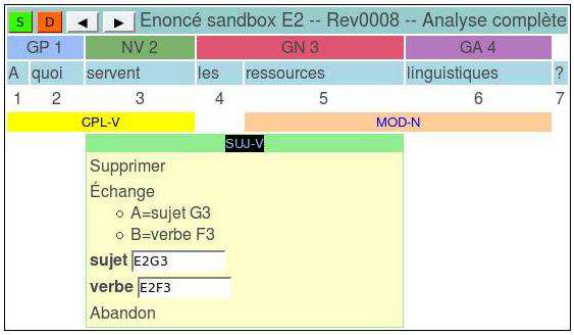

Figure 4: Modifying a dependency

Actually, the same idea has been used for creating new dependencies: the process starts by selecting

the first anchor by double-clicking on a chunk/form, selecting a type and role in the contextual menu, then selecting in a similar way the second anchor and possibly the third one. At the end, as shown in Figure 5, a menu is displayed, listing all pertinent information about the dependency to be created and asking for creation or cancellation. So, in practice, all editing actions may be performed by double-clicking on entities or selecting options in closed contextual lists, reducing the risks of errors. Still, errors remain possible (such as anchor inversion) but warnings about potential errors may help.

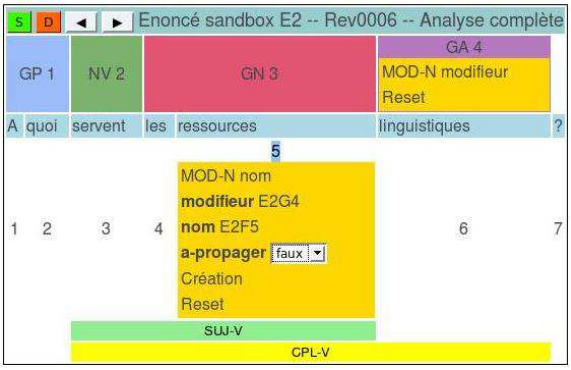

Figure 5: Creating a new dependency

Each edition action creates a new version of the sentence with an incremented revision number. The version is saved as an XML file with an automatically generated comment providing information about the nature of the action, its author, the date, ... It is possible to browse the various revisions of a sentence (using navigation arrows) but it is only possible to edit the last one.

The sentence being edited is locked during the very brief time where the edition action is validated. In case of trouble, a surviving lock expires after a few minutes. The sentence cannot be edited when locked. More subtly, a sentence is also not editable if not corresponding to the last revision, for instance because of an edition made by someone else.

4.3 Handling bug reports

Bug reports may be attached to a sentence. More precisely, several bug reports may be attached to a same bug, either because the same problem occurs several times for distinct sentences or because there exists a thread of discussion about the bug. The reports may be seen while browsing the sentences but also through a dedicated view allowing to search them according to various criteria. A report model (with an underlying XML DTD) specifies the following (possibly optional) fields:

id of the report and **bug id** of the associated bug
localization reference provided by the triple corpus+sentence+revision
author and **date** of the last edition
status of the report (open, closed, unknown, rejected)
type of error (in segmentation, chunk, dependency, ...)
diagnostic : description of the problem
pre and **post** (local) situation before and after correction
fix : nature of the proposed correction

Already 776 bug reports have been imported in the system. The Figure 6 shows the (translated) content of one of them, concerning the initial revision of sentence `E558` in corpus `questions_amaryllis`.

ID: 765 BID: 765
LOC: questions_amaryllis:E558:r000
AUTHOR: christelle
DATE: 2007-08-24 11:17:54
STATUS:o (open)
CLASS: GF
TYPE: R (Relation)
DIAG: we have two coordinations and not just one.
PRE: COORD(F1,NV1,F11)
POST: COORD(F1, ,NV)
FIX: it was a binary coordination.

Figure 6: Example of bug report

5 Evolutions

EASYREF is already used but it should be further developed during the remaining of the **Passage** project. First, the **EASy** format should be enriched, in particular to be closer to the emerging ISO TC37 SC4 standards (Ide et al., 2003). Forms should be built upon tokens referring spans of the original documents through standoff pointers, following the *Morphosyntactic Annotation Framework* [MAF] (Clément and Villemonte de La Clergerie, 2005). Besides chunks, constituency should

be completed by allowing nested recursive groups as proposed in the *Syntactic Annotation Framework* [SynAF], following the TIGER model. Structured content represented by feature structures could be attached to forms, groups, and possibly dependencies. Even with these extensions, visualization and edition should remain possible, in particular by adapting the multi-line view for groups (as done for dependencies) and opening entities to display their content. New functionalities have also to be added to EASYREF, including:

- downloading raw corpora (possibly resulting from a search);
- uploading an annotation set, provided by one of the participating parser to **Passage**;
- automatic merging of annotation sets (applying a *ROVER*);
- comparing two sets of annotations, for instance between a parser set and the rover one, or between two revisions of a same set. This functionality is already partially implemented, allowing us to compare the **EASy** reference set with the FRMG set (Boullier et al., 2005) as illustrated in Figure 7. The first line displays the reference chunks and the second one the FRMG chunks, with the color code easing the identification of mismatching chunks. Comparing dependencies is more complex: both sets of dependencies are actually mixed, the text color and weight indicating the status of the dependencies, with normal black text indicating the dependencies that belong to both sets, bold blue text for those only in the reference set, and red blue text for those only in the FRMG set;
- building virtual corpora, by saving selections of sentences. They would be useful to focus on a specific syntactic phenomena. As a preliminary step in this direction, EASYREF already allows to build a per-user per-session sentence selection (available in the sidebar, Figure 2);
- displaying various statistics about the quantity and quality of annotations, about their evolution over time, about the rover, ...

Several issues have also to be addressed, the main one concerning **scalability**, with the wish to be able to handle very large annotated corpora of several hundred millions words. A first option seems to be to structure the corpora in units adapted for the various tasks (viewing, editing, versioning, ...). We propose to structure into thematic corpus collections, corpus, blocks (about a few thousand sentences for processing), segments (about a few tens of sentences for viewing) and sentences (for editing and versioning). The second step is to move towards native XML databases such as XINDICE or EXIST that provide efficient XPath-based query and update mechanisms (Bird et al., 2006). Specialized indexes, for instance based on suffix arrays, may be needed for efficient content-based queries.

Robustness is another issue to address when running a centralized collaborative environment that should not get trapped into unexpected states. An user cannot easily restart the system and his own actions may have undesirable impacts on critical data.

The next related issue obviously concerns **security**. The current version of EASYREF implements a basic user right management, only been tested with a restricted audience. A larger audience involves a deeper analysis of the security risks and a more fine-grained right management component. For instance, a policy could enforce that a participant can only upload, browse and edit its own annotation set while being allowed to browse the rover set. In the context of an evaluation campaign, a participant should not be allowed to browse the reference set. The rover team should be allowed to browse all annotation sets but only edit the rover set. Similar policies would also work for manual annotations: two people annotating the same corpus, not having access to the other set, and a supervisor merging the two sets.

The next issue arises from the nature of EASYREF that requires a live connection to a centralized server, which may not be always very practical and may also involve too high an overload on the server. *Offline sessions* could be an option, either by using a local program (in particular by adapting existing annotating systems) or by deploying several instances of EASYREF to move forwards a more decentralized peer-to-peer infrastructure(Balasubramanya et al., 2006). However, in both cases, mechanisms have to be implemented to synchronize the annotations and to solve the potential conflicts (as done for versioning systems such as SUBVERSION). It may be noted that the conflicts may be solved by comparing two sets of annotations, a functionality already

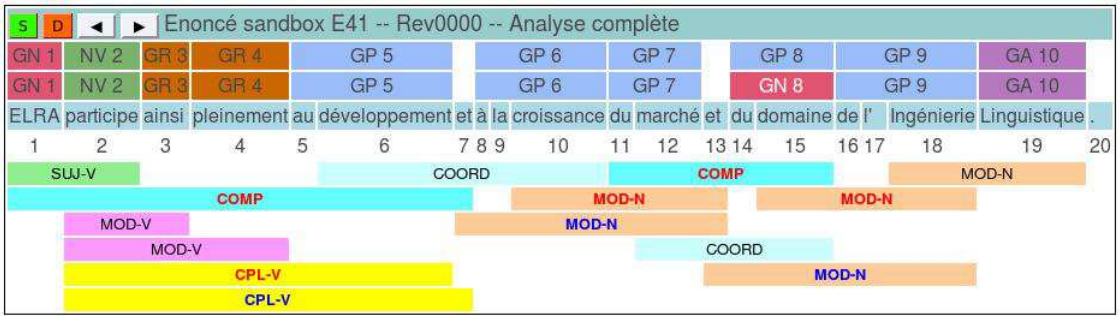

Figure 7: Comparing two annotation sets (reference and FRMG)

partially implemented.

6 Conclusion

We believe that an extended version of EASYREF will become an important infrastructure in order to coordinate the efforts of around 10 teams in the context of the **Passage** action. Two instances of EASYREF have already been deployed for the benefit of **Passage**'s participants, one for correcting the existing reference annotation set and the other one to annotate from scratch a small set of 500 new sentences for the 2007 evaluation campaign. We should very soon activate a public instance to progressively

- make available some sets of annotations, that could be used to train parsers or run knowledge acquisition tasks;
- motivate parser developers to download their annotations and compare them with the existing ones;
- favor the use of a standardized syntactic representation;
- improve the reference annotations, either by using a rover over the available annotation sets or by getting manual corrections (even if only for the simpler cases).

References

X. Artola, A. Diaz de Ilarraza, N. Ezeiza, K. Gojenola*, A. Sologaistoa, and A. Soroa. 2004. EULIA: a graphical web interface for creating, browsing and editing linguistically annotated corpora. In *proc. of LREC'04*.

Magesh Balasubramanya, Michael Higgins, Peter Lucas, Jeff Senn, and Dominic Widdows. 2006. Collaborative annotation that lasts forever: Using peer-to-peer technology for disseminating corpora and language resources. In *Proc. of the fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, May.

Steven Bird, Yi Chen, Susan B. Davidson, Haejoong Lee, and Yifeng Zheng. 2006. Designing and evaluating and XPath dialect for linguistic queries. In *22nd International Conference on Data Engineering (ICDE)*, pages 52–61, Atlanta, USA, April.

Pierre Boullier, Lionel Clément, Benoît Sagot, and Éric Villemonte de La Clergerie. 2005. « simple comme easy :-) ». In *Proceedings of TALN'05 EASy Workshop (poster)*, pages 57–60, Dourdan, France, June. ATALA.

Lionel Clément and Éric Villemonte de La Clergerie. 2005. MAF: a morphosyntactic annotation framework. In *proc. of the 2nd Language & Technology Conference (LT'05)*, pages 90–94, Poznan, Poland, April.

N. Ide, L. Romary, and Éric Villemonte de La Clergerie. 2003. International standard for a linguistic annotation framework. In *Proceedings of HLT-NAACL'03 Workshop on The Software Engineering and Architecture of Language Technology*. Journal version submitted to the special issue of JNLE on Software Architecture for Language Engineering.

Xiaoyi Ma, Haejoong Lee, Steven Bird, and Kazuaki Maeda. 2002. Models and tools for collaborative annotation. In *Proc. of LREC'02*, Las Palmas, Spain.

Patrick Paroubek, Isabelle Robba, Anne Vilnat, and Christelle Ayache. 2006. Data, annotations and measures in EASY - the evaluation campaign for parsers of french. In ELRA, editor, *proc. of LREC'06*, pages 315–320, Genoa, Italy, May.