In empirical approaches to linguistics, corpus analysis has become an indispensable method for gaining insights into many areas of linguistic inquiry, from lexical semantics and grammar to psycholinguistics and discourse pragmatics. Apart from more computationally oriented tasks associated with corpora, such as lemmatization, tagging, and parsing, corpus-linguistic research revolves around a number of routine procedures such as searching a corpus for a particular phenomenon, counting, organizing, and displaying the results. Among these elementary tasks, the creation of concordances, i.e., formatted displays of all the occurrences of a particular type in a corpus, may be considered the most fundamental task. Thus reliable, fast, and user-friendly search-and-retrieval software is of great value to any researcher working in this field.

The aim of this article is to provide a brief overview of what current search-and-retrieval software has to offer. We will examine ten software packages and characterize them along three dimensions:

1) **Functionality**, i.e., what can they do?
2) **Performance**, i.e., how much data can they handle and how fast do they operate?
3) **Usability**, i.e., how user-friendly are they?

The ten programs are: **MONOCONC PRO 2.2**, **WORDSMAIh TOOLS 4**, **CONCORDANCE**, **MULTI LANGUAGE CORPUS TOOL**, **CONCAPP 4**, **ANTCONC 1.3**, **ACONCORDE**, **SIMPLE CONCORDANCE PROGRAM**, **CONCORDANCER FOR Windows 2.0** and **TEXTSTAT 2.6**. The first three are commercial solutions, the remaining seven are freeware. In our description of a program's functionality, we assess a number of features to address issues such as: What operations are possible? What types of input/output formats are supported? How fine-grained can a query be defined? How can an obtained result be sorted and displayed? In short: What is the program capable of doing? In more detail, our list of criteria is structured...
Daniel Wiechmann and Stefan Fuhs

to go from input to output. The criteria in the sub section ‘input’ investigate the flexibility of loading and using input data. Which character encoding is supported? Do users have to modify their data by altering file extensions or directory structures? Which information are present after loading the corpus? The next section investigates various tools and means to search for patterns in the data. What kind of methods are available to match strings? How flexible are the searches, i.e., can wildcards be used and is regex supported? Here, we also consider more advanced tools for corpus analyses, like the creation keyword lists and stop lists. Under the broad headline of frequency information and collocation we investigate the functions provided to survey corpus data. What kind of information is provided about frequency bands? Can the user create collocations and specifically adjacent n-grams from his corpus data? A major advantage of all concordancers is their ability to sort and structure the context of search results. In the section of ‘search result modification and display’ we observe those functions relevant to this task: How flexible can the context be defined? Is it possible to sort using several levels? Can the search results be made more transparent by hiding the corpus mark-up (i.e., suppress the tags)? Finally, we assess the output formats. If the data are to be used for further analyses, they often have to be saved in a certain encoding and format — the more option the software offers, the less external work the user has to put into it (the answers to these questions are summarized in Table 3 in the Appendix).

To evaluate the performance of all programs in a comparable way, we conducted a number of tests on two identical systems, each consisting of an Intel Pentium 4 processor clocked at 3 GHz and equipped with 1 GB of RAM. The operating system on both systems was Microsoft Windows XP Professional (Service Pack 2), which had been patched up-to-date prior to testing using the Windows update function. Specifically, we retrieved lexical items that differ with respect to their frequency bands (low frequency item: haggardly; medium: of, high: the) from four different corpora of varying size:

1) Brown_F: N ~ 100.000 words
2) Brown_full: N ~ 1 million words
3) BNC_spoken_part: N ~ 10 million words
4) BNC_full: N ~ 100 million words

We are aware of the fact that our test system is underpowered for some of the tasks we performed; often, insufficient RAM prevented a successful completion of a task that involves larger amounts of data but we still consider it to be a reasonable test system. The reason for this is that such a system represents current standards of office-machines in
2006 and this test allows us to directly compare a program’s resource management; i.e., we assess a program’s performance in an environment with RAM limitations. The individual performance differences are summarized in Table 2 of the appendix. In this context it is interesting to point out that not a single one of the programs tested was able to perform all searches, although MonoConc Pro 2.2 came close.

To provide a point of reference, we also used a simple Perl script to perform the searches. The script opened every file in the directory one-by-one, matched a regular expression including 100 characters context left and right in a line-by-line fashion and wrote the target string to an output file, while counting the total number of tokens. The results show that the queries are not impossible per se on our test system. Instead, the gap in performance between the script and the programs considered here is probably best explained by the difference in functionality and convenience. The programs enable the user to sort and investigate the context of a matched string in various ways, they don’t just write them to files. This is basically the service that concordancing software provides. As such, the amount of data placed in memory will differ from a write-down-and-forget type amount of our script and it would be inappropriate to make a direct comparison of the performance measures.

Nevertheless, from the viewpoint of a professional corpus linguist, it would be desirable for the concordance software to carry out large queries for quantitative analyses without crashing. If memory limitations are an obstacle to advanced context-sorting and display then a feasible option for concordancing software would be to include a simplified search function in the program that writes matched strings to a file instead of handling them in memory. Admittedly, in doing so the user would forfeit a fair amount of comfort and functionality, but he would be able to carry out the query.

Of course, this overview cannot substitute more detailed discussions of the ten applications. In our discussion, we will restrict ourselves to a presentation of what we believe are the most noteworthy aspects of a given program. A detailed contrastive overview of all programs tested can be found at the end of this article in Tables 1 to 3. This should enable the reader to get a more complete picture and also allows him to quickly check whether or not a given program provides a particular function, whether it can be used with larger amounts of data, or how much help and support is offered, to name only a few potentially relevant factors.

Michel Barlow’s MonoConc Pro 2.2 (MCP 2.2), published by Athelstan, is probably the best-known commercial concordancing package on the...
market (along with WordSmith Tools). It is available at a price of approx. EUR 72.00 (USD 85.00) for a single user license and approx. EUR 464.00 (USD 550.00) for a 15 PC license valid for 2 years.

MCP 2.2 offers a remarkable set of functions including (several variants of) creating concordances, collocate and frequency lists, identifying word clusters as well as keyword identification by means of corpus comparison. Of all programs tested, MCP 2.2 makes available the most elaborate collection of search types including batch-, tag-, and regular expression-searches. In addition to loading offline corpora, MCP 2.2 allows loading an online corpus by specifying the URL. The only major drawback of this program is that it does not supply any kind of statistical association measures, which may be of interest, for example, in the context of collocational analyses (Barlow/Athelstan offer a separate software package, *Collocate*, for this purpose, which was not tested here).

In our performance test, MCP 2.2 turned out to be the most powerful program in terms of resource management. It was the only program capable of creating a full concordance for our mid-frequency item *of*, and only failed the hardware-intensive task of doing the same for the highly frequent *the*. Although it spends a few minutes collecting tag information when the corpus to be examined is loaded (1.5 minutes for
Concordancing software

113

N ~ 10 million and proportionally longer for larger corpora), MCP 2.2 still holds the second rank in terms of speed among all programs tested. Also, MCP 2.2 is very approachable. All functions are directly accessible from its single main function bar, allowing anybody familiar with the usual design of windows applications to navigate through the user interface very intuitively.

In sum, MCP 2.2 is a comprehensive and affordable software package, which is easily accessible, offers a great many functions that can be richly configured and which exhibits only few weaknesses. It is, hence, highly recommendable for beginning and advanced users alike, provided they have an additional software package for statistical analysis.

A demo version of MCP 2.2, which imposes a cap of 20 search hits, is obtainable at the official website.

http://www.athel.com/mono.html

WordSmith Tools 4

Mike Scot’s WordSmith Tools 4 (WST4), published by Oxford University Press, provides a software package consisting of a very wide range of applications relevant for corpus-linguistic investigations and is without doubt the one suite with the most extended functionality. It comes at a price of approx. EUR 76.50 (GBP 51.95) (single user) or approx. EUR 386.00 (GBP 262.90) (10 user site).

Like MCP 2.2, WST4 excels in terms of the range of functions it offers. It can create concordances, frequency lists, collocate tables and perform keyword analyses. Possibly the most noteworthy feature is its powerful grab tool, ‘WebGetter’, which allows the creation of corpora on the fly. This is especially useful, because such corpora can be specified along a number of parameters including languages to be considered. Figure 2 illustrates the corpus generation process of a 201,000 words corpus all about fishes:

Also for the analysis of collocations, it provides a number of association scores (special MI, log likelihood, z-score, MI3). On the downside, WST4 does not support regular expressions, thereby foregoing a powerful retrieval tool. In the explicit 200+ manual, there is not a single mention of the term.1 WST 4 is a little faster than MCP 2.2, yet not quite as efficient in terms of resource management: It was not possible to create a full concordance for of or the on our test-platform, when the corpus was the full BNC (N ~ 100 million).

In a nutshell, WST 4 is a highly recommendable program for beginners and more advanced users, especially those who are looking for an all-in-one solution.
WST 4 can be ordered at the website of Oxford University Press. No demo or trial version is available.

http://www.oup.co.uk/isbn/0-19-459400-9

Concordance

The third and final commercial product in this overview is Concordance, written and published by R. J. C. Watt and is available for approx. EUR 83.50 (USD 99.00) for a single user license / plus approx. EUR 33.50 (USD 40.00) for each additional copy/license.

Concordance supports Unicode, can load a corpus from various files and supports on-the-fly changes of corpus files. Simple text searches as well searches using regular expressions can be carried out.

Albeit the most expensive program, Concordance cannot compete with MonoConc Pro 2.2 (MCP 2.2) and WordSmith Tools 4 (WST4) in terms of functionality or resource management. In the attempt to test
its performance when dealing with larger amounts of data (N > 10 million), the program crashed due to memory errors and we were not able to obtain any results. Although Concordance was written with the investigation of smaller amounts of text in mind, this seems to be a major drawback of an otherwise decent program that allows intuitive handling.

As a brief look at Table 2 shows, Concordance is also considerably slower than its competitors.

In our view, researchers interested in quantitative corpus-linguistics, who want to investigate large data sets might be better advised to opt for either WST 4 or MCP 2.2. We recommend that users consider the trial version first, which can be used without restriction for 30 days, to examine whether Concordance can be used for the task at hand.

Concordance can be downloaded and licensed at the official website. A 30-day trial is available.

http://www.concordancesoftware.co.uk/

Multi Language Corpus Tool (MLCT)

Scott Piao’s Multi Language Corpus Tool (MLCT) is a very versatile JAVA application available as freeware from the University of Lancaster. MLCT offers a wide range of functions including parallel concordancing, i.e., it allows simultaneous inspection of two corpora for compara-
tive or contrastive purposes. It allows the investigation of a wide variety of languages including Chinese and supports a great number of formats including (little and big) Unicode. Furthermore, MLCT is one of the few applications capable of calculating n-grams (up to n = 6) and employs the largest range of association scores to rank them.

On the performance side, it was capable of dealing with smaller corpora (e.g., Brown, N = 1 million). Interestingly, as table 2 shows, MLCT was faster than any other commercial product.

MLCT offers a lot for a freely available program, even if it does not quite provide the range of functions and comfortable display and modification capabilities of WST 4 or MCP 2.2. Furthermore, it should be noted that the author provided valuable help and answers to questions regarding MLCT’s performance. For instance, a batch file is provided to optimize memory capacities allocated to the tool.

Researchers focusing on contrastive studies should take a look at this solution (especially if they are investigating languages that require Unicode formats such as Chinese).

MLCT can be downloaded at the following website:
http://www.lancs.ac.uk/staff/piaosl/research/download/download.htm1
ConcApp 4

ConcApp 4 (CA4) was developed by Chris Greaves and is available as freeware from the author’s website. It has an easy-to-use interface, which allows the access of all functions from a single tool bar. Concordances are generated extremely quickly, faster even than in any other (commercial) software package. The range of functions is, however, rather limited, allowing relatively simple searches only and providing no support of regular expressions. For the analysis of word associations (n-grams up to a size of 5) the program can be supplemented by a related product by the same author for a flat charge of approx. EUR 8.50 (USD 10.00). However, this upgraded version, ConcGram, was not investigated here. Furthermore, we were unable to query corpora larger than Brown (N ~ 1 million).

Researchers who usually use a corpus as a source of natural examples with no ambition to engage in serious quantitative investigation will appreciate CA4 for its intuitive handling and speed. Like many other programs presented in this overview, it is better suited for quick and dirty estimation of how often and in what context a given string is typically used than for a full-fledged large-scale analysis.

Figure 5. ConcApp 4 search menu
In summary, given CA 4’s easy-to-use interface and its excellent speed properties, we recommend this program at least to anyone dealing with relatively small sets of data. CA 4 is available at the author’s website:


**AntConc 3.1.1**

AntConc 3.1 (AC3.1) is a freeware concordancer developed and published by Laurence Anthony. It comprises several tools for creating concordances and text analysis accessible from a unique well-structured and intuitive user interface.

This interface uses a flat structure in which all modules are accessible through tabs and user input is done through a unified mask. This all-in-one approach to the user interface might first feel a little unusual to users coming from an MS Windows background. AC 3.1 allows the user to perform a wide variety of tasks: creating concordances, concordance plots, word clusters, n-grams, word lists and keyword lists. In turn, each of these modules allows a wide range of options whose full utilization is only limited by the hardware, not by the software. AC 3.1’s strength lies in its sophisticated text analysis that surpasses creating simple concordances; its functionality is paired with useful statistical measurements and its support of various character encodings and formats in both input and output.

It should be noted that a new version of AntConc (3.1.2) has recently been published, which could not be considered at the time of this review. However, the revision history notes of 3.1.2 state that, among other things, processing speed has been increased, which will likely result in a better performance profile than the one we have established for 3.1.1. Considering it is freeware, AC 3.1 comes with a surprising range of functions and thus it is also of interest to the professional corpus linguist, certain memory limitation notwithstanding. We particularly recommend AC 3.1 to corpus linguists who work with corpora consisting of multiple files and who require creating n-grams.

AC 3.1 is available at the author’s website:

http://www.antlab.sci.waseda.ac.jp/software.html

**Aconcorde**

Aconcorde (ACC) is a freeware program developed by Andrew Roberts. It is written in Java and therefore it can be run directly on all platforms for which the Java Runtime Environment is available; no in-
Figure 6. N-gram creation in Antconc 3.1.1
stallation is required. ACC is a simple concordancing tool that is special-
ized on helping the user with corpus-illustrated work. While it does not
offer a range of tools for quantitative text analysis comparable to the
other software packages discussed here, it focuses on providing compre-
hensive support for working with Arabic texts. This is reflected on sev-
eral levels: the user interface can be switched to Arabic, the character
encoding supports Unicode as well as specific Arabic fonts and text ori-
entation can be mirrored vertically. The current version of ACC imposes
the unfortunate limitation of only allowing the user to run queries on
one file at a time and it lacks absolute and relative frequency information
for retrieved patterns in the KWIC window. It does, however, automatic-
cally create a word list of the text surveyed, together with absolute fre-
cuencies and type/token counts of all word forms. A new version with
extended functionality and performance is under development and news
on its current development stage are available at the author’s website.
First screen shots suggest that some of the current limitations will be
addressed in the upcoming version.

We recommend this program to linguists and language teachers who
need quick examples from small texts for corpus illustrated work only
and who do not wish to delve into the depths of a large and complex
program suite.

ACC is available at the author’s website:
http://www.andy-roberts.net/software/

**Simple Concordance Program 4.08 (SCP)**

Simple Concordance Program (SCP) – nomen est omen – is designed
to make concordance retrieval easy. This freeware program has been
developed by Alan Reed. SCP focuses on basic functions for creating
concordances and word lists. Consequently, means of statistical text
analysis are only marginally supported. The user interface is structured
along three tabs: concordance, word list and statistics. Within the con-
cordance search tab, the user is presented with a choice of different
match methods for search strings, such as matching an entire word, suf-
fices, prefixes or any linear order of characters. In addition, several pat-
terns can be combined using either logical relations or specification of
linear order.

To enable working with languages other than English, SCP provides
an on-screen-keypad with special characters. In addition to creating con-
cordances, SCP creates word lists and provides some initial measure-
ments for investigating the type/token distribution. Both keyword lists
and stop word lists can be defined with the former presenting very flexi-
Figure 7. *SCP user interface*
Daniel Wiechmann and Stefan Fuhs

ble functions for fine-tuning the list. While the general tabbed structure
of SCP makes intuitive access possible, the rest of the user interface is
slightly more idiosyncratic. However, possible uncertainties regarding
the function of some icons are usually resolved through the exhaustive
video tutorial, which is available for download at the same site as the
program. An unfortunate drawback to an otherwise well-done program
is that input files are capped at 25 thousand lines. This affects corpora
like the British National Corpus, with file lengths greatly exceeding the
aforementioned cap and quantitative analyses cannot be carried out
without first splitting up large files.

SCP is a good choice for linguists who need quick concordances with-
out wanting to work through a pile of functions first.

SCP is available at the following website:
http://web.bham.ac.uk/a.reed/textworld/scp/

Concordancer for Windows 2.0

Concordancer for Windows 2.0 (CW 2) can be counted among the
classic concordancers. Although the development of this program was
discontinued ten years ago, there are still good reasons for considering
it here. Firstly, it is freeware and secondly it offers a customizable 5-slot
advanced query editor that allows for detailed pattern searches. CW 2
supports loading multiple file corpora, but requires all files to have the
.txt file extension. Alphanumerical word and sentence operators are de-
finable in this context and carriage return and line breaks escape charac-
ters can be incorporated as well. The advanced query editor deserves a
more detailed discussion. Plain text patterns for 5 customizable slots can
be defined in fields representing syntagmatic word order. In turn, each
box can be filled by several patterns in paradigmatic relation, for in-
cease to match different lemmas of the same lexeme. In addition, the
user can define the number of intermediate words allowed between each
pair of adjacent patterns as well as their linear order. As a result, the
query can be used to match fairly complex and flexible sentential pat-
tterns.

WC 2 was written for a Windows 3.11 environment and the user inter-
face is likely to feel a little old-fashioned to today’s users. In addition,
common functions tend to be hidden in unusual places. Saving a concor-
dance, for instance, is done through a command called ‘copy’, available
only by right-clicking the concordance results. An unfortunate limitation
is posed by CW 2’s rigid cap at 16000 word types and concordance lines.
This artificial limitation is understandable considering the hardware en-
vironment the program was originally designed for, but it is unnecessary
for state-of-the-art systems and delimits both the corpus size and set of
search patterns that the tool can cope with. We recommend CW 2 for
Concordancing software

Figure 8. CW 2's advanced word search query

users who are either investigating small corpora or who are searching for very specific patterns where the 16000-line limitation does not become a problem. The fine-grained word search options still makes it a valuable tool today. Also, CW 2 offers a collocate display that can display the collocates of several word searches simultaneously.

WC2 is available at several websites, including the following:
http://www.flwi.ugent.be/nl/upload/courses/bdfrancq/WCONCORD.EXE
http://opinion.nucba.ac.jp/~davidlee/devotedtocorpora/WCONCORD.ZIP

TextSTAT 2.6

TextSTAT 2.6 (TS2.6) is published by Mathias Hünning and it is distributed as freeware. The program is written in Python and the user is presented a choice of either downloading the source code or a ready-to-use binary for MS Windows, the latter being the version tested here. TS 2.6 is geared towards quick and direct corpus queries and easy structuring and surveying of the search results rather than a full-fledged quantitative analysis. Corpora can be filled from various sources: newsgroups, websites and several types of offline files, including MS Word and OpenOffice Writer files. The user can specify an URL and the number of pages the web spider is to retrieve from within the domain. In a similar fashion, the program can gather posts from newsgroups given a server and the group. The context of keywords can be defined in characters and the ‘citation’ function allows the user to display the matched string in a
greatly extended context, consisting of several lines and the file path of
the host file. Searches can be conducted using either plain text or regular
expressions, the latter being carried out by Python's internal \texttt{re} module.
The plain text queries can be modified by toggling case sensitivity and a
'whole-word' constraint. A very convenient feature is the query editor:
the user can simply type plain text with regular wild cards and the engine
converts this to a regex string, which is displayed after the search opera-
tion. This step offers the advantage of making searches more transpar-
ent; TS 2.6 always uses regex to match and thus the finer points of the
query are not lost to the internal ongoings of the concordance program.

Figure 9. \textit{ConcApp 4 search menu}

As stated above, TS 2.6 does not focus on more advanced tools for
concordance analysis. Hence, features like N-gram creation, cluster func-
tions, keyword identification and keyword distribution plots are absent.
However, it is possible to create a word list from the corpus data and
sort it by frequency. These results can be exported into various formats,
including text files and MS Excel files for further analysis. Moreover,
the program allows the user to search for word forms either by matching
a string or by specifying a frequency range. The results can be sorted by
frequency, alphabetically and, deserving particular emphasis, retrograde.
Concordancing software

The latter offers a convenient way to create frequency lists of words ending in a particular suffix.

TS 2.6 is available at the following website:
http://www.niederlandistik.fu-berlin.de/textstat/

University of Jena

Notes

* The order of authors is arbitrary. We thank Anatol Stefanowitsch and Stefan Th. Gries for their comments. All remaining errors are ours. Correspondence address: daniel.wiechmann@uni-jena.de and stefan-fuhs@uni-jena.de.

1. To give an example as to what more detailed tests might reveal, we would like to mention a flaw in MonoConc Pro 2.2 pointed out by Stefan Gries: when the tag search ”&AJ0 &AJ0 &NN?” is submitted using the spoken part of the British National Corpus: When the output is sorted by R1, i.e., the first item on the right of the search term, and context is suppressed, MonoConc delivers the following output in line 1646: “<w AJ0>distinguished<w AJ0> ... citive<w NN2>characteristics”. That is, MonoConc cuts off crucial parts of the desired results. Similarly, some programs have problems with more complex regex searches, the correctness of which was tested by using the ‘match’ command in Perl.

2. ‘Tag searches’ allow meta-information from annotated corpora to be included in the search string. A ‘batch search’ is helpful when one wants to search for a whole list of search strings simultaneously. ‘Regular expressions’ are usually used to give a concise description of a set, without having to list all elements. To test a program’s handling of regular expressions, we used the following string ‘(go|went|gone|goes)s+([a–z]+[^a–z]+)[0,4]:mad’, which describes a lemmatised collocation GO mad with up to 4 words in between (e.g., “people who have gone mad” or “people who went computer-mad”).


4. Note that the next higher level, which MLCT was not able to handle, was ten times the Brown corpus size. Thus, the only thing we can report here with respect to MLCT’s maximum corpus size is that it is somewhere below N = 10 million.

Appendix:

Abbreviations used:
AC 3.1 AntConc 3.1.1
ACC Aconcorde 0.4.1
CA 4 ConcApp 4
CC Concordance
CW 2 Concordancer for Windows 2.0
MCP 2.2 MonoConc Pro 2.2
MLCT Multi Language Corpus Tool
SCP Simple Concordance Program 4.08
TS 2.6 TextStat 2.6
WST 4 WordSmith Tools 4
<table>
<thead>
<tr>
<th>Stats</th>
<th>Platform</th>
<th>Publisher</th>
<th>Language</th>
<th>Price</th>
<th>Developer</th>
<th>Help</th>
<th>Global help file</th>
<th>Manual</th>
<th>Installation directories</th>
<th>Save directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCP 2.2</td>
<td>PC (Win 95 and higher)</td>
<td>Athisman</td>
<td>English</td>
<td>$85 single $500 15 PC</td>
<td>Michael Barlow</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>not necessary</td>
<td>pre-definable</td>
</tr>
<tr>
<td>WST 4</td>
<td>PC (Win 95 and higher)</td>
<td>OUP</td>
<td>English</td>
<td>GBP 51.95 single GBP 262.90 10 users</td>
<td>Mike Scot</td>
<td>yes</td>
<td>yes</td>
<td>excellent 217 pages</td>
<td>not necessary</td>
<td>prompted</td>
</tr>
<tr>
<td>CC</td>
<td>PC (Win 2000/XP)</td>
<td>R.J.C. Watt</td>
<td>English</td>
<td>$99 single $40 for subsequent copies</td>
<td>R.J.C. Watt</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>not necessary</td>
<td>prompted</td>
</tr>
<tr>
<td>MLCP</td>
<td>JAVA Runtime Environment</td>
<td>na</td>
<td>English</td>
<td>Freeware</td>
<td>Scott Piao</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>not necessary</td>
<td>prompted</td>
</tr>
<tr>
<td>CA 4</td>
<td>PC (Win 98 and higher)</td>
<td>na</td>
<td>English</td>
<td>Freeware</td>
<td>Chris Greaves</td>
<td>no</td>
<td>no</td>
<td>online help</td>
<td>not necessary</td>
<td>prompted</td>
</tr>
<tr>
<td>TS 2.6</td>
<td>Windows, Linux, MacOS X</td>
<td>na</td>
<td>English</td>
<td>Freeware</td>
<td>Matthias Huming</td>
<td>no</td>
<td>yes</td>
<td>not necessary</td>
<td>prompted</td>
<td>prompted</td>
</tr>
<tr>
<td>AC 3.1</td>
<td>Windows (also v2.2.2. for Linux)</td>
<td>na</td>
<td>English</td>
<td>Freeware</td>
<td>Laurence Anthony</td>
<td>no</td>
<td>yes</td>
<td>not necessary</td>
<td>not necessary</td>
<td>prompted</td>
</tr>
<tr>
<td>ACC</td>
<td>JAVA Runtime Environment</td>
<td>na</td>
<td>English, Arabic</td>
<td>Freeware</td>
<td>Andrew Roberts</td>
<td>no</td>
<td>no</td>
<td>not necessary</td>
<td>not necessary</td>
<td>prompted</td>
</tr>
<tr>
<td>SCP 4</td>
<td>MacOS X, Win 95/XP</td>
<td>na</td>
<td>English</td>
<td>Freeware</td>
<td>Alan Reed</td>
<td>yes</td>
<td>yes &amp; video tutorial</td>
<td>installation file</td>
<td>prompted</td>
<td>prompted</td>
</tr>
<tr>
<td>WC 2</td>
<td>Win 98/XP</td>
<td>na</td>
<td>English</td>
<td>Freeware</td>
<td>Zdenek Martinek, Les Siegrist</td>
<td>yes</td>
<td>no</td>
<td>installation file</td>
<td>prompted</td>
<td>prompted</td>
</tr>
<tr>
<td>Table 2. Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Brown F: N=100k</td>
<td>Brown 1/10mil: N=1 million</td>
<td>BNCspoken: N=10 million</td>
<td>BNCtail: N=100 million</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Search items</td>
<td>low freq</td>
<td>mid freq</td>
<td>high freq</td>
<td>low freq</td>
<td>mid freq</td>
<td>high freq</td>
<td>low freq</td>
<td>mid freq</td>
<td>high freq</td>
</tr>
<tr>
<td>Perl script</td>
<td>search (sec)</td>
<td>1</td>
<td>0</td>
<td>3616</td>
<td>1</td>
<td>1</td>
<td>36349</td>
<td>0</td>
<td>130770</td>
<td>254928</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>3615</td>
<td></td>
<td></td>
<td>36106</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>n/a</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MCP 2.2.</td>
<td>search (sec)</td>
<td>1</td>
<td>0</td>
<td>3694</td>
<td>1</td>
<td>1</td>
<td>36400</td>
<td>0</td>
<td>218929</td>
<td>493258</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6954</td>
<td></td>
<td></td>
<td>69723</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>n/a</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>WST 4.0</td>
<td>search (sec)</td>
<td>1</td>
<td>0</td>
<td>3692</td>
<td>1</td>
<td>1</td>
<td>36324</td>
<td>0</td>
<td>218252</td>
<td>493511</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6958</td>
<td></td>
<td></td>
<td>69668</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>1</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td>10</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>search (sec)</td>
<td>1s</td>
<td>0</td>
<td>3676</td>
<td>1</td>
<td>1</td>
<td>36843</td>
<td>0</td>
<td>216992</td>
<td>465344</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6317</td>
<td></td>
<td></td>
<td>62690</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>n/a</td>
<td>4s</td>
<td></td>
</tr>
<tr>
<td>AC 3.1</td>
<td>search (sec)</td>
<td>1s</td>
<td>0</td>
<td>3706</td>
<td>1</td>
<td>1</td>
<td>36472</td>
<td>0</td>
<td>218984</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6977</td>
<td></td>
<td></td>
<td>69869</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>n/a</td>
<td>2s</td>
<td></td>
</tr>
<tr>
<td>SCP</td>
<td>search (sec)</td>
<td>1s</td>
<td>0</td>
<td>3696</td>
<td>1</td>
<td>1</td>
<td>36406</td>
<td>0</td>
<td>218984</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6976</td>
<td></td>
<td></td>
<td>6997</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>n/a</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CC</td>
<td>search (sec)</td>
<td>1</td>
<td>0</td>
<td>3694</td>
<td>1</td>
<td>1</td>
<td>36552</td>
<td>0</td>
<td>218984</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6906</td>
<td></td>
<td></td>
<td>69245</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>n/a</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>MLCT</td>
<td>search (sec)</td>
<td>1</td>
<td>0</td>
<td>3511*</td>
<td>2</td>
<td>17</td>
<td>38</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>5890</td>
<td></td>
<td></td>
<td>53512</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>31288</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>CA 4</td>
<td>search (sec)</td>
<td>1</td>
<td>0</td>
<td>3706</td>
<td>1</td>
<td>1</td>
<td>36863</td>
<td>0</td>
<td>70702</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6976</td>
<td></td>
<td></td>
<td>70702</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>CW 2</td>
<td>search (sec)</td>
<td>1s</td>
<td>0</td>
<td>3706</td>
<td>1</td>
<td>1</td>
<td>36863</td>
<td>0</td>
<td>70702</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>6976</td>
<td></td>
<td></td>
<td>70702</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
<tr>
<td>AC 3.1</td>
<td>search (sec)</td>
<td>1s</td>
<td>0</td>
<td>3666*</td>
<td>1</td>
<td>1</td>
<td>3615</td>
<td>0</td>
<td>70702</td>
<td>n/a</td>
</tr>
<tr>
<td></td>
<td>hits</td>
<td></td>
<td></td>
<td>1000</td>
<td></td>
<td></td>
<td>70702</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>saving (sec)</td>
<td>n/a</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>10</td>
<td>n/a</td>
<td>n/a</td>
<td></td>
</tr>
</tbody>
</table>

Note: Performance measures cannot be provided in those cases where benchmarking failed due to RAM limitations. In other cases, a query provided no hits and the criterion ‘saving time’ could not be applied. Both cases are marked as ‘non-applicable’ (n/a).

358

359
Table 3. Range of functions

<table>
<thead>
<tr>
<th>Input</th>
<th>MCP 2.2</th>
<th>WST 4</th>
<th>Con-cordance</th>
<th>MLCP</th>
<th>CA 4</th>
<th>AC 3.1</th>
<th>ACC</th>
<th>SCP</th>
<th>CW 2</th>
<th>TS 2.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicode support</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>search/load</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>multiple files</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>incl. subdirectories</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>on-the-fly corpus change</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpus</th>
<th>No of texts in MB, sentences, words, characters</th>
<th>number of selected texts</th>
<th>type/token (ratio), characters, sentences</th>
<th>shows full corpus in window</th>
<th>number of files, length in characters</th>
<th>type/token frequency of word list, file size, character encoding</th>
<th>files, lines, type/token freq.</th>
<th>files, mb, sentences, words</th>
<th>type/tokens, number of files, total file size</th>
</tr>
</thead>
<tbody>
<tr>
<td>text search</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>regular expression</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>tag search</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>batch search</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>toggle case sensitivity</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>wild cards</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>key words</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>stop words</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Queries</th>
<th>absolute frequencies relative frequencies dispersion (plot)</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
<th>yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-gram</td>
<td>n-gram span 2-5 words, reverse, 3 consecutive searches</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>up to 6-grams</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>collocate space</td>
<td>definable span L50-R50</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>statistics</td>
<td>4 association measures</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
Table 3. (continued)

<table>
<thead>
<tr>
<th>possible context type</th>
<th>a. characters</th>
<th>b. words</th>
<th>c. sentences</th>
<th>d. lines</th>
<th>e. full line</th>
<th>f. characters</th>
<th>g. characters</th>
<th>h. characters</th>
<th>i. words</th>
<th>j. non-definable</th>
<th>k. characters</th>
<th>l. characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>context size o-40, p 0-7, q 0-10, r 0-10</td>
<td>yes, up to 3 levels</td>
<td>yes, up to 3 levels</td>
<td>full line</td>
<td>definable</td>
<td>yes, up to 3 levels</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes, up to 3 levels</td>
<td>no</td>
<td>no</td>
<td>yes, up to 9 levels</td>
</tr>
<tr>
<td>levelled sorting</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>distribution plotting</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>suppress tags suppress search terms</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Output

| formats | txt, html | txt, xls, xml, program format | txt, html, program format | txt | rtf, txt, html | txt, html, xls, program format | txt, html | rtf, html, txt | txt | doc, xls, txt |

Notes:

a. The criterion ‘incl. subdirectories’ states whether or not the user can specify a top-level directory in which all files are searched, regardless of the embedded directory structure.

b. The criterion ‘on-the-fly corpus change’ means that the list of texts loaded for query can be modified in respect to individual items, without the need to unload it and load it again.

c. The criterion ‘tag search’ states whether or not the program provides an in-build function to use the corpus mark-up as part of the search query, without the user having to incorporate it manually in every query.

d. The criterion ‘n-gram’ is to be understood to mean adjacent n-grams.
Offenlegung der Inhaber und Beteiligungsverhältnisse gem. § 7a Abs. 1 Ziff. 1, Abs. 2

Ziff. 3 des Berliner Pressegesetzes: Gisela Cram, Rentnerin, Berlin; Dr. Annette Lubasch, Ärztin, Berlin; Elsbeth Cram, Pensionärin, Rosengarten-Alvesen; Dr. Hans-Robert Cram, Verleger, Kleinmachnow; Margret Cram, Studienrätin i. R., Berlin; Verena Graß, Schülerin, Leimen; Brigitta Duvenbeck, Oberstudienrätin, Bad Homburg; Liselotte Schuchardt, Ärztin, Berlin; Dr. Georg-Martin Cram, Unternehmens-Systemberater, Stadtbergen; Jens Cram, Student, Stadtbergen; Renate Tran, Zürich; Gudula Gädeke M.A., Atemtherapeutin/Lehrerin, Tübingen; John-Walter Siebert, Pfarrer, Walheim; Dr. Christa Schütt, Arztin, Mannheim; Dorothee Seils, Apothekerin, Stuttgart; Gabriele Seils, Journalistin, Berlin; Walter Cram, Architekt, Mexico DF (Mexiko); Ingrid Cram, Betriebsleiterin, Tuxpan/Michoacan (Mexiko); Sabina Cram, Mexico DF (Mexiko); Dr. Clara-Eugenie Seils, Oberstudienrätin i. R., Reppenstedt; Christoph Seils, Journalist, Berlin; Angelika Crisolli, kaufm. Angestellte, Hohenstein; Susanne Cram-Gomez, Mexico DF (Mexiko); Kurt Cram, Großhändler, Cancun (Mexiko); Silke Cram, Wissenschaftlerin, Mexico DF (Mexiko).