Originally published in: Nagata, Masaaki/Pedersen, Ted (Eds.): Proceedings of the ACL Interactive Poster and Demonstration Sessions.
26 June 2005, University of Michigan. – Stroudsburg: Association for Computational Linguistics, 2005. Pp. 109-112.

A Flexible Stand-Off Data Model with Query Language for Multi-Level Annotation

Christoph Müller
EML Research gGmbH
Villa Bosch
Schloß-Wolfsbrunnenweg 33
69118 Heidelberg, Germany
mueller@eml-research.de

Abstract

We present an implemented XML data model and a new, simplifie query language for multi-level annotated corpora. The new query language involves automatic conversion of queries into the underlying, more complicated MMAXQL query language. It supports queries for sequential and hierarchical, but also associative (e.g. coreferential) relations. The simplifie query language has been designed with non-expert users in mind.

1 Introduction

Growing interest in richly annotated corpora is a driving force for the development of annotation tools that can handle multiple levels of annotation. We fin it crucial in order to make full use of the potential of multi-level annotation that individual annotation levels be treated as *self-contained modules* which are independent of other annotation levels. This independence should also include the storing of each level in a separate file If these principles are observed, annotation data management (incl. level addition, removal and replacement, but also conversion into and from other formats) is greatly facilitated.

The way to keep individual annotation levels independent of each other is by definin each with direct reference to the underlying basedata, i.e. the text or transcribed speech. Both sequential and hierarchical (i.e. embedding or dominance) relations between markables on different levels are thus only expressed *implicitly*, viz. by means of the relations of their basedata elements.

While it has become common practice to use the stand-off mechanism to relate several annotation levels to one basedata file it is also not uncommon to fin this mechanism applied for relating markables to other markables (on a different or the same level) directly, expressing the relation between them *explicitly*. We argue that this is unfavourable not only with respect to annotation data management (cf. above), but also with respect to *querying*: Users should not be required to formulate queries in terms of structural properties of data representation that are irrelevant for their query. Instead, users should be allowed to relate markables from all levels in a fairly unrestricted and ad-hoc way. Since querying is thus considerably simplified exploratory data analysis of annotated corpora is facilitated for all users, including non-experts.

Our multi-level annotation tool MMAX2[1] (Müller & Strube, 2003) uses implicit relations only. Its query language MMAXQL is rather complicated and not suitable for naive users. We present an alternative query method consisting of a simpler and more intuitive query language and a method to generate MMAXQL queries from the former. The new, *simplifie* MMAXQL can express a wide range of queries in a concise way, including queries for associative relations representing e.g. coreference.

2 The Data Model

We propose a stand-off data model implemented in XML. The basedata is stored in a simple XML fil

[1] The current release version of MMAX2 can be downloaded at http://mmax.eml-research.de.

Proceedings of the ACL Interactive Poster and Demonstration Sessions,
pages 109–112, Ann Arbor, June 2005.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE words SYSTEM "words.dtd">
<words>
 ...
<word id="word_1064">My</word>
<word id="word_1065">,</word>
<word id="word_1066">uh</word>
<word id="word_1067">,</word>
<word id="word_1068">cousin</word>
<word id="word_1069">is</word>
<word id="word_1070">a</word>
<word id="word_1071">F</word>
<word id="word_1072">B</word>
<word id="word_1073">I</word>
<word id="word_1074">agent</word>
<word id="word_1075">down</word>
<word id="word_1076">in</word>
<word id="word_1077">Miami</word>
<word id="word_1078">.</word>
 ...
<word id="word_1085">she</word>
...
</words>
```

Figure 1: `basedata` fil (extract)

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE markables SYSTEM "markables.dtd">
<markables xmlns="www.eml.org/NameSpaces/utterances">
 ...
<markable id="markable_116" span="word_1064..word_1078"/>
 ...
</markables>
```

Figure 2: `utterances` level fil (extract)

which serves to identify individual tokens[2] and associate an ID with each (Figure 1).

In addition, there is one XML fil for each annotation level. Each level has a unique, descriptive name, e.g. `utterances` or `pos`, and contains annotations in the form of <markable> elements. In the most simple case, a markable only identifie a sequence (i.e. *span*) of basedata elements (Figure 2).

Normally, however, a markable is also associated with arbitrarily many user-define attribute-value pairs (Figure 3, Figure 4). Markables can also be discontinuous, like `markable_954` in Figure 4.

For each level, admissible attributes and their values are define in a separate annotation scheme fil (not shown, cf. Müller & Strube (2003)). Freetext attributes can have any string value, while nominal attributes can have one of a (user-defined closed set of possible values. The data model also supports associative relations between markables: *Markable set* relations associate arbitrarily many markables with each other in a transitive, undirected way. The `coref_class` attribute in Figure 4 is an example of how such a relation can be used to represent a coreferential relation between markables (here: `markable_954` and `markable_963`, rest of set

[2]Usually words, but smaller elements like morphological units or even characters are also possible.

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE markables SYSTEM "markables.dtd">
<markables xmlns="www.eml.org/NameSpaces/pos">
 ...
<markable id="markable_665" span="word_1064" pos="PRP$"/>
<markable id="markable_666" span="word_1065" pos=","/>
<markable id="markable_667" span="word_1066" pos="UH"/>
<markable id="markable_668" span="word_1067" pos=","/>
<markable id="markable_669" span="word_1068" pos="NN"/>
<markable id="markable_670" span="word_1069" pos="VBZ"/>
<markable id="markable_671" span="word_1070" pos="DT"/>
<markable id="markable_672" span="word_1071" pos="NNP"/>
<markable id="markable_673" span="word_1072" pos="NNP"/>
<markable id="markable_674" span="word_1073" pos="NNP"/>
<markable id="markable_675" span="word_1074" pos="NN"/>
<markable id="markable_676" span="word_1075" pos="IN"/>
<markable id="markable_677" span="word_1076" pos="IN"/>
<markable id="markable_678" span="word_1077" pos="NNP"/>
<markable id="markable_679" span="word_1078" pos="."/>
 ...
<markable id="markable_686" span="word_1085" pos="PRP"/>
...
</markables>
```

Figure 3: `pos` level fil (extract)

```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE markables SYSTEM "markables.dtd">
<markables xmlns="www.eml.org/NameSpaces/ref_exp">
 ...
<markable id="markable_953" span="word_1064" type="poss_det"/>
<markable id="markable_954" span="word_1064,word_1068" type="np"
  coref_class="set_3"/>
<markable id="markable_955" span="word_1070..word_1074" type="np"/>
<markable id="markable_956" span="word_1071..word_1073" type="pn"/>
<markable id="markable_957" span="word_1077" type="pn"/>
 ...
<markable id="markable_963" span="word_1085" type="pron"
  coref_class="set_3"/>
...
</markables>
```

Figure 4: `ref_exp` level fil (extract)

not shown). *Markable pointer* relations associate with one markable (the *source*) one or more *target* markables in an intransitive, directed fashion.

3 Simplifie MMAXQL

Simplifie MMAXQL is a variant of the MMAXQL query language. It offers a simpler and more concise way to formulate certain types of queries for multi-level annotated corpora. Queries are automatically converted into the underlying query language and then executed. A query in simplifie MMAXQL consists of a sequence of *query tokens* which are combined by means of *relation operators*. Each query token queries exactly one basedata element (i.e. word) or one markable.

3.1 Query Tokens

Basedata elements can be queried by matching regular expressions. Each basedata query token consists of a regular expression in single quotes, which must *exactly* match one basedata element. The query

```
'[Tt]he'
```

matches all definit articles, but not e.g. *ether* or

there. For the latter two words to also match, wildcards have to be used:

```
'.*[Tt]he.*'
```

Sequences of basedata elements can be queried by simply concatenating several space-separated[3] tokens. The query

```
'[Tt]he  [A-Z].+'
```

will match sequences consisting of a definit article and a word beginning with a capital letter.

Markables are the carriers of the actual annotation information. They can be queried by means of string matching and by means of attribute-value combinations. A markable query token has the form

```
string/conditions
```

where `string` is an optional regular expression and `conditions` specifie which attribute(s) the markable should match. The most simple 'condition' is just the name of a markable level, which will match all markables on that level. If a regular expression is also supplied, the query will return only the matching markables. The query

```
[Aa]n?\s.*/ref_exp
```[4]

will return all markables from the `ref_exp` level beginning with the indefinit article.

The `conditions` part of a markable query token can indeed be much more complex. A main feature of simplifie MMAXQL is that redundant parts of conditions can **optionally** be left out, making queries very concise. For example, the *markable level name* can be left out if the name of the *attribute* accessed by the query is unique across all active markable levels. Thus, the query

```
/!coref_class=empty
```

can be used to query markables from the `ref_exp` level which have a non-empty value in the `coref_class` attribute, granted that only one attribute of this name exists.[5] The same applies to the names of *nominal attributes* if the *value* specifie in the query unambiguously points to this attribute. Thus, the query

```
/pn
```

can be used to query markables from the `pos` level which have the value `pn`, granted that there is exactly one nominal attribute with the possible value `pn`. Several conditions can be combined into one query token. Thus, the query

```
/{poss_det,pron},!coref_class=empty
```

returns all markables from the `ref_exp` level that are either possessive determiners or pronouns and that are part in some coreference set.[6]

3.2 Relation Operators

The whole point of querying corpora with multi-level annotation is to relate markables from different levels to each other. The reference system with respect to which the relation between different markables is established is the sequence of basedata elements, which is the same for all markables on all levels. Since this bears some resemblance to different *events* occurring in several *temporal* relations to each other, we (like also Heid et al. (2004), among others) adopt this as a metaphor for expressing the **sequential** and **hierarchical** relations between markables, and we use a set of relation operators that is inspired by (Allen, 1991). This set includes (among others) the operators `before`, `meets` (default), `starts`, `during/in`, `contains/dom`, `equals`, `ends`, and some inverse relations. The following examples give an idea of how individual query tokens can be combined by means of relation operators to form complex queries. The example uses the ICSI meeting corpus of spoken multiparty dialogue.[7] This corpus contains, among others, a `segment` level with markables roughly corresponding to speaker turns, and a `meta` level containing markables representing e.g. pauses, emphases, or sounds like breathing or mike noise. These two levels and the basedata level can be combined to retrieve instances of *you know* that occur in segments spoken by female speakers[8] which also contain a pause or an emphasis:

```
'[Yy]ou know' in (/participant={f.*} dom /{pause,emphasis})
```

[3] Using the fact that `meets` is the default relation operator, cf. Section 3.2.

[4] The space character in the regular expression must be masked as `\s` because otherwise it will be interpreted as a query token separator.

[5] If this condition does not hold, attribute names can be disambiguated by prepending the markable level name.

[6] The curly braces notation is used to specify several OR-connected values for a single attribute, while a comma *outside* curly braces is used to AND-connect several conditions relating to different attributes.

[7] Obtained from the LDC and converted into MMAX2 format, preserving all original information.

[8] The firs letter of the `participant` value encodes the speaker's gender.

Relation operators for **associative** relations (i.e. markable set and markable pointer) are `nextpeer`, `anypeer` and `nexttarget`, `anytarget`, respectively. Assuming the sample data from Section 2, the query

```
/ref_exp nextpeer:coref_class /ref_exp
```

retrieves pairs of anaphors (right) and their direct antecedents (left). The query can be modifie to

```
/ref_exp nextpeer:coref_class (/ref_exp equals /pron)
```

to retrieve only anaphoric *pronouns* and their direct antecedents.

If a query is too complex to be expressed as a single query token sequence, *variables* can be used to store intermediate results of sub-queries. The following query retrieves pairs of utterances (incl. the referring expressions embedded into them) that are more than 30 tokens[9] apart, and assigns the resulting 4-tuples to the variable `$distant_utts`.

```
(/utterances dom /ref_exp) before:31- (/utterances dom /ref_exp)
   -> $distant_utts
```

The next query accesses the second and last column in the temporary result (by means of the zero-based column index) and retrieves those pairs of anaphors and their direct antecedents that occur in utterances that are more than 30 tokens apart:

```
$distant_utts.1 nextpeer:coref_class $distant_utts.3
```

4 Related Work

In the EMU speech database system (Cassidy & Harrington, 2001) the hierarchical relation between levels has to be made explicit. Sequential and hierarchical relations can be queried like with simplifie MMAXQL, with the difference that e.g. for sequential queries, the elements involved must come from the same level. Also, the result of a hierarchical query always only contains either the parent or child element. The EMU data model supports an association relation (similar to our markable pointer) which can be queried using a => operator.

Annotation Graphs (Bird & Liberman, 2001) identify elements on various levels as arcs connecting two points on a time scale shared by all levels. Relations between elements are thus also represented implicitly. The model can also express a binary association relation. The associated Annotation Graph query language (Bird et al., 2000) is very explicit, which makes it powerful but at the same time possibly too demanding for naive users.

The NITE XML toolkit (Carletta et al., 2003) define a data model that is close to our model, although it allows to express hierarchical relations explicitly. The model supports a labelled pointer relation which can express one-to-many associations. The associated query language NXT Search (Heid et al., 2004) is a powerful declarative language for querying diverse relations (incl. pointers), supporting quantificatio and constructs like `forall` and `exists`.

5 Future Work

We work on support for queries like 'pairs of referring expressions that are a certain number of referring expressions apart'. We also want to include wild cards and proximity searches, and support for automatic markable creation from query results.

Acknowledgements

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany.

References

Allen, James (1991). Time and time again. *International Journal of Intelligent Systems*, 6(4):341–355.

Bird, Steven, Peter Buneman & Wang-Chiew Tan (2000). Towards a query language for annotation graphs. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation,* Athens, Greece, 31 May-June 2, 2000, pp. 807–814.

Bird, Steven & Mark Liberman (2001). A formal framework for linguistic annotation. *Speech Communication*, 33:23–60.

Carletta, Jean, Stefan Evert, Ulrich Heid, Jonathan Kilgour, J. Robertson & Holger Voormann (2003). The NITE XML toolkit: fl xible annotation for multi-modal language data. *Behavior Research Methods, Instruments, and Computers*, 35:353–363.

Cassidy, Steve & Jonathan Harrington (2001). Multi-level annotation in the EMU speech database management system. *Speech Communication*, 33:61–78.

Heid, Ulrich, Holger Voormann, Jan-Torsten Milde, Ulrike Gut, Katrin Erk & Sebastian Pado (2004). Querying both time-aligned and hierarchical corpora with NXT search. In *Proceedings of the 4th International Conference on Language Resources and Evaluation,* Lisbon, Portugal, 26-28 May, 2004, pp. 1455–1458.

Müller, Christoph & Michael Strube (2003). Multi-level annotation in MMAX. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue,* Sapporo, Japan, 4-5 July 2003, pp. 198–207.

[9] A means to express distance in terms of markables is not yet available, cf. Section 5.