



Cambridge Sketch Engine

Using Corpus Query Language (CQL) ^(1.3)

Understanding and using Corpus Query Language (CQL) (version 1.1)

This guide outlines how Corpus Query Language can be used in order to carry out complex searches using Sketch Engine. It is intended to be used in conjunction with the other documentation (e.g. [Getting Started](#) and [Advanced Help](#)), and so presumes an element of prior knowledge of Sketch Engine.

This guide is a working document - please contact corpus@cambridge.org if you have any queries or suggestions. All comments welcome!

CONTENTS

1. [What is Corpus Query Language \(CQL\) and what can you do with it?](#)
2. [Using CQL with Sketch Engine](#)
3. [Basic principles of CQL](#)
4. [Searching for words, lemmas and Parts of Speech \(POS\)](#)
5. [Searching for adjacent words, lemmas or POS](#)
6. [Searching for more than one word /lemma/POS at the same time](#)
7. [Understanding and using wildcards](#)
8. [Using wildcards](#)
9. [Combining words and tags](#)
10. [Leaving gaps in your searches](#)
11. [Excluding elements](#)
12. [Optional elements](#)
13. [Searching for punctuation](#)
14. [Using CQL to look at grammatical variation](#)
15. [FAQs: CQL search strings](#)

1. What is CQL and what can you do with it?

Corpus Query Language (CQL) is a powerful function that allows you to search the Corpus for complex grammatical patterns.

In Sketch Engine, typically, you can search the Corpus by typing words into the query box. As mentioned in the [Getting Started](#) guide, along with the default **simple** search, it is possible to run a **lemma**, **phrase**, **word form**, or a **CQL** search. Typing in your own queries using CQL gives you greater control over the patterns you search for.

CQL uses a particular notation (described in more detail below) in order to run these searches.

CQL allows you to:

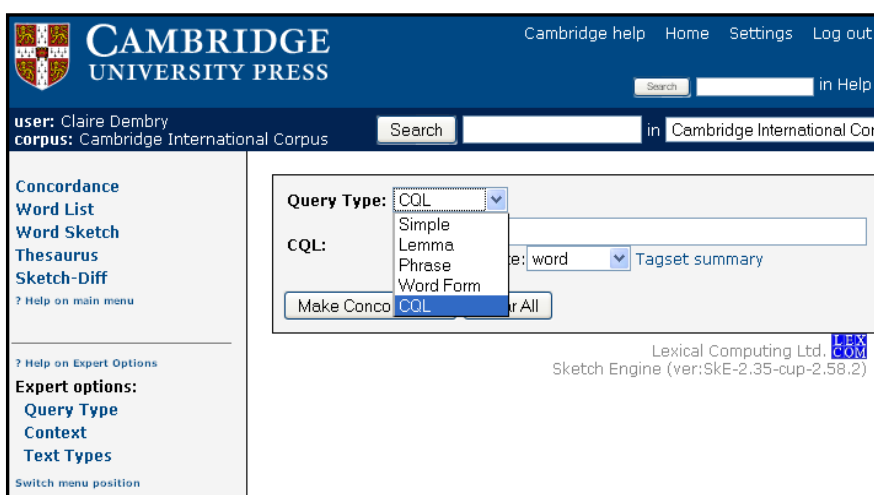
- **Search for grammatical constructions.** For example, it is possible to search for present progressive verbs, or all adjectives occurring directly before nouns.
- **Search for gapped constructions.** For example, it is possible to search for "*more X than*" and return results such as "*more important than*" and "*more likely than*"
- **Search for words that contain a particular string of characters.** For example, it is possible to search for all words ending in "*-ical*" or starting with "*pre-*"

CQL also allows you to run many other searches; these are described in more detail in the subsequent sections.

2. Using CQL with Sketch Engine

To enter a CQL query in Sketch Engine:

- Choose the corpus you'd like to work with from the Sketch Engine homepage
- Display the *Query Type* menu, by selecting it from the lower left hand side menu
- Select CQL from the dropdown, as shown below.
- Type your CQL search string into the *CQL:* box situated underneath the dropdown.



The screenshot shows the Cambridge University Press Sketch Engine interface. At the top, there is a navigation bar with links for 'Cambridge help', 'Home', 'Settings', and 'Log out'. Below this, the user's name 'Claire Dembry' and the selected corpus 'Cambridge International Corpus' are displayed. A search box is present with a 'Search' button. On the left side, there is a menu with options: 'Concordance', 'Word List', 'Word Sketch', 'Thesaurus', 'Sketch-Diff', and 'Expert options'. The 'Expert options' menu is expanded, showing 'Query Type' selected. A dropdown menu for 'Query Type' is open, listing 'Simple', 'Lemma', 'Phrase', 'Word Form', and 'CQL'. The 'CQL' option is highlighted. Below the dropdown, there is a text input field for the CQL query, a 'Tagset summary' dropdown, and buttons for 'Make Concordance' and 'For All'. At the bottom right, the version information 'Sketch Engine (ver:SkE-2.35-cup-2.58.2)' is visible.

TIP: When using CQL, it's useful to open the [list of grammar tags](#) in a separate window, for reference.

To find the list of grammar tags (i.e. *Pos tags*) in Sketch Engine, click on *Tagset summary* found underneath the search box, as shown opposite.

3. Basic principles of CQL

- Each element of a query has to be enclosed in **square brackets**: []
- **Quotation marks**: " " enclose specific search items. Usually this is the particular word or lemma (e.g. "*dog*" or "*make*"), or the particular Part of Speech (POS) tag, (e.g. the tag for adverbs – "*RB*", or the tag for possessive pronouns – "*PP*").
- The type of search, e.g. for a word, lemma or tag also needs to be specified, by typing **word=** or **lemma=**, etc in the query.
- For example, to search for the word *find* using CQL, you would type: **[word="find"]**

A search for the word find would therefore look like this if you typed it into Sketch Engine:	<p>Query Type: <input type="text" value="CQL"/></p> <p>CQL: <input type="text" value='[word="find"]'/></p> <p><input type="button" value="Make Concordance"/> <input type="button" value="Clear All"/></p>
--	--

4. Searching for words, lemmas and Parts of Speech (POS)

Following from the example set out above with *find*, searching for single words, parts of speech or lemmas are easy! This is best demonstrated using examples:

<p>To search for the word <i>mind</i> type: [word="mind"] To search for the word <i>cake</i> type: [word="cake"]</p> <p>To search for the tag for adjectives (which is JJ) type: [tag="JJ "] To search for the tag for coordinating conjunctions (which is CC) type: [tag="CC"]</p> <p>To search for the lemma <i>be</i> type: [lemma="be"] To search for the lemma <i>run</i> type: [lemma = "run"]</p>
--

(Of course, any word, POS or lemma can be substituted in between the inverted commas). Remember to look at the [grammatical tagset](#) for a list of all of the POS tags used in Sketch.

5. Searching for adjacent words, lemmas or POS

As mentioned in [Section 3](#) (basic principles of CQL), square brackets delineate search elements. To search for two or more adjacent elements, simply write the search string one after another. This is shown below:

To search for the adjacent words *mind* and *your* type: **[word="mind"][word="your"]**

To search for adjectives (JJ) with an adjacent coordinating conjunction (CC), type: **[tag="JJ "][tag="CC"]**

To search for the lemma *be* adjacent to the lemma *run*, type: **[lemma="be"][lemma="run"]**

NOTE: Adjacent elements don't have to be of the same type. For example, you can search for:

- a POS followed by a word (in this case, adjective + *man*): **[tag="JJ "][word="man"]**

This returns results such as *lovely man*, *old man*, *strange man*.

Also, you can search for any number of elements, providing they are separated by square brackets:

- to search for the words *very* + *quickly* + the lemma *go*: **[word="very"] [word="quickly"] [lemma="go"]**

- to search for a singular noun + modal + the word *go*: **[tag="NP"] [tag="MD"] [word="go"]**

This returns results such as *Jim might go*, *Laura could go*, *Tony will go*.

Using this principle we can begin to see how it is possible to build up complex queries.

6. Searching for more than one word /lemma/POS at the same time

It is possible to search for more than one word, POS or lemma at the same time, using CQL. For example, you can search for the words *small*, *tiny* and *miniscule* and get results for all of these together. Or, you can search for both adjectives and adverbs and see the results together.

To do this, you need include the words or POS within the same square brackets. The words need to be separated by a vertical bar: |

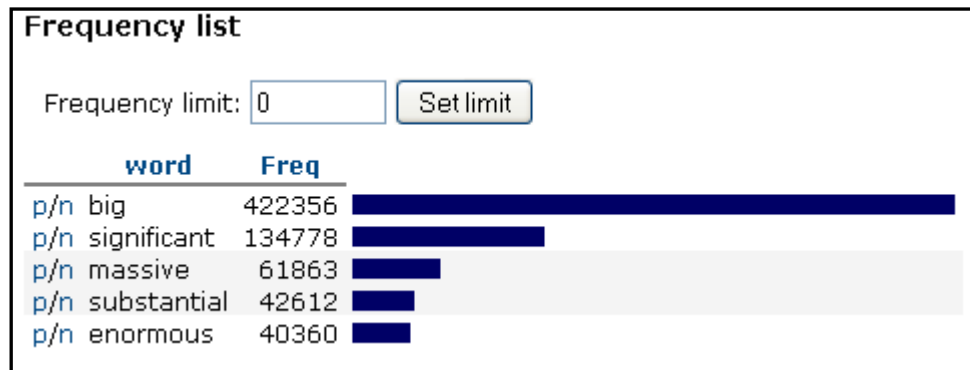
This bar can be found here on most keyboards as shown circled in red:



The CQL search string for the combined results for *small*, *tiny* and *miniscule* would therefore look like this:
[word = "small|tiny|miniscule"]

This same principle applies for two POS tags. E.g. the search string for the combined results for adjectives and adverbs would look like this: **[tag="JJ|RB"]**

Searching for more than one word using CQL is quite neat as by clicking on *Node forms* under *frequency* on the left hand side menu bar you can generate a graph for the results, as shown below.



CQL search string: [word="enormous|big|massive|significant|substantial"]

7. Understanding wildcards

Wildcards are used in order to substitute for any other character or characters in a string. The CQL wildcards that can be used in Sketch Engine are shown below:

wildcard	use	example string	example results
.	full stop	substitutes for one character only	[word="to."] <i>top, tot, too, toy, tow</i>
*	asterisk	repeats a preceding wildcard – can be used in conjunction with full stop	[word="*.ing"] <i>going, working, finding, saying, calling</i>
!	exclamation mark	Excludes an element - see Section 10	[lemma="fast" & tag != "AJ0"] fast as a noun, verb and adverb, but not as an adjective.
?	question mark	Makes an element optional - see Section 11	[word="did"] [word="not"]? [word="go"] <i>did not go and did go</i>
/	forwardslash	Protects the elements sometimes used as wildcards, e.g. punctuation. See Section 12	[word="\."] this will return instances of full stops found in the texts.

8. Using wildcards

We can use wildcards to make searches less complex. For example, if we wanted to search for all nouns, we could use the string **[tag="NN|NNS|NP|NPS"]** – i.e. search for the values where the tag is NN or NNS or NP or NPS.

However, it would be much easier to instead search for **[tag="N.*"]** This search means: search for the values where the tag starts with N and is followed by any other letter. Therefore, this search would therefore find all tags that start with N (i.e. those shown in the table below)

NN	noun, singular or mass
NNS	noun plural
NP	proper noun, singular
NPS	proper noun, plural

We can also use the **.*** wildcard to search for words that have letters in common.

For example, **[word=".*ing"]** would return: *walking, running, saving, exploiting*
[word="pre.*"] would return: *prefer, pretence, prelude*
[word="b.*n"] would return: *bun, bitten, bargain*

NB- these wildcards can also be used in other places within Sketch. See Using the [Cambridge Learner Corpus](#) and [Advanced Help](#) for more details on using wildcards along with error tags and wordlists

9. Combining words and tags

It is possible to be more specific about the word you are searching for. Words and POS can be combined to search for, e.g. *impact*, **only** as a verb, or *record* **only** as a noun.

This is achieved using the ampersand symbol: &

The symbol must appear **within** the square brackets of a search string.

For example, to search for the lemma "impact" as any noun: **[lemma = "impact" & tag = "N.*"]**
To search for the lemma "impact" as any verb: **[lemma = "impact" & tag = "V.*"]**
To search for the word "record" as a noun: **[word = "record" & tag = "N.*"]**

These search items can also be combined, to search then more than one word/lemma/POS.

For example, to search for the words "flood" or "dam" as a verb: **[word="flood|dam" & tag="V.*"]**
To search for the word "baby" as a noun or verb: **[word="baby"&tag="N.*|V.*"]**

10. Leaving gaps in your searches

It is possible to leave gaps in your search. For example, you may want to search for two words occurring next to each other, but with a number of words between them. This is possible in two ways:

Leave an empty set of brackets to indicate a word, (shown here in red):

For example: **[word="make"] [] [tag="J.*"]** will return: *make the best, make a big, and make it easy.*

[tag="V.*"] [] [] [tag="RB"] will return: *hear from you soon, happened to me recently and enjoyed it very much.*

[lemma="be"] [] [] [] [word="*.ing"] will return: *is visiting us next Thursday, and was given to the hospital.*

The number of empty brackets you leave relates to the number of spaces that are searched for.

NB – searching in this way means that these are not “optional” gaps. A search with three sets of empty square brackets will not return results with any less than three words between your search words, i.e. instances where only one or two of the empty brackets are filled will not be included.

However, it is possible to search for a number of optional spaces, and also to search for larger spans without having to type in numerous sets of empty square brackets.

To do this, add curly brackets after your empty square brackets. Inside these, write the number of spaces you’d like to leave:

E.g.: **[lemma = "break"] []{5} [lemma="it"]** will find examples with exactly 5 words between *break* and *it*.

[tag = "CC"] []{9} [word="finished"] will find examples with exactly 9 words between *break* and *it*.

You can make the fixed number of spaces between your search items into a range of spaces by adding a comma between the numbers inside the curly brackets.

For example, **[lemma = "break"] []{2,5} [lemma="it"]** will find examples with between 2 and 5 spaces between *break* and *it*, such as: *breaks and consequently it, break his arm now it, and broken my bed jumping on it.*

[tag = "CC"] []{1,3} [word="finished"] will find examples with between 1 and 3 spaces between the conjunction (CC) and the word *finished*, such as: *and I finished, and and will have finished, and and she has just finished.*

11. Excluding elements

It is possible to specify elements to exclude from your search. To do this, you need to use the exclamation mark symbol: !

When used preceding the equals sign, the exclamation mark means *does not equal*.

The following query will find *fast* as a noun, verb and adverb, but not as an adjective¹:

[lemma="fast" & tag != "AJ0"]

The next example finds *dream* followed by anything other than the word *about*.

[lemma="dream"] [word != "about"]

The next examples find all forms of *break* or *smash* followed by a gap of 1 to 3 words and then *table* not as a verb. **[lemma = "break|smash"] [{1-3} [lemma != "table"]**

12. Optional elements

It is possible to specify elements that may be optional in a search string (i.e. those that may or may not be present). To do this, you need to use the question mark symbol: ?

When used after an element (outside the brackets) it indicates that the element inside the square brackets is optional.

In this search string, the word *did* is optional:

[word="did"]? [word="not"] [word="go"] e.g. *why not go?*, and *However, we did not go there.*

In this search string, the word *not* is optional:

[word="did"] [word="not"]? [word="go"] e.g. *so I did go on the walk,* and *he did not go far enough.*

In this search string, the word *go* is optional:

[word="did"] [word="not"] [word="go"]? e.g. *he did not invent it,* and *we did not go home.*

¹ It should be noted that the methods described here detail how to search using POS tags only – the application of these POS tags by Sketch Engine can sometimes be incorrect, and this is something to take into consideration.

13. Searching for punctuation

As some punctuation symbols serve as wildcards (as we have seen already with e.g. ? and !), when using CQL we need to indicate that we would like to search for them as punctuation rather than as a wildcard.

To do this, simply type a backslash before the punctuation: \

The punctuation must appear in the usual CQL format, as shown below.

Searching for *which* preceded by a comma: [word = "\", "[word = "which"]
 Searching for *now* followed by an exclamation mark: [word = "now"][word = "\!"]

15. Using CQL to look at grammatical variation

The patterns below capture the underlined parts of the constructions in the examples only. They do not, for example capture any possible intervening adjectives or negation (e.g. *I am not playing*, *I quickly played*). See Section 11 (on *optional elements*) and Section 9 (on *gapped searches*) for information on adding intervening elements. See also 15b on some common intervening elements that you may like to add.

VERBS

Grammatical construction	Example	
simple present	<i>I <u>play</u></i>	[tag="V.*P V.*Z"]
simple past	<i>I <u>played</u></i>	[tag="V.*D"]
future (with <i>will</i>)	<i>I <u>will play</u></i>	[word="will" & tag="MD"] [tag="V.*"]
present progressive	<i>I <u>am playing</u></i>	[tag="VBP VBZ"] [tag="V.*G"]
past progressive	<i>I <u>was playing</u></i>	[tag="V.*D"] [tag="V.*G"]
future progressive	<i>I <u>will be playing</u></i>	[word="will" & tag="MD"] [tag="VB"] [tag="V.*G"]
present perfect	<i>I <u>have played</u></i>	[tag="VH.*"] [tag="V.*N V.*D"]
past perfect	<i>I <u>had played</u></i>	[tag="VH.*"] [tag="V.*N V.*D"]
future perfect	<i>I <u>will have played</u></i>	[word="will" & tag="MD"] [tag="VH"] [tag="V.*N V.*D"]
present perfect progressive	<i>I <u>have been playing</u></i>	[tag="VH.*"] [tag="VBN"] [tag="V.*G"]
past perfect progressive	<i>I <u>had been playing</u></i>	[tag="VHD"] [tag="VBN"] [tag="V.*G"]
future perfect progressive	<i>I <u>will have been playing</u></i>	[word="will" & tag="MD"] [tag="VH"] [tag="VBN"] [tag="V.*G"]

This list is by no means exhaustive, but hopefully you should be able to form your own verbal CQL search strings by analogy, using the table above as a resource.

15b. Adding intervening elements to the patterns above

Intervening element	CQL
negation (e.g. <i>not</i> , <i>n't</i>)	[word="not n't" & tag="RB"]
adverb (e.g. <i>quickly</i> , <i>usually</i>)	[tag="RB"]
any element	[]
(any) modal verb	[tag="MD"]
specific modal verb (in this case, <i>would</i>)	[word="would" & tag="MD"]

16. Notes on using CQL

While using CQL allows you to perform very powerful searches, it should be used with care when searching in the Cambridge Learner Corpus. The POS tagger used on the CLC can make some errors in its application of these tags due to the nature of the language used in the exam scripts. For this reason, where possible, you should include the word forms in your search, rather than the POS only.

17. FAQs: CQL search strings

This is a working list of CQL queries that have been requested recently. It is by no means complete or perfect, but is instead an ongoing document. Please contact corpus@cambridge.org if you have any queries or suggestions.

Below are the raw CQL searches only. Usually to see useful frequency results, click on *frequency*, or one of the *frequency* options (e.g. *node forms*, *node tags*) from the resulting concordance page.

1. Most frequent verbs directly before *will*

CQL search string: [tag="V.*"] [word="will" &tag="V.*|MD"]

2. Most frequent verbs directly after *will*

CQL search string: [word="will" &tag="V.*|MD"] [tag="V.*"]

3. Most frequent verbs in the span 0-4 around *will*

CQL search string: [word="will" &tag="V.*|MD"] [][0,3] [tag="V.*"]

5. Present progressive verbs

CQL search string: [lemma="be"] [tag="VBG"]

6. Verbs found 0-3 after present progressives

CQL search string: [lemma="be"] [tag="VBG"] [][0,3] [tag="V.*"]

7. Verbs found -3-0 before present progressives

CQL search string: [tag="V.*"] [][0,3] [lemma="be"] [tag="VBG"]

8. How frequent are *I'm hoping / I'm planning to vs I'm going to ...*

CQL search string: [word="I"] [word="m|am"] [word="planning|hoping"] [word="to"]

9. Verbs directly before *going to*

CQL search string: [tag="V.*"] [word="going"] [word="to"]

10. Verbs directly after *going to*

CQL search string: [word="going"] [word="to"] [tag="V.*"]

11. Verb + *to* + infinitive

CQL search string: [tag="V.*"] [tag="TO"] [tag="V.*"]

12. Verb + verb + *ing* combinations

CQL search string: [tag="V.*"] [tag="V.*"] [word=".*ing"]

13. adjective + noun (e.g. last year, tall man):

CQL search string: [tag="J.*"] [tag="N.*"]

14. Finding adj before ANY noun:

CQL search string: [tag="J.*"]

Then, *filter* and then change the span to 0-1 to find things after the adjective

Then, type in [tag="N.*"] in the search box. Then click on *node forms*.

15. Comparative adjectives

CQL search string: [tag="JJR"]

16. Searching for a word as a particular POS (e.g. so as a past tense verb, rather than just as a verb).

E.g. *significantly* as an adverb.

CQL search string: [word = "word in here" & tag = "tag in here*"]

So, [word="significantly" & tag="R.*"]

17. Frequencies of adjectives filling the gap "more X than"

CQL search string: [word="more"] [tag="J.*"] [word="than"]

18. Adjectives filling the gap "less X than"

CQL: [word="less"] [tag="J.*"] [word="than"]

19. Adjectives ending in -ic

CQL search string: [word = ".*ic" & tag = "J.*"]

20. Adjectives ending in -ical

CQL search string: [word = ".*ical" & tag = "J.*"]

21. Comparative adjectives ending -er

CQL search string: [word= ".*er" & tag = "JJR"]

22. Most frequent of a list of words

Searching using CQL is quite nice for this, because then you can get a lovely graph, as shown below.

CQL search string:

[word= "enormous|big|massive|tremendous|considerable|serious|significant|substantial" & tag= "J.*"]

Frequency list

Frequency limit:

word	Freq	
p/n significant	118535	
p/n considerable	25874	
p/n substantial	21714	
p/n serious	21587	
p/n big	10432	
p/n massive	6526	
p/n enormous	5442	
p/n tremendous	1998	

Showing page 1
the last page