Fast syntactic searching in very large corpora for many languages

Miloš Jakubíček[‡†], Adam Kilgarriff[†], Diana McCarthy[†] , Pavel Rychlý[‡†]
{xjakub, pary}@fi.muni.cz
adam@lexmasterclass.com, diana@dianamccarthy.co.uk

[‡]Masaryk University, Brno, Czech Republic
[†]Lexical Computing, Ltd., Brighton, United Kingdom

Abstract. For many linguistic investigations, the first step is to find examples. In the 21st century, they should all be found, not invented. Thus linguists need flexible tools for finding even quite rare phenomena. To support linguists well, they need to be fast even where corpora are very large and queries are complex. We present extensions to the CQL 'Corpus Query Language' for intuitive creation of syntactically rich queries, and demonstrate that they can be computed quickly within our tool even on multi-billion word corpora.

Keywords: corpus search, large corpora, CQL, syntactic search

1 Introduction

It is important for much linguistic research to find datasets relating to hunches and hypotheses. Linguists and system developers would like to be able to find large numbers of examples quickly and easily.

Our tool computes the Corpus Query Language (CQL) (Christ and Schulze, 1994) queries rapidly on large corpora using the Manatee corpus manager system as its backend which is based on stream processing techniques and has been described in (Rychlý, 2000). We have large (more than 100 million-word) corpora loaded into the tool and accessible over the web for 25 languages and the number rises month by month (Reddy et al., 2010). For English, Italian and German we have very large (more than 1.5 billion-word) corpora, and for English, also a corpus of 5.5 billion words (Pomikálek et al., 2009).

We have recently augmented the CQL formalism with extensions to the 'within' operator and a new 'containing' operator, to better support queries relating to phrase structure. In this paper we first summarise standard CQL, then describe the extensions to the 'within' operator, then introduce the 'containing' operator, and finally indicate computation times for sample queries.

For the same purpose, the 'meet' and 'union' operators have been extended so that they can be used at any token position in the query as we describe in this paper.

For the bulk of the paper we assume a corpus which is lemmatised and part-of-speech tagged (as these processes are fast and available for many languages, and have been applied to most of our corpora) but not parsed.

2 Related Work

There are numerous other corpus query tools available. Here we give brief references to several that are either widely used on large corpora or which concentrate on syntactic search.

- the IMS Stuttgart CorpusWorkBench (Christ and Schulze, 1994), limited to 2 billion-words corpora,
- a system presented in (Davies, 2009) which uses standard relational database technology, not closely related to syntactic search,
- tools being part of the Tigersearch project (König and Lezius, 2000), currently unmaintained,
- the Gsearch corpus query system (Steffan et al., 2001),

3 Core CQL

A CQL query is a pattern which may match a token or series of tokens in the corpus. Each token is assigned a set of *attributes* (word form, lemma, part-of-speeech tag etc.) and each corpus might be assigned a set of *structures*. Structures may identify any sequence of tokens and are typically used to mark up documents, paragraphs, sentences, utterances, syntactic phrases of various kinds and named entities. Zero-length structures can also be used. Structures are marked up in standard XML, as `<structure>`, and may carry *structure attributes*.

Basic CQL queries take the form of a sequence consisting of:

```
[attribute="value"]
```

specifications, where *attribute* is a token attribute and `value` is a *regular expression* over the attribute which might use the standard notation of Kleene star/Kleene plus, repetition, set restrictions and various other as defined by the underlying PCRE library (Hazel, 1997). A simple query searching the corpus for the phrase *be looking forward to* + *gerund* is:

```
[lemma="be"][word="looking"][word="forward"][word="to"][word=".*ing"]
```

The query matches *was looking forward to meeting*, *am looking forward to participating* but not *am looking forward to it*.

A single token denoted in the query by [. . .] can contain more restrictions combined using the logical 'and' (`&`) and 'or' (`|`) operators, so that we can improve the example above by specifiying that the last position must be a verb:

```
[word=".*ing" & tag="V.*"]
```

and hereby prevent matching phrases like *looking forward to Beijing*.

Not only an attribute value, but a whole position can be subject to regular expressions, enabling queries suitable for basic phrase structure matching (e. g. *those two flowers, winter flowers, a flower*):

```
[tag="SPEC.*"]{0,2} [tag="N.*"]+
```

Structures can be employed in a query using the notion of `<str>`, `</str>` or `<str/>` which marks the beginning, the end and the whole structure respectively and can contain structure-attribute constraints.

Hence, when looking for nouns at the very beginning of a closed set of documents determined some attribute, we can formulate the query in the following way

```
<doc id="dialect10"> [tag="N.*"]
```

Finally, any position can be assigned a numeric identifier constrained by a global condition at the end of the query:

```
1:[word=".*"] 2:[word=".*"] & 1.tag = 2.tag
```

which will match any two consecutive words with the same value for the `tag` attribute.

4 `within` and `containing` queries

The CQL as described by (Christ and Schulze, 1994) has established a `within <structure>` query limiting the search *within* the boundaries of *structure*. This ensured that when matching e. g. noun phrases using a query of the following form:

```
[tag="N.*"]+ within <s/>
```

the matched sequence will never cross the sentence boundary delimited by structure `s` (which could be e. g. a sentence and takes this meaning in examples given in this paper).

In our recent development, we have extended the `within` query so that the argument can be any subquery of the CQL language. This represents a substantial extension as it makes it possible to contruct complex nested queries matching e. g. for any subpart of a bigger sequence, let it be a structure or an embedding of a sequence into another one. It also makes many queries much easier to craft, speeding up the linguist's work and more closely linking theory to query.

`within` allows us to match any constituent of a phrase structure rule:

```
PP → P NP
     ⇓
  [tag="P.*"]
within [tag="P.*"] [tag="N.*"]+
```

As a dual option to the `within` query, a new `containing` operator has been introduced. It has the same syntax but returns not the embedded, but the embedding sequence. In this respect it allows us to match a phrase structure:

```
PP → P NP
     ⇓
[tag="P.*"] [tag="N.*"]+
containing [tag="P.*"]
```

It may be nested, even together with the `within` operator, enabling truly complex queries which are still computed efficiently.

Some of the notable examples of queries which might be created using the extended `within` and new `containing` operator entail[1]:

4.1 particles

Verb particles can be split up from the verb and thus matching them requires involving queries similar to:

```
[tag="PR.*"] within [tag="V.*"] [tag="AT0"]?  [tag="AJ0"]*
[tag="(PR.?|N.*)"] [tag="PR.*"] within <s/>
```

4.2 relative clauses

Relative clauses introduced by *that* can be distinguished from *that* used in a specifier position by following query:

```
[lemma="that"] within [tag="AT0"]* [tag="AJ0"] [tag="N.*"]
[]{0,20} [tag="V.*"] within <s/>
```

4.3 idioms

Searching for idioms (Bušta, 2008) is especially complex in free word order languages where verbal idioms consist of several parts spread more or less arbitrarily over the whole sentence, as can be demonstrated on the case of the Czech idiom

hnout někomu žlučí – "to get somebody's goat"
word-by-word translated as:
hnout "move" [V, infinitiv]

[1] if not specified otherwise, the BNC tagset (Burnard, 1995) is assumed as given on https://trac.sketchengine.co.uk/wiki/tagsets/bnc.

někomu “somebody” [N, dativ]
žlučí “bile” [N, instrumental].

A unique solution to this problem lies in using a nested `containing` query:

```
<s/> containing [lemma="hnout"] containing [tag=".*c3.*"]
containing [word="žlučí"]
```

4.4 phrasal verbs

We can match phrasal verbs by querying:

(a) `[tag="PR.*"] within [tag="V.*"] [tag!="V.+"] [tag="PR.*"]`
or, depending on the desired match, in the dual way:
(b) `[tag="V.*"] [tag!="V.+"] [tag="PR.*"] containing [tag="PR.*"]`

5 `meet` and `union` queries

A completely separate query type with the semantics inspired by functional programming paradigm is represented by so called `meet` and `union` queries. A `meet` query takes the form of

```
(meet [attribute=value] [attribute=value] -N +M)
```

and matches words satisfied by the `attribute=value` restriction on the first position if and only if there is a token satisfying the costraints given on the second position in a `-N` to `+M` context.

The result of two `meet` queries can be combined using the `union` operator:

```
(union (meet ...)  (meet ...))
```

Given this, one can match all occurrences of *give* in context of *up* together with *drop* in the context of *out* using the following query:

```
(union
     (meet [lemma="give"] [word="up"] -10 10)
     (meet [lemma="drop"] [word="out"] -5 5)
)
```

Note that the `union` operator might be nested as well.

Formerly, a CQL query was either an ordinary query (as described in Section 3) or a single (but possibly nested) `meet/union` query – their mixing was not allowed. We extended the CQL formalism in a way that enables a `meet/union` query to be placed at any position in the standard query syntax. Thus, we can exploit both the extended `meet/union` and `within/containing` queries to make a contextual search for e. g. all sentences where the verb *have* occurs in a -5 to $+5$ context of a pronoun and concurrently some noun occurs in a -3 to $+3$ context of the adjective *blue*:

```
<s/>
containing (meet [lemma="have"] [tag="P.*"] -5 5)
containing (meet [tag="N.*"] [lemma="blue"])
```

6 Parsed Corpora

Our larger corpora do not currently contain phrase structure or similar markup, and for this reason the paper focuses on queries which do not assume it. However we plan to add some varieties, in particular for named entities and for base noun phrases. (Full parsing is appealing until the logistics are considered: there are currently no parsers that can parse a billion words without a very large amount of time and effort.)

Base-np markup integrated with the new operators will make the linguist’s life easier for many queries.

Table 1: Query performance evaluation – corpora legend: ○ BNC (110M tokens), ● BiWeC (version with 9.5G tokens), ∗ Czes (1.2G tokens)

query	# of results	time (m:s)
○ `[lemma="time"]`	179,321	0.07
○ `[lemma="t.*"]`	14,660,881	3.12
○ Ex. 4.1 (particles)	1,219,973	33.36
● Ex. 4.1 (particles)	97,671,485	32:26.48
○ Ex. 4.2 (relclauses)	510,829	1:7.44
● Ex. 4.2 (relclauses)	36,080,966	79:38.33
∗ Ex. 4.3 (idioms)	66	1:6.86
○ Ex. 4.4a (phrasal verbs)	1,257,327	6.54
● Ex. 4.4a (phrasal verbs)	88,494,217	8:2.37
○ Ex. 4.4b (phrasal verbs)	1,241,575	6.23
● Ex. 4.4b (phrasal verbs)	81,159,268	8:14.69
○ Ex. 5 (meet/union)	3	8.47
● Ex. 5 (meet/union)	1457	7:13.12

7 Evaluation of performance

Extending the Corpus Query Language would be useless if evaluating complex `within` and `containing` queries would become infeasible. However, our measurements confirm the powerful efficiency of the underlying corpus manager Manatee. The results are summarized in Table 1: we first provide performance evaluation for some basic queries and the for those listed in previously given examples. All measurements were done in a command-line interface on an AMD Opteron 2.4 GHz machine, BNC (Burnard, 1995) and BiWeC (Pomikálek et al., 2009) corpora were used for querying except for Example 3 where the Czech corpus CZES (Horák et al., 2009) was used.

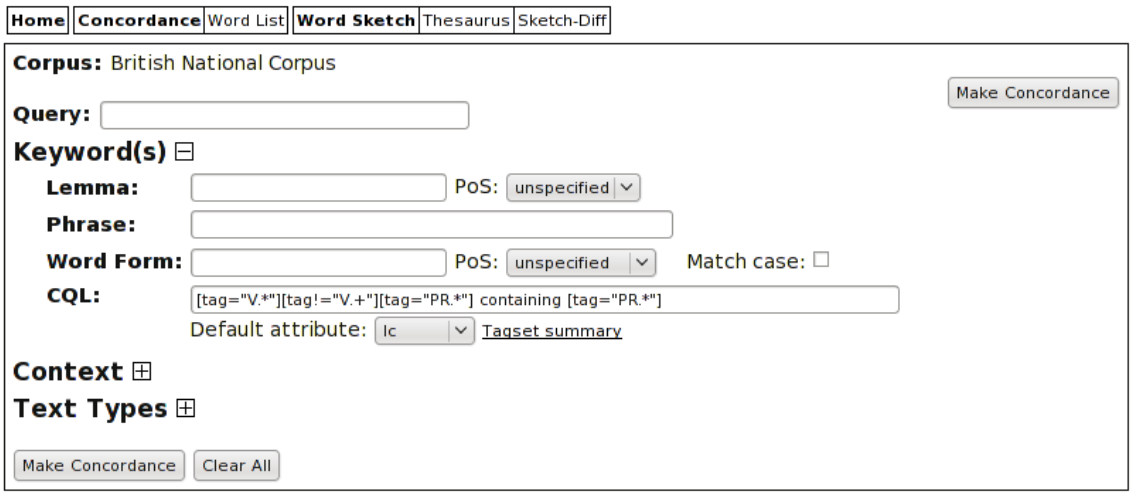

Figure 1: User interface of the system: query dialogue

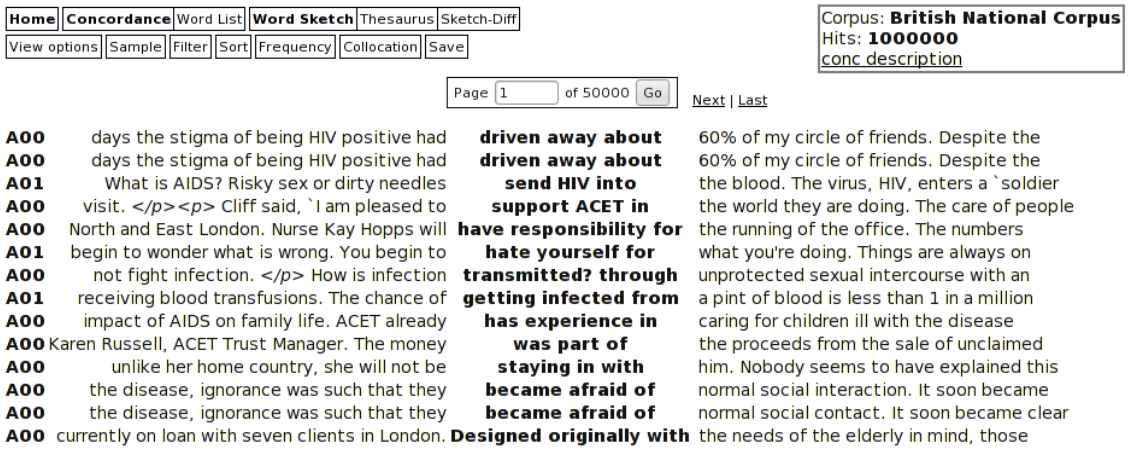

Figure 2: User interface of the system: concordance results

8 Conclusions and future work

In this paper we have presented an extension to the Corpus Query Language (CQL) which makes it possible to perform fast syntactic searches in very large corpora using the `within` and `containing` or `meet` and `union` operator.

These new enhancements enrich the current corpus management system and provide lexicographers with new ways how to utilize their tagged but unparsed corpora.

In the coming years we plan to implement further extensions to the CQL based on the needs of our users and our future research in the field of corpus management systems and databases.

Acknowledgments

This work has been partly supported by the Ministry of Education of CR within the Center of basic research LC536 and in the National Research Programme II project 2C06009 and by the Czech Science Foundation under the projects P401/10/0792 and 407/07/0679. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement Nr. 248307.

References

Burnard, L., editor (1995). *Users Reference Guide for the British National Corpus*. Oxford University Computing Service.

Bušta, J. (2008). Computing idioms frequency in text corpora. In *Proceedings of Recent Advances in Slavonic Natural Language Processing 2008*, pages 71–74, Brno, Czech Republic. Masaryk University.

Christ, O. and Schulze, B. M. (1994). *The IMS Corpus Workbench: Corpus Query Processor (CQP) User's Manual*. University of Stuttgart, Germany, http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench.

Davies, M. (2009). Relational databases as a robust architecture for the analysis of word frequency. In *In What's in a Wordlist?: Investigating Word Frequency and Keyword Extraction*, pages 53–68, Ashgate.

Hazel, P. (1997). *PCRE – Perl-compatible regular expressions*. University of Cambridge, United Kingdom, http://www.pcre.org.

Horák, A., Rychlý, P., and Kilgarriff, A. (2009). Czech word sketch relations with full syntax parser. In *After Half a Century of Slavonic Natural Language Processing*, pages 101–112, Brno, Czech Republic. Masaryk University.

König, E. and Lezius, W. (2000). A description language for syntactically annotated corpora. In *Proceedings of the COLING Conference*, pages 1056–1060, Saarbrücken, Germany.

Pomikálek, J., Rychlý, P., and Kilgarriff, A. (2009). Scaling to Billion-plus Word Corpora. *Advances in Computational Linguistics*, 41:3–13.

Reddy, S., Kilgarriff, A., and Pomikálek, J. (2010). A corpus factory for many languages. In *To appear in LREC*, Malta.

Rychlý, P. (2000). *Korpusové manažery a jejich efektivní implementace*. PhD thesis, Fakulta informatiky, Masarykova univerzita, Brno.

Steffan, C., Corley, M., Keller, F., Crocker, M. W., and Trewin, S. (2001). Finding syntactic structure in unparsed corpora: The gsearch corpus query system. In *Computers and the Humanities 35:2*, pages 81–94.