Evaluating Query Languages for a Corpus Processing System

Elena Frick*, **Carsten Schnober***, **Piotr Bański***†

*Institute for the German Language (IDS)
R5 6–13, 68161 Mannheim, Germany

†Institute of English Studies
University of Warsaw, Poland

{frick | schnober | banski}@ids-mannheim.de

Abstract

This paper documents a pilot study conducted as part of the development of a new corpus processing system at the Institut für Deutsche Sprache in Mannheim and in the context of the ISO TC37 SC4/WG6 activity on the suggested work item proposal "Corpus Query Lingua Franca". We describe the first phase of our research: the initial formulation of functionality criteria for query language evaluation and the results of the application of these criteria to three representatives of corpus query languages, namely COSMAS II, Poliqarp, and ANNIS QL. In contrast to previous works on query language evaluation that compare a range of existing query languages against a small number of queries, our approach analyses only three query languages against criteria derived from a suite of 300 use cases that cover diverse aspects of linguistic research.

Keywords: Query Language, Evaluation, Corpus Analysis, KorAP

1. Introduction

This paper presents an evaluation of three corpus-query languages (QLs) against a number of criteria informed by user surveys. QLs are defined here as formal languages in which users communicate their requests to the corpus processing software. The evaluation takes place at the Institut für Deutsche Sprache (IDS) in Mannheim and is part of development of a new corpus processing platform KorAP, which is planned to succeed COSMAS II (Bodmer, 2005), in order to address the needs of linguists and lexicographers, today and in the future, and to be flexible and scalable enough to deal with increasing corpora sizes and the demands of modern corpus-linguistic practice (Bański et al., 2012). The work described in this paper is part of the project design phase and serves as a basis for the selection of a QL for the new platform. It is also part of the ISO TC37 SC4/WG6 activity on the suggested work item proposal "Corpus Query Lingua Franca" (Bański and Witt, 2011). This is a pilot study that formulates the initial criteria and that we plan to extend to other QLs.

2. Query languages under evaluation

There exists a large number of QLs designed for text corpus querying, which makes it hardly feasible to generate a comprehensive overview in a limited space. For the initial evaluation, we have chosen three QLs as representatives of different language families: the COSMAS II query language (Bodmer, 1996), the query language of Poliqarp (Przepiórkowski et al., 2004) and the ANNIS Query Language (AQL) (Rosenfeld, 2010).

2.1. COSMAS II QL

The COSMAS II QL (Bodmer, 1996) has been designed specifically for the COSMAS II system that serves as the interface to the DeReKo corpus (Kupietz et al., 2010). This QL with a flexible and powerful syntax allows the user to create from very simple to very complex queries that look for single words, words with special properties, and various combinations thereof. It provides a variety of operators for defining distances between words and allows to specify positions in relation to sentence and paragraph boundaries. Operators for case-sensitive search and search for lemma are also available.

A unique feature of the COSMAS II QL are operators that allow the user to change the scope of the match in the output, as illustrated in examples 1 and 2. Queries 1a, 1b and 1c express the same request, for any verb followed by *Sonne* at a distance of 1 to 3 tokens, but queries 1b and 1c use operators #LINKS and #ALL that allow for the results to be presented in different ways (cf. outputs in 2a, 2b, 2c).[1]

1. (a) {C} MORPH(V) /+w1:3 Sonne
   (b) {C} #LINKS(MORPH(V) /+w1:3 Sonne)
   (c) {C} #ALL(MORPH(V) /+w1:3 Sonne)
2. (a) Draußen **scheint** die helle **Sonne.**
   (b) Draußen **scheint** die helle Sonne.
   (c) Draußen **scheint die helle Sonne.**

The COSMAS II QL also implements Boolean operators and allows wildcards, though restricted to the surface text. Apart from full text and annotation search, search in metadata is supported as well.

2.2. Poliqarp QL

Poliqarp (Przepiórkowski et al., 2004) is a corpus indexing and search tool initially developed for the IPI PAN corpus (`http://korpus.pl`) that has later been extended and is now used for the National Corpus of Polish (`http://nkjp.pl`) and distributed from SourceForge. The syntax of Poliqarp's QL is based on that of the CQP (Corpus Query Processor) developed for the IMS Open Corpus Workbench (Christ, 1994).

A search query formulated in this QL is a sequence of terms, each of which can be an attribute-value pair or just a string corresponding to a word or a part of it. It is possible

[1] We use {A}, {C} and {P} to mark the query strings of, respectively, AQL, COSMAS II, and Poliqarp.

for the user to use regular expressions, to specify the distance between the matches and to constrain them to special structures such as phrases and sentences. Some characteristic features of the Poliqarp QL are the ability to handle ambiguous morphosyntactic interpretations and to search for syntactic and semantic heads of phrases. Furthermore, it provides the possibility to use variables and the 'match any token' operator '[ ]', as shown in queries 3 and 4. Query 3 uses a variable for searching for all duplicates occurring in a text, and query 4 looks for a token that follows the sequence *Ich freue mich*.

3. {P} [orth=$1] [orth=$1]

4. {P} Ich freue mich [ ]

2.3. ANNIS QL

AQL (Rosenfeld, 2010), as a representative of tree-based languages, is especially useful for searching hierarchical structures. The '>' operator can be used to find a node dominating another, specifically labeled, node. Properties such as 'the order of' and 'the distance between' can serve as search criteria. By means of the pointing relation operator '->', it is possible to search for direct or indirect relationships between two nodes, such as verb-subject dependencies or coreference and anaphoric relations. Example 5 illustrates this with a query that searches for a finite verb and a noun that is linked to it by a dependency relation that indicates subjecthood.

5. {A} pos="VVFIN" & pos="NN" & #2 -> dep[label="SUBJ"]#1

One special feature of AQL is the possibility to specify namespaces for tags, as shown in query 6, which searches for attributive adjectives in TreeTagger (Schmid, 1994) annotations. This is useful when identical tags coming from different annotation sources need to be distinguished, as in the case of the KorAP engine.

6. {A} tt:pos="ADJA"

Like Poliqarp, AQL makes it possible to search the text, all annotations, and metadata using regular expressions.

3. Methodology

This section presents our approach to the evaluation, provides an overview of the testing environment and the testing process, and places our contribution in the context of the previous work reported in the literature.

3.1. Evaluation criteria

According to (Jarke and Vassiliou, 1985), the evaluation of QLs can be divided into two main categories: functionality and usability. The latter is determined by subjective aspects such as intuitiveness and learnability, which depend to a large extent on the user's previous knowledge. Evaluating these subjective criteria requires methods that differ structurally from those developed for the functionality-related criteria. The study documented here focused on the functional aspect of the evaluation in order to establish the core properties of the QL that is going to be adopted for the KorAP platform.

In order to establish the criteria for the evaluation, we have created a use case analysis expressing various potential user requests in natural language. Having consulted grammarians and lexicographers at IDS Mannheim, we have been able to compile an inventory of requests that users in our target group may wish to express, starting from querying single words and phrases and scaling up to complex linguistic phenomena and nested structures. At this point, it has to be noted that our use cases include a certain bias: we are looking for a QL that handles multiple, potentially conflicting annotations, and, in particular, concurrent tokenizations, cf. (Bański et al., 2012) for a more detailed description of the aims of the project.

From these use cases, a number of criteria have emerged for a functionality-based evaluation of QLs. These criteria can be categorised into eight types that are further discussed below: plain-text search (4.1), annotation-based search (4.2), constraints on the size and position of matches (4.3), Boolean operations (4.4), universal quantification and implication (4.5), fuzzy and pattern search (4.6), metadata access (4.7), and display directives (4.8).

3.2. Testing process

From the collection of 300 abstract sample queries, actual queries have been generated in each of the evaluated QLs. Each of these has been tested in a corresponding test system, i.e. a Poliqarp (version 1.3.11) (Janus and Przepiórkowski, 2007) and an ANNIS2 (version 2.2.0) (Zeldes et al., 2009) installation. For the COSMAS II queries, we have used the freely available web interface (version 1.7).

The COSMAS II system accompanies the DeReKo corpus (Kupietz et al., 2010). In order to make the testing conditions maximally comparable, we have concentrated on a snapshot of the German Wikipedia, included into DeReKo in 2005 and consisting of about 200,000 documents with ca. 50,000,000 tokens, and converted that resource into the format accepted by Poliqarp, and part of it into the format native to ANNIS. This test corpus has been annotated with three different annotation tools, each outputting different types of annotations and labels: TreeTagger, Connexor Machinese Phrase Tagger[2], and Xerox Incremental Parser (XIP)[3].

3.3. Previous research

Studies on the functionality of QLs can be generally classified into two groups: the first group investigates linguistic phenomena in certain types of corpora and shows what QL features are required to search these phenomena. For example, Mírovskỳ (2008), on the basis of the Prague Dependency Treebank, created a list of functional features that a QL must have in order to effectively query this corpus. In our study, we also created a list of functional requirements for a QL. It overlaps partly with that of Mírovskỳ (2008), but concerns some additional issues that are not covered yet

[2] http://www.connexor.com/nlplib/?q=mpt

[3] http://open.xerox.com/Services/XIPParser

Feature	COSMAS II	Poliqarp	AQL
Single-word search	Yes	Yes	Yes
Multi-word sequences	Yes	Yes	Yes
Word combinations	Yes	Yes	Yes
Search by partial matching	Yes (wildcards)	Yes (RegEx)	Yes (RegEx)

Table 1: Plain-text search

in related work, like "metadata access" or "display directives". Our focus does not lie on querying a specific corpus or annotation format, but on covering all possible facets of linguistically motivated searches in order to find a QL that satisfies the end users' requirements.

The second group of research endeavours, to which our survey is complementary, evaluates the functionality of QLs against concrete query examples. In this approach, some authors focus on a single QL and demonstrate its expressive power, as Cassidy (2002) does for XQuery, Bouma (2010) for Prolog, and Kepser (2003) for fsq (Finite Structure Query). Others compare several QLs for their functionality (cf. (Lai and Bird, 2004; Cassidy et al., 2000)). In either case, a very small number of query examples is taken into account for the evaluation of QLs and those approaches focus mainly on special selected issues, such as hierarchical and sequential navigation (cf. (Lai and Bird, 2004)) or querying spoken dialogs (cf. (Cassidy et al., 2000)). The challenge of our approach lies in the analysis of a relatively small number of QLs but on the basis of a very detailed suite of 300 query examples that cover diverse aspects of linguistic research.

4. Results

This section presents results of the application of the evaluation criteria, listed according to the categories enumerated in section 3.1, and illustrated with a series of tables.[4]

4.1. Plain-text search

The criteria in this category (cf. Table 1) evaluate the QLs' capabilities for searching through surface text only. They investigate simple search for single words, word sequences and combinations, as well as search for words by partial matching. All the evaluated QLs fulfill these criteria except for COSMAS II that shows drastic limitations in searching for words by partial matching due to its lack of regular expressions support. For example, a query for words that start with any vowel has to be expressed by searching for words beginning with each vowel separately and the unification of the results with the conjunction OR (=ODER) (cf. query 7). In Poliqarp and AQL, regular expressions make it possible to formulate the same request more elegantly (cf. 8).

7. {C} A* ODER a* ODER (...) ODER Y* ODER y*

8. {P} "[AaEeUuIiOoÜüÖöÄäYy].*"

[4]Various categorisations of the criteria are possible here, depending on one's focus; we have chosen what we believe to be a reader-friendly ordering.

4.2. Annotation-based search

Corpora often contain various types of annotations apart from the surface text. These can include simple 1:1-relations, i.e. one tag for each token, or more complicated structures such as syntactic hierarchies or dependency graphs. Searching such structures requires a QL explicitly providing structure-related queries. This is the case for AQL, while COSMAS II and Poliqarp provide partial solutions in this regard (cf. Tables 2 and 3).

Feature	COSMAS II	Poliqarp	AQL
Search in annotations	Yes	Yes	Yes
Search across annotation layers	Yes	Yes	Yes
Search by annotation source	No	No	Yes

Table 2: Annotation-based search

Both COSMAS II and Poliqarp support search in annotation content and across different annotation layers, as shown in queries 9 and 10. Query 9 looks for Named Entities, whereas query 10 looks for the word *Lehrer* followed by a finite verb, within a sentence. However, both COSMAS II and Poliqarp lack the possibility to 'namespace' annotation content in the way that AQL does (cf. query 6 above).

9. {C} MORPH(NE)

10. {P} [orth=Lehrer] [pos=VVFIN] within s

Moreover, in contrast to AQL (cf. query 5), COSMAS II and Poliqarp do not support directed relations pointing from one node to another (cf. Table 3).

Feature	COSMAS II	Poliqarp	AQL
Directed relations	No	No	Yes
Hierarchical relations	Yes	Only subordination	Yes
Coverage relations	Yes	No	Yes

Table 3: Relationship between spans

Regarding structural relations such as dominance (searching for A containing B) and its reverse, subordination (searching for A within B), Poliqarp provides the 'within' operator that constrains matches to a dominant structure (cf.

query 11) and does not allow queries such as “find a noun phrase containing an adjective”.

11. {P} [pos=ADJA] within np

Both COSMAS II and AQL provide a variety of operators for specifying different types of coverage relations between spans (inclusion, full coverage, right and left alignment, right and left overlap). But AQL has an advantage because it also provides operators for specifying the type of relationship. This is illustrated by queries 12 and 13. The former looks for a sentence node directly dominating a noun phrase. The latter additionally includes sentence nodes that dominate a noun phrase indirectly.

12. {A} cat="S" & cat="NP" & #1 > #2

13. {A} cat="S" & cat="NP" & #1 >* #2

4.3. Constraints on the size and position of matches

This category (cf. Table 4) includes four criteria: they deal with specifying (i) the word order, (ii) distances between segments, (iii) positions of segments within a structure, and (iv) the extent of sentences and phrases. Not all of the evaluated QLs satisfy these criteria fully. For example, regarding the word order, COSMAS II is limited to express indirect precedence and does not allow queries such as “find an auxiliary verb followed by a main verb in the infinitive, potentially with one or more words in between”. The distance between matches always has to be specified (cf. 14). Poliqarp offers a more comfortable way for expressing such queries, by using the ‘match any token’ operator ‘[ ]’ in combination with the Kleene star (cf. 15). AQL also features a simple way for specifying the word order by using the direct and indirect precedence operators ‘.’ and ‘.*’ (cf. 16).

14. {C} MORPH(VRB fin a) /+w10 MORPH(VRB inf v)

15. {P} [pos=VAFIN] [ ]* [pos=VVINF]

16. {A} pos="VAFIN" & pos="VVINF" & #1 .* #2

The three QLs assume different strategies in queries intended to find items occurring in any order. COSMAS II disregards precedence information: the search in (cf. 17) matches the three target words occurring in any sequence. Poliqarp appears to adopt almost a mirror strategy: it implicitly assumes that the sequence of terms in the query corresponds to the surface sequence of the matches. This is why a search for three words occurring in an arbitrary sequence has to mention all orderings (query 18). AQL separates the information concerning the targets and their ordering, and expects the precedence information to be stated separately, as in (cf. 19). In this respect, AQL provides the user with the greatest amount of flexibility.

17. {C} Zukunft UND Gegenwart UND Vergangenheit

18. {P} Zukunft [ ]* Gegenwart [ ]* Vergangenheit | Zukunft [ ]* Vergangenheit [ ]* Gegenwart | Gegenwart [ ]* Vergangenheit [ ]* Zukunft | Gegenwart [ ]* Zukunft [ ]* Vergangenheit | Vergangenheit [ ]* Zukunft [ ]* Gegenwart | Vergangenheit [ ]* Gegenwart [ ]* Zukunft

19. {A} "Zukunft" & "Gegenwart" & "Vergangenheit" & (#1 .* #2 & #2 .* #3 | #1 .* #3 & #3 .* #2 | #2 .* #1 & #1 .* #3 | #2 .* #3 & #3 .* #1 | #3 .* #2 & #2 .* #1 | #3 .* #1 & #1 .* #2)

Defining distances is not as trivial as one might expect, either. Poliqarp’s is the only evaluated QL that is able to specify a minimum distance between terms. For this purpose, regular expressions are applicable, as shown in query 20, which searches for a sequence of an article and a noun with at least one word between them.

20. {P} [pos=ART] [ ]+ [pos=NN]

Specifying various positions of tokens within a structure is supported extensively by COSMAS II. The following examples illustrate queries looking for the word *heute* at the beginning (cf. 21a and 21b), at the end (cf. 22a and 22b), and in the third position in a sentence (cf. 23), while the query in example 24 matches heute at the beginning of a paragraph. Query 25 matches *heute* at the end of a sentence as well as before a colon, while example 26 expresses a query that looks for *heute*, but not at the beginning of a sentence and not after a colon.

21. (a) {C} <sa> /w0 heute
    (b) {C} heute #IN(L) <s>

22. (a) {C} heute /w0 <se>
    (b) {C} heute #IN(R) <s>

23. {C} <sa> /+w2 heute

24. {C} heute:pa

25. {C} heute:se

26. {C} heute:-se

AQL allows a search for an element at the beginning or at the end of a containing structure by using leftmost (cf. 27) and rightmost (cf. 28) child operators, as well as operators for the left (cf. 29) and right (cf. 30) alignments. It is also possible to search for an element at a specific distance from the beginning of the structure (cf. 31). The search for elements not adjacent to an edge of a containing structure is not supported.

27. {A} cat="NP" & "der" & #1 >@l #2

28. {A} cat="S" & cat="NP" & #1 >@r #2

29. {A} cat="S" & "Ein" & #1_l_#2

30. {A} cat="S" & pos="VVPP" & #1_r_#2

31. {A} cat="S" & node & pos="NN" & #1_l_#2 & #2.#3

The last evaluation criterion in this category deals with the possibility to define the extent of a structure by specifying the number of segments (e.g. tokens, nodes) covered by this structure. It is supported by AQL and COSMAS II, but with some restrictions. COSMAS II allows for the definition of sentence length by specifying the number of words, but the minimal length can be specified only if the maximum is specified as well (cf. 32).

Feature	COSMAS II	Poliqarp	AQL
Define word order	Yes, but specifying the distance required	Yes, implicitly	Yes, explicitly
Define distances	Min. only in combination with max.	Yes	Min. only in combination with max.
Define positions	Yes	No	Yes, but restricted
Define the extent of sentences or phrases	Min. only in combination with max.	No	By tokens or directly dominated nodes; min. only in combination with max.

Table 4: Constraints on the size and position of matches

32. {C} #BEG(<s>) /5:50w,s0 #END(<s>)

In AQL, the extent of a structure can be specified in two ways: either by defining the amount of directly dominated children that this structure has or by defining the token span covered by it. Example 33 searches for a sentence containing exactly five tokens while query 34 looks for an "S" node that dominates maximally five others. It is not possible to specify only the minimal number of tokens or nodes.

33. {A} cat="S" & #1:tokenarity=5

34. {A} cat="S" & #1:arity=1,5

4.4. Boolean operations

This section reports evaluation results concerning the use of Boolean operations: conjunction, disjunction and negation. While the former two of those operations are fully supported by all three QLs, negation is not always expressible, depending on its scope (cf. Table 5).
In Poliqarp and AQL, it is a trivial task to negate features of a token, e.g. 'pos != ART'. This can be combined with any other condition, as in examples 35 and 36 that look for the word der whose part of speech is not 'ART'. However, negation applied to higher-order structures, for instance syntactic trees, is more complex. Neither Poliqarp nor AQL allow for negating specific segments as in "find word A but only if word B does not occur in the same text, sentence or phrase". COSMAS II, on the other hand, does not always allow negation at the level of features of tokens, as e.g. in "find a token with a part-of-speech specification other than 'article'" – this needs to be expressed implicitly by allowing for all possible values except 'ART'. In comparison, the search for der that is not an article is completely unproblematic (cf. 37).

35. {A} tok = "der" & pos != "ART" & #1_=_#2

36. {P} [orth = der & pos != ART]

37. {C} der %w0 MORPH(ART)

Search in structures and hierarchies is what tree-based languages, in this paper represented by AQL, are designed for, and therefore perform best. Nevertheless, AQL does not provide the possibility to negate the dominance relation, e.g. the query "find all noun phrases that do not dominate any adjectives" is not feasible in AQL. The same is true for negating subordination relations, e.g. "search for an adjective that is not dominated by a noun phrase". This also holds for Poliqarp that does not provide operators to negate dominance and subordination.
Concerning negation of span relations, COSMAS II differs from Poliqarp and AQL in that it offers the negation operator '%' for searching for non-overlapping spans, and makes it possible to negate dominance, e.g. to specify elements that are disallowed within a sentence (cf. 38).

38. {C} <s> %w0 &haben

4.5. Universal quantification and implication

Our use case collection contains requests such as "find a sentence containing verbs in the first person singular only" or "find a noun phrase containing more than one noun, but all nouns must be in singular". In order to recast them as QL statements, operators for universal quantification and strict implication are required (cf. (Marek et al., 2008)). None of the evaluated corpus processing systems supports this functionality. As has been noted in the literature, implementing such operators is not an easy task and may have serious consequences for system performance. Introducing such functionality is a tempting challenge that we are considering for KorAP, following the example set by the creators of the Stockholm Tree Aligner (Volk et al., 2007).

4.6. Fuzzy and pattern search

Regular expressions provide a lot of expressive power, which is why the lack of support for them on the part of COSMAS II (cf. Table 7) is a disadvantage that makes it impossible to reflect some of the user needs (cf. section 4.1). AQL supports regular expressions at the level of the textual content of individual tokens (cf. 39), annotations (cf. 40), and metadata. Poliqarp's QL is able to apply regular expressions also to higher-level constructs for the purpose of specifying the order of tokens and thus the content of syntactic constituents. A search for a sequence beginning with a determiner and ending with a noun, with any number of adjectives between them, can be specified in Poliqarp as shown in 41.

39. {A} /.*platz/ (or: tok=/.*platz/)

40. {A} pos=/VV.*/

41. {P} [pos=DET] [pos=ADJ]+ [pos=NN]

For cases where only a fragment of a query should receive a case-insensitive match, the operators '/i' and '$' can be used in Poliqarp and COSMAS II, respectively (cf. 42 and

Feature	COSMAS II	Poliqarp	AQL
Negation within a segment on its features	Yes, but only in combinations with multiple features (query 37)	Yes	Yes
Negation of segments (search for A but not B)	Yes	No	No
Negation on dominance	Yes	No	No
Negation on subordination	No	No	No
Non-overlap	Yes	No	No

Table 5: Negation

43). AQL does not provide dedicated case-related operators, but can handle such requests through regular expression sets (cf. 44).

42. {P} Institut für deutsche/i Sprache

43. {C} Institut für $deutsche Sprache

44. {A} "Institut" & "für" & /[Dd]eutsche/ & "Sprache" & #1 . #2 & #2 . #3 & #3 . #4

In contrast to COSMAS II, Poliqarp and AQL allow for the use of variables. This opens up the possibility to express e.g. "agreement constraints" or "multiple node relations with respect to one node" (König at all. 2003). The Poliqarp query in example 45 illustrates this by searching for a word repetitively in different inflectional forms within a sentence. The AQL query in 46 uses variables to find a noun and a verb phrase within the same sentence.

45. {P} [base=$1 & orth=$2] [ ]* [base=$1 & orth=$3] within s

46. {A} cat="S" & pos="NN" & cat="VP" & #1>#2 & #1>#3

The feature that distinguishes Poliqarp from both the other evaluated QLs is its support for handling ambiguous morphosyntactic interpretations within a single annotation layer. Allowing the user to choose between different equality operators ('=', '==', '~', '~~') is a serious advantage for queries such as "find a noun where at least one case interpretation given by a morphological analyser (before disambiguation) is nominative" (cf. 47) or "find a segment where all its interpretations in the given context (disambiguated interpretations) are relative pronouns" (cf. 48).

47. {P} [pos=NN & case~nom]

48. {P} [pos=="PREL.*"]

We have called the last criterion in Table 6 'search by alternatives'. This refers to the ability of a QL to automatically expand a query e.g. by including words that are spelled or sound similar to the given term (search by similarity) or by searching a term and all its synonyms, hypernyms, or hyponyms (search by thesaurus). Support for such functionality is a useful feature in a QL, especially for corpora that contain raw material from a variety of sources. 'Search by alternatives' makes it possible to satisfy requests such as "find the word *Galerie*, including mistypes such as *Gallerie*" or "find verbs occurring as predicates for *Hund* or its hyponyms like *Schäferhund, Collie, Dackel*, etc". This functionality is partially present in other QLs such as CATMA[5], but none of the evaluated QLs supports it.

4.7. Metadata access

Both Poliqarp and AQL are capable of using document metadata to constrain matches (cf. Table 7). If the keyword 'meta' is used at the end of a query, followed by a specification of metadata attributes and values, Poliqarp restricts the scope of that query to e.g. texts by a certain author and published in a certain range of years (cf. 49).

49. {P} [orth=Frau] meta author=Goethe & created>=1800 & created<=1825

COSMAS II does not allow for joint searches across annotations and metadata. However, it provides the '#IN' operator to query metadata: example 50 illustrates how to find text titles ('<üh>' for German 'Hauptüberschrift') containing the word *heute*.[6]

50. {C} heute #IN <üh>

Besides "formal" metadata describing the title, author, genre of a text, etc., it may sometimes be useful to access information about the statistical properties of elements (sentences, tokens, word forms, pos, etc.) occurring in a document from the level of the QL. Naturally, this requires a number of preconditions to be fulfilled, concerning the nature of the queried resource (static or dynamic), the frequency of (re)indexing, etc. It is worth noting that some QLs, notably CATMA, support such functionality (the query **freq=1** selects any word occurring only once in a text (Schüch, 2010)).
Among the three evaluated QLs only Poliqarp claims to partially support this functionality by allowing the user to specify the minimum frequency threshold using the keyword 'min<cmin>' so that "only results which occurred at least <cmin> times in the matches should be displayed"

[5]CATMA (`http://www.catma.de/`) allows to search for orthographically similar words and even to adjust the degree of similarity. This can be illustrated by the query `simil="Haus" 90%` that selects any word which is, according to some pre-defined criteria, 90% similar to the word *Haus* (Schüch, 2010).

[6]The letter 'ü' is straightforwardly obtained from German keyboards. However, it has to be noted that in the interest of full accessibility, the QL projected for KorAP will use a restricted set of characters.

Feature	COSMAS II	Poliqarp	AQL
RegEx	No, but wildcards	Yes	Yes
Case-sensitivity	Yes	Yes	With RegEx
Ambiguities within a single annotation layer	No	Yes	No
Variables	No	Variables for feature values	Node variables
Search by alternatives	No	No	No

Table 6: Fuzzy and pattern search

Feature	COSMAS II	Poliqarp	AQL
Query metadata only	Yes	No	No
Constrain search by metadata	No	Yes	Yes
Search by statistical properties	No	Min. frequency threshold	No

Table 7: Metadata access

(Buczyński, 2007). Given the limitations of our data set, we were not able to test this in practice.

4.8. Display directives

The criteria in Table 8 concern the output of search results. From a general point of view, it is disputable whether a QL should include directives for formatting the results, or whether that task should be delegated to the user interface. Nevertheless, it is useful at least to mark the information foci for the interface to act on, and this is what Poliqarp does by means of a dedicated marker that allows to align matched segments in columns. Query 51 searches for attributive adjectives that occur after a sequence comprising a preposition and the word *immer*. The '^'-operator before [pos="ADJA"] results in the output being split into four columns containing the left context, the left part of the match (the preposition and "immer"), the right part of the match (the adjectives) and the right context.

51. {P} [pos=APPR] immer ^[pos=ADJA]+

Furthermore, the syntax of Poliqarp allows the user to specify how many hits should be displayed (cf. 52) and to define the parameters for grouping and sorting them (cf. 53).

52. {P} [pos=ADJA & orth="a.*"] count 100

53. {P} [orth=Tag] group by case

As already mentioned in section 2.1, the COSMAS II QL is able to highlight user-defined parts of the match (cf. examples 1 and 2).

5. Summary

The present paper relates the results of a pilot study that we have conducted in order to establish the features that we would like to see in the QL adopted for the KorAP platform. The study is far from concluded, but we believe that the results achieved so far are worth sharing with the community and may generate useful discussion.

We see the main outcome of this work as twofold: firstly, the evaluation criteria that we have generated, based on user requirements, provide a generic measure for a functionality-based comparison of QLs that identifies their weaknesses and strengths. The second result is the actual evaluation of three QLs representing different language families, by applying the aforementioned criteria.

Given that the three QLs come from different traditions and have been created with different data models in mind, it is natural to see that they differ in their capabilities. Nevertheless, we have attempted to highlight limitations that are common to many usage scenarios. They include restricted support for negation (cf. 4.4), the lack of universal quantifiers and the impossibility to express implication (cf. 4.5). Additionally, specifying distances between tokens is, at the moment, not sufficiently robust and operators for specifying the extent of structures are not flexible enough (cf. 4.3). Desirable functionalities such as e.g. variables (cf. 4.6) and search by statistical properties (cf. 4.7) are not, or are only partially implemented, allowing for some very specific queries but not covering all searching aspects summarized under the appropriate criterion. Search by alternatives (cf. 4.6) is not provided by any of the evaluated QLs, which is due to the fact that this is not implemented in the respective systems.

Having identified these limitations allows to us to conclude that neither of the evaluated QLs satisfies our requirements for a new corpus analysis platform. This means that we will not re-use either of the evaluated QLs per se, but instead design a new QL that is based on existing ones, at least function-wise. The QL closest to our needs, especially given the data model assumed for KorAP, is AQL2, and the focus in the next phase of our research will be on how feasible it may be to adopt the AQL (or at least its most essential features, possibly enhanced by desirable features of other QLs), in view of the limited scalability of the system that underlies ANNIS2[7].

[7] It is important to stress that ANNIS2 and AQL are under active development, aimed, among others, at addressing its systemic shortcomings and extending the query language (Victor Roselfeld and Amir Zeldes, p.c.)

Feature	COSMAS II	Poliqarp	AQL
Highlight user-defined parts of matches	Yes	No	No
User-defined match alignment	No	Yes	No
Define the context size	Only in UI	Only in UI	Only in UI
Define the number of displayed hits	Only in UI	Yes	Only in UI
User-defined sorting and grouping hits	Only in UI	Yes	No

Table 8: Display directives

6. Future work

In our subsequent work, we intend to focus on AQL and from this position, to extend the range of the evaluated QLs and to extend the area of the evaluation.

Two important QL features have not yet been investigated: usability and performance. For the former aspect, quantitative studies that measure features such as intuitiveness and learnability on trial users would be helpful for an improved QL evaluation. At a later stage, empirical experiments such as those suggested by (Kaufmann and Bernstein, 2010) and (Graaumans, 2004), will be considered in order to fine-tune for usability.

For performance evaluation, comparable measures need to be implemented; this is an involved task because the mere time-consumption of a query depends on many aspects where the actual implementation may be more significant than the language itself. However, performance is important for KorAP as it is designed to process corpora with more than 50 billion tokens. Minimally, a computational cost estimation will be necessary, possibly by comparing the computational complexity of queries (cf. (Christ and Schulze, 1995)).

7. Acknowledgements

The KorAP project, in the context of which the present study is conducted, is funded within the Senate Committee Competition (SAW) programme of the Leibniz Association.

We wish to thank the anonymous LREC reviewers – we have attempted to address all their remarks. We are also grateful to Marc Kupietz and Andreas Witt, for their advice and helpful comments, Florian Zipser for his assistance in testing the ANNIS platform, and Amir Zeldes for prompt reactions to our queries as well as important remarks on an earlier version of this paper. Finally, we would like to thank Helge Krause and Franck Bodmer for their assistance with COSMAS II queries.

8. References

Bański, P. & Witt, A. (2011). Do linguists need a corpus query lingua franca? Presentation given at the ISO TC37 meeting, June.

Bański, P., Fischer, P. M., Frick, E., Ketzan, E., Kupietz, M., Schnober, C., Schonefeld, O. & Witt, A. (2012). The new IDS corpus analysis platform: Challenges and prospects. In *Proceedings of LREC-2012*, Istanbul, May.

Bodmer, F. (1996). Aspekte der Abfragekomponononente von COSMAS-II. In Jüttner, I. & Neumann, R. (Eds.), *LDV-INFO 8. Informationsschrift der Arbeitsstelle Linguistische Datenverarbeitung*, pages 112–122. IDS.

Bodmer, F. (2005). COSMAS II. Recherchieren in den Korpora des IDS. *Sprachreport*, 21(3):2–5.

Bouma, G. (2010). Querying linguistic corpora with Prolog. *Semantic Approaches in Natural Language Processing*.

Buczyński, A. (2007). Statistical extension to the poliqarp search engine. In *Twelfth ESSLLI Student Session*, pages 47–54, Dublin.

Cassidy, S., Welby, P., McGory, J. & Beckman, M. (2000). Testing the adequacy of query languages against annotated spoken dialog. In *Proceedings of the Speech Science and Technology Conference*, pages 428–433.

Cassidy, S. (2002). XQuery as an annotation query language: a use case analysis. In *LREC-2002, Gran Canaria*.

Christ, O. & Schulze, B. (1995). Ein flexibles und modulares Anfragesystem für Textcorpora. *Tagungsberichte des Arbeitstreffens Lexikon+ Text*.

Christ, O. (1994). A modular and flexible architecture for an integrated corpus query system. In *Proceedings of the 3rd Conference on Computational Lexicography and Text Research. Budapest, Hungary*.

Graaumans, J. (2004). A qualitative study to the usability of three XML query languages. In *Proceedings of the conference on Dutch directions in HCI*, page 6. ACM.

Janus, D. & Przepiórkowski, A. (2007). Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 85–88. ACL.

Jarke, M. & Vassiliou, Y. (1985). A framework for choosing a database query language. *ACM Computing Surveys (CSUR)*, 17(3):313–340.

Kaufmann, E. & Bernstein, A. (2010). Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):377–393.

Kepser, S. (2003). Finite structure query: A tool for querying syntactically annotated corpora. In *Proceedings of the tenth EACL conference*, volume 1, pages 179–186. ACL.

Kupietz, M., Belica, C., Keibel, H. & Witt, A. (2010). The german reference corpus DeReKo: A primordial sample for linguistic research. In *LREC-2010. Malta*.

Lai, C. & Bird, S. (2004). Querying and updating treebanks: A critical survey and requirements analysis. In

Proceedings of the Australasian Language Technology Workshop, pages 139–146.

Marek, T., Lundborg, J. & Volk, M. (2008). Extending the TIGER query language with universal quantification. In *Proceeding of KONVENS*, pages 3–14.

Mírovskỳ, J. (2008). PDT 2.0 requirements on a query language. *Proc. 46th ACL*, pages 37–45.

Przepiórkowski, A., Krynicki, Z., Dębowski, Ł., Woliński, M., Janus, D. & Bański, P. (2004). A search tool for corpora with positional tagsets and ambiguities. In *Proceedings of LREC-2004*, pages 1235–1238.

Rosenfeld, V. (2010). An implementation of the Annis 2 query language. Technical report, Humboldt-Universität zu Berlin.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of international conference on new methods in language processing*, volume 12, pages 44–49. Manchester, UK.

Schüch, L. (2010). *User Manual for CATMA 3.0.*

Volk, M., Lundborg, J. & Mettler, M. (2007). A search tool for parallel treebanks. In *Proceedings of the Linguistic Annotation Workshop*, pages 85–92, Prague, Czech Republic, June. Association for Computational Linguistics.

Zeldes, A., Ritz, J., Lüdeling, A. & Chiarcos, C. (2009). Annis: A search tool for multi-layer annotated corpora. In *Proceedings of Corpus Linguistics*, pages 20–23.