Talc 2000
July 19-23, Graz, Austria

XKWIC: A powerful concordancer for research (Part II)

Workshop by
David Lee (david_lee00@hotmail.com)
& Paul Rayson (paul@comp.lancs.ac.uk)
Lancaster University

1.1 The Xkwic input file format

XKWIC[1] uses files prepared to a 'vertical' format such as the following:

word	pos	jpos	Lemma	sem	file**
There	EX	EX	THERE	Z5	w/W_ac_hum/A04
is	VBZ	VVBZ	BE	A3+	w/W_ac_hum/A04
no	AT	DD	NO	Z6	w/W_ac_hum/A04
need	NN1	NN1	NEED	S6+	w/W_ac_hum/A04
to	TO	TO	TO	Z5	w/W_ac_hum/A04
be	VBI	VABI	BE	Z5	w/W_ac_hum/A04
intimidated	VVN	VV0P	INTIMIDATE	E5-	w/W_ac_hum/A04
by	II	II	BY	Z5	w/W_ac_hum/A04
the	AT	DD	THE	Z5	w/W_ac_hum/A04
formality	NN1	NN1	FORMALITY	A6.2+	w/W_ac_hum/A04
of	IO	IO	OF	Z5	w/W_ac_hum/A04

(**The field called "file" actually includes more than the filename. The three types of information are: *mode/genre/filename*, where *mode* can be either spoken or written, *genre* is one of the abbreviated genre labels used in the research corpus, and *filename* is the 3-character BNC filename/ID.)

The fields and their labels (in this case *word*, *pos*, *jpos*, *lemma, sem*, and *file*) are first chosen by the user and 'registered' with XKWIC, so that the program 'knows' the fields you want to base your searches on. The number of such fields depends, of course, on how many kinds of tag or information fields you want to search by. Queries based on any combination of the fields (using full regular expressions and Boolean operators) can be then be made (e.g. you can specify that you want a concordance of the word *need* but only if it is a noun, and only if it is found in the "written, academic prose, humanities" genre, and only if in the specific file called "A04"). In addition, a limited number of what XKWIC calls 'structural attributes' (i.e. text organisational units such as sentences and paragraphs, and representational attributes such as bold and italics, all of which are typically rendered in SGML in contemporary corpora) may also be encoded. These cannot be logically combined with the fields just described above ('word', 'pos', 'jpos' etc. which are 'positional attributes') but may be referred to and used in queries to a limited degree (see discussion of 'within s' below). Thus, when not referred to specifically, structural attributes are 'invisible' to XKWIC rules, neither restricting nor blocking them.

What follows is a brief explanation of the XKWIC query syntaxes. Full explanations may be found in the manuals and references.

1.2 Key to the XKWIC query syntax

`.`	matches any single character
`*`	(Kleene star or closure operator) matches sequences of arbitrary length (including zero) of its preceding argument (e.g. `[word="R.*"]` will match any word beginning with capital 'R' and followed by zero or more characters).
`+`	matches sequences of at least length 1 of its preceding argument (e.g. [word="test.+"] will match *testing, tested, tests*, etc., but not *test* itself.
`?`	(omission operator) makes the preceding argument optional (e.g. `walks?` matches *walk* and *walks*, with *s* being the preceding argument in this case)
`|`	(disjunction operator) matches arguments on both sides of the operator (e.g. `[pos="I.*|R.*"]` matches all prepositions and adverbs).
`!`	(negation operator)
`[abcd]`	(square brackets when used for listing) makes every character enclosed within the brackets an alternative (e.g. `"[Bb]all"` matches *Ball* and *ball*; e.g.2. `[abcd]` is equivalent to `[a|b|c|d]`; e.g.3 `[A-Za-z]` matches all letters of the alphabet).
`[]`	denotes any word form (`[]*` thus matches zero or more arbitrary word forms)
`{}`	(interval operator) This occurs in 3 forms: `{n}`: exactly `n` repetitions of previous expression `{n,}`: at least `n` repetitions of previous expression `{n,m}`: between `n` and `m` repetitions of previous expression e.g. `[pos="R.*"]{1,3}` will match at least one and at most 3 adverbs.
`%c`	makes the preceding expression case insensitive (e.g. `[word="my"%c] matches` *my, My, mY,* `and` *MY*.)
`<s>`	matches any sentence boundary marker (i.e. the punctuation marks `!`, `"`, `.`, `:`, and `?`)
`\`	('quote' character) makes XKWIC treat the following character(s) literally or in special way. (e.g. `[pos="\?"]` matches all instances of the question mark: the quote character "\" forces XKWIC to treat the symbol `"?"` not as the omission operator, but literally, as the symbol to be matched. Another function of "\" is to enable foreign characters (e.g. those with diacritics, like the German umlaut) to be searched (e.g. to find the word `Spätzle`, the query may be written as: "`Sp\344tzle`" (where "344" is the octal code of a specific character set) or "`Sp\"atzle`" (in the LATEX format)

[1] XKWIC is the Motif-based graphical interface to CQP, the concordancer engine itself, which is part of the IMS Corpus Workbench, a set of tools developed at the Institut für maschinelle Sprachverarbeitung at the University of Stuttgart. Detailed information can be obtained from http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench

`[label]:` allows agreement or value congruence between two 'positions/words' (or, technically 'attribute expressions'), e.g. the rule:

```
y:[pos="I.*"] [pos=","] [word=y.word]
```

matches cases of repeated prepositions separated by a comma (e.g. "`This will be shown in, in the next slide`"). Whatever value for '`word`' the labelled expression (i.e. in this example, `[pos="I.*"],` labelled by the arbitrary label "`y:`") takes, the same value will be matched in the subsequent expression referencing that label (i.e. `[word=y.word],` where "`y.word`" here not is a literal string but refers to whatever value the previously referenced labelled expression took).

`MU((meet ...))` This is an entirely optional syntax prefix which simply makes XKWIC run more quickly and efficiently on some kinds of query (viz. those that consist of only 1 (without the '*meet*' syntax) or 2 arguments (with '*meet*'). Where used in the rules below, it is put in superscript so as to make reading of the algorithms easier.

`within s` This syntax suffix (tagged on to the end of a query) restricts matches to those which lie within a sentence boundary (i.e. between the structural attributes encoded as `<s>` and `</s>`), and is thus only logically necessary for rules which span two or more word units. Thus, a rule looking for "an adjective followed by a noun" (e.g. attributive adjectives) will **not** match cases where a sentence ends with an adjective and the following one begins with a noun (e.g. Nana's delighted_JJ. Mum_NN1! isn't she? [KB3]).

1.3 Comparison of XKwic & WordSmith

	Xkwic	**WordSmith**
Platforms/OSs	UNIX, Linux, Java	MS Windows©
Price	Free for educational use	Not free, but inexpensive
Ease of installation	Not easy	Easy
Ease of setting up corpus/texts	Needs reformatting of texts and indexing	On-the-fly processing of any text(s)
Speed	Fast even on large corpora (because pre-indexed)	Depends on size of corpus, but generally slower than Xkwic
User-friendliness	Steep learning curve, but powerful and complex searches can be made	Reasonably easy to learn
Query Syntax	Powerful – full regular expression searches	Less complex searches
Discontinuous constructs	Easy to capture using interval operators, scoping restrictions & label-matching feature	Fiddly to capture – require workarounds to refer to intervening context (e.g. phrasal verb with NP between V.* and particle)
SGML handling	Some structural attributes can be used to scope queries (e.g. 'within s'), but limited in number	Some handling of SGML entities – also to a limited degree
Whole-text browsing	Not really possible	Possible – integrated browser
Referencing of File IDs and other information fields	Information fields need to be explicitly linked to every single word – quite fiddly & wasteful of space, but allows sophisticated sub-corpus searches limited only by your coding	File IDs easily referenced if used as filenames, but other information fields will need to be coded as part of filename (e.g. W_conv_KSW) and further pre- or post-processing required if sub-corpus searches needed
Advanced Features	No frills – only frequency distributions	Excellent statistical analyses: Keywords (χ^2, log likelihood), Key Keywords (dispersion), other text-formatting tools
Collocational Searches	Can be POS-based: e.g. frequency distributions of all noun tokens up to 3 words to left of node	Word-based
Concordance Output for presentation	If File IDs needed, output will need further processing	Output generally useable/presentable straightaway

Conclusion:

- Xkwic's main advantage: speed, sophisticated query syntax, sub-corpus searches
- Well worth learning if you have time and determination or need to count linguistic features which are otherwise impossible to capture.

1.4 Example Queries

An illustrative example of the use of XKWIC syntax and yet another of its features is in the rule for capturing all **punctuation marks**:

```
[pos!="[A-Z].*"]
```

(which is equivalent to `[pos=".|\.\.\.|__UNDEF__"]`)

Punctuation marks in the CLAWS tagset[2] are simply copied over to the part of speech column (and all other annotation fields): which means that they are all non-alphabetic. The above rule exploits this fact by capturing all POS tags where the first character is not a member of the set 'A to Z' in order to match all punctuation marks. The alternative way of doing this, also shown above, is to capture all POS tags with only one character, or three dots (the ellipsis marker), or the tag "__UNDEF__" (which marks structural SGML entities not recognised by XKWIC due to software limits).

Another illustration of the query syntax is in the following XKWIC rule for **'total number of words'**:

```
[pos="[A-Z].*" & pos!=".*[0-9][0-9]|FU"] | [pos=".*[23456]1"]
```

This rule is specific to the CLAWS C7 tagset. It counts all non-punctuation marks, non-multiwords, non-fragmented-words, and non-CLAWS-unrecognised words, then adds a count of the **first** parts of all multiwords (i.e. any POS tag ending with two digits, where the last digit is '1'). In other words, the rule captures all CLAWS-recognised 'words', treating multiwords as one unit. This probably differs from the way most people count 'words', especially in the exclusion of word fragments or unrecognised words (POS= "FU"), which are especially common in spoken texts.

1.5 The search algorithms

For comparison, references are made below to the way Biber counted his linguistic features in his 1988 book *Variation across Speech and Writing*.

1. past tense:

```
[pos="V.+D.?"]
```

(equivalent to: `[pos="V.*D" | pos="VBDZ" | pos="VBDR"]`, i.e. all lexical verb *-ed* forms, including *had* and *did,* plus *was* and *were*.)

Biber identified past tense as follows: "Any past tense form that occurs in the dictionary, or any word not otherwise identified that is longer than six letters and ends in ed#. Past tense forms have been edited by hand to distinguish between those forms with past participial functions and those with past tense functions" (p.223).

With CLAWS C7 tags, however, the identification of past tense forms is more straightforward, as *-ed* forms with past participial functions are differently tagged (viz. V.N) by CLAWS.

2. third person personal pronouns (excluding *it*):

```
[pos="PPH[SO]."]|[word="[h']is"%c]|[word="[h']er.*"%c &
pos=".*PP[GX].*"]|[word="their"%c]|[word=".*msel[fv].*"%c]
```

3. agentless passives:

Rule 1/4:

```
[pos="VB.*"][pos!="V.*|.|N.*|P.*|DD.*|CS.*|AT|AT1|APPGE"]{0,4}
[pos="VVN"] [pos="I.*|R.*" & word!="by"%c]{0,3} [word!="by|."]{0,2}
[word!="by"%c] within s
```

Rule 2/4:

```
[pos="VB.*"][pos="I.*" & word="to|in"] [pos!="V.*|."]{0,4}
[pos="VVN"][pos="I.*|RR" & word!="by"%c]?
[word!="by"%c]{0,4}[word!="by"%c] within s

These were edited by hand.
```

In Rule 2/4, I have expanded Biber's rules in order to catch those cases where (mostly parenthetical) prepositional phrases such as "in fact", "in other words", "in no way", and "to some extent" (which are blocked by Rule 1) to come in between BE and the passive form.

Rule 3/4 (Question forms):

```
<s>[pos="VB.*"][]{0,3}[pos="N.*|P.*|AT.*|APPGE"][pos="V.*N"][]{0,4}
[word!="by"%c] within s

Checked manually.
```

Rule 4/4:

```
Manual additions: a small number of manually identified cases (not
covered by the above rules) spotted during checks on other features
(e.g. WHIZ_VBN).
```

Notes:

`[w/W_b_bus/A8G]`

1. The manual additions (which were not captured by the above rules) included cases such as the following:

   `The data was converted to average daily figures and comparisons made in this way.`

   `[w/W_na_pol/HCC]`

   (→ "comparisons were made in this way")

[2] In some versions of the BNC tagged with the C7 tagset, however, punctuation marks have been given tags beginning with Y, such as YCOM, YQUE and YSTP for the comma, question mark and full-stop respectively, instead of having the actual punctuation mark represent itself.

Furthermore, the figure stated…probably also includes a number of those who wanted restrictions placed on the traditional model of sole practice [w/W_na_pol/J1P]

(→ "wanted restrictions to be placed on…")

Also manually included were newspaper headlines, which stereotypically use passive constructions without a 'BE':

Workers kicked in teeth says TUC boss

MPs misled over arms says John Smith

Sane sex attacker jailed for nine years

2. Biber's algorithms did not allow for passive constructions separated by parenthetical adverbs which are set off by commas. The above algorithms do.

4. ***that* adjective complements** (e.g. *I'm glad that you like it*):

```
[word!="so"][pos="JJ"][pos="FU|UH|R.*|."]{0,5}[pos="CST"]
```

Notes:

1. Biber's algorithm did not prohibit 'so' from preceding the adjective. Thus, degree complement clauses, as in the following example, would have been included:

   The steps which will have to be taken are, in my view, so grave, that it becomes a question whether any one party can carry them through [A6G]

   Strictly speaking, these modify the degree adverb rather than the adjective, and therefore should not be included. I have adjusted the algorithm accordingly.

2. No 'within s' limitation (i.e. 'restrict algorithm to match only within a sentence') was put, as a manual examination showed that all the examples which crossed a sentence boundary were from poems, where the run-on lines were treated as sentences by CLAWS.

3. Allowing the optional elements ([pos="FU|UH|R.*|."]{0,5}) to intervene between the adjective and 'that' meant that the results had to **edited by hand,** to weed out incorrect matches. Those who want a 'quick and easy' algorithm may omit this part.)

5. ***that* relativizer in subject function** (e.g. *the dog that bit me*):

```
([pos="N.*|PN1"]|[word="any|those"])[pos="CST"][pos="R.*"]?[pos="V.*"] within s
```

Biber's original label for this feature and the next was '*that* relatives on subject/object position'. I have renamed this and the other related features to make the meaning more transparent. I have extended Biber's algorithm by including pronouns (instead of just nouns).

6. ***that* relativizer in object function** (e.g. *the toy that I bought*):

```
([pos="N.*|PN1"]|[word="any|those"])[pos="CST"][pos="R.*"]?[pos="D.*|PP.S.|APPGE|PPH1|J.*|N.*2|NP.*|NNB|AT.*|M.*"] within s
```

I have extended Biber's algorithm by including pronouns (instead of just nouns). As Biber warns, this algorithm does not distinguish between *that*-complements to nouns and true relative clauses.

7. **stranded prepositions (e.g. *the candidate that I was thinking of*):**

```
[pos!= "."] a:[pos = "I.*"][pos = "." & pos!= "\"|\(|:"] [word!= "for" & word!=a.word]
```

The above rule significantly improves on Biber's algorithm in 4 ways:

(1) *'Example' parentheticals are excluded*: The restriction [word!="for"] rules out cases of parentheticals like 'for instance/example' used immediately after prepositions: e.g. "babies **of**, for instance, Pakistani mothers". In such cases, the preposition before the comma is not really 'stranded' and should therefore not be counted.

(2) *Repeated prepositions are excluded*: The 'label reference' feature of XKWIC used in the above rule is a very powerful and efficient way to weed out rogue examples such as:

Are you still completely confident **in, in** finishing? [s/conv/KCV/003302]
Well I'm blowed if I saw it **on, on** that receipt. [s/conv/KBF/015303]

Such repeated prepositions are, of course, very common in spontaneous spoken discourse, as well as in fictional representations of natural speech:

Then you are not angry **about — about** the duke? [w/prose/CCD/CBride]

Such cases might be included in measures of disfluencies or repetition, but do not constitute stranded prepositions.

(3) *Prepositions which occur between punctuation marks (this includes sentence-initial prepositions) are excluded*:

e.g. **Unlike**, however, the 1958 Notting Hill riots, few of those involved...

this system in relation to sex equality rights had lain dormant, **as**, indeed, had the potential significance of the Community principles

A manual check showed that these may be safely disregarded.

(4) *Prepositions occurring before colons are excluded*:

Based on a careful examination of concordance examples, the vast majority (99.2%) of such instances are not examples of stranded prepositions, but instead illustrate various orthographic conventions:

e.g. `Other householders MUST put refuse for collection` **`into:`** `— TIED PLASTIC SACKS which must be strong enough...`

`Send orders` **`to:`** `Daily Mirror...`

`she developed a marketing plan which aims` **`to:`** `increase general awareness of the...`

In addition, there were the ubiquitous cases of headers in the e-mail texts:

`e.g.` **`From`** `: Date : Tue, 4 Jan 94 10:34:28 GMT`

Biber's algorithm does not weed out **any** of the above cases, which in my spoken corpus amounted to more than 26% of the total cases when the restrictions were not there, and in my written data amounted to more than 38%. This suggests that if I had counted stranded prepositions the way Biber did, I would have obtained higher counts, but with a higher error rate.

8. phrasal coordination (*noun* and *noun*; *adj* and *adj*; *verb* and *verb*; *adv* and *adv*):

```
a:[pos="N.*|J.*|V.*|R.*" & pos!="NP.*|NNB"] [word="and|an'"%c &
pos="CC"][pos=a.pos]
```

"NP1" would have included, for example, *Tyne and Wear*, *John and Mary*, and "NNB" would have counted *Mr and Mrs*. Thus, proper nouns and terms of address are excluded from the algorithm. The above rule uses the labelling function of XKWIC to ensure that both sides of the coordinator are the same part of speech (i.e. `adj` and `adj`, `noun` and `noun`, etc.).

9. clause coordination:

In Biber 1988, this feature was called 'independent' clause coordination, but his rules actually counted co-ordinated subordinate clauses as well. Since the main aim is to count '*and*' used as clausal coordinator (as opposed to a phrasal coordinator, which is counted separately) there is no reason to restrict the counts to only independent clauses. Also, there is no reason to limit oneself (as Biber does) to the word *and*. I have thus included *but, or* and *nor* as well.

Rule 1/2:

```
[pos!="[A-Z].*"] [pos="CC.*"] ([word="it|so|then|you"%c] |[word
="there"][jpos="V.B.*"]|[jpos="PD.*|PP.S.*"])
```

This captures those cases where a coordinator occurs after a non-clause-punctuation mark (e.g. commas), and also where it occurs after a semi-colon and colon.

Rule 2/2:

```
[pos!="[A-Z].*"] [word="[A-Z].*" & pos="CC.*"]
```

By restricting cases to those where a coordinator begins with a capital letter, this rule captures all clause-initial cases.

10. attributive adjectives (e.g. *the big horse*):

Biber's method appears to be to treat adjectives as being either attributive or predicative. There is, however, a third class of 'leftovers', which, strictly speaking, is neither, but can be treated as predicative for practical purposes. This is the class of adjectives such as *mad* in "He made me *mad*". One way of classifying adjectives is therefore to first count all attributive adjectives using the rules given below. Subtracting these from the set of all adjectives then leaves us with the predicative and 'leftover' adjectives. This method is recommended because attributive adjectives are easier to automatically identify. The rules used for finding attributive adjectives are:

```
(a) [pos="J.*"][pos="J.*|N.*|PN1|M.*"] within s
(b) [word="the|a|an"%c] [pos="J.*"]
    [pos!="J.*|C.*|N.*|R.*|PN1|V.*|M.*"] [pos!="N.*|C.*|PN1"]{3}
    within s
(c) [word="the|a|an"%c] [pos="J.*"] [pos="R.*|."]{0,3} [pos="V.*"]
    within s
(d) [pos="J.*"][pos="CC.*|RR|RG[RT]?"] [pos="J.*"] [pos="N.*|PN1|MC"]
    within s
```

Rule (d) represents an improvement to Biber's algorithm, allowing a succession of adjectives with a conjunction or certain adverbs in between:

- "The brave and courageous people"
- "…with certain very clear objectives in pushing forward"

1.6 References & Websites

Xkwic Website: http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/

Brew, Chris & Marc Moens (1999) *Data Intensive Linguistics*. HCRC Language Technology Group: University of Edinburgh. (Edition: 15 Feb 1999). Available as HTML at http://www.ltg.ed.ac.uk/~chrisbr/dilbook or as gzipped Postscript at http://www.ltg.ed.ac.uk/ chrisbr/dilbook.ps.gz

Christ, Oliver (1994) A modular and flexible architecture for an integrated corpus query system. *Proceedings of COMPLEX'94: 3rd Conference on Computational Lexicography and Text Research* (Budapest, July 7-10 1994). Budapest, Hungary. pp23-32.

Christ, Oliver, Bruno Schulze, Anja Hofmann & Esther König (1999) *The IMS Corpus Workbench: Corpus Query Processor (CQP) User's Manual*. Institute for Natural Language Processing, University of Stuttgart. (CQP version 2.2)

~~~ * ~~~
~~~