CQPweb — combining power, flexibility and usability in a corpus analysis tool

Andrew Hardie
Lancaster University

CQPweb is a new web-based corpus analysis system, intended to address the conflicting requirements for usability and power in corpus analysis software. To do this, its user interface emulates the BNCweb system. Like BNCweb, CQPweb is built on two separate query technologies: the IMS Open Corpus Workbench and the MySQL relational database. CQPweb's main innovative feature is its flexibility; its more generalised data model makes it compatible with any corpus. The analysis options available in CQPweb include: concordancing; collocations; distribution tables and charts; frequency lists; and keywords or key tags.

An evaluation of CQPweb against criteria earlier laid down for a future web-based corpus analysis tool suggests that it fulfils many, but not all, of the requirements foreseen for such a piece of software. Despite some limitations, in making a sophisticated query system accessible to untrained users, CQPweb combines ease of use, power and flexibility to a very high degree.

Keywords: concordancer, corpus analysis tool, Corpus Workbench, database, web interface

1. Introduction

It is widely understood, though perhaps not explicitly stated quite as often as it should be, that the practice of corpus linguistics is utterly dependent on the availability of computer software capable of effectively performing concordance queries and other analytic procedures. Without such software, corpus linguistics may accurately be considered as a "pseudo-procedure", in the words of Abercrombie (1965). It is not the corpus alone that allows novel and interesting research questions to be addressed: rather, it is the combination of corpus and search software. But while concordancers and other corpus analysis tools "are powerful aids to the linguist, they also, crucially, limit and define what we can do with a corpus"

(McEnery & Hardie 2012: 36). Overcoming the limits that are thus implicitly placed on corpus-based research requires continual improvement in both the *power* and the *usability* of the available corpus analysis software.

This paper introduces CQPweb, a new web-based corpus analysis system intended to help address these issues. CQPweb has been designed to emulate the widely-used BNCweb interface to the British National Corpus (Lehmann et al. 2000, Hoffmann & Evert 2006, Hoffmann et al. 2008). CQPweb's main innovative feature is its flexibility, in that it is compatible with any corpus, rather than being bound to a particular dataset such as the BNC. As a front-end to extremely powerful data management tools including the IMS Open Corpus Workbench (CWB; see Christ 1994) and the MySQL relational database system, CQPweb is itself a powerful tool; since it implements the friendly and accessible user-interface design popularised by BNCweb, it combines that power with a high degree of usability.

The paper is structured as follows. In Section 2.1 I review some historical developments in corpus analysis tools from the dual perspectives of power and usability, extending the account previously presented in McEnery & Hardie (2012), and arguing that the current "fourth generation" of concordancers (of which BNCweb and CQPweb are both representatives) have made the greatest progress to date to combine power and usability. However, as outlined in Section 2.2, BNCweb lacks *flexibility* in the sense that it is bound to a single corpus. The data model used by CQPweb, which achieves cross-corpus flexibility without losing the power and usability that characterise BNCweb, is described in Section 3.1, while some other features of CQPweb's architecture are outlined in Section 3.2; the system's capabilities are overviewed in Section 4. Finally, in Section 5 I attempt a cursory evaluation of CQPweb, in particular comparing what it achieves to the requirements for a future corpus web-interface laid out by Hoffmann & Evert (2006), but also considering the system's present limitations — especially in terms of flexibility — and giving some indication of additional functionality planned for future versions.

2. Corpus analysis tools: Some background

2.1 Usability versus power

The development of corpus linguistics as a distinct practice within linguistics has arguably been both enabled and constrained by the software tools available to corpus analysts at different points in time (see McEnery & Hardie 2012: Chapter 2 for an outline). The capabilities of these various analysis tools have, in turn, been shaped by two very different demands: the need for power and the need for usability (a third demand, for *flexibility*, will be discussed in the following section).

The level of corpus-based research activity expanded dramatically in the late 1980s and early 1990s. This boom was driven largely by new PC software such as the Kaye concordancer (Kaye 1990), the Longman Mini Concordancer (Chandler 1989) and Micro-OCP (Hockey 1988). The popularity of these concordancers — which McEnery and Hardie refer to as "second-generation", following the "first generation" of mainframe-based software — was in turn enabled by the spread of the IBM-compatible PC. This second generation, and later third-generation tools such as WordSmith (Scott 1996), MonoConc (Barlow 2000) and AntConc (Anthony 2005), were driven in part by the need for tools that are usable by the great majority of linguists who are not also computer programmers, and who do not necessarily have a great deal of detailed technical knowledge of computers. For this reason, these second- and third-generation concordancers run on a desktop computer under a Windows, Macintosh or desktop Linux operating system, rather than requiring remote login to a server; and are controlled via a graphical user interface (GUI) similar to a word processor, spreadsheet, or other familiar piece of desktop software, rather than requiring the use of the command line.

However, this usability or user-friendliness — while hugely successful in opening up corpus analysis to non-technical-specialists, including students and scholars in other branches of linguistics or other disciplines — has typically come at the expense of power. The *power* of a corpus analysis tool can be considered from two perspectives. First, a powerful tool may be defined as one that can query a very large corpus (on the order of tens or hundreds of millions of words), and can do so efficiently, i.e. within a practical time span. Query speed is in part dependent on the computer hardware. But even on high-end hardware, to achieve query speed on very large datasets, a program must generally work with *indexed* corpus data rather than raw text files (a corpus index is a data structure which allows the query program to locate matches for a query without searching sequentially through all the text of the corpus). Second, a powerful tool may be defined as one that allows complex and sophisticated queries, especially queries that go beyond just searching for particular strings of characters. For example, a tool where annotations such as grammatical tags or text headers can be queried via an abstract data model may be considered more powerful than a tool where such markup must be queried via a search for its literal form in the original files. Alternatively, a more powerful tool might allow additional forms of analysis beyond just concordancing. As a general rule, the most powerful software tools for corpus analysis have been *less* user-friendly. Perhaps the paradigmatic example of this tendency is CWB and its concordancer, the Corpus Query Processor (CQP). CQP is extremely powerful: it can search very large corpora very quickly, and has a sophisticated query language. But CQP is a command-line program controlled by typing commands into a query parser; when a query has run, its results are printed to the command line. It is thus a rather intimidating and strange-seeming program for the very audience of non-technical-specialists so successfully reached by desktop query tools that focus on usability.

Of course, a technically-savvy user, familiar with a Unix-like command-line, would not find CQP unfriendly. But the overwhelming majority of scholars who might be interested in using corpus methods are non-technical users who would find CQP unapproachable. Even *within* some of the more usability-oriented tools, the most powerful features are often the least accessible to the novice or the non-technical user. An example is WordSmith Tools. WordSmith is capable of indexing a corpus for later analysis, which makes subsequent queries much faster. But this indexing process is typically outside the range of operations understood by the average non-specialist WordSmith user — and so such users are restricted to a subset of the power that WordSmith actually possesses.

Some technically sophisticated corpus researchers recommend a "do-it-yourself" approach to corpus analysis tools, where rather than exploiting an off-the-shelf concordancer, the analyst instead writes their own computer programs to process their data. That the corpus user should know how to program is argued forcefully by Biber et al. (1998: 254) and by Gries (2010), among others (see also Mason 2001, Weisser 2009). Clearly this "do-it-yourself" approach to corpus analysis software is the most powerful imaginable *if* power is equated to maximum scope for adaptability — although "do-it-yourself" programs may run slowly if they do not incorporate the "tricks", such as indexing, needed for high speed on large datasets. However, this approach falls short in terms of usability for the majority of potential corpus analysts, who either cannot or do not wish to learn computer programming. It should be clear that given a choice between (i) learning to program or (ii) not using corpus data, the majority of potential corpus analysts will — understandably — opt for (ii).

For such researchers, usability will always be more critical than power. But we should not minimise the problems of privileging usability over power. The nature of the query tool used in a corpus analysis both enables and constrains the range of research questions that may reasonably be addressed. As McEnery & Hardie (2012: 37) argue, the potential of corpus data

> […] is unlocked by tools that allow linguists to manipulate and interrogate the corpus data in linguistically meaningful ways. The availability of tools that are relevant to specific research questions remains a crucial limiting factor in corpus linguistics.

An analyst whose toolbox is limited to the power of a GUI-based desktop concordancer is therefore also limited in the questions they can research. Most obviously, such an analyst may be limited in their choice of corpora: even if a larger corpus may be most appropriate to their research goals, they will be constrained to work with corpora no larger than the maximum amount of data their concordancer has the power to handle.

How can we address the competing demands of usability and power? One way is by means of a client/server software model. In this approach, a *client* program is responsible for interacting with the user. The user composes their query in the client, but the client does not actually search the corpus: instead the query is communicated to the *server* program which is responsible for running it. The client and server programs *might* run on the same computer — but they do not have to. Instead, they can run on different computers and communicate across a network. It is thus possible for the client to provide a user-friendly GUI on a desktop computer, while the server is a powerful (but non-user-friendly) tool running on a more capable machine, and, possibly, serving a multitude of clients.

The use of a client/server model for corpus tools is not new. The SARA software used to search the original release of the BNC (Aston & Burnard 1998) operates on this model, consisting of a server program called sarad which users never needed to deal with directly, together with the SARA client for Windows.[1] Xaira, the successor to SARA, has the same architecture. However, more recently, in what McEnery & Hardie (2012) dub the "fourth generation" of concordancers, the client/server model has been used across the World Wide Web. In this case, the client software is no more than an interactive web page. Typically, the user enters their query into a hypertext form, which is then submitted to the web server; the actual search program, or "back-end" software, runs on the web server, and its output is returned to the client in the form of a page of HTML that can be displayed in the user's browser.

In some cases, fourth-generation concordancers are used for copyright reasons — to allow a corpus to be made openly accessible across the web without giving users access to the underlying text, where doing so would breach the original text producers' rights. They also decouple the issue of corpus searching from the limits of the memory and processing power of the user's desktop computer. Likewise, because of the move to the web, fourth-generation tools automatically run on every operating system — unlike many third-generation tools which were available on only one operating system. Then too, the "price of entry" in terms of technical competence is much lower for a web-based concordancer than for a third-generation tool. Even very user-friendly third-generation tools such as WordSmith or AntConc may be difficult to install and run for users with little knowledge of computers. With fourth-generation concordancers, there is nothing to install — since most desktop computers come with at least one browser pre-installed. Furthermore, using web pages with forms as the primary user interface gives fourth-generation tools an instant advantage in usability, because even non-technically-aware users are very likely, in the modern world, to make regular use of web pages with forms for a wide range of activities, from online banking to social networking. As a result of all these factors, fourth-generation concordancers arguably represent the greatest progress made to date in satisfying simultaneously the need for power *and* the need for usability.

The best-known fourth-generation corpus analysis tools include Wmatrix (Rayson 2008), SketchEngine (Kilgarriff et al. 2004), and the corpus.byu.edu system (Davies 2005, Davies 2009, Davies 2010). There are also a large number of "one-off" systems, where a web-based tool has been constructed solely to afford access to a particular corpus on a single website. For instance, the primary means of accessing resources such as the PELCRA reference corpus of Polish and the Hellenic National Corpus is via such an interface.[2] Technically, fourth-generation systems can be classed into two groups: those that use a relational database system as their back-end, and those that use an indexing and query system such as CWB/CQP or the Xaira server. Relational databases are not specifically designed for corpus analysis, but can easily be adapted to this purpose by constructing database tables where each row in the table represents some unit of linguistic interest — typically a single word-token. These tables are then searched by converting the user's query request to a database search language such as the Structured Query Language (SQL); the results are then returned in the format of a concordance, frequency list or other standard output. The PELCRA interface is SQL-based, for instance (Uzar et al. 2004). Mark Davies' corpus.byu.edu interface — which allows access to several different corpora including the BNC, the Corpus of Contemporary American English and the Corpus del Español — probably represents the most sophisticated system built on a relational database. In the other category, one of the most advanced systems is SketchEngine, a multi-corpus system whose back-end software is a CWB/CQP-compatible program called Manatee (Kilgarriff et al. 2004). Looking at systems that, unlike SketchEngine, are not simply CWB-compatible but use CWB itself as their back-end, we find a large number of "one-off" corpus access tools, but also more generalised tools, such as Serge Sharoff's corpus.leeds.ac.uk system, via which a range of very large corpora in different languages are made available. However, the CWB-based system which arguably goes furthest in the combination of usability and power (in both senses) is BNCweb.

2.2 From BNCweb to CQPweb: The role of flexibility

The fourth-generation concordancers mentioned above, while usable and powerful, vary in their *flexibility*, considering flexibility specifically in the sense of the range of corpora that they can work with. Third-generation tools typically allow a user to work with whatever corpus data they have on their computer. Some fourth-generation tools approach this level of flexibility, most notably Wmatrix, which operates primarily on text uploaded by users. But most uses of the client/server model lead to a situation where one corpus or a small number of corpora are set up on the server by the administrator of the server software, and then accessed passively by the users of the client program. Users cannot install their own data. The fourth-generation systems thus lack flexibility that the third generation possessed. In the extreme case, such as BNCweb, there is no choice of dataset at all. The genesis of CQPweb as an advance on BNCweb, which I will consider in this section, can be viewed as an exercise in restoring (some but not all of) the flexibility that was lacking.

The original version of BNCweb (Lehmann et al. 2000), using a SARA back-end, was one of the earliest fourth-generation concordancers. From the outset BNCweb provided power (in the range-of-possible-analyses sense) beyond the affordances of SARA by using a relational database to implement functions such as: a tool to view the distribution of a query's results across BNC text categories; the ability to sort concordances by an adjacent word or part-of-speech (POS) tag; and a highly-configurable system for extracting statistical collocations from a concordance. Each of these functions operated by reformatting some output of the SARA concordance query into an SQL database table on the fly; this table could then be queried to produce results for the user.

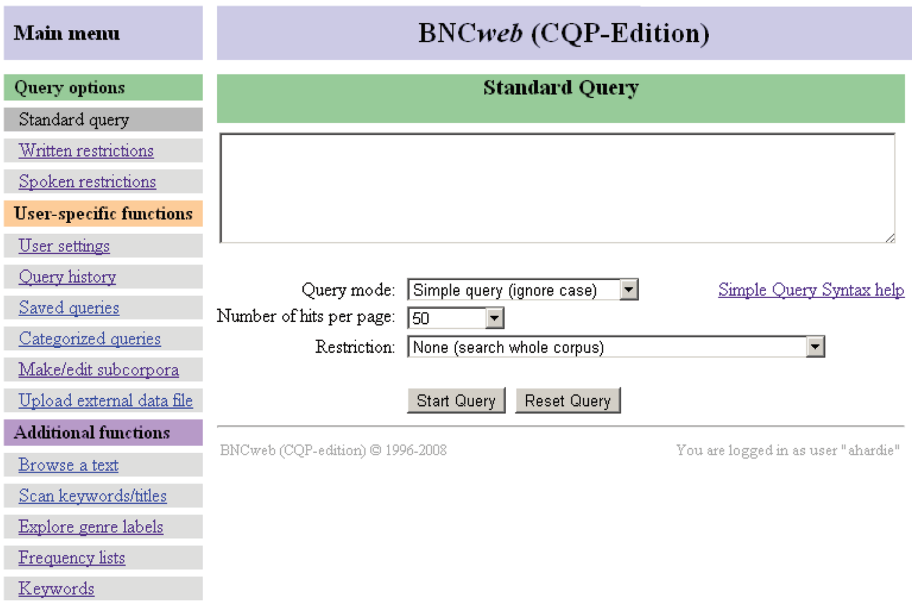
BNCweb has evolved significantly over time; the biggest change has been a shift in the back-end from sarad to CWB/CQP, described by Hoffmann & Evert (2006). Although Hoffmann and Evert describe substantial alterations to the internal working of BNCweb, the changes to the interface seen by the user were slight. The main difference is that most queries are now specified by means of a simplified query language dubbed Common Elementary Query Language (CEQL), designed by Stefan Evert. This gives access to the most useful features of CQP while greatly reducing the complexity of the formalism that users need to learn. Apart from this change, there have also been incremental advances as new features are added or existing features refined. An example is given by Smith et al. (2008), who detail a function where a user can download a query result, thin it manually or automatically, and then re-upload it into the BNCweb system. A comprehensive and detailed description of BNCweb's capabilities has been published as Hoffmann et al. (2008).

As the foregoing overview should make clear, BNCweb provides both a high degree of power — in the sense of speed in querying a 100 million word corpus, and also in the sense of the wide range of functions and scope of the query language — and a high degree of usability through the web-based user interface. As has been noted, however, BNCweb does not provide the flexibility to work with multiple datasets. BNCweb is tightly bound to the BNC in a number of ways, via assumptions built into the program which are "known" to be true for the BNC but might well not be true for other corpora. For example, BNCweb "knows" that each word in the BNC is annotated for POS tag and lemma, that these annotations have been indexed into the CWB back-end, and that frequency lists of them must be created and stored in the database. It is also "known" that the BNC is primarily divided into spoken and written sections and that separate frequency lists for each will be needed. BNCweb "knows" that, in the markup of the BNC, the <u> tag is used to indicate utterances and its "who" attribute is used to indicate speaker identity; it thus "knows" that a database of "u-who" is needed and is to be used in spoken-corpus restricted queries. Many more examples could be added; all these things that BNCweb "knows" are obviously untrue of many other corpora.

As a user of BNCweb, I found the fact that it could not be used with other corpora both completely expected in light of its design goals and rather frustrating. My impression, gained from teaching corpus linguistics over a period of some years, was that the inconsistency of interface across corpus analysis tools was a positive barrier to learning. Consider a student who has learned the basics of corpus linguistics using BNCweb. If, subsequently, the demands of their course mean they need to work with, say, the FLOB corpus (Hundt et al. 1998), they will have to switch to another concordancer such as WordSmith or AntConc. Given the fairly large differences of user interface, they will need to spend a not insignificant amount of time re-acclimatising to the basic procedures of what menu options to select, what buttons to press, and so on, to get a concordance or a collocation list in this new tool.

From a pedagogical perspective, time spent learning to use a *second* corpus tool is dead time. Nothing educationally productive is accomplished. The student already understands what a concordance is, what collocations are, and so on. Mastering a new tool has value if the new tool opens up *additional* methodological options. But nothing is added methodologically when students learn an alternative sequence of buttons to press to get the same basic outputs. In fact, this reduces the amount of time that can be spent teaching the difficult and/or interesting parts of corpus linguistics.

To address this problem, I began in late 2008 to program a new tool that would allow students to apply the interface skills they had learnt on BNCweb with any of the corpora commonly used in teaching corpus linguistics at Lancaster University. I gave it the name CQPweb; this is of course modelled on BNCweb, to acknowledge its inspiration, but it also captures the system's basic nature as a generalised web-interface to CQP.[3] CQPweb does not descend from BNCweb in the sense of being a direct derivative of its code-base; rather, it was newly written from the ground up, in PHP rather than the Perl used by BNCweb, but closely following BNCweb as a model.

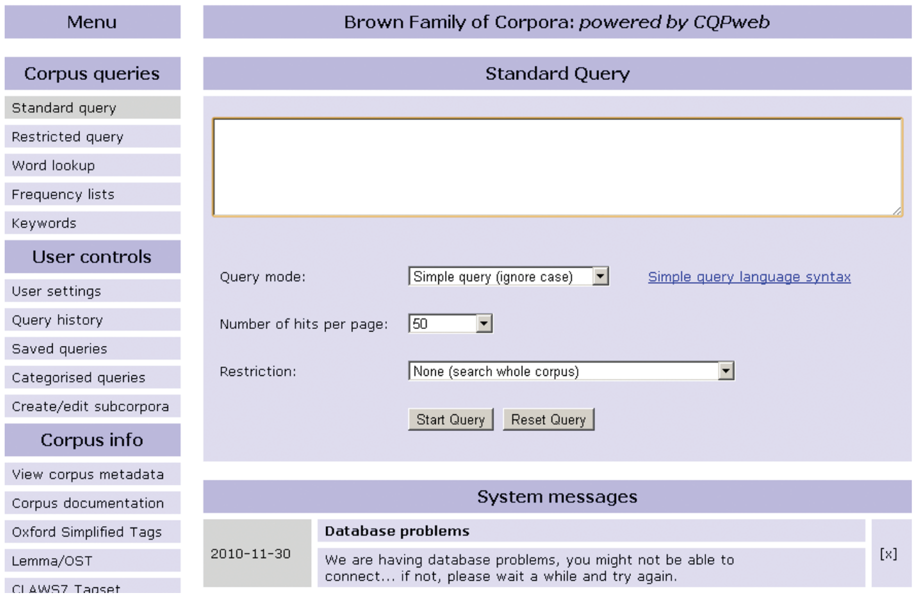


Query mode: Simple query (ignore case)
Number of hits per page: 50
Restriction: None (search whole corpus)

BNCweb (CQP-edition) © 1996-2008

You are logged in as user "ahardie"

Figure 1. The welcome-page of BNCweb (compare Figure 2)



Query mode: Simple query (ignore case)
Number of hits per page: 50
Restriction: None (search whole corpus)

2010-11-30 **Database problems**
We are having database problems, you might not be able to connect... if not, please wait a while and try again.

Figure 2. The welcome-page of a corpus indexed in CQPweb

Screenshots of the main search page (which also serves as the welcome page) of BNCweb and CQPweb are given in Figures 1 and 2 respectively, and illustrate the difference between them. The overall layout of the interface is as close to identical as possible. The goal is that a user familiar with BNCweb should be able to use CQPweb without further training. However, some differences result from the need to generalise the interface to any corpus. For example, where BNCweb has options for "Written restrictions" and "Spoken restrictions", CQPweb has an option for "Restricted query" — because unlike BNCweb, CQPweb cannot "know" that speech versus writing is the top-level distinction for this particular corpus. Differences of this magnitude, introduced in the process of generalisation, are found throughout CQPweb. They have proven unproblematic for most users.

Initially, the goals of CQPweb were purely pedagogical and students were the main target users. However, it subsequently became clear that researchers found the system useful as well. There are two areas where CQPweb has proven particularly useful. First, it has been used to give broad access via the web to newly-developed corpora. For instance, the BE06 corpus (Baker 2009) is made publicly available via Lancaster University's CQPweb server. The usability that CQPweb has inherited from BNCweb has also made it suitable as a conduit by which corpus techniques can be made accessible to linguists with no experience working with corpora, and likewise to scholars in other humanities and social science fields. For example, by indexing the keyboarded text of Early English Books Online on Lancaster's CQPweb server, we have made it possible for historians to apply the full range of corpus methodologies to this dataset, complementing the usual historical methods of text analysis.

CQPweb is released under the GNU General Public Licence, which permits use, modification and redistribution, but requires redistributed or modified versions to be placed under the same licence. Several different research institutions have set up CQPweb servers in the period since its initial release. Subsequent to its creation, CQPweb has been adopted as the main graphical user interface of CWB/CQP. In their discussion of CQP, Hoffmann & Evert (2006: 180) note that "the Corpus Workbench suffers from the lack of a user-friendly graphical interface to the query processor". CQPweb has come to fill this gap, at least to some extent.

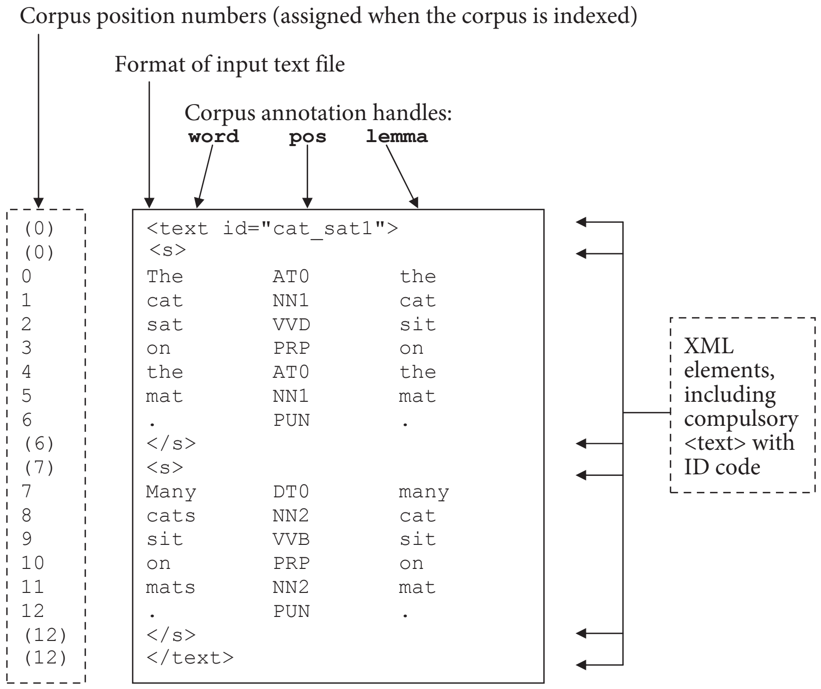
3. Architecture and technology

3.1 Data model

CQPweb shares the fundamental data model of CWB. A corpus is understood as consisting of a stream of tokens, where each token is assigned an integer number that represents its position in the corpus, starting at zero. For example, if a corpus consists solely of the sentence *The cat sat on the mat*, then the token position numbers are 0, 1, 2, 3, 4 and 5. Punctuation marks are treated as independent tokens.

These position numbers are only used internally; the user need never deal with them. For a corpus to be indexed in CWB, it must be laid out in plain-text files, in vertical format, with one token per line. Alongside each token, in columns separated by tab characters, different fields of word-level annotation can optionally be added. Typical word-level annotations include POS tags, lemmata, and semantic tags. CWB creates a separate index for each column. In the indexing process, the column is encoded as a sequence of binary integers, each of which represents a word-form in the lexicon built for that column; a reverse index is also created listing all occurrences of each word-form so that they can be swiftly located (for more details regarding CWB indexing see Christ 1994, Evert & Hardie 2011; all these details are, of course, hidden from the CQPweb user). Each column is given a "handle" in the course of indexing, which in CQPweb must be a "C word".[4] XML markup to indicate ranges (such as sentence start and end points) can also be included, where each XML tag is placed on a line on its own (and does not "count" as a token in the numeric sequence); these XML tags may have attributes in the usual format. All data must be encoded as either ASCII, ISO-8859, or UTF-8.[5]

In addition to these basic rules for a CWB corpus, CQPweb adds certain additional requirements. The corpus must be divided into *texts*, which must be indicated with <text> elements. If the corpus cannot meaningfully be broken down


Corpus position numbers (assigned when the corpus is indexed)
Format of input text file
Corpus annotation handles:
word
pos
lemma
XML elements, including compulsory <text> with ID code

```
(0)    <text id="cat_sat1">
(0)    <s>
0      The     AT0     the
1      cat     NN1     cat
2      sat     VVD     sit
3      on      PRP     on
4      the     AT0     the
5      mat     NN1     mat
6      .       PUN     .
(6)    </s>
(7)    <s>
7      Many    DT0     many
8      cats    NN2     cat
9      sit     VVB     sit
10     on      PRP     on
11     mats    NN2     mat
12     .       PUN     .
(12)   </s>
(12)   </text>
```

Figure 3. The CQPweb data model: CWB input format

into texts, a single pair of <text>…</text> tags is still needed, enclosing the entire corpus. Furthermore, each text must have an ID code, represented within the XML tags by an "id" attribute; ID codes are handles and thus must also be "C words". Figure 3 shows the layout of a minimal CQPweb corpus, with BNC-style POS tags and lemmata as optional annotations, and optional <s> tags indicating sentences.

The stream of tokens, which is always indicated by the handle "word", is always present in a CWB-indexed corpus. But other "columns" may or may not be present, and may differ from corpus to corpus. So when a corpus is indexed in CQPweb, information on the annotations that are present is stored in the system database (this is distinct from BNCweb, where the nature of the corpus is "known" and hard-coded). For example, some of the information stored in the database of Lancaster's CQPweb installation for the BE06 corpus (Baker 2009) is as shown in Table 1.

Table 1. The CQPweb data model: Annotation metadata, exemplified for the BE06 corpus

Corpus	Handle	Description	Tagset	External URL
be2006	pos	Part-of-speech tag	CLAWS7 Tagset	http://ucrel.lancs.ac.uk/claws7tags.html
be2006	hw	Lemma	Lemma	
be2006	semtag	Semantic tag	USAS Tagset	http://ucrel.lancs.ac.uk/usas/
be2006	class	Simple tag	Oxford Simplified Tags	http://www.natcorp.ox.ac.uk/docs/URG/codes.html#klettpos

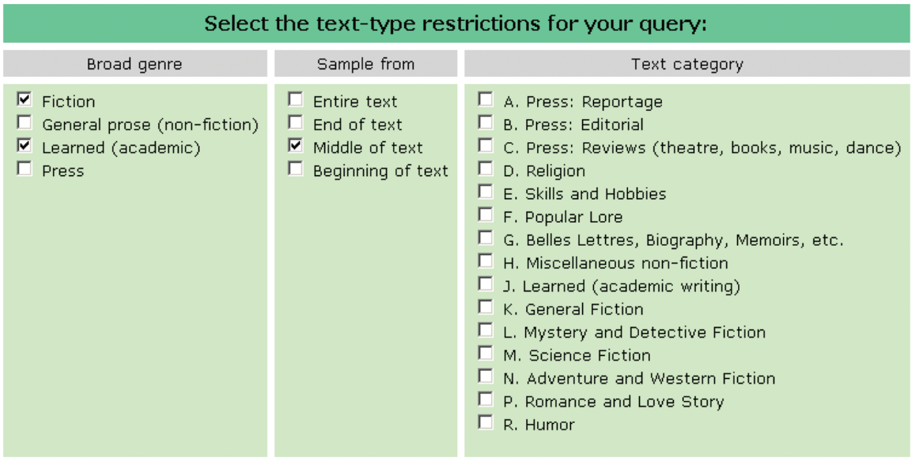
Since BE06 has been annotated by both the CLAWS POS tagger (Garside et al. 1987) and the USAS semantic tagger (Rayson et al. 2004), both these annotations are represented in the data model, as well as lemmata and a "simple tag" which complies with the simplified POS tags present in the BNC (which have been unofficially dubbed "Oxford Simplified Tags" in homage to their origin at Oxford University Computing Services). The information stored under "Description" and "Tagset" is entered by the system administrator when the corpus is indexed. It is the information given here that is actually presented to the user when they interact with the streams of annotations accessed via the handles "word", "pos"' "hw", "semtag" and "class". Whenever a function is accessed that can operate on any annotation (for example, viewing a frequency list), the user is presented with a web form in which they must specify the annotation they wish to work with. The options on the web form are generated from the "Description" column of the database table shown in Table 1. The name of the tagset used by that annotation, and the location of a website where that tagset can be viewed, may also be specified if available. Elsewhere in the database, for each corpus a *primary annotation* is specified. This is the single annotation that is treated specially (for example, it is visualised in the concordance, and can be used for concordance sorting). As noted above, the "special" annotation in BNCweb is the POS tag, but CQPweb makes no assumption about this, and a different annotation — or none at all — can be specified as "primary" on a per-corpus basis.

The final major part of CQPweb's data model is that each corpus has a text-level metadata database. Since *metadata database* is both unattractive and cumbersome as a description, I will henceforth use the term 'metadatabase' for the collection of text-level information associated with each CQPweb corpus. The metadatabase, which must be loaded when a corpus is indexed, has one row for each text in the corpus, with each row labelled by the text identifier used in the <text> tags of the input text (see Figure 3). As many fields of metadata as required can be added as columns in this table. Fields can be of two sorts: *free text* where the content of the field may be different for every text, and *classifications* where the field takes one of a limited number of values, each of which represents a category in some classification scheme. Typically, a metadatabase would contain information extracted from a corpus file header. But if the corpus lacks text-level metadata, a minimal metadatabase can be automatically generated which contains *no* fields of metadata — only the text identifiers. Since the number and nature of the fields is expected to differ from corpus to corpus, a record of the metadatabase's structure (number of columns, their names, and so on) is also held in the CQPweb database. A partial example of a metadatabase is given in Table 2.

The column designations in a metadatabase are handles; like the handles for annotations, they are associated with descriptions elsewhere in the database, and it is the descriptions that the user actually sees. Table 2 exemplifies both types of text metadata. The "title", "author" and "date" columns are free text — they contain bibliographic information for each specific text. The "textcat", "genre" and "sampled" columns are classifications. They contain one of a finite set of values, where each value indicating a category. Since BE06 follows the Brown Corpus sampling frame (Baker 2009: 317), the main classification schemes are the fifteen-genre system of that sampling frame ("textcat"), and the broader four-genre system into which the Brown sampling frame categories are often collapsed ("genre"). The category labels are handles, and can be associated with longer descriptions in the database. This system allows an interface for placing text-level restrictions on a query to be generated dynamically, using whatever classification metadata a corpus happens to possess, in a checkbox-based format identical to the "Written restrictions" function of BNCweb. The interface generated for BE06 is shown in Figure 4. A query

Table 2. The CQPweb data model: A sample of the metadatabase for BE06

text_id	textcat	genre	title	author	sampled	date
A01	A	press	Love is All Around Us; Ocean Colour Scene Turn up For Lemon; Factory boss and the :70m homes plan; Young mum dies one month after leukaemia comes back	Aberdeen Evening Post	all	Nov 21 2004
A02	A	press	Bid to end Gypsy land scam; He'll be cleared says Blair; Pounds 100 to buy an ID card and a Pounds 2,500 fine if you don't	Daily Mail	all	Nov 30 2004
A03	A	press	Patients 'fleeced' by hospital ban on mobile phones; Pounds 106,000pa: What family doctors earn despite most refusing to work unsociable hours; The great migrant riot farce	Daily Mail	all	Nov 30 2006
A04	A	press	3 black watch heroes die in suicide attack; bomb injures 8 soldiers; Arafat in a coma; Bush off, george; straw rules out british attack on iran; Civil war on cuts; 3 black watch heroes killed by suicide bomber; lorry blast as iraqis launch mortar blitz; Den and buried; pervy star gets eastenders axe … And it's final	Daily Star	all	Nov 5 2004
A05	A	press	'GET OUR BOYS OUT… NOW'; Ken to spend £100,000 in new Trafalgar battle over Mandela statue; Man wanted after gay-hate knife attack; Pounds 340m payout as Caz ties the knot with JP Morgan	Evening Standard	all	Nov 30 2004
A06	A	press	Small shops in revolt against Post Office; Tsunami families left in legal limbo: Families in limbo; Ministers at war over pub closing time	The Guardian	all	Jan 15 2005



Select the text-type restrictions for your query:

Broad genre	Sample from	Text category
☑ Fiction	☐ Entire text	☐ A. Press: Reportage
☐ General prose (non-fiction)	☐ End of text	☐ B. Press: Editorial
☑ Learned (academic)	☑ Middle of text	☐ C. Press: Reviews (theatre, books, music, dance)
☐ Press	☐ Beginning of text	☐ D. Religion
		☐ E. Skills and Hobbies
		☐ F. Popular Lore
		☐ G. Belles Lettres, Biography, Memoirs, etc.
		☐ H. Miscellaneous non-fiction
		☐ J. Learned (academic writing)
		☐ K. General Fiction
		☐ L. Mystery and Detective Fiction
		☐ M. Science Fiction
		☐ N. Adventure and Western Fiction
		☐ P. Romance and Love Story
		☐ R. Humor

Figure 4. Restricted query interface for BE06 in CQPweb

run with the restrictions shown would find *only* results in fiction or academic texts that were sampled from the middle of a longer document.

This somewhat lengthy discussion of CQPweb's data model has hopefully illustrated how it achieves the inter-corpus flexibility that is the primary point of distinction between it and BNCweb, without losing either power or usability: namely, CQPweb's data model, an extension to the general CWB data model, includes a full internal description of all the things about each corpus that BNCweb is programmed to "know" about the BNC, especially relating to annotations and text-level metadata. Some other aspects of the architecture of CQPweb, which it mostly shares with BNCweb, are detailed in the following section.

3.2 Software architecture

CQPweb operates across the web on a client/server model. Its foundation is a set of CWB-indexed corpora. CQPweb is capable of interacting with the CWB utility programs to create and manage the indexes. A relational database (MySQL) is employed alongside the CWB indexes. In addition, CQPweb makes use of the CEQL module from CWB's Perl interface to parse the syntax of simple queries into CQP-syntax queries. Figure 5 shows how the different components of CQPweb fit together .

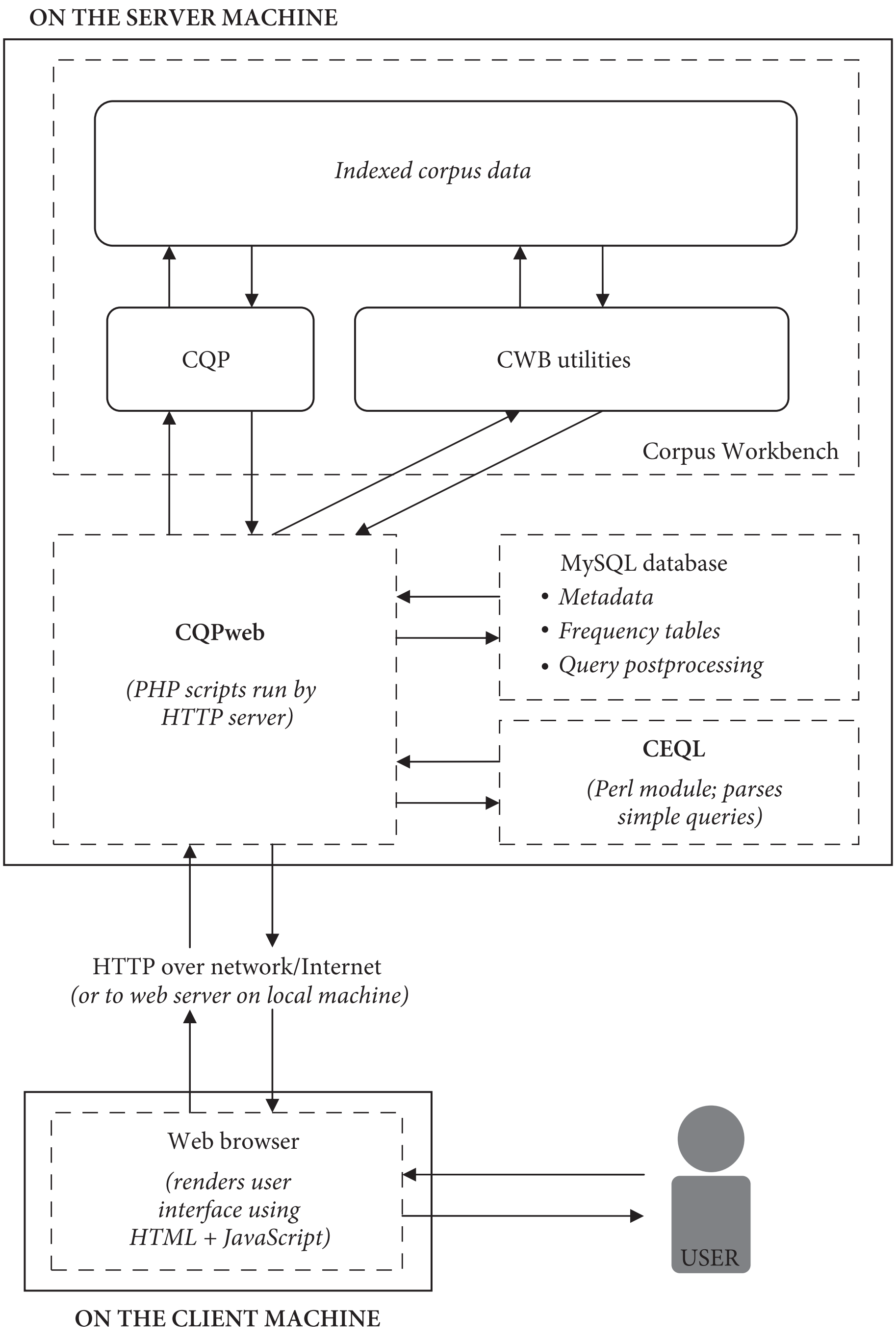


Figure 5. General architecture of the CQPweb system

4. What CQPweb can do

4.1 What the user can do with CQPweb

When a user logs on to CQPweb they are presented with a list of available corpora. Selecting a corpus brings them to the welcome screen (as in Figure 2), from where they can perform a query or choose another menu option. At any point they can return to the main menu and select a different corpus to work with. Within a given corpus, the major functions of CQPweb replicate and generalise those of BNCweb. These include providing the concordancing power of CQP both in full and via the simpler CEQL language. For instance, using CQP-syntax allows regular expression-based searching, whereas CEQL makes a subset of regular-expression syntax available in the form of simplified wildcards such as <?> for "any one character" or <*> for "any string of characters". Both types of query can address annotations as easily as the wordform. A query which imposes conditions on both word-form and part-of-speech annotation has the following form in CQP-syntax:[6]

```
[word="dogs"%c & pos="NN2"]
```

The corresponding CEQL is much simpler, and highly mnemonic due to its use of the underscore character (which is traditionally used to associate words with POS tags):

```
dogs_NN2
```

CEQL also gives access to a second annotation stream, indicated by putting a query pattern in {braces}.

As illustrated by Figure 4, both types of query can be restricted by textual metadata categories. All queries (and many database operations) are cached, making them much faster to run a second or subsequent time. When a query has been generated, a range of 'postprocesses' are available — additional procedures by which the result may be analysed further. Each postprocess is controlled via a user-friendly web form. The main query postprocesses currently available in CQPweb are as follows:

- *Thinning*. Queries can be thinned randomly or pseudo-randomly to a set number of results or to a percentage of their original size.
- *Collocation*. The user can move from a concordance to a table of statistically-generated collocates. Collocations can be calculated on any annotation, i.e. on POS or semantic tags or lemmata if they are available; a choice of statistics including log-likelihood and mutual information are available and other parameters, such as minimum frequency thresholds, can be configured. Clicking on any collocate creates a query containing just those hits from the original query that have the collocate in question in their context.

- *Distribution*. The user can generate a table or bar chart showing the relative frequencies of query hits across the categories of a given textual classification (as modelled in the metadatabase; see Section 3.1). Clicking on the name of a given category produces a new query thinned to just hits within that category.
- *Categorising*. The "Categorise query" function allows the user to define their own analytic labels, and then categorise the query results according to those labels. A categorised query can be split into several different queries, each containing the hits assigned to a particular label. One typical use of this function is to annotate the results of a query with low precision, to filter out undesired hits.
- *Sorting*. Queries can be sorted on the "hit" word or a selected position before or after; the sort can be by word or by the primary annotation. Sorting can also thin a query, since it is possible to specify a pattern that the word/annotation at the sort position must match. A random sort is also available.
- *Frequency breakdown*. The frequency breakdown of a query shows how often particular forms occur as the "hit" word (and/or as the primary annotation of the "hit" word). For example, after a query for words beginning with <n>, the frequency breakdown would show the frequency of each such word-form within that set of hits. Clicking on any form then produces a query thinned to just the matching subset of hits.
- *Multiple postprocesses*. The output of running a postprocess on a query result is itself a query result, to which further postprocesses can be applied. The history of the currently-active query result is shown on-screen at all times.

Any query can be saved within CQPweb, or downloaded to the user's computer as a plain-text table; the format of the download file is configurable.

Users can also access frequency lists — both for the corpus they are working with and for subcorpora they have defined; there are a number of methods for defining a subcorpus, including using categories from the metadatabase, or adding only texts that contain at least one result for a given query. Frequency lists for different subcorpora can be compared using the keywords function. This implements the standard keyness procedure of testing the difference in frequencies of each item on the lists for statistical significance (Scott 1996: 236, Rayson 2008: 527), in this case using the log-likelihood test. "Keywords" is something of a misnomer here, since in CQPweb the keyness procedure can be applied just as easily to any available stream of annotation. This is a generalisation of the BNCweb system, in which keyness can be calculated on words, word/POS combinations, or lemmata. Unlike BNCweb, CQPweb can have no built-in understanding of what annotations it makes sense to use for a keyness analysis; so some discrimination by the user may be necessary here.[7] A further point of generalisation in CQPweb is that keywords can be calculated *across corpora*. The frequency lists of a given corpus or subcorpus can be declared "public" by the system administrator; a public frequency list can then be used anywhere across the system.

CQPweb is both language-independent and writing-system-independent. The features listed above can be applied regardless of the language of the corpus. Writing-system-independence is achieved by use of Unicode (in the form of UTF-8) throughout, although the main ISO-8859 character sets are also supported for backward-compatibility. For scripts such as Chinese, where there are no explicit word breaks, tokenisation is a necessary prerequisite for indexing into CQPweb, since tokens are a primary unit of the CWB data model. To date, corpora in the Arabic, Bengali, Chinese, Cyrillic, Devanagari and Latin writing systems have been analysed using CQPweb. The concordance layout supports right-to-left text ordering for Arabic, Hebrew, or other such writing systems. For all scripts, the entering of text (such as when composing a query) and rendering of corpus data is the responsibility of the client browser. In general this means that CQPweb will work in the way the user is accustomed to from other applications they run, depending on their computer's operating system, available Unicode fonts, and input methods.

4.2 What the system administrator can do with CQPweb

For any given CQPweb installation one or more user accounts are designated as system administrators. Administrators are given behind-the-scenes control of the system — this is also accomplished via a web interface, laid out similarly to the query interface seen by the ordinary user. Most of the operations necessary to manage a CQPweb installation are accessible via this interface. They fall into two broad groups: managing corpora (including installing, customising, and deleting); and managing users (creating and deleting user accounts and managing their privileges). In general, while no compromises on usability have been made in the parts of CQPweb that the ordinary user sees, some compromises on user-friendliness have been made in this part of the system: the assumption is that system administrators will have a degree of technical expertise (for reasons explained in Section 5.2 below). For example, to index a corpus using the web interface, the administrator needs at least some understanding of the underlying data model (see Section 3.1) which the ordinary user does not need.

Indexing a corpus via the web interface requires one or more input files in the specified format, and an input file for the metadatabase. The nature of the annotations in the corpus data, and the design of the metadatabase, must be specified via the web forms that control indexing. Alternatively, corpora that have been indexed using CWB on the command line can be subsequently imported into CQPweb. Once a corpus has been set up, a range of configuration options are available to the administrator. For example, a corpus can be designated "invisible", which means it does not appear in CQPweb's main menu of corpora: users need to know the actual URL to get to it. Corpora can be assigned to categories, which are then used to split up the main menu. Moreover, the stylesheet used by a corpus can be set. Nearly all aspects of the visual appearance of CQPweb are specified by a CSS stylesheet. CQPweb comes with twelve different-coloured variants of the same basic stylesheet. But administrators can load additional CSS files of their choosing. This is not merely a matter of aesthetics. It is important for psychological reasons. Using a different stylesheet for each corpus on a CQPweb server is an aid for visual discrimination — it helps the user know what corpus they are working with at a given moment. If a user moves around among many corpora in a single session, this can be very useful.

Other options relate to the annotations of each corpus. Notably, the CEQL query language has become configurable in CQPweb. Originally, in BNCweb, the CEQL syntax of underscores and braces gave access to the POS tag, lemma and simple-tag annotations of the BNC. However, it is quite possible that a corpus might lack these particular annotations, and CQPweb does not assume that they are present. Instead, the administrator can configure CEQL on a per-corpus basis to link any available annotation (or none, if none are present) to each of the CEQL syntax shortcuts detailed above. Of course, to keep the interface consistent it is recommended that POS tag, lemma and simple-tag annotations be linked to the CEQL shortcuts following the BNC model *if they are present*. But this lies within the choice of the administrator.

The other main set of administrator tools relate to user account administration. Most of these functions currently rely on CQPweb running under the Apache web server. User accounts can be created or deleted, passwords set or reset, and limits placed or adjusted on the size of the temporary databases each user can create. Most importantly, the administrator has full control over users' rights of access to particular corpora. This allows user access to be adjusted according to the sensitivity of the corpus (in terms of copyright or licensing conditions). So, for instance, a corpus made up solely of public-domain texts (e.g. nineteenth-century fiction) would be low-sensitivity, and there would be no reason not to allow anyone with a user account to access it. But a corpus made available by its creators only to licensed users — for example, any of the datasets on the ICAME CD-ROM — could be restricted to accounts of persons whose institutions possess a licence.

No claims are made for the security of these access right limitations. CQPweb is in general only as secure as the web server it runs on. That said, assuming a reasonable security model on the server, CQPweb can be a suitable tool for affording limited access to moderately sensitive data. When a corpus cannot be legally redistributed at all, allowing access to the data through a limited-context web-based concordancer is the only way that other researchers can be allowed to study the corpus — a practice with obvious benefits in terms of permitting the replication of research results. Because CQPweb is flexible enough to support many different (kinds of) corpora, it can function as a suitable off-the-shelf tool for providing a web interface of this sort. The expense in time and effort of developing a new tailor-made interface for each corpus to be made available via the web can thus be avoided.

5. Strengths and limitations of CQPweb

5.1 Evaluating CQPweb

Before evaluating CQPweb, it is necessary to decide on what basis it *should* be evaluated. One obvious route for evaluation is performance, as measured, for instance, by the time required to perform a certain set of (complex) queries. This is the approach taken by Mark Davies (n.d.) in evaluating the corpus.byu.edu architecture (Davies 2005, Davies 2009, Davies 2010; see also Section 2.1 above), a fourth-generation concordance system with many of the same goals as CQPweb in terms of usability, flexibility, and power. This approach to evaluation is clearly appropriate for Davies' software, which runs only on one system, namely the corpus.byu.edu server. But CQPweb is, by contrast, openly-available software intended to be installed on many different computers, from laptops to high-powered servers. The speed with which a query runs is only partly dependent on the software; it also depends on hardware. An evaluation of performance, thus defined, that is based on any single installation is therefore not likely to be meaningful for the same software on another computer. If this kind of performance-based evaluation is not productive, how might CQPweb be meaningfully evaluated? One possibility is to compare what it is capable of, to forecasts made in the past of what a future corpus analysis tool *should* be capable of. The forecast most relevant to CQPweb is that by Hoffmann and Evert.

In the article that introduces the CQP-edition of BNCweb, Hoffmann & Evert (2006: 191–194) also lay out a "white paper" for a proposed future corpus tool which they label CORPORAweb or Cweb for short. Cweb is envisaged as a tool similar to BNCweb, but capable of working with any corpus — that is, the primary advance envisioned is identical to the primary novel feature of CQPweb. CQPweb's capabilities can therefore be evaluated by examining to what degree it fulfils the requirements for Cweb laid out in Hoffmann and Evert's "white paper".

1. *Multiple corpora*. The primary requirement for Cweb was that it should support "a broad range of (text) corpora, provided that their structure is reasonably

similar to that of the BNC" (Hoffmann & Evert 2006: 191). CQPweb more than meets this requirement, as corpora are not required to be similar to the BNC in any particular way.

2. *TEI compatibility*. Hoffmann & Evert (2006: 191) "envisage Cweb to be compatible with any corpus that is encoded in an XML format and whose structure conforms to the TEI". CQPweb partially meets this requirement. The XML format of a TEI (Text Encoding Initiative) corpus cannot be indexed into CQPweb directly: in TEI tokens are indicated by <w> elements (or not at all) rather than as lines, and metadata is stored in the header of each file rather than as a separate table. There are, as yet, no tools built in to CQPweb for mapping from TEI format to CQPweb input format. However, it is computationally trivial to generate suitable input files from a TEI-compliant corpus, perhaps using XSLT.
3. *Easy corpus setup*. Hoffman & Evert (2006: 192) suggest that "[e]ven users without programming or system administration skills should be able to configure Cweb for use with a new corpus". CQPweb mostly fulfils this requirement, as corpus setup is accomplished via the web interface, although creating the input files may require programming expertise. However, CQPweb falls short by restricting corpus-setup functions to the administrator-user(s); I will return to this point in Section 5.2 below. Hoffmann and Evert also suggest that users should be able to specify which XML elements are displayed in concordances; this feature is under development for CQPweb but not yet complete, but again this functionality will be configurable by administrators only, due to its complexity.
4. *Simple query language*. CQPweb fulfils this requirement by incorporating CEQL; but Hoffmann & Evert (2006: 192) also recommend that "there should be a smooth migration path from basic simplified queries over an extended syntax […] to the full-fledged CQP language" which is not the case in CQPweb; the choice is straightforwardly between CEQL and CQP-syntax with no intermediate language.
5. *Compatibility with BNCweb*. Closely following the user interface and affordances of BNCweb is a requirement that CQPweb fulfils in full. However, one extra function proposed for Cweb (the ability to create templates of multiple postprocesses that can be activated as single actions) does not yet exist in CQPweb.
6. *User management*. Hoffmann & Evert (2006: 193) say that there should be "a convenient way of adding new users and setting individual access restrictions"; CQPweb fulfils this requirement, as outlined in Section 4.2.
7. *Client/server architecture on any platform*. CQPweb by its very nature replicates BNCweb's client/server model as per Hoffmann and Evert's recommendations,

and it can be installed on a desktop computer, as they suggest, though a Unix-like environment is required for a CQPweb server. So at present CQPweb mostly but not fully meets this requirement.

8. *Modular architecture*. Hoffmann & Evert (2006: 193–194) suggest that if its code has a sufficiently modular architecture, technically sophisticated users will be able to customise Cweb; the customisations they suggest are user-written XSLT stylesheets to determine the format of data displays, or user-written query language parsers. Although it has been designed in as modular a style as possible, CQPweb permits neither of these customisations.

Arguably then, CQPweb fulfils three of Hoffmann and Evert's criteria (1, 6 and 7) in full or nearly in full; four in part (2, 3, 4 and 5); and one criterion (8) is not fulfilled at all. One feature not mentioned by Hoffmann and Evert is scalability, that is, the ability to work with very large corpora without major reductions in speed; this is an aspect of the *power* of a concordancer (see Section 2.1). CQPweb does perform fairly well on this score, querying corpora of hundreds of millions of words without much loss of speed. In fact, speed falls in line not with the growth of the corpus itself, but rather with the growth of certain associated database tables. For instance, as the number of texts in the corpus — and thus the number of entries in the metadatabase — grows, many CQPweb functions that reference text metadata become slower. Taking measures to optimise queries on the metadatabase may address this weakness.

So, evaluated against Hoffmann and Evert's white paper, CQPweb does rather well. However, there are other bases of evaluation against which CQPweb performs less well. For instance, CQPweb might be judged on the extent to which it has added *new* corpus analysis procedures to the toolbox that is available via user-friendly concordancers. By this criterion CQPweb fares rather badly: frequency lists, concordancing, collocations, keywords, thinning, sorting and so on are all long-established techniques. Yet arguably this criterion is the most important of all; as McEnery & Hardie (2012: 41–43) point out, many analytic techniques have been developed and described in the literature but are not available to the non-programmer because no user-friendly corpus analysis tools integrates them. For example, the 'multi-dimensional' analysis of Biber (1988), the 'collocational networks' technique of Phillips (1989), and the 'collostructional' analysis of Stefanowitsch & Gries (2003) are all widely known and influential methods, but no current concordancer supports them.[8] CQPweb does not yet do anything to address this state of affairs.

5.2 Evaluating flexibility

Flexibility (as defined in Section 2.2) is CQPweb's main advance on its predecessor BNCweb. The degree to which CQPweb succeeds in combining flexibility with the usability and power of other fourth-generation tools is therefore worthy of particular consideration. In the sense I have considered it in this paper, *flexibility* is primarily the possibility of using a tool with any corpus. CQPweb's data model (see Section 3.1) gives it this flexibility. It is capable of representing internally a corpus with any amount of any type of word-level annotation, or none; with any amount of text-level metadata, or none; in any language and any writing system. Moreover, the web interface includes specific affordances for dealing with that data model in the corpus indexing process (Section see 4.2). On these grounds, CQPweb may be deemed an extremely flexible concordancer.

However, there are some important caveats here. Firstly, because the flexibility emerges from the data model, it is necessary for anyone administering a CQPweb server — but not the end-users — to have a reasonable understanding of that data model, as well as the technical skills to convert the corpus (meta-)data to be indexed into the appropriate input formats. So, the price of flexibility is added complexity for the administrator. In practice, this is not a serious impediment because running a CQPweb server is already a technically complex task for other reasons. Namely, setting up the various systems on which a CQPweb server depends requires a substantial degree of technical know-how: these dependencies include a Unix-like operating system (on Windows, the Cygwin environment is required; future development will remove this dependency), the MySQL database server, the PHP and Perl languages, and of course CWB.

Secondly, flexibility for the system administrator — who can set up corpora — is not the same thing as flexibility for the end-user — who cannot. This is a feature CQPweb shares with most fourth-generation concordancers and one that makes them, from the point of view of the user, less flexible than the less-powerful third-generation desktop-based systems. It is certainly not impossible in principle for ordinary users to be able to index their own data in a system like CQPweb, and enhancements to make this possible are currently at the planning stage. However, there are numerous practical hurdles to overcome: how is the format of non-expert users' input data to be validated? How can the CQPweb data model be presented to users in a way that is accessible to them? How can the use of system resources be effectively managed across potentially very many users, given that indexing a large corpus can lock up a great deal of disk space and processing power? The generalisability of whatever solution is adopted will also be crucial here. Wmatrix (Rayson 2008), the current fourth-generation system most oriented towards users' own data, offers a single tool-chain consisting of a POS tagger and semantic tagger for English. By contrast, CQPweb's language-independence will require all such toolchains to be wholly configurable. This will necessarily increase the complexity of CQPweb's dependencies on other software.

Finally, the dimension of flexibility on which I have placed greatest weight in this paper is generalisability across corpora. But there exist other aspects of flexibility.[9] For example, no client/server system can ever offer the level of total control over the process of analysis that the use of a specially-written program does; the need for this type of flexibility underlies the recommendations cited in Section 2.1 that corpus linguists should learn (at least some) computer programming. Another kind of analytic flexibility is that available to a researcher who has direct, local access to the full, plain text of a corpus, allowing them to examine, move around, or manipulate the data at will. Neither of these forms of flexibility is targeted by CQPweb, but they are important affordances in many kinds of research nonetheless.

5.3 Other limitations and planned developments

CQPweb continues to develop; some indications of planned enhancements have been given in the preceding discussion, and adding new analysis techniques to the toolbox it makes available will surely also form part of this future work. Some further existing limitations and planned developments are worth mentioning.[10]

CQPweb naturally inherits the limitations of the underlying CWB platform. The most serious of these is the maximum corpus size, which is 2.1 billion words on a 64-bit system (less on older 32-bit systems: see Evert & Hardie 2011). This limit is imposed by the index format, which will be revised in future versions to remove the limit. Beyond this issue, there are two clear areas where CQPweb is currently lacking, and a range of other possible avenues for enhancement.

The online user documentation ("help pages") in CQPweb is still very limited. This is something of a flaw in a tool that aims at a high degree of usability, but it has not been a major problem to date, because CQPweb's main group of target users are usually accustomed to the better-documented BNCweb and thus do not need detailed help. However, as time goes on there will be more CQPweb users who have not migrated from BNCweb, and better online help will be an absolute necessity. Ideally, help pages would be dynamically generated so that their content can be customised to the active corpus.

CQPweb's support for XML annotation in the underlying corpus data is currently rather weak. XML elements such as <s> tags for sentences can be indexed, and then referred to in CQP-syntax queries. But the database does not keep track of what XML elements exist, nor is their metadata represented in the data model. Nor can XML elements be rendered in the concordance or extended context views. Amending these issues is the most urgent enhancement required for future versions of CQPweb. The administrator will be able to configure XML visualisations that, when active, will generate a customisable representation of a given XML element at the point where it occurs. So, for example, it will be possible to create a visualisation for a time-alignment point in a spoken text, encoded as XML, that would manifest the time-code as a "link out" to a sound-file for that part of the text stored somewhere on the web. Likewise, once XML regions and annotations are represented in the data model, it will be possible to restrict queries to XML-delimited subsets of the corpus in the same way as it is possible, in BNCweb, to restrict a query to utterances whose speakers meet certain demographic criteria.

Some other enhancements are already under development. Two worth mentioning are: (a) the addition of an interface to the R statistical software, to exploit R's unparalleled capabilities for statistical analysis and graphs; and (b) a reworking of the concordance-rendering module to support the analysis of morphologically-glossed field-linguistic data in the classic "three-line-example" format. Other planned extensions remain to be implemented: support for concordancing across parallel corpora; support for visualising dispersion of the hits in a query result; support for the analysis of word or annotation n-grams; extending the collocation function to collocation by grammatical relation (e.g. collocations between head noun and modifying adjective, or between verb and object) as well as by proximity; support for indexing and querying dependency-parsed or constituency-parsed corpora; and, finally, support for programmatic interaction with CQPweb. This last feature will make it possible to create customised web-tools with CQPweb as their back-end. This will address, to some degree, Hoffmann and Evert's (2006) modularity criterion.

In contemplating such future developments, there is an ever-present tension between extending CQPweb and maintaining compatibility with BNCweb. Obviously, the interface must be adjusted to accommodate any new features. Yet if the interface changes too much, then the original virtue of CQPweb — that it replicates a tool that many users are familiar with — is diluted or lost. This is, in essence, the same tension as exists for all corpus analysis tools between power and usability. Though CQPweb goes a long way towards satisfying both demands, it ultimately does not escape the dilemma entirely.

6. Summary

CQPweb has a number of advantages as a tool for corpus analysis. Like BNCweb it combines *power* and *usability* by bringing CWB/CQP together with relational database functions within a client/server model, providing a simple but powerful query language and a range of useful query postprocesses. Unlike BNCweb, it does all this *flexibly*, so that any corpus can be indexed into the system. By emulating BNCweb's user interface, CQPweb achieves *familiarity* for a great many target users — most notably, students and other novice corpus analysts; this is a major factor in its user-friendliness. For teaching purposes it is a particular advantage that methodological skills learned using BNCweb are immediately transferable to CQPweb. The system is also *scalable*, working effectively with corpora whose size is on the order of hundreds of millions of words.

While I have attempted in this paper to turn a critical (and evaluative) eye on CQPweb as well as extolling its virtues, I would ultimately argue that simply by making a sophisticated corpus query system accessible to the non-technical user via the web, CQPweb simultaneously meets the two main requirements of a corpus analysis tool — power and usability — to a very high degree. This is not to deny that CQPweb also has weaknesses and limitations, many of which have been discussed in Section 5. In particular, while very flexible in terms of the range of corpora it can work with, CQPweb falls short in other aspects of flexibility, as detailed in Section 5.2, especially from the non-administrator user's point of view. Ultimately, so long as corpus analysts are faced with competing and irreconcilable technical and methodological requirements, which vary depending on exactly what they want to accomplish and on what data is to be used, then the choice of corpus software must always be a compromise; it is thus extremely useful for a range of software options to be available. If nothing else, then, CQPweb surely represents a useful addition to the range of compromises currently available to the corpus analyst.

Notes

1. Many server programs have names ending in <d> for 'daemon'; 'daemon' is another term for a server program.

2. The public interfaces in question are http://korpus.ia.uni.lodz.pl for PELRCA and http://hnc.ilsp.gr/en/ for the Hellenic National Corpus. Related, but distinct, are web-based query systems which query the World Wide Web, using a web search engine as their back-end; such systems are beyond the scope of this paper, but see, for example, Renouf (2003).

3. BNCweb was more than merely a model. The process of creating the CQPweb system benefitted at critical points from the good advice of the creators of BNCweb (CQP edition), Sebastian Hoffmann and Stefan Evert.

4. A "C word" is a label that complies with the rules for identifiers in C and similar programming languages. It must consist *only* of unaccented upper- or lowercase letters, Latin-alphabet digits, and the underscore character. For technical reasons, all "handles" in CQPweb must be C words.

5. CQPweb treats all text as UTF-8 internally, but can translate underlying ISO-8859 text to UTF-8 on the fly. If CQPweb is used with a version of CWB prior to 3.2, UTF-8 regular expressions are not fully supported.

6. The NN2 tag indicates a plural noun (distinguishing plural nouns from third person singular verbs when searching for a word like *dogs* is one very useful application of part-of-speech tags).

7. For instance, in some of the non-English corpora indexed on Lancaster University's CQPweb server, one annotation is an English gloss of each word; a keyness analysis on this annotation would not be meaningful.

8. However, SketchEngine's 'word sketches' approximate collostructional analysis to some degree.

9. I gratefully acknowledge the insight of an anonymous reviewer who suggested this point.

10. At the time of writing, CQPweb is at version 3.0.5; version numbers higher than this may be expected to implement some or all of the features here designated as developments for 'the future'.

References

Abercrombie, D. 1965. "Pseudo-procedures in linguistics". In D. Abercrombie (Ed.), *Studies in Phonetics and Linguistics*. Oxford: Oxford University Press, 114–119.

Anthony, L. 2005. "AntConc: A learner and classroom friendly, multi-platform corpus analysis toolkit". In *Proceedings of IWLeL 2004: An Interactive Workshop on Language e-Learning*. Tokyo: Waseda University, 7–13.

Aston, G. & Burnard, L. 1998. *The BNC Handbook: Exploring the British National Corpus with SARA*. Edinburgh: Edinburgh University Press.

Baker, P. 2009. "The BE06 Corpus of British English and recent language change". *International Journal of Corpus Linguistics*, 14 (3), 312–337.

Barlow, M. 2000. *MonoConc Pro*. Houston, TX: Athelstan.

Biber, D. 1988. *Variation across Speech and Writing*. Cambridge: Cambridge University Press.

Biber, D., Conrad, S. & Reppen, R. 1998. *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge: Cambridge University Press.

Chandler, B. 1989. *Longman Mini Concordancer*. Harlow: Longman.

Christ, O. 1994. "A modular and flexible architecture for an integrated corpus query system". In *Papers in Computational Lexicography (COMPLEX '94)*, Budapest, Hungary, 22–32.

Davies, M. 2005. "The advantage of using relational databases for large corpora: Speed, advanced queries and unlimited annotation". *International Journal of Corpus Linguistics*, 10 (3), 307–334.

Davies, M. 2009. "The 385+ million word corpus of contemporary American English (1990–2008+): Design, architecture and linguistic insights". *International Journal of Corpus Linguistics*, 14 (2), 159–190.

Davies, M. 2010. "More than a peephole: Using large and diverse online corpora". *International Journal of Corpus Linguistics*, 15 (3), 412–418.

Davies, M. n.d.: online. *Comparison of the BYU Corpus Architecture, Corpus Workbench (SketchEngine), and Corpus Workbench (BNCweb)*. Available at: http://corpus.byu.edu/architecture.asp (accessed December 2010).

Evert, S. & Hardie, A. 2011: online. "Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium". In *Proceedings of Corpus Linguistics 2011*. Available at: http://www.birmingham.ac.uk/research/activity/corpus/publications/conference-archives/2011-birmingham.aspx (accessed November 2012).

Garside, R., Leech, G. & Sampson, G. (Eds.). 1987. *The Computational Analysis of English*. London: Longman.

Gries, S. Th. 2010. "Methodological skills in corpus linguistics: A polemic and some pointers towards quantitative methods". In T. Harris & M. M. Jaén (Eds.), *Corpus Linguistics in Language Teaching*. Frankfurt am Main: Peter Lang, 121–146.

Hockey, S. 1988. *Micro-OCP (OCP Version 2)*. Oxford: Oxford University Press.

Hoffmann, S. & Evert, S. 2006. "BNCweb (CQP-Edition) — the marriage of two corpus tools". In S. Braun, K. Kohn & J. Mukherjee (Eds.), *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*. Frankfurt am Main: Peter Lang, 177–195.

Hoffmann, S., Evert, S., Smith, N., Lee, D. Y. W. & Berglund Prytz, Y. 2008. *Corpus Lnguistics with BNCWeb — a Practical Guide*. Frankfurt am Main: Peter Lang.

Hundt, M., Sand, A. & Siemund, R. 1998: online. *Manual of Information to Accompany the Freiburg–LOB Corpus of British English ('FLOB')*. Englisches Seminar. Freiburg: Albert-Ludwigs-Universität Freiburg. Available at: http://khnt.hit.uib.no/icame/manuals/flob/index.htm (accessed November 2012).

Kaye, G. 1990. "A corpus-builder and real time concordance browser for an IBM PC". In J. Aarts & W. Meijs (Eds.), *Theory and Practice in Corpus Linguistics*. Amsterdam: Rodopi, 137–162.

Kilgarriff, A., Rychly, P., Smrz, P. & Tugwell, D. 2004. "The Sketch Engine". In G. Williams & S. Vessier (Eds.), *Proceedings of Euralex 2004*. Bretagne, France: Université de Bretagne-Sud, 105–116.

Lehmann, H-M., Schneider, P. & Hoffmann, S. 2000. "BNCweb". In J. Kirk (Ed.), *Corpora Galore: Analysis and Techniques in Describing English*. Amsterdam: Rodopi, 259–266.

Mason, O. 2001. *Programming for Corpus Linguistics*. Edinburgh: Edinburgh University Press.

McEnery, T. & Hardie, A. 2012. *Corpus Linguistics: Method, Theory and Practice*. Cambridge: Cambridge University Press.

Phillips, M. 1989. *Lexical Structure of Text*. Birmingham: University of Birmingham.

Rayson, P. 2008. "From key words to key semantic domains". *International Journal of Corpus Linguistics*, 13 (4), 519–549.

Rayson, P., Archer, D., Piao, S. L. & McEnery, T. 2004. "The UCREL semantic analysis system". In *Proceedings of the Workshop on Beyond Named Entity Recognition Semantic Labelling for NLP Tasks, in Association with 4th International Conference on Language Resources and Evaluation (LREC 2004)*, 25th May 2004, Lisbon, Portugal, 7–12.

Renouf, A. 2003. "WebCorp: Providing a renewable data source for corpus linguists". In S. Granger & S. Petch-Tyson (Eds.), *Extending the Scope of Corpus-Based Research: New Applications, New Challenges*. Amsterdam: Rodopi, 39–58.

Scott, M. 1996. *WordSmith Tools*. Oxford: Oxford University Press.

Smith, N., Hoffmann, S. & Rayson, P. 2008. "Corpus tools and methods, today and tomorrow: Incorporating linguists' manual annotations". *Literary and Linguistic Computing*, 23 (2), 163–180.

Stefanowitsch, A. & Gries, S. Th. 2003. "Collostructions: Investigating the interaction between words and constructions". *International Journal of Corpus Linguistics*, 8 (2), 209–243.

Uzar, R., Pęzik, P. & Levin, E. 2004. "Developing relational databases for corpus linguistics". In B. Lewandowska-Tomaszczyk (Ed.), *Practical Applications in Language and Computers: PALC 2003*. Frankfurt am Main: Peter Lang, 93–104.

Weisser, M. 2009. *Essential Programming for Linguistics*. Edinburgh: Edinburgh University Press.

Author's address

Andrew Hardie
Department of Linguistics and English Language
Lancaster University
Lancaster
LA1 4YL
United Kingdom

a.hardie@lancaster.ac.uk