

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

Andrew Hardie
Lancaster University
a.hardie@lancaster.ac.uk

Abstract

CQPweb is a new web-based corpus analysis system. Its main purpose is to address the tension between the requirements for usability and power in corpus analysis software. To do this, CQPweb emulates the widely-used BNCweb interface to the British National Corpus. Like BNCweb, CQPweb uses a client/server model; the client software is a straightforward web browser, while the webpage-based interface gives mediated access to more powerful but less user-friendly software on the server. Like BNCweb, CQPweb is built on two separate query technologies. The IMS Open Corpus Workbench (CWB) and its powerful Corpus Query Processor (CQP) are used, respectively, to index corpora, and to retrieve solutions from the corpus data using a powerful and flexible query formalism with full regular-expression support. Alongside CWB and CQP, the MySQL relational database system is used to manage textual metadata and to implement query postprocesses such as concordance sorting and the calculation of collocations. CQPweb’s main innovative feature relative to BNCweb is that it is compatible with any corpus, rather than being bound to a particular dataset; this is made possible by its more generalised data model, in which no assumptions are made about the form of the data in any particular corpus.

The analysis options made available to the ordinary user in CQPweb include concordancing with either CQP syntax or the simplified CEQL syntax; extraction of collocations; creation of distribution tables and charts; analysis of user-defined corpora via frequency lists and keywords or key tags; and other functions. However CQPweb has also been designed to assist the system administrator in the management of corpus data (including the process of indexing corpora into CQPweb) and of user accounts (including managing users’ varying access rights to different corpora on the system).

An evaluation of CQPweb against criteria that were 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; the system clearly is possessed of limitations, most notably that non-administrator users cannot add their own data to the system, and that while it provides powerful functionality, it as yet does little to extend the usual toolbox of concordances, frequency lists, collocations and keywords that most or all concordancers offer. That said, however, by virtue of making a sophisticated and powerful query system (i.e. CQP with added features from the relational database) accessible to the untrained user through a web-based interface, CQPweb combines ease of use, power and flexibility to a very high degree.

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 processing corpus data by carrying out concordance queries and performing 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 that let us explore research questions which might otherwise be almost unimaginable. 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 and Hardie 2011). 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 and Evert 2006, Hoffmann et al. 2008). CQPweb’s main innovative feature relative to BNCweb is 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)¹ and the MySQL relational database system,² 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, arguing that the current ‘fourth generation’ of concordancers (of which BNCweb and CQPweb are both representatives) have made the greatest progress to date to provide both power and usability at the same time. However, as outlined in section 2.2, despite its impressive range of capabilities BNCweb lacks *flexibility* in the sense that it is bound to a single corpus – the limitation that CQPweb seeks to overcome. The data model used by CQPweb, which allows cross-corpus flexibility to be achieved 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 and Evert (2006), but also considering the system’s present limitations and giving some indication of additional functionality planned for future versions of the software.

2. Corpus analysis tools: some background

2.1. Usability versus power

The development of corpus linguistics as a distinct practice within linguistics and language studies has arguably been both enabled and constrained by the software tools available to corpus analysts at different points in time (see McEnery and Hardie 2011: 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.

The amount of corpus-based research being undertaken expanded drastically in the late 1980s and early 1990s. This boom was driven largely by newly available PC concordance software such as the Kaye concordancer (Kaye 1990), the Longman Mini Concordancer (Chandler 1989) and Micro-OCP (Hockey 1988). The popularity and strong democratising effect of these concordancers – which McEnery and Hardie refer to as ‘second-generation’, following

¹ <http://cwb.sourceforge.net>

² <http://www.mysql.com>

the ‘first generation’ of largely mainframe-based analysis software – was in turn enabled by the spread of the IBM-compatible PC. This second generation, and later the 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 about how computers work. For this reason, these second- and third-generation concordancers run on a desktop computer under a Windows or Macintosh operating system, rather than requiring remote login to a server; and are controlled via a graphical user interface (GUI) similar in principle 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 (at least) two perspectives. First, a powerful tool may be defined as one that can query a large or very large corpus (on the order of tens or hundreds of millions of words), and that moreover can do so efficiently, i.e. within a practical time span. Query speed is in part dependent on the computer hardware being used, of course. But even on high-end hardware, to achieve this kind of speed on very large datasets, a program must generally work with *indexed* corpus data rather than the actual 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 for complex and sophisticated queries, especially queries that go beyond just searching for particular strings of characters in the data files of a corpus. For example, a tool where annotations such as grammatical tags or text headers can be queried via an abstract data model can 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 those that are *less* user-friendly. Perhaps the paradigmatic example of this tendency is the IMS Open Corpus Workbench (CWB; see Christ 1994) and its concordancer, the Corpus Query Processor (CQP). CQP is extremely powerful: it can search very large corpora very quickly, and has a query language which is flexible and sophisticated in the kinds of search criteria it can describe. But CQP is a command-line program which, when running, is controlled by typing commands into a query parser; when a query has run, its results are simply printed to the command line (perhaps via a pager program). It is thus a rather intimidating and strange-seeming piece of software for users who are not familiar with the use of command-line tools and with regular-expression query languages – that is, for the very audience of beginners and non-technical-specialists so successfully reached by desktop query tools that have focused on usability.

Of course, a technically-savvy user who is familiar with the environment of a Unix-like command-line, and associated search programs such as *grep*,³ would not find CQP anywhere near so unfriendly. But the overwhelming majority of linguists, or of other scholars who might be interested in using corpus methods, are in the group of non-technical users who would find CQP unapproachable. While CQP is a particularly good example of a non-user-

³ *grep* is a command-line utility available by default on Unix and similar operating systems such as Linux or Mac OS X. It searches through one or more text files for a given string (which may be defined in regular-expression syntax), and prints out every line that it finds containing one or more instances of the search target. It can be seen as, in effect, a no-frills concordancer.

friendly system, there do naturally exist other approaches or tools that require very high degrees of technical know-how from the user. 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 on that data much faster. But this indexing process is typically outside the range of operations understood by the average beginner or 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 have recommended 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 and query 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. While nowadays the statistical programming language R is often promoted (especially by Gries 2009a, 2009b) as suitable for linguists who need to write such programs, other languages such as Perl (Weisser 2009), Python or PHP, or even C, C++ or Java (Mason 2001), can equally well be used for this purpose. Clearly this ‘do-it-yourself’ approach to corpus analysis software is the most powerful imaginable *if* power is equated to flexibility and adaptability – although not necessarily if power is equivalent to speed and efficacy in very large corpora; ‘do-it-yourself’ programs may run slowly if they do not incorporate the ‘tricks’, such as data indexing, needed for high speed on large datasets. However, equally clearly, this approach is completely lacking in usability for the very large number of potential corpus analysts, within linguistics and more broadly in the humanities and social sciences, who either cannot or do not wish to learn computer programming. It should be very 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).⁴

For such researchers, usability will always be the more critical of the two factors. But we should not underestimate the problematic nature of a decision to privilege 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 and Hardie (2011) argue,

[T]he corpus alone solves few (if any) problems for a linguist. Its potential 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. Over time, an increasing number of tools have become available, and this has expanded the range of research questions that may be addressed using a corpus. But a range of research questions do still lie beyond the reach of what can be done with a corpus at the present time.

An analyst whose toolbox is limited to the power of a GUI-based desktop concordance package is therefore also limited in the questions they can ask. 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 (and hardware) has the power to deal

⁴ My view as stated here is informed by several years of teaching introductory corpus linguistics to undergraduate students in the UK. The experience of those whose teaching has been in other educational systems or with students at other levels may differ, naturally.

with. Of course much valuable work can be done, and has been done, working with small corpora. But only a subset of interesting research questions can be addressed in this way.

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, two separate programs are used. The *client* program is responsible for interacting with the user. The user composes their query in the client software, but the client does not actually search the corpus: instead the query is communicated to the *server* program which is responsible for running the query. It is possible for the client and server programs to run on the same computer – but they do not have to. Instead, they can be set up on different computers and communicate across a network or the Internet. For this reason, it is possible for the client to provide a user-friendly GUI on a simple 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 version of the BNC (Aston and Burnard 1998) operates on exactly this model, consisting of a server program called *sarad*⁵ which most users never needed to deal with directly, together with the SARA client for Windows. Xaira, the successor to SARA,⁶ has the same architecture. However, more recently, in what McEnery and Hardie (2011) dub the ‘fourth generation’ of concordancers, the client/server model has been used across the specific medium of the World Wide Web. In this case, the client software is in fact no more than an interactive web page, rather than a program that must be installed on the user’s local computer. 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, is run by the web server, and its output is returned to the client computer in the form of a page of HTML that can be rendered in the user’s browser.

Fourth-generation concordancers have been developed and deployed for a number of reasons. In some cases, they 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 – all that is needed is sufficient power to render the results in a browser, which most machines have. Likewise, because of the move to the web, fourth-generation tools automatically run on every operating system – unlike many third-generation tools which were (and are) 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 – except, of course, the web browser, but most desktop computers come with at least one browser pre-installed. Furthermore, using one or more web forms as the primary user interface gives fourth-generation tools an instant advantage in usability over even the friendliest of third-generation tools, because even non-technically-aware users are very likely, in the modern world, to make regular use of web forms for a wide range of activities, from email to online banking to social networking sites. As a result of all these factors, fourth-generation concordancers considered *en masse*

⁵ Many server programs have names ending in <d> for *daemon*; *daemon* is another term for a server program.

⁶ <http://xaira.sourceforge.net>

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, 2009, 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 or the Hellenic National Corpus is via such an interface.⁷ 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 a powerful corpus indexing and query system such as CWB/CQP or the Xaira server program. Relational databases are of course 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 table are then searched by converting the user’s query request to a database search language such as the *Structured Query Language* (SQL); the database results are then returned in the format of a concordance, frequency list or other standard corpus analysis output. The PELCRA interface is SQL-based, for instance (Uzar et al. 2003). However, 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, a large number of ‘one-off’ corpus access tools have used CWB/CQP as their back-end; one system that is more generalised is Serge Sharoff’s ‘Leeds CQP’ tool⁸ via which a range of very large corpora in different languages are made available. Likewise the back-end of SketchEngine is a CWB/CQP-compatible program called Manatee (Kilgarriff et al. 2004). But 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 original version of BNCweb (Lehmann et al. 2000) was one of the earliest fourth-generation concordancers – that is, one of the first to apply the client/server model across the web. Its back-end for performing actual corpus searches in the British National Corpus was the SARA server. However, BNCweb from the outset provided additional power (in the range-of-possible-analyses sense) by using a relational database to implement functions such as: a tool to view the distribution of a query’s results across text categories as given by the BNC’s file headers; the ability to sort concordances by an adjacent word or part-of-speech (POS) tag; and a highly-configurable system for extracting collocations from a concordance via one of a range of statistical measures. Each of these functions operated by reformatting some output of the SARA concordance query into an SQL database table on the fly; this new table could then be queried to produce results for the user.

As is typical for any complex software system, BNCweb has evolved significantly over time, reflecting both developments in the software and updates to allow it to work with subsequent

⁷ See <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 do live queries of (some part of) the World Wide Web, using a web search engine such as Google as their back-end; such systems are beyond the scope of this paper, but see, for example, Renouf (2003).

⁸ <http://corpus.leeds.ac.uk>

versions of the BNC (most recently the XML edition released in 2007).⁹ The biggest change has been a change in the back-end from *sarad* to CWB/CQP, described by Hoffmann and Evert (2006). Although Hoffmann and Evert describe moderately substantial changes to the internal working of BNCweb, from an outside perspective, the most notable characteristic of this change was how slight an effect it had on the interface seen by the average user. Apart from this major change, there have also been many more incremental advances as new features are added or existing features refined. An example is given by Smith et al. (2008), who detail the addition of 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); it suffices here to give an overview of its scope. As noted above, concordances are provided via CQP. But users do not have to contend with the full complexity of CQP query syntax, although that complexity is optionally available. Rather, most queries are done by means of a simplified query language dubbed the 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. For instance, all CQP search strings are regular expressions, whereas CEQL makes a subset of the regular-expression syntax available in the form of simplified wildcards such as `<?>` for ‘any one character’ or `<*>` for ‘any string of characters’. A query which imposes conditions on both wordform and on the BNC’s C5 part-of-speech annotation¹⁰ would have the following form in CQP:

```
[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
```

The CEQL syntax also gives access to the lemmata encoded for each word in the BNC files, which are queried by surrounding the lemma in {braces}, and to the simple POS tags (indicating major wordclass only without any morphosyntactic detail) that have been added alongside the original C5 tags in more recent versions of the corpus. Another simplification for the end-user is found in BNCweb’s interface to the metadata in the BNC’s TEI-conformant file headers. A query can be restricted to particular genres or text-types in the written corpus, or to particular social categories of speaker in the spoken corpus, using a straightforward checkbox-based web form.

As well as the distribution, sorting and collocation functions described above, queries can be thinned randomly, sorted randomly, or manually annotated within the BNCweb interface. Queries can be saved within BNCweb or downloaded in plain-text format. Furthermore, the user has access to overall corpus frequency lists, as well as frequency lists for subcorpora they have defined themselves. Keyword lists can also be calculated by comparing these subcorpus frequency lists. There is in addition a tool for browsing the text of any BNC file. Users also have available to them a complete history of their own queries. Finally, all queries

⁹ <http://www.natcorp.ox.ac.uk/XMLedition>

¹⁰ See <http://ucrel.lancs.ac.uk/claws5tags.html>. 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).

are saved in a cache, so that if the same query is run again, it is simply retrieved from the cache. This adds a great deal to the speed of BNCweb, for instance, if it is being used for a practical teaching session, where a group of several students are likely to be running very similar queries multiple times within the space of an hour or two. All these features are supported by a combination of CWB/CQP with the SQL database.

As the foregoing overview should make clear, BNCweb provides both a high degree of power – both in the sense of speed in querying a 100 million word corpus, and in the sense of the wide range of functions and flexibility of the query language – and a high degree of usability through the web-based user interface. It is also notable in that, unlike many other fourth-generation concordancers, it has been released under a free/open-source software licence (the GNU General Public Licence¹¹), allowing anyone with a copy of the BNC to install and run it on their own computer or website. As a result it is very widely used.¹² BNCweb is, then, a very successful system, accomplishing exactly what it was intended to as an interface to the BNC. What it does *not* provide is the flexibility to work with many different datasets.

BNCweb is tightly bound to the BNC in a number of ways. Usually, this takes the form of 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, the part of BNCweb that manages the corpus frequency lists held in the database ‘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. The system also ‘knows’ that the BNC is primarily divided into spoken and written sections and that separate frequency lists for each will be needed at all times. It ‘knows’ that the POS tags are usually considered the most important word-level annotation in the BNC, and so POS tags are visualised within the concordance display, and can be used as a filter on sorting, lemmata are not visualised and cannot be so used. Features on the BNC metadata and tagsets are also ‘known’. At places in the code where the user can select a value for part-of-speech, the structure of the C5 tagset is built into the code. The code that generates the web form containing the spoken and written query restrictions has knowledge built into it of what the relevant metatextual categories are. BNCweb likewise ‘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’ about the corpus it analyses are things that 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 understandable 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 (a pedagogical strategy which I and many others have followed with undergraduates on Linguistics and/or English Language degrees). If, subsequently, the demands of the course they are studying 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

¹¹ <http://www.gnu.org/licenses/gpl.html>

¹² At time of writing, the installation of BNCweb at Lancaster University alone has 5,048 registered users.

time re-acclimatising themselves 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. All these practical skills must be learnt, in many cases, from scratch; the average undergraduate course in corpus linguistics is too short for a newcomer to the field to get under their skin the methodological know-how which would let them easily adapt to any corpus tool.

From a pedagogical perspective, time spent learning to use a *second* corpus tool is dead time. Nothing educationally productive is being 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. When class time is limited, this kind of dead time is a severe problem as it 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 corpus tool that would allow students to apply the interface skills they had learnt on BNCweb with any of the corpora we commonly use in teaching corpus linguistics at Lancaster University. I gave it the name *CQPweb*; this is of course modelled on BNCweb, to acknowledge its inspiration,¹³ but it also captures the system’s basic nature as a generalised web-interface to CQP. 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,¹⁴ but closely following BNCweb as a model. In only a relatively few cases was code borrowed *verbatim* – most notably in the case of the SQL statements used to calculate collocation statistics, to assure compatibility of these statistics between BNCweb and CQPweb (although the issue of collocation statistics is a vexed one on many broader levels; see Evert 2004, 2008).

¹³ It is worth noting at this point that BNCweb was more than merely an inspiration or model. The process of creating the CQPweb system benefitted at critical points from the good advice of the creators of the BNCweb (CQP edition), Sebastian Hoffmann and Stefan Evert.

¹⁴ Debates about the relative merits of different programming languages are proverbially unproductive. However, in this particular case, I would note that the mere fact of using a *different* programming language made the reimplementation of BNCweb as CQPweb much easier. I had to look carefully at every single line of Perl code for the core BNCweb functions, giving me the opportunity to consider whether or not, as well as translating it to PHP, an adaptation was necessary to generalise its operation to work with any corpus. If I had been simply copying and then modifying the Perl code, it would have been much harder to spot the points where generalising changes were needed.

This is a pre-publication draft MS. You can cite it as “Hardie (forthcoming)”. In the event of any difference between this MS and the future published version, the published version is to be considered definitive.

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

Figure 2. The welcome-page of a corpus indexed in CQPweb (compare fig. 1)

Screenshots of the main search pages (which also serves as the welcome page) of BNCweb and CQPweb are given in figures 1 and 2 respectively, and serve as a representative example of the difference between them. The overall layout of the interface is as close to identical as possible. The goal is that a user who is familiar with BNCweb should be able to use CQPweb

without any further training. However, there are some differences resulting from the need to generalise the interface to any corpus, most notably in the organisation of the main menu: some options are in a slightly different order, and some have different names. 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 a relevant distinction for this particular corpus, nor that these two sections have different bases for restriction (textual metadata versus speaker metadata, in BNCweb). 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 access via the web to corpora which cannot be distributed in textual form due to copyright restrictions. As noted above, this is a common application for web-based corpus query tools. For instance, the BE06 corpus (Baker 2009) is made publicly available solely via Lancaster University’s CQPweb server. The high degree of 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 Online¹⁵ on Lancaster’s CQPweb server, we have made it possible for historians to apply the full range of corpus methodologies to this dataset, complementing and enhancing the usual historical methods of text analysis.

CQPweb, like BNCweb, 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 time since its initial release. Subsequent to its creation, CQPweb has been adopted as the main graphical user interface of CWB/CQP and is now available from the same source code repository.¹⁶ In their discussion of CQP, Hoffmann and Evert (2006: 180) note that ‘the Corpus Workbench suffers from the lack of a user-friendly graphical interface to the query processor, which is only available as a command-line application’. CQPweb has come to fill this gap, at least to some extent.

3. Architecture and technology

3.1. Data model

As a front-end to CQP, CQPweb shares the same fundamental data model as 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 (and, if the corpus has been processed by tokenising software, other word-break adjustments can be represented within CWB format: for instance, to split *isn’t* or *you’re* into *is n’t* and *you ’re*). 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

¹⁵ <http://eebo.chadwyck.com>

¹⁶ <http://cwb.svn.sourceforge.net/viewvc/cwb>

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. Each column is given a ‘handle’ in the course of indexing, which in CQPweb must be a ‘C word’.¹⁷ 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 of course have attributes in the usual format. All data must be encoded as either ASCII, ISO-8859, or UTF-8.¹⁸

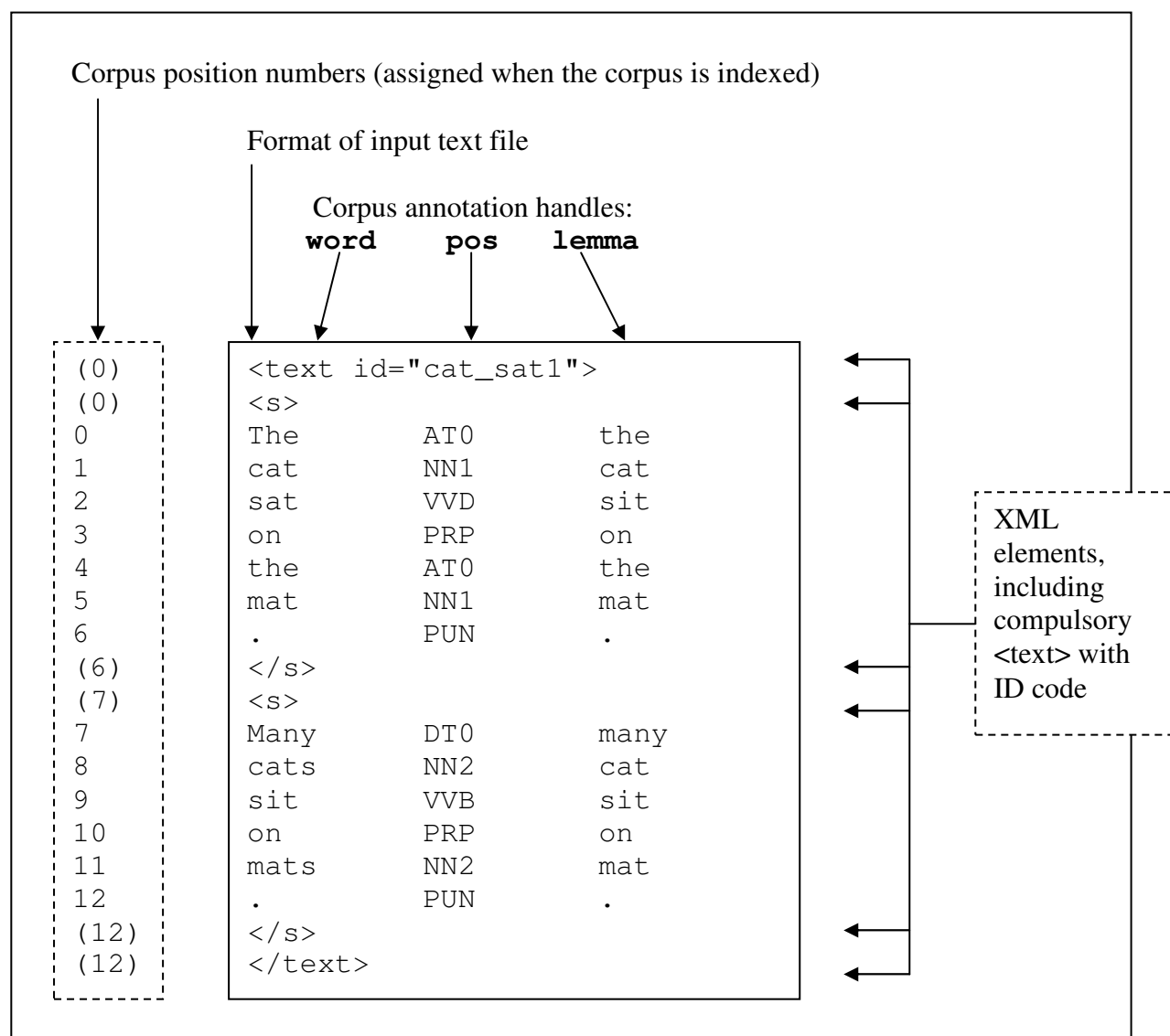


Figure 3. The CQPweb data model: CWB input format

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>

¹⁷ A ‘C word’ is a label that complies with the rules for identifiers in C and similar programming languages. This basically means that 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.

¹⁸ CQPweb treats all text as UTF-8 internally, but can translate underlying ISO-8859 text to UTF-8 on the fly. Note that if CQPweb is used with a version of CWB prior to 3.2, UTF-8 regular expressions are not fully supported.

elements. If the corpus cannot meaningfully be broken down 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’ element; 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.

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

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

Since BE06 has been annotated by both the CLAWS POS tagger¹⁹ and the USAS semantic tagger,²⁰ 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). Note that 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; from these, CQPweb generates ‘help’ links particular to the annotation of a given corpus on its main menu. Elsewhere in the database, for each corpus a *primary annotation* is specified. This is the 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

¹⁹ <http://ucrel.lancs.ac.uk/claws>

²⁰ <http://ucrel.lancs.ac.uk/usas>

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 is expected to 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.

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

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

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 relevant classification schemes are the fifteen-genre system of that sampling frame (indicated in *textcat* by fifteen one-letter labels), and the

broader four-genre system into which the Brown sampling frame categories is often collapsed (indicated under *genre* by the labels *press*, *gen_prose*, *learned* and *fiction*). These 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 compatible with the ‘Written restrictions’ function of BNCweb. The interface generated for BE06 is shown in figure 4. A query run with these restrictions would find *only* results in fiction or academic texts that were sampled from the middle of a longer document.

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

Figure 4. Restricted query interface for BE06 in CQPweb

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

As already noted, CQPweb like BNCweb operates across the web on a client/server model. Its foundation is a set of CWB-indexed corpora that can be queried using CQP. CQPweb is also capable of interacting with the CWB utility programs to create and manage the actual indexed data. Alongside the corpus data, a relational database is also employed; the particular system used is MySQL. In addition, CQPweb makes use of the CEQL module of CWB’s Perl interface to parse the syntax of simple queries into CQP-syntax queries. An overview of how the different components of CQPweb fit together is given in figure 5.

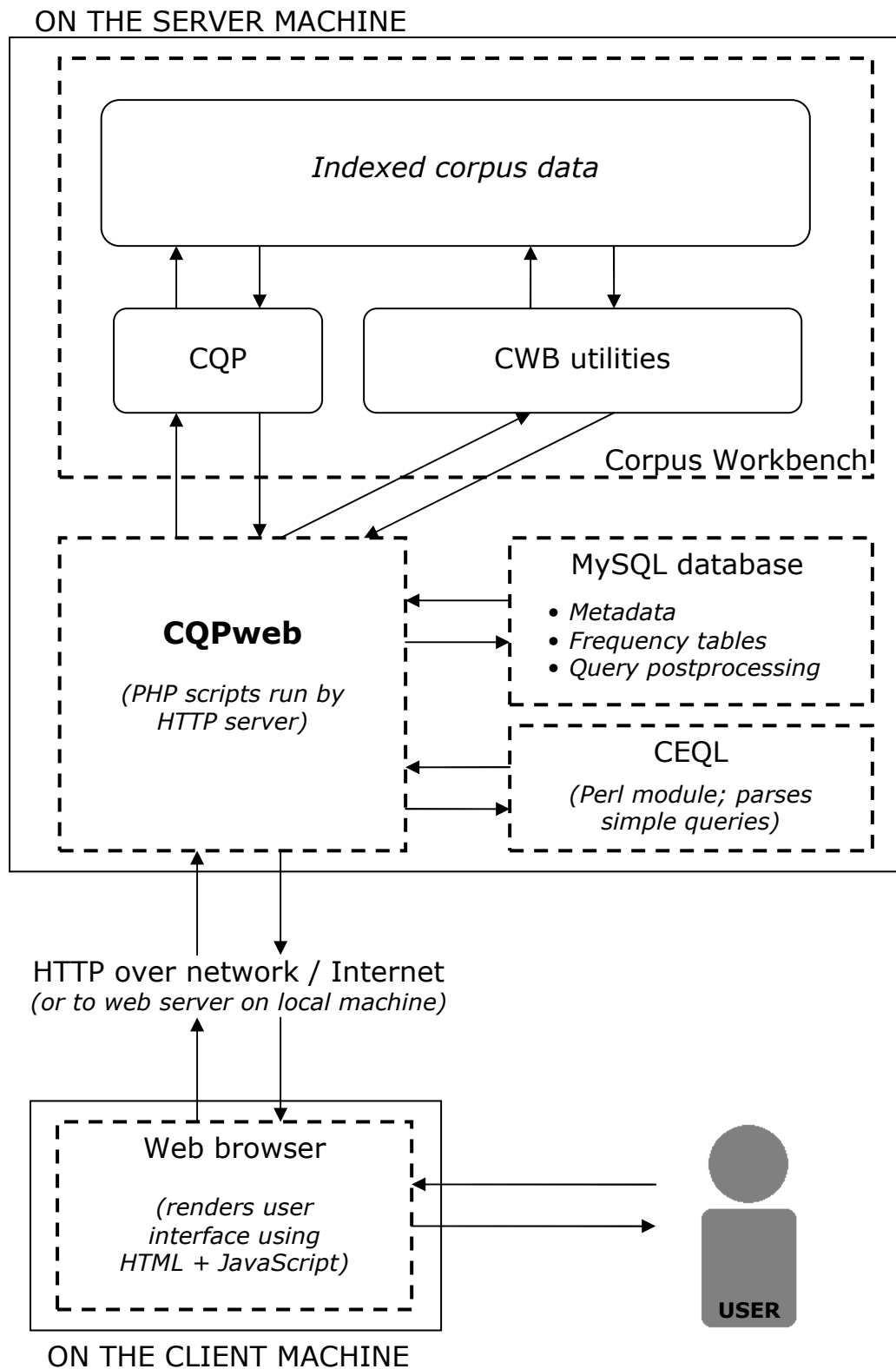


Figure 5. General architecture of the CQPweb system

4. What CQPweb can do

4.1. What the user can do with CQPweb

When a user logs on to CQPweb they are typically presented with a list of available corpora (see 4.2 for more details). Selecting a corpus brings them to the welcome screen (see figure 2), from where they can perform a query or choose another menu option; however, 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 (as already discussed in section 2.2). These include providing the concordancing power of CQP both in full and via the simpler CEQL language; in both cases, queries can address any indexed stream of annotations as easily as the wordform. Furthermore, as illustrated by figure 4, queries 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 results of a CQP query may be analysed further. Each postprocess is controlled via a user-friendly and BNCweb-compatible web form, just like the basic query interface shown in figure 2. 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. (Many of the other postprocesses – sorting, collocation, distribution, and so on – can also have the effect of reducing the query to a subset of hits, but on a principled basis rather than randomly.)
- *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 the query hits across the categories of a given textual classification (as modelled in the metadatabase; see 3.1). Clicking on the name of a given category produces a new query thinned to just the hits within texts in that category.
- *Categorising*. The ‘Categorise query’ function allows the user to define their own set of analytic labels, and then categorise the query results according to those labels. When this has been done, a query can be split into several different queries, each containing the this assigned to a particular label. One typical use of this function might be to annotate the results of a query with low precision, to distinguish the desired hits from the undesired, allowing the latter to be filtered out before further analysis.
- *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 (see 3.1; normally this is a POS tag). Sorting can also thin a query, since it is possible to specify a pattern that the word/annotation at the sort position must match to be shown. A random sort is also available, as well as the facility to restore the original corpus order.
- *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 in <n>, the frequency breakdown would show the frequency of each such wordform within that set of hits.

Clicking on any form then produces a query thinned to just the subset of hits that match that form.

- *Multiple postprocesses.* The output of running a postprocesses on a query result is itself a query result, to which further postprocesses can be applied. The history of the currently-active particular query result (i.e. the original query, and everything done to the results by way of applying postprocesses) is shown on-screen at all times.

Any query can be saved within CQPweb, or alternatively downloaded to the user’s computer in the format of a plain-text table suitable for importing into word-processor, spreadsheet, or database software; the format of this plain-text file is configurable.

Apart from queries and postprocesses, users can view 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 including only texts that contain at least one result for a given query. Moreover, 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 1997: 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 stream of annotation – POS, lemma, semantic tag – that has been indexed and recorded in the database. 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.²¹ A further point of generalisation in CQPweb is that keywords can be calculated *across corpora*. It is possible for the frequency lists of a given corpus or subcorpus to be declared ‘public’ by the system administrator. A public frequency list can be used anywhere across the system. So, for instance, if the frequency list of a spoken corpus is declared public, it could be used as the reference corpus for a keyness analysis looking at a subcorpus of fiction texts created within a completely different corpus.

CQPweb is both language-independent and writing-system-independent. That is, the features listed above can be applied regardless of the language of the corpus (although, naturally, features based on annotation can only be used if the annotation is actually present for a given corpus). Writing-system-independence is achieved, naturally, by use of Unicode (in the form of UTF-8) throughout, although the main ISO-8859 character sets are also supported as a form of backward-compatibility. It should be noted that for scripts such as Chinese, where there are no obvious 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 corpora in Arabic, Hebrew, or other such writing systems. For all scripts, the entering of text (such as in 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.

²¹ 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 terribly meaningful.

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. For example, to index a corpus using the web interface, the administrator needs at least some understanding of the underlying data model (see 3.1) which the ordinary user does not need. The web interface makes many operations more convenient for the administrator, but a degree of technical expertise is still required.

Indexing a corpus via the web interface requires one or more input files in the specified format, and an input file for the metadatabase, to be inserted into CQPweb’s ‘upload area’ – a directory on the server machine that CQPweb controls and makes visible to the administrator within the web interface. 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. Corpora that have been indexed directly 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. More significantly, the stylesheet to be used by a corpus can be set. Nearly all aspects of the visual appearance of CQPweb – colours, fonts, table grid design, and so on – are specified by a CSS stylesheet. CQPweb comes with twelve built-in ‘skins’, which are essentially different-coloured variants of the same basic stylesheet. But administrators can load in additional CSS files, which can deviate as much as desired from the default visual style shown in figures 2 and 4. Essentially, the aesthetics of CQPweb are entirely configurable – but 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 semi-instinctively what corpus they are working with at a given moment. If a user is moving around among many corpora in a single session, this feature can be very useful.

Other options relate to the annotations of each corpus. Notably, the CEQL query language has become configurable in CQPweb. As explained in section 2.2, in BNCweb the CEQL syntax of underscores and braces gives 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. So, for example, the `_XXX` syntax, which in BNCweb searches for the POS tag XXX, can be set to search for a semantic tag, or a gloss, or whatever else might be present. 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 the user administration functions currently rely on CQPweb running under the Apache web server;²² some others are available on any web server. User accounts can be created or deleted, passwords set or reset, and limits placed or adjusted on the size of the temporary MySQL databases each user can create (for example, when running the collocation or sort functions). Most importantly, the administrator has full control over users’ rights of access to particular corpora. This works as follows. Every user account is assigned to one or more user groups. For each corpus on the system, the administrator then specifies which user groups can access it. 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 only be made available to users known to be affiliated to institutions with a licence.

Of course, this assumes that user accounts are allocated rather freely; it would naturally be possible to give out accounts only to researchers or students within one specific institution, in which case such gradations of access rights would not be needed. The limiting case of this would be a copy of CQPweb running on one individual’s personal computer, in which case only a single user account (that of the administrator) is needed.

It should be noted that 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;²³ it provides functionality for managing user accounts, but it assumes that keeping these accounts secure is not its responsibility. That said, assuming a reasonable security model on the server, CQPweb can be a suitable tool for affording limited access to moderately sensitive data. As noted previously, when a corpus cannot be legally redistributed at all (e.g. if it was collected *without* requesting permission to redistribute from the copyright holders; see Davies 2010 for some discussion of this point), 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. The generality of CQPweb makes it flexible enough to support many different (kinds of) corpora, and for this reason CQPweb can function as a suitable off-the-shelf tool for providing a web interface of this sort – and, furthermore, a relatively inexpensive one, since the program itself and all its dependencies are all free/open-source software. The expense in time and effort of developing a new tailor-made interface for each corpus to be made available via the web can be avoided.

5. Evaluating CQPweb

5.1. Evaluation

In evaluating CQPweb, it is first necessary to decide on what basis it *should* be evaluated. One obvious route of evaluation is performance, as measured, for instance, by the time required to perform a certain of (complex) queries. This is the approach taken by Mark Davies (n.d.) in evaluating the corpus.byu.edu architecture (Davies 2005, 2009, 2010; see

²² <http://httpd.apache.org/>

²³ For instance, CQPweb will work over either HTTP or HTTPS according to how the web server it is running on has been set up. In the former case, username and password security is likely to be rather low.

also 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’ system, which runs only on one system, namely the corpus.byu.edu server. But CQPweb – and CWB/CQP in general – are, by contrast, openly-available software packages intended to be installed and run 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 is also a factor of the speed of the 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 and 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 and Evert 2006: 191). CQPweb more than meets this requirement, as corpora are not required to be similar to the BNC in any particular way. All that is necessary is that the corpus can be expressed as appropriate input files for indexing under the CQPweb data model.

(2) *TEI compatibility*. The BNC uses (one variant of) the XML markup scheme(s) defined by the Text Encoding Initiative (TEI)²⁴ and Hoffmann and Evert ‘envisage *Cweb* to be compatible with any corpus that is encoded in an XML format and whose structure conforms to the TEI’ (2006: 191). CQPweb partially meets this requirement. The XML format of a TEI corpus cannot be indexed into CQPweb directly: in the TEI tokens are indicated by <w> elements (or not at all) rather than as lines, and, moreover, metadata is stored in the header of each file rather than as a separate table. Moreover there are, as yet, no tools built in to CQPweb for mapping from TEI format to CQPweb input format; this task is left to the system administrator. However, it would be computationally trivial to generate suitable input files from a TEI-compliant corpus, perhaps using XSLT to extract, on one hand, the body of each text as a vertical input file with the <w> elements converted to tab-delimited lines within the requisite <text>...</text> tags, and on the other, a tab-delimited metadata table collecting together information from the headers of the various texts.

(3) *Easy corpus setup*. Hoffman and Evert suggest (2006:192) 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; however, some programming skill may be necessary for the process of converting corpora in other formats into CQPweb input format, and in any case an understanding of the data model is required to effectively index a corpus. However, CQPweb

²⁴ <http://www.tei-c.org>

falls short by restricting corpus-setup functions to the administrator-user(s). In an ideal world, ordinary users would also be able to index their own data. But given that indexing a large corpus can lock up a great deal of disk space and processing power, the restriction to administrators is a practical necessity on a CQPweb server with many users. 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 and Evert also recommend that ‘there should be a smooth migration path from basic simplified queries over and extended syntax [...] to the full-fledged CQP language’ (2006: 192) 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 of BNCweb, and offering the same range of query postprocesses, is the requirement of the white paper that CQPweb fulfils most completely; this was, in fact, its *raison d’être*. However, one extra function proposed for *Cweb* (the ability to create templates of multiple postprocesses that can be activated as single actions) does not exist in CQPweb.

(6) *User management.* Hoffmann and 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. However, CQPweb purposefully goes against Hoffmann and Evert’s suggestion that users should be able to configure the visual appearance of the *Cweb* interface; as noted above, fonts, colours and so on are determined by the administrator on a per-corpus basis, for all users, in order to allow different corpora and/or different CQPweb installations to be visually distinct; I would argue that this is a more valuable use of the aesthetic side of the interface than allowing individuals to determine the appearance.

(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 for *Cweb*, and it can be installed on a desktop computer, as they suggest. However, CQPweb does still require a Unix-like operating system (i.e. not Windows); future development will hopefully allow this requirement to be removed so that Windows machines can run CQPweb installations. So at present CQPweb mostly but not fully meets Hoffmann and Evert’s requirements on this point.

(8) *Modular architecture.* Hoffmann and Evert suggest (2006: 193-194) 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 simple query language parsers (in Perl). Although it has been designed in as modular a style as possible, CQPweb permits neither of these customisations. Future extensions to administrator’s interface may allow some customisation especially as relates to the management of corpus input files, but it is not expected that this functionality will ever be exposed to the ordinary user.

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 2.1). CQPweb does perform fairly well on this score, querying corpora of 600+ million words without much loss of speed (and of course the use of a query cache helps prevent loss of speed especially in the classroom context). In fact, speed seems to fall not in line with the growth of the corpus itself, but rather in line 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 (whereas a much larger corpus with the same number of texts would not suffer the same kind of slowdown). Taking measures to optimise the MySQL queries that reference the metadatabase may address this current weakness in CQPweb.

So, evaluated against Hoffmann and Evert’s white paper, CQPweb does rather well. However, in fairness it should be noted that Hoffmann and Evert are considering the future concordancer strictly as a successor to BNCweb. 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 and Hardie (2011) point out, there are many analytic techniques that have been developed and described in the literature that 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 and Gries (2003) are all widely known and influential methods, but no current concordancer supports them.²⁵ CQPweb does not yet do anything to address this state of affairs, rather sticking to the core procedures that most corpus analysis tools have long possessed.

5.2. Other limitations and planned developments

CQPweb continues to develop; some indications of planned enhancements have been indicated in the preceding discussion, and adding new analysis techniques to the toolbox it makes available will surely also form part of this future work. However, some further planned developments are worth mentioning.²⁶

Only two current features of BNCweb are not implemented in CQPweb; one, the ability to upload a query (see Smith et al. 2008) will be added in a future version, but the other, the ‘browse text’ function, is not currently planned for CQPweb at all, given that CQPweb’s purpose is, at least in part, to allow corpus searching to be done *without* exposing the full texts. Arguably text browsing could be added if it was possible to disable it on a per-corpus basis; however, even so it is not a high-priority for future development. Beyond these features, 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 rather limited. This is something of a flaw in a tools that aims at a high degree of usability, but it has not been a major problem to date, simply because CQPweb’s main group of target users are usually

²⁵ However, SketchEngine’s ‘word sketches’ do approximate collostructional analysis to some degree.

²⁶ At the time of writing, CQPweb is at version 2.17; version numbers higher than this may be expected to implement some or all of the features here designated as developments for ‘the future’.

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 should 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-user 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 CQPweb. Likewise, once XML regions and annotations are represented in the data model, it should be possible to restrict queries to XML-element-defined subsets of the corpus in the same way as it is possible, in BNCweb, to restrict a query to utterances whose speakers meet certain criteria (such as age or social class).

Some other enhancements are already under development, although not yet complete in the current version of CQPweb. Two worth mentioning are: (a) the addition of an interface to the R statistical software, to allow use of R’s unparalleled capabilities for statistical analysis and graphical display within some of CQPweb’s analysis functions; and (b) a reworking of the concordance-rendering module to support the analysis of field-linguistic data in the classic ‘three-line-example’ format, for use with corpora where the primary annotation is a morpheme-by-morpheme gloss rather than a POS tag. Other planned extensions remain to be implemented. These include support for concordancing across parallel corpora; CWB has extensive support for aligned parallel data, none of which is currently accessible via CQPweb – but clearly it should be so accessible. Likewise, the following are at present only in the planning stage: support for visualising (and statistically analysing) dispersion of the hits in a query result; support for the analysis of word or annotation n-grams; extending the collocation function to allow it to look at collocation by grammatical relation (e.g. collocations between head noun and modifying adjective, or between verb and object) as well as by straightforward proximity; support for indexing and querying parsed corpora (whether dependency-parsed or constituency-parsed); an intermediate level of user privileges, between normal user and administrator, that gives oversight over a group of other user accounts (for use by teachers allowing them to monitor their students’ work); and, finally, support for programmatic interaction with CQPweb. This last feature is intended to allow other tool-designers to program against a function-oriented interface to corpus queries and the other user-facing capabilities of CQPweb. Commands composed and sent via HTTP to a special interface page will produce output data in plain-text format that can be processed according to the programmer’s liking. It will thus be possible to create custom-designed web-tools that use CQPweb as their back-end – much as CQPweb itself uses CWB, but with all the extra functionality of CQPweb also available. This will address, to some degree, the modularity criterion in Hoffmann and Evert’s (2006) white paper.

In contemplating such future developments to CQPweb, there is an ever-present tension between extending the capabilities of CQPweb, and maintaining compatibility with BNCweb. Obviously, if a feature is added, the interface must be adjusted to accommodate this new

feature. Yet if the interface changes too much, then the original virtue of CQPweb – that it replicates a well-known and widely-used 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 – and, I would argue, in principle *cannot* – 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 and analysed. In particular, CQPweb does not demand any especial annotation or metadata in a corpus; it works with whatever is there, and its data model can accommodate corpora with no annotation at all, or texts with no metadata, just as easily as a corpus with BNC-level annotation. 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. In particular, for teaching purposes it is an immediate 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 (although its speed depends to a considerable extent on the hardware it is running on).

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, though it does combine flexibility with power and usability, its flexibility has one very sharp boundary: it does not allow users to upload and analyse their own data, but rather restricts them to working with corpus resources set up for them by a system administrator. By contrast, many desktop-computer-based concordancers are rather less powerful but make it easy for users to analyse their own data. Ultimately, so long as corpus analysts are faced, when choosing a software tool, 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 tool 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.

References

- Abercrombie, D. 1965. ‘Pseudo-procedures in linguistics’, in D. Abercrombie (ed.) *Studies in Phonetics and Linguistics*, pp. 114–9. Oxford: Oxford University Press.
- 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*, pp. 7–13. Tokyo: Waseda University.

This is a pre-publication draft MS. You can cite it as “Hardie (forthcoming)”. In the event of any difference between this MS and the future published version, the published version is to be considered definitive.

Aston, G and 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, Texas: Athelstan.

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

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

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

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

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): 307–34.

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): 159–90.

Davies, M. 2010. 'More than a peephole: Using large and diverse online corpora', *International Journal of Corpus Linguistics* 15 (3): 412–8.

Davies, M. (n.d.) *Comparison of the BYU Corpus Architecture, Corpus Workbench (SketchEngine), and Corpus Workbench (BNCweb)*. Available online at <http://corpus.byu.edu/architecture.asp>, accessed 17th December 2010.

Evert, S. 2004. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Unpublished Ph.D. thesis. University of Stuttgart.

Evert, S. 2008. 'Corpora and collocations', in A. Lüdeling and M. Kytö (eds) *Corpus Linguistics: An International Handbook*, pp. 1212–48. Berlin: Mouton de Gruyter.

Gries, St. Th. 2009a. *Quantitative Corpus Linguistics with R*. London and New York: Routledge.

Gries, St. Th. 2009b. *Statistics for Linguistics with R: A Practical Introduction*. Berlin: Mouton de Gruyter.

Gries, St. 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*, pp. 121-146. Frankfurt am Main: Peter Lang.

- Hockey, S. 1988. *Micro-OCP (OCP Version 2)*. Oxford: Oxford University Press.
- Hoffmann, Sebastian & Stefan Evert. 2006. "BNCweb (CQP-Edition) - The Marriage of Two Corpus Tools." In: Sabine Braun, Kurt Kohn & Joybrato Mukherjee (eds.) *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*. Frankfurt am Main: Peter Lang. 177-195.
- Hoffmann, Sebastian, Evert, Stefan, Smith, Nicholas, Lee David YW and Ylva Berglund Prytz. 2008. *Corpus linguistics with BNCweb – a practical guide*. Frankfurt am Main: Peter Lang
- Hundt, M., Sand, A. and Siemund, R. 1998. *Manual of information to accompany the Freiburg–LOB Corpus of British English ('FLOB')*. *Englisches Seminar*. Freiburg: Albert-Ludwigs-Universität Freiburg. Available online at: <http://khnt.hit.uib.no/icame/manuals/flob/index.htm>
- Kaye, G. 1990. 'A corpus-builder and real time concordance browser for an IBM PC', in J. Aarts and W. Meijs (eds) *Theory and Practice in Corpus Linguistics*, pp. 137–62. Amsterdam: Rodopi.
- Kilgarriff, A., Rychly, P., Smrz, P. and Tugwell, D. 2004. 'The Sketch Engine', in G. Williams and S. Vessier (eds) *Proceedings of Euralex 2004*, pp. 105–16. Bretagne, France: Université de Bretagne-Sud.
- Lehmann Hans-Martin, Peter Schneider & Sebastian Hoffmann. 2000. "BNCweb." In: John 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 and Hardie, A (2011) *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): 519–49.
- Renouf, A. 2003. 'WebCorp: providing a renewable data source for corpus linguists', in S. Granger and S. Petch-Tyson (eds) *Extending the Scope of Corpus-based Research: New Applications, New Challenges*, pp. 39–58. Amsterdam: Rodopi.
- Scott, M. 1996. *WordSmith Tools*. Oxford: Oxford University Press.
- Smith, Nicholas, Sebastian Hoffmann & Paul Rayson. 2008. "Corpus Tools and Methods, Today and Tomorrow: Incorporating Linguists' Manual Annotations." *Literary and Linguistic Computing*. 23:2. 163-180.

This is a pre-publication draft MS. You can cite it as “Hardie (forthcoming)”. In the event of any difference between this MS and the future published version, the published version is to be considered definitive.

Stefanowitsch, A. and Gries, St. Th. 2003. ‘Collostructions: investigating the interaction between words and constructions’, *International Journal of Corpus Linguistics* 8 (2): 209–43.

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

Weisser, M. 2009. *Essential programming for linguistics*. Edinburgh: Edinburgh University Press.