SACR: A Drag-and-Drop Based Tool for Coreference Annotation

Bruno Oberle
LiLPa (EA 1339), Université de Strasbourg
14 rue Descartes, F-67084 Strasbourg Cedex, France
oberleb@unistra.fr

Abstract

This paper introduces SACR, an easy-to-use coreference chain annotation tool, which is used to annotate large corpora for Natural Language Processing (NLP) applications. Coreference annotation is usually considered as costly both in terms of time and human resources. So, in order to find the easiest annotation strategy, we will first of all compare several annotation schemes implemented in existing tools. Since interface ergonomics is also an important part of our research, we then focus on identifying the most helpful features to reduce the strain for annotators. In the next section of the paper, we present SACR in details. This tool has been developped specifically for coreference annotation, and its intuitive user interface has been designed to facilitate and speed up the annotation process, making SACR equally suited for students, occasional and non-technical users. In order to create coreference chains, elements are selected by clicking on the corresponding tokens. Coreference relations are then created by drag-and-dropping expressions one over the other. Finally, color frames around marked expressions help the user to visualize both marked expressions and their relations. SACR is open source, distributed under the terms of the Mozilla Public License, version 2.0, and freely available online.

Keywords: coreference annotation, annotation tool, coreference chain, interface ergonomics

1. Introduction and Context

The development of statistical methods in automatic language processing leads to an increased need for annotated resources. In particular, training algorithms that aim to detect coreference[1] requires manually annotated texts. Because such a task is costly in terms of time and human resources, there are few annotated corpora with coreference relations. In French, there is only one such a corpus large enough to be used with supervised methods: the ANCOR corpus (Muzerelle et al., 2014), with 488,000 lexical units, 116,000 referring expressions (only nouns and pronouns) and 51,300 relations. But it contains only spoken French.

The "Democrat" project[2] aims, among others, at providing a large corpus annotated with coreference chains. It is expected to have one million words and 100,000 elements of coreference chains. Several annotation strategies have been discussed (Landragin et al., 2017) in order to balance scientific needs with annotation speed.

The main purpose of this paper is to introduce SACR[3], a tool specifically designed to facilitate and speed up annotation of coreference chains. We first compare coreference annotation strategies used by different tools, both in terms of annotation scheme (section 2) and interface usability (section 3), before turning to a presentation of SACR (section 4), and how it is used by annotators in the Democrat project.

[1]There is coreference when two linguistic expressions refer to the same *referent*, that is, the same entity. For example, in *My cat is drinking milk. It is really thirsty*, both *my cat* and *it* refer to the same entity (the cat I own) and thus are *coreferential*. All expressions that refer to the same entity are said to be in a *coreference chain*.

[2]*DEscription et MOdélisation des Chaînes de Référence: outils pour l'Annotation de corpus (en diachronie et en langues comparées) et le Traitement automatique* "Description and modelling of reference chains: tools for corpus annotation (with diachronic and cross-linguistic approaches) and automatic processing".

[3]SACR is an acronym for *Script d'Annotation des Chaînes de Référence* "Script for Coreference Chain Annotation". It is freely available at `http://boberle.com/projects/sacr`.

2. Coreference chain annotation

According to (Habert, 2005), any annotation task may be divided into three steps; for coreference annotation, these steps are:

1. delimiting and marking referring expressions (that is, a chunk of text that *refers* to some entity, the *referent*, in the extralinguistic world),
2. annotating a set of features for each referring expression (*e.g.* the part of speech of the syntactic head or its grammatical function),
3. linking coreferential expressions to build coreference chains (that is, the set of all the linguistic expressions that refer to the same referent).

We could also add a fourth step which would consist in annotating relations and/or chains (*e.g.* with the type of the referent: a person, an organization, a location, an idea, etc.).

Some of these stages may be automated (Landragin, 2011; Poudat and Landragin, 2017), chiefly the annotation of parts of speech, for which well performing tools exist. The other steps require a manual annotation, since no tool is (yet) effective enough to rely on, at least for French. Coreference annotation is the most demanding among these steps, and this is what we will discuss in this section.

Several annotation schemes may be used to annotate coreference relations. First, each relation between two coreferential expressions may be annotated separately (for example, between "cat" and "animal" then between "animal" and "it"). Chains are then built afterwards by transitivity (if "animal" is coreferential with both "cat" and "it", that means that "cat" and "it" are also coreferential). This scheme offers fine-grained annotation possibilities, since it is possible to add a set of features for each relation (is the relation anaphoric? cataphoric? etc.). This method is tedious, though, and a quicker way is to put all the coreferential expressions in one set (a coreference chain), without annotating individual relations. The annotator simply mark "cat", "animal" and "it" as belonging to the same set.

Both schemes are possible with most of the tools that allow coreference annotation. Glozz (Widlöcher and Mathet, 2012) and Analec (Landragin et al., 2012) define a three-level model: *units* are used to mark chunks of texts, that is, in our case, referring expressions; *relations* to annotate binary relations; *schemata* to make coreference chains, by linking either the relations or the units. The other tools have similar features, with different names: MMAX2 (Müller and Strube, 2006) calls the Glozz-like relations "pointer-type relations" and the schemata "set-type relations" where BRAT (Stenetorp et al., 2012; Bra, 2014) uses the terms "binary relations" and "equivalence relations". GATE (Cunningham et al., 2013; Cunningham et al., 2011) seems to allow only generic sets, and not binary relations.

Annotation with schemata in Glozz and Analec has been tested during the MC4 project[4] (Landragin, 2011; Mélanie-Becquet and Landragin, 2014); but this requires to build a schema for each new coreference chain, which involves an extra work.

Rather than building a schema for each chain, a better strategy is to build coreference chains from annotations: they can be easily and automatically deduced if the name of the referent is recorded as a feature of each referring expression: a chain is then the set of referring expressions that have the same referent name. For example, if the expressions "cat", "animal" and "it" have the same referent name "John's cat" recorded as a feature, then it must be that they are in the same coreference chain.

This is a change of perspective: annotators do not build the chain themselves, but focus on finding the referent for each referring expression. This has been found to be quicker and easier than the building of chains *via* the schema-like strategy and is the method currently used in the Democrat project (Landragin et al., 2017).

Consequently, the tool used to annotate must only allow marking referring expressions and recording referent names. Almost every annotation tool can be used as long as annotators can mark tokens and add a feature set to them, so that a tool like UAM CorpusTool (O'Donnell, 2008), which does not (yet) allow annotation of relations, can be used. Even a basic XML editor like Oxygen[5] may be used.

The Democrat project is currently using TXM (Heiden, 2010), a platform to which Analec has been added as an extension. Referent names are added as a feature to each referring expression, and the coreference chains are created automatically afterwards, as schemata computed from these names. But adding and managing dozens of referent names is tedious and the process may be optimized with a dedicated user interface.

3. User interface ergonomics

Usually annotation tools are developed with little attention for the comfort of annotators (Fort, 2012), even if annotating requires both intellectual and physical efforts, and if user interface should take both into consideration (Müller and Strube, 2006).

Physical efforts come first from the fact that annotators must type a referent name for each referring expression. This can be facilitated by offering a default unique name when the referring expression is marked, by auto-completing the name when the referent has been entered previously or even by copying the name from an expression to another by a simple drag-and-drop operation. This also has the advantage to prevent typing errors and name misspellings. Some annotation schemes require all referring expressions to be annotated, and not only those that are related to other expressions, because having all the referring expressions marked allows more refined analysis, like comparing isolated expressions in contrast to coreferential ones. One annotation strategy to ease the work of annotators is to allow them to type a code (like "SI" for "singleton"[6]) when the expression is not related to another: annotations marked with this code will not be included when building chains.

Delimiting and marking referring expressions can also be at times irritating with some tools that do not tokenize the text in "words". While this can be useful for some languages such as Chinese, it is most often an issue when trying to mark an expression around a apostrophe or a comma.

A better visualization of marked expressions helps to reduced the cognitive load. Gate and Analec just highlight marked expressions with only one color, while MMAX offers the possibility to surround them with brackets. BRAT uses colors, but it is Glozz which is the most helpful tool here since it draws colored frames around marked expressions. This is especially important when expressions are nested, and it often happens, when all referring expressions are to be annotated, and not only those which are part of a chain. But Glozz shows relations as arrows, which tend to obstruct the text beneath them.

Keeping track of all the referents encountered so far in a text is difficult yet necessary to be able either to link coreferential expressions or to type exactly the same referent name entered for previously encountered referents. Glozz and Analec (when used with schemata) require the creation of a schema for each chain, so chains are clearly identified. Referents that do not give rise to a chain (a schema) are more problematic. Analec, when used as an extension of TXM (without schemata but with referent names), offers an auto-completing list of referent names. GATE and MMAX2 have similar features. But sometimes the mere referent name is not enough for annotators who need a list of all the expressions associated with a referent, in order to decide how to annotate a new referring expression. Only Glozz offers users an easy way to list all the elements in a given chain.

[4] *Modélisation Contrastive et Computationnelle des Chaînes de Coréférence*, project from the French *Centre National de la Recherche Scientifique*.

[5] `https://www.oxygenxml.com/`

[6] A *singleton* is a referring expression that is not related to another one.

Several authors (Poudat and Landragin, 2017; Felt et al., 2010) have pointed out that some annotations can be made automatically, for example the annotation of parts of speech. CCASH (Felt et al., 2010) is entirely built around this idea, and has integrated modules (called "widgets") that automate simple annotation tasks. But it is difficult to create custom widgets. Tools that have no such integrated modules may be paradoxically more flexible since they allow the use of other programs *via* a common exchange format. For example, referring expressions may be manually marked with one dedicated tool, parts of speech may be automatically added with some other specialized tool, etc. In the Democrat project (Poudat and Landragin, 2017), using a chunker, like SEM (Tellier et al., 2012), has been considered to pre-annotate the text, so annotators would only have to create coreference relations.
All these highlights are not found at once in general purpose annotation and analysis tools, but require a dedicated tool, specifically designed to ease the coreference chain annotation process. SACR was designed in the light of this idea.

4. Presentation of SACR

4.1. Overview

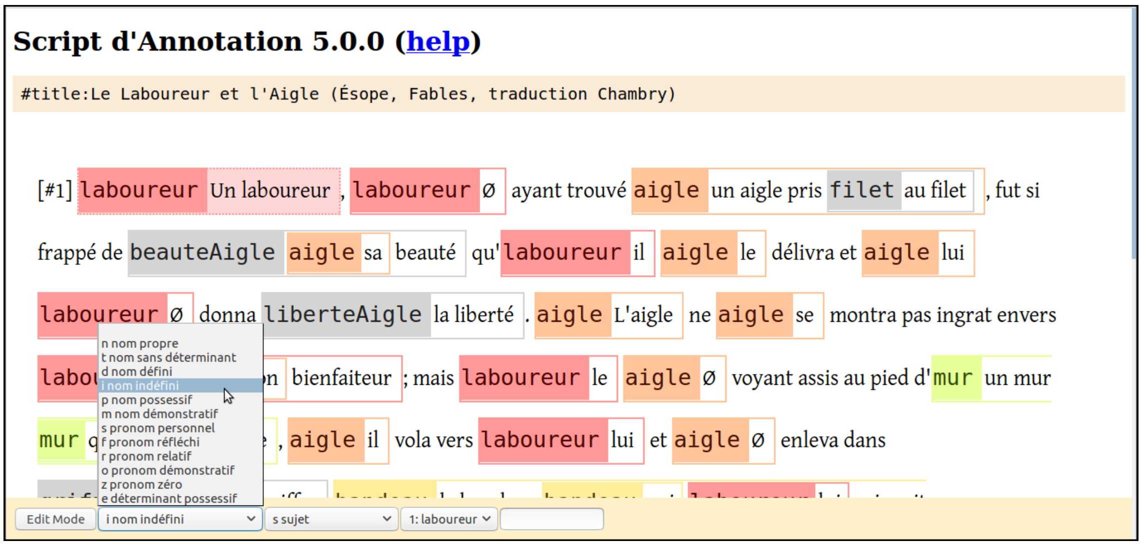
SACR (figure 1) is an open source, ready-to-use annotation tool. Written in HTML, CSS and JavaScript, it is a single-page application that works with Firefox or Chromium/Google Chrome. It can be used immediately, with no installation overhead, by virtually anyone on any platform. It is available online, but can also be downloaded for offline use.
Referring expressions are marked by clicking on their first and last words, coreference relations are made by dragging one expression and dropping it on another. There is almost no learning time and the tool is thus well suited for students of literary background or non-technical annotators as well as more expert users.
It is dedicated to annotation and users are expected to use some other tools (for example TXM) for the analyses. This has allowed us to optimize its interface specifically for the annotation process.

4.2. Creating annotations

Markables may be words (that is, a serie of letters) or characters for languages where characters are words (*e.g.* Chinese). A click on the first and last markables is enough to make a referring expression with a default unique name. Rather than having to type the referent name for each expression, a simple drag-and-drop operation is used to copy the name: this create a new coreference relation, which is symbolized by the coloring of the two linked expressions. When adding more related expressions, they get the same color, so each chain is identified with a unique color, while singletons (referring expressions not related to any other one) are left gray. Using eye-catching colors for the most prominent referents allows to easily identify them and to rely heavily on the drag-and-drop without having to carefully read and type referent names (*e.g.* if a pronoun "he" refers to "Paul", and "Paul" is in red, just drag the "he" and drop it on one of the red elements). Default names may be changed (chiefly for the most prominent referents), but this is not necessary, and annotators may want to rely only on the colors.
When it comes to annotate features like parts of speech or grammatical functions (or check them if they are the output of an automatic annotating system), a special edition mode allows annotators to use shortcuts (*e.g.* "d" for a noun with a definite article, "p" for a personal pronoun, etc.): the next expression is automatically selected and presented in the middle of the screen when annotators have made their choice. This way, one keystroke is enough for each annotation so that annotating or checking a feature for one hundred referring expressions requires only one hundred keystrokes, and nothing more.
SACR does not impose a predefined annotation scheme: users can use as many features as they want (parts of speech, grammatical functions, morphological informations, number and type of modifiers, named entity types, speakers (for oral transcriptions), etc.) and choose the tagset and the corresponding shortcuts. The annotation scheme is saved into a separate file and so may be used for several texts or corpora.

4.3. Helping the annotator

Annotation visualization is an important part of SACR. Marked expressions are surrounded by colored frames, allowing an easy view of several levels of nested expressions (for instance, a genitive in a relative clause). This is especially critical when all the referring expressions of a text must be marked.
A popup window (figure 2) lists all the referents and all the expressions associated with each referent. So annotators can look for already annotated expressions and decide whether they must attach, or not, the expression they are currently working on to a previously defined referent. Drag-and-dropping is also possible to link two somewhat distant expressions. This list may also be used to check if an expression has not been related to the wrong coreference chain.
Referring expressions may be filtered in several ways. It is possible to show only expressions from a given chain. Users can also search expressions with specific features, for example all expressions with a certain part of speech (all pronouns, etc.) or grammatical function (all subjects, etc.), if these features are annotated. The search can be made with a literal or with a regular expression, so it is possible to combine criteria (*e.g.* searching all nouns and pronouns). Such searches are useful to check annotations at the end of the annotation process, for example by looking for expressions lacking any value for a feature (which means that an annotator has forgotten to annotate it).
SACR does not perform any automatic annotation, but its simple import/export text format allows converting to and from other tools that can perform such annotations. A nominal chunker may be used as a pre-treatment so that annotators do not need to mark referring expressions, but only to link them. Several tools have been tested, including TreeTagger (Schmid, 1994; Schmid, 1995) and SEM (Tellier et al., 2012), but they are limited and add more work (too many wrong results) than they solve, since correcting

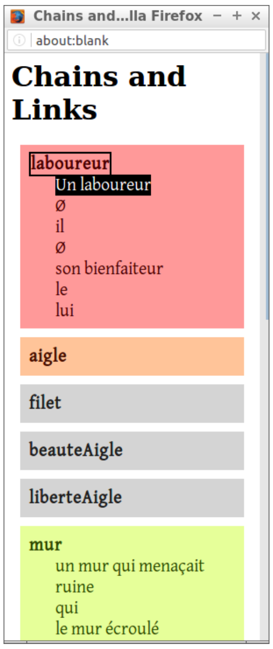


Script d'Annotation 5.0.0 (help)

```
#title:Le Laboureur et l'Aigle (Ésope, Fables, traduction Chambry)
```

[#1] laboureur Un laboureur , laboureur Ø ayant trouvé aigle un aigle pris filet au filet , fut si frappé de beauteAigle aigle sa beauté qu'laboureur il aigle le délivra et aigle lui laboureur Ø donna liberteAigle la liberté . aigle L'aigle ne aigle se montra pas ingrat envers labou … n bienfaiteur ; mais laboureur le aigle Ø voyant assis au pied d'mur un mur mur q … , aigle il vola vers laboureur lui et aigle Ø enleva dans

Figure 1: Annotating coreference and parts of speech with SACR.



Chains and Links

laboureur
Un laboureur
Ø
il
Ø
son bienfaiteur
le
lui

aigle

filet

beauteAigle

liberteAigle

mur
un mur qui menaçait ruine
qui
le mur écroulé

Figure 2: List of all referring expressions grouped by chains. Elements of a chain can be collapsed or expanded, and it is possible to drag-and-drop any element of the list from or to the main window, or vice-versa, to add a coreference relation.

wrong annotations requires an additional cognitive workload; furthermore, referring expressions that are not detected by the tool may be missed by annotators if they rely too much on the tool output. Features like parts of speech may also be annotated with other tools, and checked within SACR.

4.4. Output format

Annotations made in SACR are stored inline, that is, within the text. The aim is to offer an easy-to-parse format that can be converted to other formats with the help of simple scripts. Such scripts exist for example to convert to and from the Glozz format (Widlöcher and Mathet, 2012) (a format in which annotations are deported and that can be imported in tools such as Analec or TXM) or the CONLL-2011 format. Inline annotations also allow the modification of the text itself, for example for spelling correction. Stand-off annotations, on the other hand, offer other advantages like the possibility to easily add annotation layers (*e.g.* output from a part-of-speech tagger and a parser). Since Glozz offers such a stand-off format and there are converters from and to the Glozz format, it is easy to combine the advantages of both type of annotations storage.
The format includes any metadata that are needed for the program (colors), the organization of the corpus (identifier, title, author, source, etc.), and even the text structure (headings of different levels; sections like "introduction" or "conclusion"; etc.).
The ouput file itself contains the text with the marked expressions enclosed between curly brackets. Features are stored near the opening bracket. Here is a short example:

```
#title:  Le laboureur et l'Aigle

{laboureur:categorie="i
nom indéfini",fonction="s
sujet",head="1",expansion="" Un
laboureur}, {laboureur:categorie="z
pronom zéro",fonction="s
sujet",head="0",expansion="" Ø}
ayant trouvé {aigle:categorie="i
nom indéfini",fonction="v compl
(verbe)",head="1",expansion="v"
un aigle pris {filet:categorie="d
nom défini",fonction="c
circonstant",head="1",expansion="" au
filet}}...
```

The file describing the features to be annotated is also a simple text file listing all the values allowed for a property, or offering a text box for annotators to complete:

```
PROP:name=categorie
n nom propre
t nom sans déterminant
d nom défini
...

PROP:name=fonction
s sujet
v compl (verbe)
n compl (nom)
...

PROP:name=head,type=head

PROP:name=expansion,type=text
```

A special type of property, called `head`, is set to let annotators choose the syntactic head of the referring expression (which is, in some annotation schemes, required to be a syntactic phrase).

Annotators may define the feature set they want, with the features and values adapted for specific tasks and/or languages. SACR may also be used without any feature set defined: in this case, annotators just have to mark referring expressions and create coreference relations.

4.5. Using SACR: User feedback and experiment

This tool has been easily used by several annotators, both students and linguists, even when they were not at ease with computers. The preferred way of annotating is to mark all the referring expressions of a paragraph: this first reading allows annotators to get the meaning of the passage. Then coreference relations are marked by drag-and-dropping expressions one over the other. Finally, when the whole text has been annotated with coreference, users may annotate features for each expression (this process may be automated by an external tool).

We conducted a small-scale experiment to compare SACR to Analec (Landragin et al., 2012) as integrated in TXM (Heiden, 2010). Three master's students in linguistics and computational linguistics have been trained to use both SACR and TXM, and have been asked to mark referring expressions and coreference relations in a 18,900 token long corpus of fables (easy to understand and annotate) and legal European texts (more challenging to understand and annotate, with referring expressions sometimes spanning several lines and nested several levels deep).

When using SACR, annotators have taken 55 % less time on average to complete the task, which means that annotation with SACR is about twice as fast as with Analec-TXM. We had expected difficult legal texts to be annotated even faster with SACR than simpler texts like fables, but this was not the case: the reduction in time is similar for both types of texts.

Beyond coreference, SACR has been used for other annotation tasks that necessitated a quick and user-friendly interface. It can handle any annotation scheme that requires to mark chunks of text and to annotate a feature set.

4.6. Limitations

SACR is dedicated to referring expression and coreference chain annotation, with or without a set of features for each expression. While it may be used to annotate other type of linguistic phenomena (named entities, specific vocabularies, etc.), its focus on coreference chains implies some limitations. Most notably, there is no possibility of discontinuous annotations because each referring expression is assumed to be a syntactic phrase. This is also the reason why there cannot be any overlapping marked expression, but only nested ones. This limitation fits the technical choice of using HTML as a way of representation. Furthermore, we have made the choice of considering coreference chain as a set of referring expressions, without the possibility of annotating each relation in particular. For example, while it is possible to annotate features for expressions A and B, is not possible to annotate specific features for the *relation* from A to B.

5. Conclusion

In this paper, we presented SACR, a new easy-to-use, annotator-friendly tool designed and optimized for coreference chain annotation. It requires no installation and is ready-to-use even by non-technical users. Marking referring expressions is done by clicking on the first and last words of an expression, and coreference chains by drag-and-dropping expressions one over the other.

6. Acknowledgements

This work was supported by the Democrat project (*DEscription et MOdélisation des Chaînes de Référence: outils pour l'Annotation de corpus (en diachronie et en langues comparées) et le Traitement automatique*), from the French National Research Agency (ANR) (ANR-15-CE38-0008).

7. Bibliographical References

(2014). BRAT: Brat Rapid Annotation Tool. `http://brat.nlplab.org`.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M. A., Saggion, H., Petrak, J., Li, Y., and Peters, W. (2011). *Text Processing with GATE (Version 6)*.

Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. (2013). Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLOS Computational Biology*, 9(2):1–16.

Felt, P., Merkling, O., Carmen, M., Ringger, E., Lemmon, W., Seppi, K., and Haertel, R. (2010). CCASH: A web application framework for efficient, distributed language resource development. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Fort, K. (2012). *Les ressources annotées, un enjeu pour l'analyse de contenu: vers une méthodologie de l'annotation manuelle de corpus.* Ph.D. thesis, Paris 13.

Habert, B. (2005). Portrait de linguiste(s) à l'instrument. *Texto!*, 10(4).

Heiden, S. (2010). The TXM Platform: Building Open-Source Textual Analysis Software Compatible with the TEI Encoding Scheme. In Ryo Otoguro, et al., editors, *24th Pacific Asia Conference on Language, Information and Computation*, pages 389–398, Sendai, Japan. Institute for Digital Enhancement of Cognitive Development, Waseda University.

Landragin, F., Poibeau, T., and Victorri, B. (2012). Analec: a new tool for the dynamic annotation of textual data. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Landragin, F., Potier, J., and Bothua, M. (2017). Annotation manuelle d'expressions référentielles: expérimentations pour simplifier les prises de décisions et optimiser le processus. In *9èmes Journées Internationales de la Linguistique de Corpus (JLC 2017)*, Grenoble, France.

Landragin, F. (2011). Une procédure d'analyse et d'annotation des chaînes de coréférence dans des textes écrits. *Corpus*, 10:61–80.

Mélanie-Becquet, F. and Landragin, F. (2014). Linguistique outillée pour l'étude des chaînes de référence: questions méthodologiques et solutions techniques. *Langages*, 195.

Müller, C. and Strube, M. (2006). Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, et al., editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a. M., Germany.

Muzerelle, J., Lefeuvre, A., Schang, E., Antoine, J.-Y., Pelletier, A., Maurel, D., Eshkol, I., and Villaneau, J. (2014). ANCOR Centre, a large free spoken french coreference corpus: Description of the resource and reliability measures. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

O'Donnell, M. (2008). Demonstration of the UAM CorpusTool for text and image annotation. In *Proceedings of the 46th annual meeting of the Association for Computational Linguistics on human language technologies: Demo session*, pages 13–16. Association for Computational Linguistics.

Poudat, C. and Landragin, F. (2017). *Explorer un corpus textuel*. Champs linguistiques. De Boeck.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*.

Schmid, H. (1995). Improvements in part-of-speech tagging with an application to german. In *Proceedings of the ACL SIGDAT-Workshop*.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107. Association for Computational Linguistics.

Tellier, I., Duchier, D., Eshkol, I., Courmet, A., and Martinet, M. (2012). Apprentissage automatique d'un chunker pour le français. In Gilles Sérasset Georges Antoniadis, Hervé Blanchon, editor, *TALN2012*, volume 2 of *Actes de la conférence conjointe JEP-TALN-RECITAL 2012*, pages 431–438, Grenoble, France.

Widlöcher, A. and Mathet, Y. (2012). The glozz platform: A corpus annotation and mining tool. In *Proceedings of the 2012 ACM symposium on Document engineering*, pages 171–180. ACM.