Coral: Corpus Access in Controlled Language*

Tobias Kuhn[1,2] and Stefan Hoefler[1]

[1]Institute of Computational Linguistics
University of Zurich, Switzerland

[2]Department of Intelligent Computer Systems
University of Malta

kuhntobias@gmail.com / hoefler@cl.uzh.ch

Abstract

This paper presents Coral, an interface in which complex corpus queries can be expressed in a controlled subset of natural English. With the help of a predictive editor, users can compose queries and submit them to the Coral system, which then automatically translates them into formal AQL statements. The paper gives an overview of the controlled natural language developed for Coral and describes the functionalities of the predictive editor provided for it. It also reports on a user experiment in which the system was evaluated. The results show that, with Coral, corpora of annotated texts can be queried easier and faster than with the existing ANNIS interface. Our system demonstrates that complex corpora can be accessed without the need to learn a complicated formal query language.

1 Introduction

In recent years, the analysis of large corpora of annotated texts has come to play an ever more important role in linguistic research. Not only has an increasing number of corpora become available but the amount of linguistic information with which they are annotated is on the rise too: corpora, nowadays, are annotated with part-of-speech tags, syntactic structure and even semantic or other linguistic information.

More complex annotations also require more complex queries if they are to be exploited effectively. The problem is that it is often not possible to make the full complexity of such queries available through simple, user-friendly web forms. At the moment, linguists can thus only use this new type of corpora effectively if they invest a considerable amount of time and effort into acquiring complicated

*This manuscript will appear in *Corpora*, issue 7.2, Edinburgh University Press.

and idiosyncratic formal query languages. Examples of user-interfaces that embed such formal query languages are XKWIC (Christ, 1994), TigerSEARCH (König et al., 2003), and more recently ANNIS (Zeldes et al., 2009). Users with no particular background in computer science would benefit from simpler and more intuitive corpus query interfaces, be they corpus linguists or users employing corpora-based systems for language learning (Brill, 1993) and for translation support[1].

In this paper, we present a way to tackle this problem: we have developed Coral, an interface in which complex corpus queries can be expressed in a controlled subset of natural English. They are then automatically translated into the underlying formal query language AQL (Zeldes et al., 2009). We show that with this interface, complex corpora can be queried effectively without much training or prior knowledge.

The remainder of the paper falls into four main parts. In section 2, we introduce the method of controlled natural language on which Coral is based. In section 3, we give an overview of the syntax and semantics of Coral's controlled English and introduce its special editor. In section 4, we present an evaluation of our approach in the form of a user experiment. In section 5, we place Coral in the context of related work.

2 Approach

The central idea of the Coral system is to employ the method of controlled natural language to provide an interface for annotated text corpora in which users can compose complex queries in a straight-forward and intuitive way without much training or prior knowledge.

Controlled natural languages (Wyner et al., 2010; Pool, 2006) are artificially defined subsets of natural languages whose vocabulary, syntax and/or semantics have been restricted in order to reduce or eliminate ambiguity and complexity. Some of these languages are completely formal and can be automatically mapped to some sort of logic. Their goal is to improve human-computer communication. Examples are ACE (Fuchs et al., 2008), CLP (Clark et al., 2005) and PENG (Schwitter and Tilbrook, 2006). Other controlled natural languages are richer and less restricted, but cannot be interpreted automatically. Caterpillar Fundamental English and ALCOGRAM are examples among many others (Adriaens and Schreors, 1992). Their goal is to improve the communication among humans, especially non-native speakers of the respective language.

The controlled natural language implemented in Coral is of the first type: it can be mapped deterministically onto a formal representation, namely onto the ANNIS Query Language AQL. Coral queries, as shown in Fig. 1, are thus at the same time statements in natural English *and* statements in a formal query language. They combine the intuitiveness of natural language with the precision of formal languages. From a linguistic perspective, such an approach is thus of twofold interest: firstly, it offers a novel way of querying annotated text corpora,

[1] e.g. `http://www.linguee.com/`

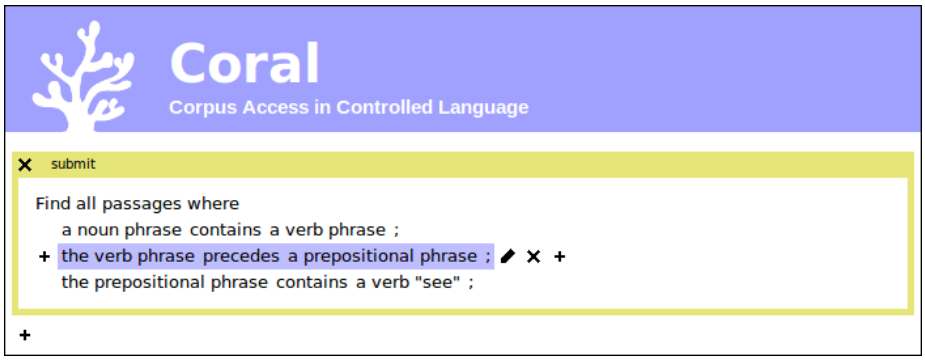

Figure 1: A screenshot of the Coral web interface.

and secondly, it uses natural language – the very object of study of linguistics – as an interface to do so.

Statements in controlled natural language are easier to read and understand than statements in formal languages (Kuhn, 2010b). However, they are still relatively hard to write: without the help of adequate authoring tools, users need to keep in mind the often numerous and complex restrictions of the controlled language they are using. Two main approaches have been suggested to tackle this problem: conceptual authoring (Power et al., 2009) and predictive editors (Tennant et al., 1983; Schwitter et al., 2003).

In both approaches, a tool supports users in composing statements incrementally and informs them about their possible next actions. With a conceptual authoring tool, users do not have direct control of the text in the controlled language. They can only trigger specific actions to change the underlying logic model. After each change, the model is verbalized in the respective controlled language and shown to the user. With a predictive editor, users have more fine-grained control of the actual syntax of the statements. A statement is composed step by step, i.e. phrase by phrase or word by word, from the start to the end of the statement. At each step, the user sees all possible options of how to continue the statement. We chose the second approach and have equipped the Coral system with such a predictive editor to support users in composing queries in Coral's controlled English (see section 3.2 below).

3 Coral

Coral allows for users to compose queries in a controlled natural language and automatically translates them into AQL statements.[2] Coral's output can then be passed on to ANNIS or any other AQL-compliant query engine (in the case

[2]We chose AQL because it was specifically designed for the composition of complex queries on multi-level linguistic corpora (Chiarcos et al., 2008).

AQL gets implemented in other systems in the future). We used version 2.1.7 of ANNIS and AQL. Coral is implemented as a web-based application that can be easily accessed with a browser.

We will now (1) provide an overview of the syntax and semantics of Coral's controlled English, (2) introduce the predictive editor offered by the Coral system and (3) briefly discuss the implementation of its grammar and lexicon.

3.1 Coral's Controlled English

Fig. 1 shows a screenshot of the Coral web interface with a sample query. Coral queries start with the phrase: *Find all passages where...* This initial phrase is followed by the actual query conditions, which take the form of one or more sentences separated by semicolons. The formal semantics of these query conditions is defined by a deterministic mapping to AQL.

The most basic component of an AQL query is one that refers to a specific token (e.g. *telephone*). The following example shows such an AQL expression (a) and its equivalent in Coral's controlled English (c).

(1) a. `token="telephone"`

c. *a token "telephone"*

Both languages also support the shorter forms *"telephone"* or `"telephone"` respectively. A search for a node with a specific attribute-value pair is realised as follows in AQL and in Coral:

(2) a. `cat="VP"`

c. *a structure has an attribute "cat" of value "VP"*

Coral supports shortcuts for specific attributes or for whole attribute-value pairs in its lexicon: e.g. *category* for `cat` or *verb phrase* for `cat="VP"` (cf. section 3.3). This option makes it possible to phrase the respective queries in a more straightforward manner:

(3) a. `cat="VP"`

c1. *a structure with the category "VP"*

c2. *a verb phrase*

To express relations, AQL uses special symbols (e.g. `.*` for precedence) together with relative addressing (`#1` referring to the first of the introduced nodes, `#2` to the second, etc.); in Coral, such relations are mapped onto verbs:

(4) a. `"a" & "telephone" & #1 .* #2`

c. *a token "a" is followed by a token "telephone"*

Table 1 provides a list of AQL's relation symbols and the verbs they map to in Coral. Verbs can be used in active or in passive voice; all of the below Coral statements are thus equivalent:

Coral	AQL
[1] *precedes* [2]	`#1 .* #2`
[1] *directly precedes* [2]	`#1 . #2`
[1] *is preceded by* [2]	`#2 .* #1`
[1] *is directly preceded by* [2]	`#2 . #1`
[1] *follows* [2]	`#2 .* #1`
[1] *directly follows* [2]	`#2 . #1`
[1] *is followed by* [2]	`#1 .* #2`
[1] *is directly followed by* [2]	`#1 . #2`
[1] *contains* [2]	`#1 >* #2`
[1] *directly contains* [2]	`#1 > #2`
[1] *is contained in* [2]	`#2 >* #1`
[1] *is directly contained in* [2]	`#2 > #1`
[1] *is identical to* [2]	`#1 _=_ #2`
[1] *includes* [2]	`#1 _i_ #2`
[1] *is included in* [2]	`#2 _i_ #1`
[1] *overlaps with* [2]	`#1 _o_ #2`
[1] *overlaps on the right with* [2]	`#1 _or_ #2`
[1] *overlaps on the left with* [2]	`#1 _ol_ #2`
[1] *is left-aligned with* [2]	`#1 _l_ #2`
[1] *is right-aligned with* [2]	`#1 _r_ #2`
[1] *shares a parent with* [2]	`#1 $ #2`
[1] *shares an ancestor with* [2]	`#1 $* #2`

Table 1: Default relations of the Coral lexicon and their AQL equivalents.

(5) c1. *"a" precedes "telephone"*

c2. *"telephone" is preceded by "a"*

c3. *"telephone" follows "a"*

c4. *"a" is followed by "telephone"*

As shown in the sample query in Fig. 1, definite noun phrases can be used as anaphoric expressions to refer to objects that have been introduced earlier in the query. To facilitate anaphora resolution in complex queries, Coral additionally permits the use of numbers as explicit identifiers:

(6) a. `cat="S" & cat="S" & #1 >* #2 & pos="V" & #1 >* #3 & #2 .* #3`

c. *a sentence (1) contains a sentence (2);*
sentence (1) contains a verb;
sentence (2) precedes the verb

AQL's unary operators are realised as prepositional phrases or as the direct object of the verb *have*:

(7) a. `cat="NP" & #1:length=6`

c1. *a noun phrase with length 6*

c2. *a noun phrase has length 6*

Negation is only possible on the level of attribute-value pairs in AQL; in Coral, this feature can be expressed by inserting *not*:

(8) a. `pos="N" & lemma!="telephone" & #1 _=_ #2`

c. *a noun with a lemma that does not have the value "telephone"*

The value of an attribute can also be indicated as a regular expression; in Coral, regular expressions are introduced by the verb *match*:

(9) a. `pos="N" & lemma=/d[aeiou]g/ & #1 _=_ #2`

c. *a noun has a lemma that matches "d[aeiou]g"*

In AQL, edges can be assigned attribute-value pairs as labels. The example below shows an edge labeled with the user-defined attribute `func`. In Coral, such edge labels are expressed as prepositional phrases with the preposition *as*:

(10) a. `cat="S" & pos="N" & #1 >[func="OA"] #2`

c. *a sentence contains a noun as an accusative object*

A special key phrase makes it possible to indicate the distance between tokens:

(11) a. `pos="N" & pos="Conj" & #2 .5,10 #1`

c. *a noun follows a conjunction at a distance of 5 to 10 tokens*

Complex queries can be constructed by combining these elements:

(12) a.
```
CAT="S" & POS=V & LEMMA=/.*ize/ & #2 _=_ #3 & #1 >* #2 &
CAT="NP" & #1 >[func=OA] #4 & POS=N & LEMMA="telephone"
& #5 _=_ #6 & #2 .5,10 #5 & #4 >* #5
```

c. *a sentence contains a verb with a lemma that matches ".*ize" and contains a noun phrase as an accusative object;*
the noun phrase contains a noun with the lemma "telephone" that follows the verb at a distance of 5 to 10 tokens;

AQL has some more features, which cannot all be discussed here. Coral covers all but some of the most recently added features of AQL.

3.2 Predictive Editor

As mentioned at the beginning of this paper, we chose the predictive editor approach to solve the problem that controlled English is easy to read but relatively hard to write. We took the editor that is part of the ACE Editor[3] and

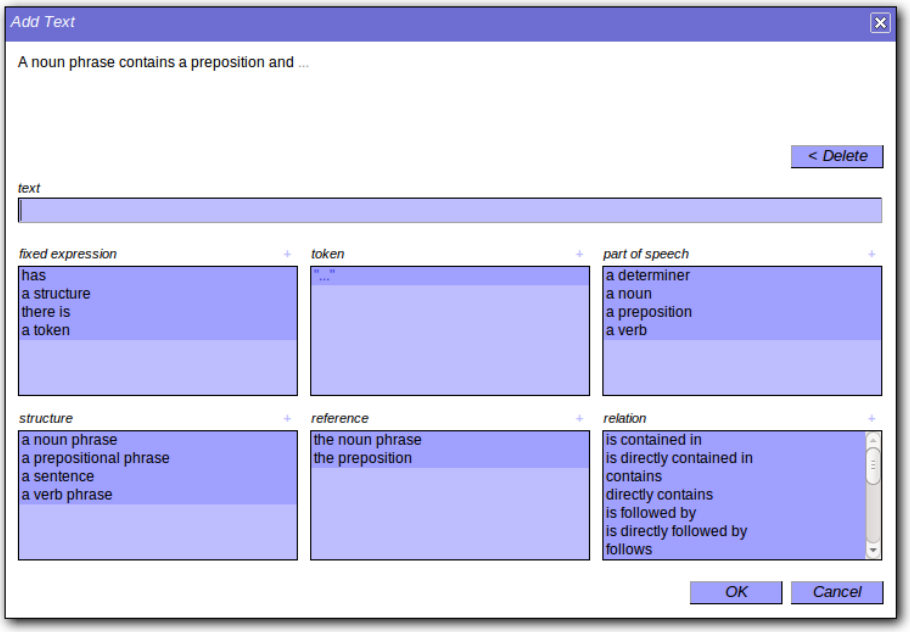

Figure 2: The predictive editor of the Coral system.

of AceWiki (Kuhn, 2009a). The source code of these systems is open[4], and the predictive editor module can be easily reused and incorporated in other systems.

Figure 2 shows a screenshot of the predictive editor in Coral. Its interface is organised as follows. At the very top, it shows the (partial) query that the users have entered so far. Underneath, it shows all ways in which the query can currently be continued. At any given point of time, users can continue the composition of their query by choosing from any of the displayed words or phrases. Of course, the availability of specific words or phrases also depends on the grammar and lexicon that have been loaded (see below).

The availability of a predictive editor thus enables users to compose query statements that comply with the restrictions of the controlled language without having to memorise these restrictions beforehand. User experiments with the AceWiki system showed that the predictive editor in question is easy to use, even without prior training (Kuhn, 2009b).

3.3 Grammar and Lexicon

The grammar describing the controlled subset of English used in Coral is written in the Codeco notation (Kuhn, 2010a). This notation is specifically designed for

[3] `http://attempto.ifi.uzh.ch/aceeditor`
[4] `https://launchpad.net/acewiki`

Type	Coral	AQL
element	*adjective phrase*	`CAT="AP"`
element	*noun phrase*	`CAT="NP"`
element	*adjective*	`POS=/ADJ.*/`
element	*proper noun*	`POS="NE"`
element	*normal noun*	`POS="NN"`
element	*noun*	`POS=/N.*/`
property	*category*	`CAT`
property	*part of speech*	`POS`
property	*lemma*	`LEMMA`
role	*the subject*	`[func="SB"]`
role	*a relative clause*	`[func="RC"]`

Table 2: Some exemplary lexicon entries of element, property and role descriptions.

controlled natural languages to be used in predictive editors. Coral's grammar consists of 51 grammar rules.

The lexicon describes the dynamic part of the language: it can easily be modified and customized for specific corpora and their tag sets. Table 2 shows some exemplary lexicon entries of element, property and role descriptions. Elements describe tokens and nodes, properties map to tag names, and roles are used for labeled edges. Since AQL does not have predefined tag names or categories, the mapping from controlled English to AQL depends on the actual tag set used by the respective corpus. For this reason, the lexicon has to be tailored towards the corpus to be used. Alternatively, more low-level expressions are possible in Coral, as shown above in example (2).

In contrast to tag names and categories, general relations like precedence and containment are predefined in AQL: these entries in Coral's lexicon do therefore not need to be adapted to the tag sets of specific corpora. However, it can still make sense to give these relations different aliases in different application areas, to allow or disallow certain synonyms, or to remove certain relations from the lexicon if they are not needed in a particular scenario. Table 1 shows the default definitions of the relations.

4 Evaluation

To test whether the approach implemented in the Coral system constitutes an improvement to existing query interfaces, we set up a user experiment. We chose ANNIS as the system to compare Coral against. ANNIS provides a graphical query builder as well as the possibility to directly write AQL code. Since both Coral and ANNIS rely on AQL, a direct comparison is possible. We will first explain the design of the experiment and then discuss the results we obtained.

4.1 User Experiment Design

The experiment focused on how easy it is to compose corpus queries in either system. The experiment was thus performed on the query interfaces only, without a corpus actually being searched: the participants were told to compose queries but they did not have the possibility to actually see any results these queries might have returned. The reason for this was to keep the design of the experiment simple, focusing on one key aspect and allowing for strict and clear evaluation. It seems natural to expect that users normally use corpus query engines in an iterative, trial-and-error-based manner, while our experiment only covers the first of such a sequence of queries. However, it is sensible to assume, we think, that the first query plays a crucial role: it must have a fair quality in order to keep the iterative process going into the right direction, and for very simple queries users would probably get frustrated if they fail to get it right on the first try. We have to leave these assumptions to future research and we concentrate here on the quality of queries written without the feedback from previous query results.

We recruited twelve participants, all with a computational linguistics[5] background (students or researchers) and a reasonably good, non-native command of English but without any special expertise in controlled natural languages. None of them had worked with ANNIS or Coral before, but eight of them had worked with systems similar to ANNIS, and three had worked with systems similar to Coral (according to the questionnaire they had to fill out after the experiment).

All participants were tested on both systems, Coral and ANNIS. In order to rule out learning effects, half of them received Coral first, while the other half started with ANNIS.

The task of the participants was to compose queries (using Coral or ANNIS) for given statements in natural language. Since these statements were not to be biased towards one of the systems, we took them from academic articles and user guides, preserving exact wording and formatting. We selected eight statements that are reasonably simple and use consistent vocabulary and divided them into three groups:

Group A:

1. Find all trees in which 'is' immediately precedes a determiner (Schulte im Walde and Zinsmeister, 2006)
2. verb *fight* followed by the noun *independence* (Rychlý, 2008)
3. Find nouns that follow a verb which is a child of a verb phrase (Bird et al., 2005)

Group B:

[5] We chose computational linguists as participants to give ANNIS a more realistic chance to outscore CORAL. As computational linguists should be familiar with formal query languages, they can be expected to produce sensible results on both systems after the short learning phases provided during the experiment.

Type	Coral	AQL
element	sentence	`CAT="S"`
element	noun phrase	`CAT="NP"`
element	verb phrase	`CAT="VP"`
element	prepositional phrase	`CAT="PP"`
element	determiner	`POS="DET"`
element	verb	`POS="V"`
element	noun	`POS="N"`
element	preposition	`POS="PREP"`
property	category	`CAT`
property	part of speech	`POS`
property	lemma	`LEMMA`

Table 3: Element and property entries of the Coral lexicon used in the experiment.

4. verb *fight* followed by any preposition (Rychlý, 2008)
5. Find noun phrases that immediately follow a verb (Bird et al., 2005)
6. Find all verb phrases that are comprised of a verb, a noun phrase, and a prepositional phrase (Bird et al., 2005)

Group C:

7. verb *fight* preceded by a noun (Rychlý, 2008)
8. locate all sentences with a preposition followed immediately by the word "the" (MacWhinney, 2010)

The Coral lexicon entries required to express these statements are shown in Table 3.

Half of the participants had to express the statements of group A in Coral and those of group B in ANNIS; the other half of the participants had to express the statements of group A in ANNIS and those of group B in Coral. The statements of group C were used as examples in the instructions.

The procedure of the experiment was as follows:

1. The participants read an instruction sheet explaining the procedure of the experiment.
2. The participants received an instruction sheet for the first system (either Coral or ANNIS), which provided them with the knowledge needed to successfully accomplish the subsequent tasks (showing only the elements and relations necessary for the tasks). The instructions used statements 7 and 8 as examples and showed how they could be expressed as queries in the respective system. Figure 3 illustrates the sample solutions that were provided for statement 8. The participants were allowed to spend 6 minutes on this step.

3. The participants then received one of the statements of group A (or B respectively) and had to compose a query for it on the given system. They were granted 2 minutes to solve the task, and they were allowed to consult the instructions. In the case of the ANNIS interface, the participants could either directly write the query code or use the graphical editor, as they preferred. This step was then repeated for the other two statements of the same group. The participants received the individual statements in different orders.

4. Steps 2 and 3 were repeated for the other system.

5. The participants filled out a questionnaire asking about their background, whether they have worked with similar systems before, and how usable they found the two systems.

We chose strict and tight time limits because the tasks were relatively simple, the participants were skilled, and perfect scores for both systems would not have allowed us to detect on which system the participants performed better. Participants were allowed to finish before the time limit was reached; we recorded the amount of time they needed for each task. Thus we were able to compare Coral and ANNIS not only with regard to the number of tasks that were solved, but also with regard to the time that was needed to complete these tasks.

4.2 User Experiment Results

The most important result is the number of tasks the participants accomplished successfully for each system. Figure 4 compares the percentages of successful tasks for each system, and Table 4 shows the concrete scores of the individual participants. With the Coral system, participants managed to solve almost twice as many tasks as with ANNIS (23 vs. 12). On average, 64% of the tasks were completed successfully with Coral but only 33% with ANNIS. In the case of ANNIS, six of the participants used the code-based way to construct all three queries, another four only used the graphical editor, and the remaining two used the code-based way for the first task and switched then to the graphical editor for the remaining two tasks. The percentage of correctly formalized queries was slightly higher with the code-based editor than with the graphical one (35% vs. 31%).

The difference of the scores (i.e. the number of correctly formalized queries) between Coral and ANNIS is statistically significant, with a p-value of 0.039 when using a Wilcoxon signed rank test (Wilcoxon, 1945). Since these scores are not single measurements but consist of three individual measurements each (i.e. Boolean measurements of whether the task was accomplished or not), we can apply more fine-grained tests on the individual tasks: applying a simple logistic regression, with the system used as the independent variable and the score (0 or 1) as the dependent one, shows a significant effect in favor of Coral (with a p-value of 0.011); the correlation between used system and resulting

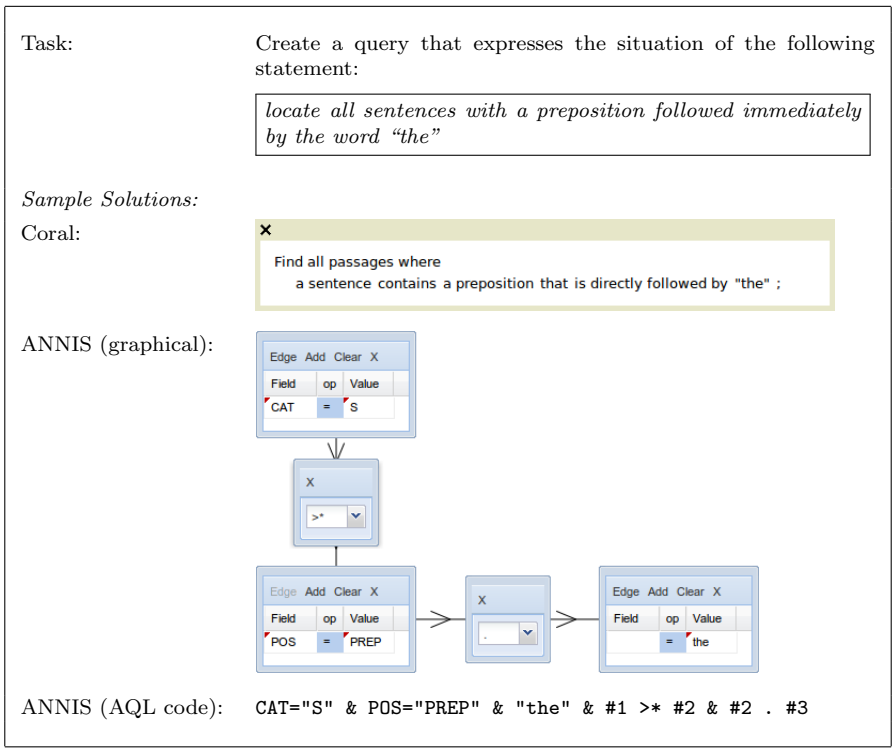

Figure 3: An exemplary task that the participants of the experiment had to accomplish.

score measured as a Pearson correlation shows a significant effect of medium strengh in favor of Coral (with a coefficient of 0.31 and a p-value of 0.0090).

Next, we can have a look at the time aspect. Figure 5 shows the total amount of time the participants spent on average on the three tasks of each system. With Coral, users required a bit more than four minutes (86 seconds per task), whereas more than five minutes were needed in the case of ANNIS (108 seconds per task). Only one participant was faster with ANNIS, while all others were faster with Coral. The difference is highly significant, with a p-value of 0.0044 when using a Wilcoxon signed rank test. The above results include the time values for all tasks regardless of whether the participant was successful or not. Restricting the attention to the successful tasks shows an even bigger difference: 70 seconds per successful task for Coral versus 102 seconds for ANNIS. The participants clearly required less time to accomplish the tasks with Coral than they needed with ANNIS.

As a further dimension, we can look at the subjective usability, i.e. at how usable the participants found the two systems. The questionnaire contained a question "How easy or hard to use did you find system X?" for each of the

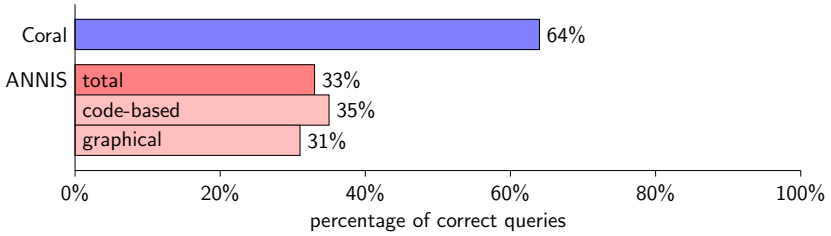

Figure 4: Average percentage of correct queries per participant and system. In the case of ANNIS, the results are shown for each of the two possibilities to construct queries (i.e. code-based vs. graphical).

Participant	Coral	ANNIS	Total
1	3	2	5
2	3	2	5
3	3	1	4
4	3	1	4
5	2	2	4
6	2	2	4
7	2	0	2
8	2	0	2
9	2	0	2
10	0	2	2
11	1	0	1
12	0	0	0
Total	23	12	35
Average	1.92	1.00	1.46

Table 4: Scores of the individual participants, sorted by their total score.

systems. They could choose between "very hard to use" (value 0), "hard to use" (1), "easy to use" (2), and "very easy to use" (3). Figure 6 shows the results. Coral got an average value of 2.33, i.e. between "easy to use" and "very easy to use". ANNIS, in contrast, was in the lower half between "hard to use" and "easy to use" with an average value of only 1.42. This difference is significant too, with a p-value of 0.027. Thus, the participant perceived Coral as being easier to use than ANNIS.

Finally, we can have a closer look at the tasks that the participants were not successful in. We were able to identify seven patterns of mistakes: (1) some solutions covered only part of what was described in the task; (2) others contained the correct entities (categories, relations, etc.) but connected them in an incorrect way; (3) some used incorrect relations (e.g. precedence instead of containment); (4) some used incorrect categories or parts of speech; (5) some

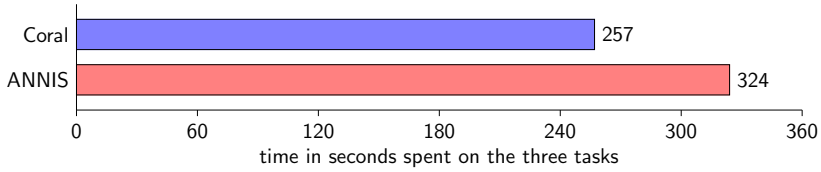

Figure 5: Average total amount of time the participants spent on accomplishing the three tasks.

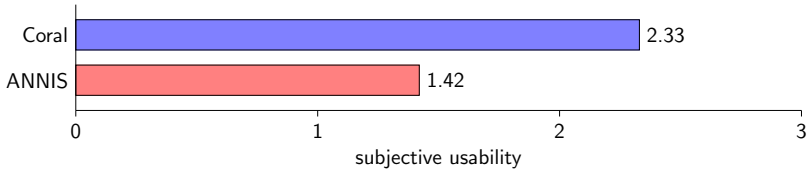

Figure 6: Degree of usability as perceived by the participants (0 means "very hard to use"; 3 means "very easy to use").

contained mistyped tokens; (6) some were empty; and finally (7) some solutions contained syntax errors (this last case was only possible when ANNIS was used in its code-based mode). Furthermore, a number of incorrect solutions could not be classified according to this scheme or contained multiple mistakes. The distribution of these error types is visualized in Figure 7.

The most apparent difference between Coral and ANNIS in terms of error types is the fact that errors due to incorrectly connected entities are frequent with ANNIS (about 11% of all tasks), while they did not occur in Coral. All participants that made this type of error chose to use the code-based way to construct the query. The graphical editor of ANNIS did not show this kind of problem. The following example should clarify why this type of error is so frequent with the code-based ANNIS interface. One participant's solution to task 5 (i.e. "Find noun phrases that immediately follow a verb") consisted of the following AQL query:

```
CAT="NP" & POS="V" & #1 . #2
```

This solution is almost correct, but it should be "`#2 . #1`" instead of "`#1 . #2`" to correctly reflect the task description. Arguably, the direction of a relation is clearer with Coral's controlled English ("*precedes*" or "*is preceded by*"?) than with AQL ("`#1 . #2`" or "`#2 . #1`"?).

Another type of problem that only occurred with the code-based interface of ANNIS were syntax errors, which could be identified as the sole cause of an incorrect solution in two cases. The fact that mistyped tokens and multiple mistakes were more frequent with ANNIS than with Coral could be an indication that some users were overwhelmed by ANNIS and did not manage to understand

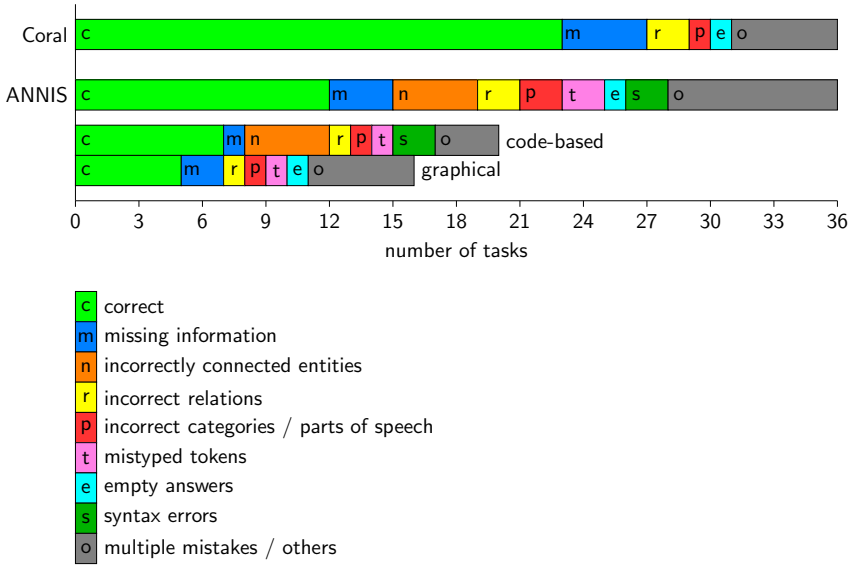

Figure 7: Number and types of errors for all of the 36 tasks that were performed on each system. For ANNIS: errors that were made with each of the two possible interfaces, i.e. the code-based and the graphical one.

the interface in the short time provided.

In summary, our results show that in the given scenario (computational linguists as users, little training, and relatively high time pressure) Coral is easier to use than ANNIS. We expect that linguists without a computer science background would exhibit an even stronger preference of Coral.

5 Related Work

A number of approaches have been presented in the past that are related to our approach.

Controlled English has been proposed for queries to the structure of software code (Würsch et al., 2010) and as a query language for ontologies (Kaufmann and Bernstein, 2007). This has been implemented in systems like GINO (Bernstein and Kaufmann, 2006) and PANTO (Wang et al., 2007). Other approaches use controlled English as a general knowledge representation language for the Semantic Web (Schwitter and Tilbrook, 2004; Kaljurand, 2007; Schwitter et al., 2008). In the area of corpus queries, however, to the best of our knowledge no research on controlled language queries has been conducted so far.

In terms of evaluation, there are some experiments that test the understandability of controlled languages (Kuhn, 2010b; Hart et al., 2008; Hallett et al., 2007; Chervak et al., 1996). They come to the conclusion that statements in

controlled English are easier to understand than other formal languages. However, this does not imply that such statements are also easier to *write* (given an appropriate editor), which is what we showed with our experiment.

Funk et al. (2007) compared the usability of an ontology editor based on controlled English and a classical ontology editor (their study is about writing declarative statements, not queries). They found a significant preference for the first system but, in contrast to what we did, they only measured the subjective usability, i.e. the participants were asked how usable they found the respective system. The subjective feeling of the participants, however, does not necessarily coincide with the actual, objective usability of the tool (they might have made mistakes without having noticed it, they might have over- or underestimated certain aspects, etc.).

6 Conclusions

Linguistically annotated text corpora will be the more useful, the more effectively researches can query them. At present, linguists interested in regularities that require complex queries are forced to learn sophisticated formal languages or complicated graphical notations if they want to access these corpora. We believe that corpus-based linguistic research would greatly profit from the availability of query methods that would combine the expressivity and power of formal query languages with greater intuitiveness and ease of use.

In this paper, we have shown that controlled natural language may prove to be a useful tool to achieve this goal. We have introduced Coral, a system in which users can express queries in a controlled subset of natural English and which then automatically translates these queries into the formal query language AQL. The evaluation we conducted showed that even relatively skilled users find it easier to compose corpus queries with Coral's predictive editor than to express them in a formal query language they were not deeply familiar with, at least on the first encounter with the system. The effects of iterative interaction with the system and of longer periods of usage experience and training remain open to be studied for the future.

Acknowledgments

We would like to thank Amir Zeldes and Florian Zipser for their help on ANNIS and AQL. We also want to thank Alexandra Bünzli, Norbert E. Fuchs, Michael Hess and Gerold Schneider for their input and feedback. Finally, we thank all participants of the experiment for their time and effort.

References

Adriaens, G. and D. Schreors: 1992, 'From COGRAM to ALCOGRAM: Toward a Controlled English Grammar Checker', in *Proceedings of the 14th*

Conference on Computational Linguistics, Vol. 2, pp. 595–601. Association for Computational Linguistics.

Bernstein, A. and E. Kaufmann: 2006, 'GINO — A Guided Input Natural Language Ontology Editor', in *The Semantic Web — ISWC 2006, Proceedings of the 5th International Semantic Web Conference* (*Lecture Notes in Computer Science* 4273), pp. 144–157. Springer.

Bird, S., Y. Chen, S. B. Davidson, H. Lee, and Y. Zheng: 2005, 'Extending XPath to support linguistic queries', in *Proceedings of the Workshop on Programming Language Technologies for XML 2005 (PLAN-X 2005)*, pp. 35–46.

Brill, E.: 1993, *A Corpus-Based Approach to Language Learning*, PhD dissertation, Philadelpha, PA, USA.

Chervak, S., C. G. Drury, and J. P. Ouellette: 1996, 'Field Evaluation of Simplified English for Aircraft Workcards', in *Proceedings of the 10th FAA/AAM Meeting on Human Factors in Aviation Maintenance and Inspection*.

Chiarcos, C., S. Dipper, M. Götze, J. Ritz, and M. Stede: 2008, 'A Flexible Framework for Integrating Annotations from Different Tools and Tagsets', in *Proceeding of the Conference on Global Interoperability for Language Resources*. Hongkong, China.

Christ, O.: 1994, 'A modular and flexible architecture for an integrated corpus query system', in *Proceedings of COMPLEX'94, 3rd Conference on Computational Lexicography and Text Research*, pp. 23–32.

Clark, P., P. Harrison, T. Jenkins, J. Thompson, and R. H. Wojcik: 2005, 'Acquiring and Using World Knowledge Using a Restricted Subset of English', in *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2005)*, pp. 506–511. AAAI Press.

Fuchs, N. E., K. Kaljurand, and T. Kuhn: 2008, 'Attempto Controlled English for Knowledge Representation', in *Reasoning Web — 4th International Summer School 2008*, pp. 104–124. Springer.

Funk, A., V. Tablan, K. Bontcheva, H. Cunningham, B. Davis, and S. Handschuh: 2007, 'CLOnE: Controlled Language for Ontology Editing', in *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007)* (*Lecture Notes in Computer Science* 4825), pp. 142–155. Springer.

Hallett, C., D. Scott, and R. Power: 2007, 'Composing Questions through Conceptual Authoring', *Computational Linguistics* **33**(1), 105–133.

Hart, G., M. Johnson, and C. Dolbear: 2008, 'Rabbit: Developing a Controlled Natural Language for Authoring Ontologies', in *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)* (*Lecture Notes in Computer Science* 5021), pp. 348–360. Springer.

Kaljurand, K.: 2007, *Attempto Controlled English as a Semantic Web Language*, PhD dissertation, Faculty of Mathematics and Computer Science, University of Tartu, Estonia.

Kaufmann, E. and A. Bernstein: 2007, 'How Useful are Natural Language Interfaces to the Semantic Web for Casual End-users?', in *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007)* (*Lecture Notes in Computer Science* 4825), pp. 281–294. Springer.

König, E., W. Lezius, and H. Voormann: 2003, 'TIGERSearch 2.1 — User's Manual'. University of Stuttgart.

Kuhn, T.: 2009a, 'AceWiki: A Natural and Expressive Semantic Wiki', in *Proceedings of the Fifth International Workshop on Semantic Web User Interaction (SWUI 2008) — Exploring HCI Challenges* (*CEUR Workshop Proceedings* 543). CEUR-WS.

Kuhn, T.: 2009b, 'How Controlled English can Improve Semantic Wikis', in *Proceedings of the Forth Semantic Wiki Workshop (SemWiki 2009)* (*CEUR Workshop Proceedings* 464). CEUR-WS.

Kuhn, T.: 2010a, 'Codeco: A Grammar Notation for Controlled Natural Language in Predictive Editors', in *Pre-Proceedings of the Second Workshop on Controlled Natural Languages (CNL 2010)* (*CEUR Workshop Proceedings* 622). CEUR-WS.

Kuhn, T.: 2010b, 'An Evaluation Framework for Controlled Natural Languages', in *Proceedings of the Workshop on Controlled Natural Language (CNL 2009)* (*Lecture Notes in Computer Science* 5972), pp. 1–20. Springer.

MacWhinney, B.: 2010, 'The CHILDES Project, Tools for Analyzing Talk, Part 2: The CLAN Programs'. Carnegie Mellon University, electronic edition.

Pool, J.: 2006, 'Can Controlled Languages Scale to the Web?', in *Proceedings of the 5th International Workshop on Controlled Language Applications (CLAW 2006).*

Power, R., R. Stevens, D. Scott, and A. Rector: 2009, 'Editing OWL through Generated CNL', in *Pre-Proceedings of the Workshop on Controlled Natural Language (CNL 2009)* (*CEUR Workshop Proceedings* 448). CEUR-WS.

Rychlý, P.: 2008, 'Building and Exploring (Web) Corpora'. `www.fi.muni.cz/~pary/emasters08-1.pdf`.

Schwitter, R., K. Kaljurand, A. Cregan, C. Dolbear, and G. Hart: 2008, 'A Comparison of three Controlled Natural Languages for OWL 1.1', in *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions* (*CEUR Workshop Proceedings* 496). CEUR-WS.

Schwitter, R., A. Ljungberg, and D. Hood: 2003, 'ECOLE — A Look-ahead Editor for a Controlled Language', in *Controlled Translation — Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Controlled Language Application Workshop (EAMT-CLAW03)*, pp. 141–150. Dublin City University, Ireland.

Schwitter, R. and M. Tilbrook: 2004, 'Controlled Natural Language meets the Semantic Web', in *Proceedings of the Australasian Language Technology Workshop 2004* (*ALTA Electronic Proceedings* 2), pp. 55–62. Australasian Language Technology Association.

Schwitter, R. and M. Tilbrook: 2006, 'Let's Talk in Description Logic via Controlled Natural Language', in *Proceedings of the Third International Workshop on Logic and Engineering of Natural Language Semantics (LENLS2006)*, pp. 193–207.

Tennant, H. R., K. M. Ross, R. M. Saenz, C. W. Thompson, and J. R. Miller: 1983, 'Menu-based Natural Language Understanding', in *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pp. 151–158. Association for Computational Linguistics.

Schulte im Walde, S. and H. Zinsmeister: 2006, 'Exercise: Searching Treebanks TIGERSearch and Tregex'. `http://www.coli.uni-saarland.de/~schulte/Teaching/ESSLLI-06/Exercises/syntax-ex.pdf`.

Wang, C., M. Xiong, Q. Zhou, and Y. Yu: 2007, 'PANTO: A Portable Natural Language Interface to Ontologies', in *The Semantic Web: Research and Applications — Proceedings of the 4th European Semantic Web Conference (ESWC 2007)* (*Lecture Notes in Computer Science* 4519), pp. 473–487. Springer.

Wilcoxon, F.: 1945, 'Individual Comparisons by Ranking Methods', *Biometrics Bulletin* **1**(6), 80–83.

Würsch, M., G. Ghezzi, G. Reif, and H. C. Gall: 2010, 'Supporting developers with natural language queries', in *ICSE '10: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pp. 165–174. ACM, New York, NY, USA.

Wyner, A., K. Angelov, G. Barzdins, D. Damljanovic, B. Davis, N. Fuchs, S. Hoefler, K. Jones, K. Kaljurand, T. Kuhn, M. Luts, J. Pool, M. Rosner, R. Schwitter, and J. Sowa: 2010, 'On Controlled Natural Languages: Properties and Prospects', in *Proceedings of the Workshop on Controlled Natural Language (CNL 2009)* (*Lecture Notes in Computer Science* 5972), pp. 281–289. Springer.

Zeldes, A., J. Ritz, A. Ldeling, and C. Chiarcos: 2009, 'ANNIS: A Search Tool for Multi-Layer Annotated Corpora', in *Proceedings of Corpus Linguistics 2009*.