AutoSearch demo: a brief overview

AutoSearch allows users to quickly upload and index annotated (with lemma- and part-of-speech) text data and search it.

Supported formats

AutoSearch has support for the annotated XML text formats FoLiA and TEI.

The input files should already be tagged with lemma and part-of-speech (PoS).

> **NOTE:** For TEI, the part-of-speech tags are expected to be in the "function" attribute of the "w" (word) tags.

Unfortunately, document metadata is not currently indexed and therefore cannot be searched. We hope to add this feature in a future update.

Annotating your text data with lemma and PoS

If your text data is not yet tagged (annotated) with lemma and part of speech, you need to convert the text to a supported format and tag it.

Plain text to FoLiA

TTNWW

This is probably the easiest option right now. If you have *plain text* (e.g. a .txt file you can view in Notepad, so not a Word document or such), you can use the Frog implementation in CLARIN TTNWW to create FoLiA XML: http://yago.meertens.knaw.nl/apache/TTNWW/.

For support, please contact: helpdesk@clarin.nl.

Install Frog locally

Alternatively, you can install Frog in your own computer environment, but for that you will need the support of a software developer: http://ilk.uvt.nl/frog/.

For support, please contact: Antal van den Bosch, a.vandenbosch@let.ru.nl.

Plain text to TEI

Converting plain text to TEI is at present, unfortunately, not straightforward. A TEI example can be found at the end of this document.

For support please contact: servicedesk@inl.nl.

Other formats (.doc, .docx, .pdf, .html, .epub, ...)

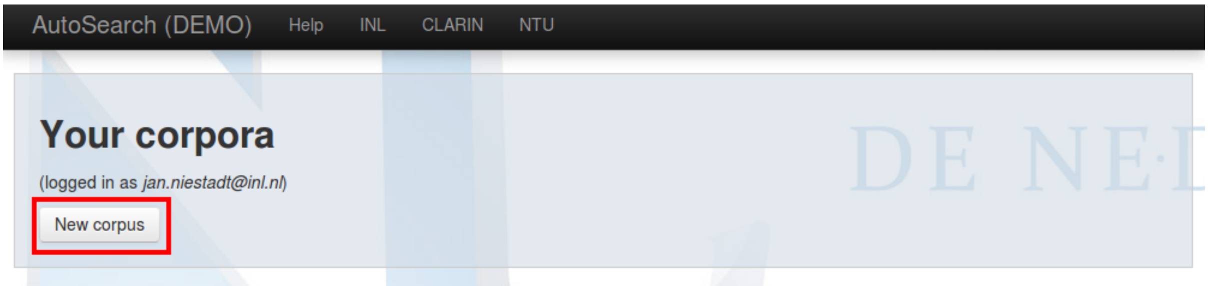
If you do not have plain text, but another document format such as the ones listed above, please contact servicedesk@inl.nl for advice about conversion.

Creating a corpus

Once you have annotated text data in one of the supported formats, you can create your corpus.

First, log in to AutoSearch using your CLARIN account at https://portal.clarin.inl.nl/autocorp/. Your corpora will be linked to this account.

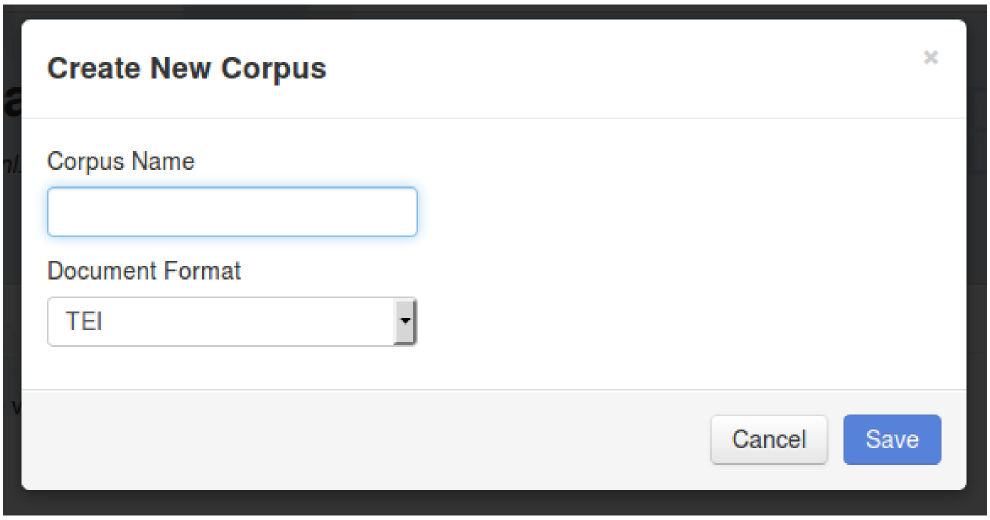
To create a corpus (you may store up to 10 of them), click the “New corpus” button:



Your corpora

(logged in as *jan.niestadt@inl.nl*)

The “Create New Corpus” dialog appears:

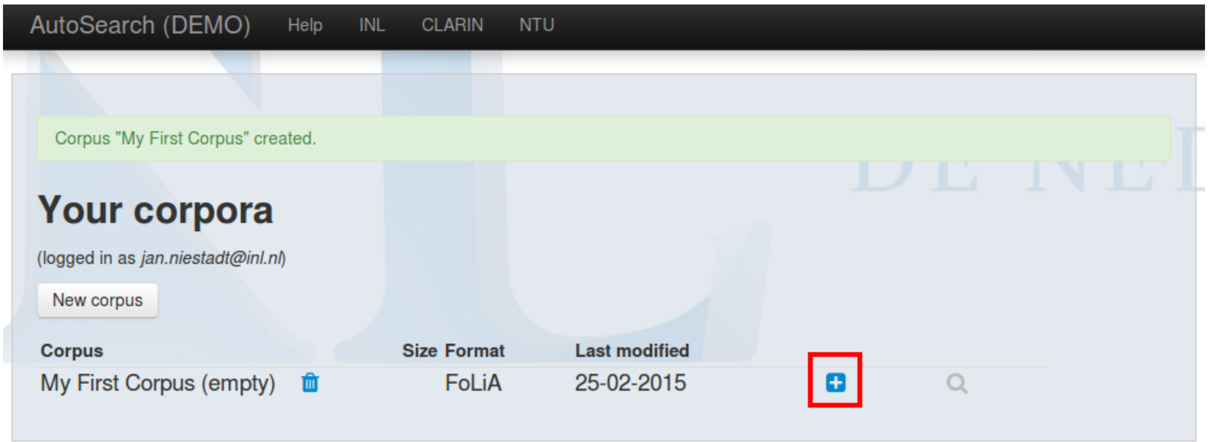


Create New Corpus

Corpus Name

Document Format: TEI

Enter a name and choose format of your text data (FoLiA or TEI), then click “Save”. An empty corpus will be created and will appear in the list. Click the “plus” icon to add some data to it:



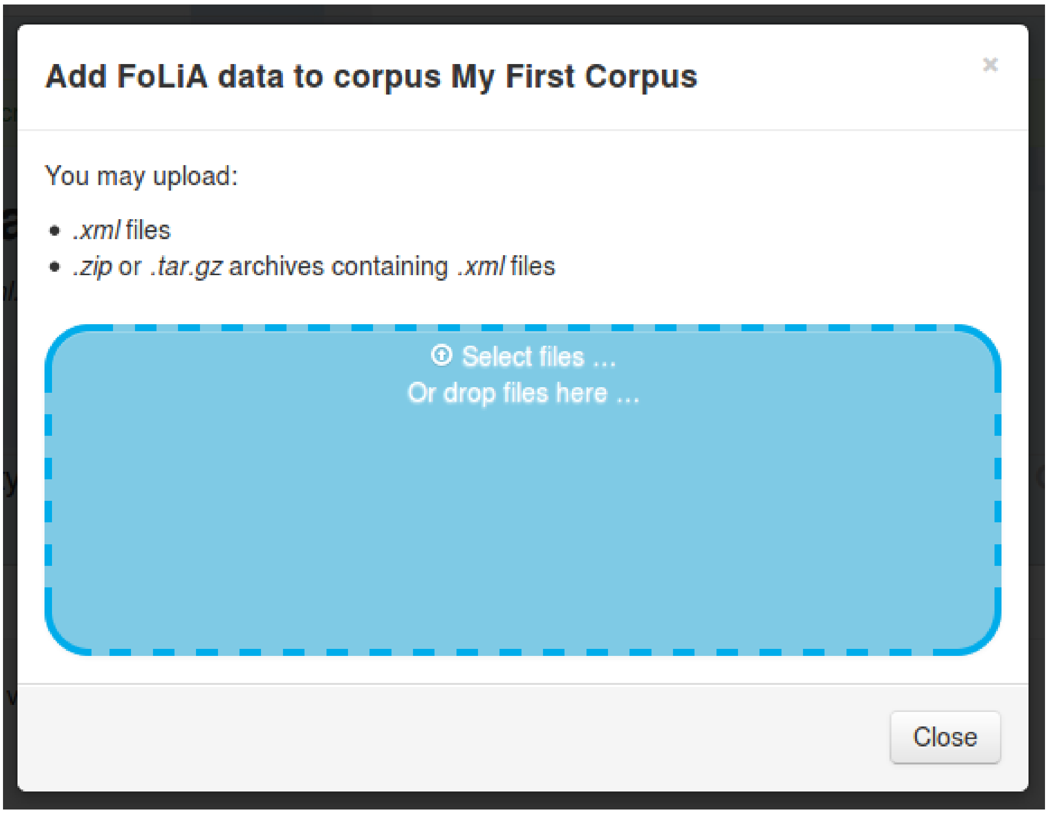
Corpus "My First Corpus" created.

Your corpora

(logged in as *jan.niestadt@inl.nl*)

Corpus	Size	Format	Last modified
My First Corpus (empty)		FoLiA	25-02-2015

The “Add data” dialog appears:



Add FoLiA data to corpus My First Corpus

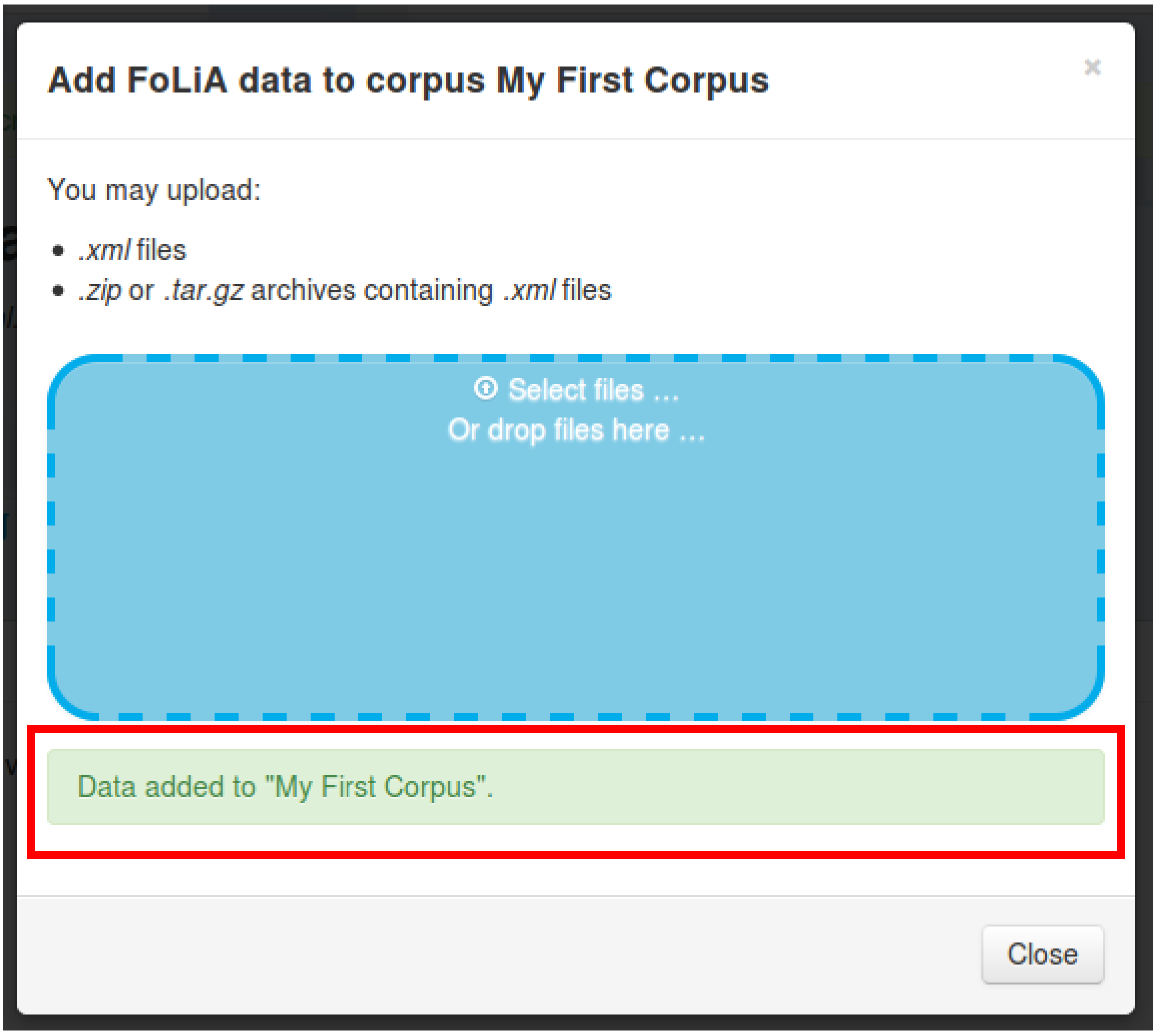
You may upload:

- *.xml* files
- *.zip* or *.tar.gz* archives containing *.xml* files

To add data, you can either click on the blue area, which will cause an “Open File” dialog to appear, allowing you to navigate to the .xml file (or archive containing .xml files) you wish to add, or you can drag & drop a file onto the blue area directly.

> **NOTE:** there is a maximum of 25 MB per uploaded file, and a maximum of 500,000 words per corpus.

The file will be uploaded to the server and indexed. Please wait until the “Data added” message appears:



Add FoLiA data to corpus My First Corpus

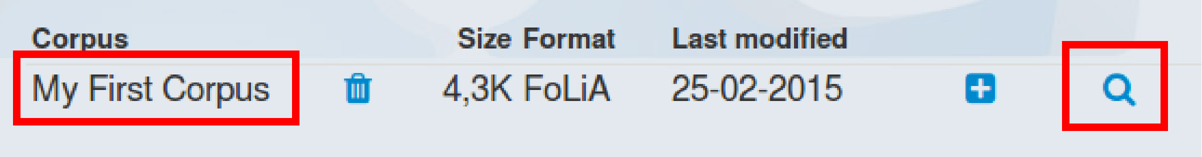
You may upload:

- *.xml* files
- *.zip* or *.tar.gz* archives containing *.xml* files

Data added to "My First Corpus".

Now you can add more files, or you can close the dialog.

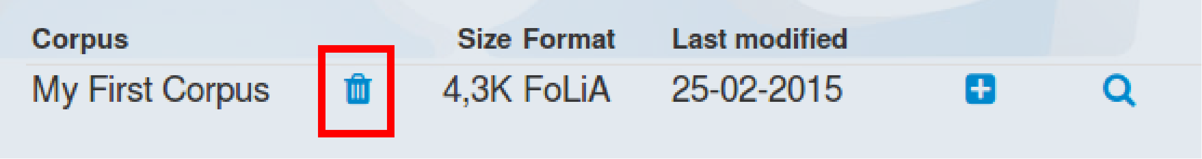
To search your corpus, click the corpus name or the magnifying glass icon:



Corpus		Size	Format	Last modified		
My First Corpus	🗑	4,3K	FoLiA	25-02-2015	⊞	🔍

This will make the corpus search interface appear (see the next section).

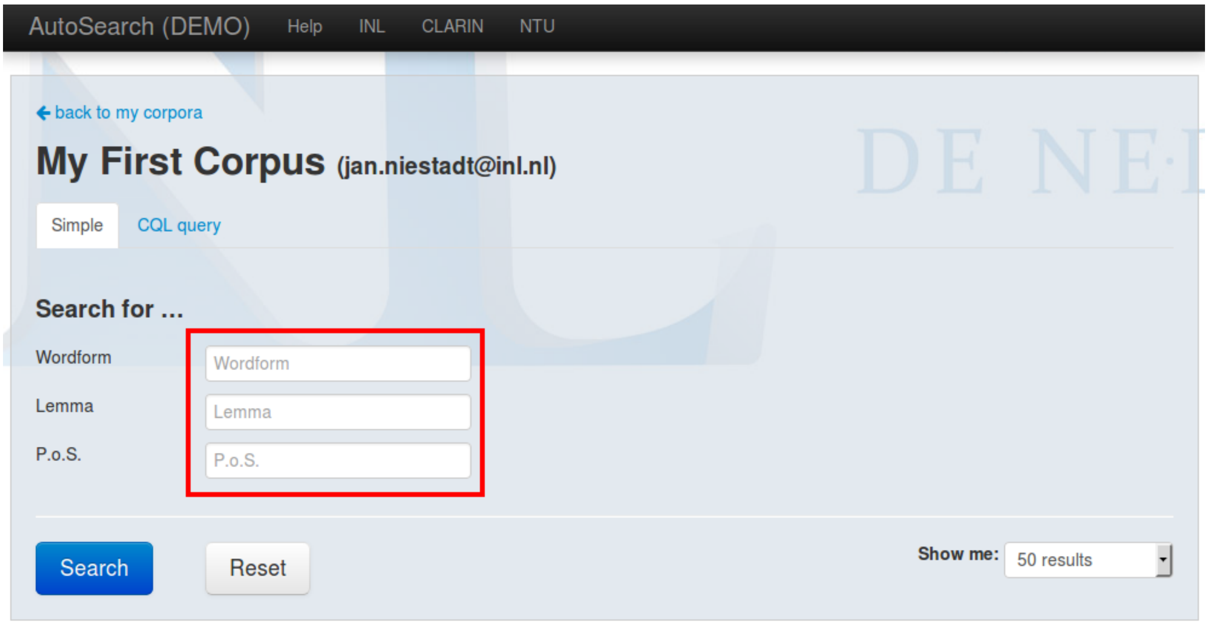
Finally, if you wish to delete your corpus, click the garbage can icon:



Corpus		Size	Format	Last modified		
My First Corpus	🗑	4,3K	FoLiA	25-02-2015	⊞	🔍

Searching a corpus

The corpus search interface allows you to query your corpus by word form, lemma or part-of-speech (PoS):



My First Corpus (jan.niestadt@inl.nl)

Show me: 50 results

Simple search

A **word form** is an occurrence of a word in the text. By typing “wandelen” (without quotes) in the word form search field, you will find the occurrences of the word “wandelen” in the text data.

Of course, *wandelen* is only one of the possible forms of the verb *wandelen*. You can search for all forms by looking for the **lemma** *wandelen*, using the lemma search field.

These fields accept wildcard characters (* for zero or more characters, ? for a single character), so typing “wan*” (again, without quotes) will search for all words that start with “wan”.

It is also possible to search for words with a specific **Part of Speech (PoS)** in the text.

> **NOTE:** the exact values you need to type in the PoS field depend on how your data is part-of-speech tagged. For example, for some annotated text data, “V*” will search for verbs, while for other annotation schemes you may have to search for “WW*”, or perhaps yet another code. Check your annotated input files to find the exact codes.

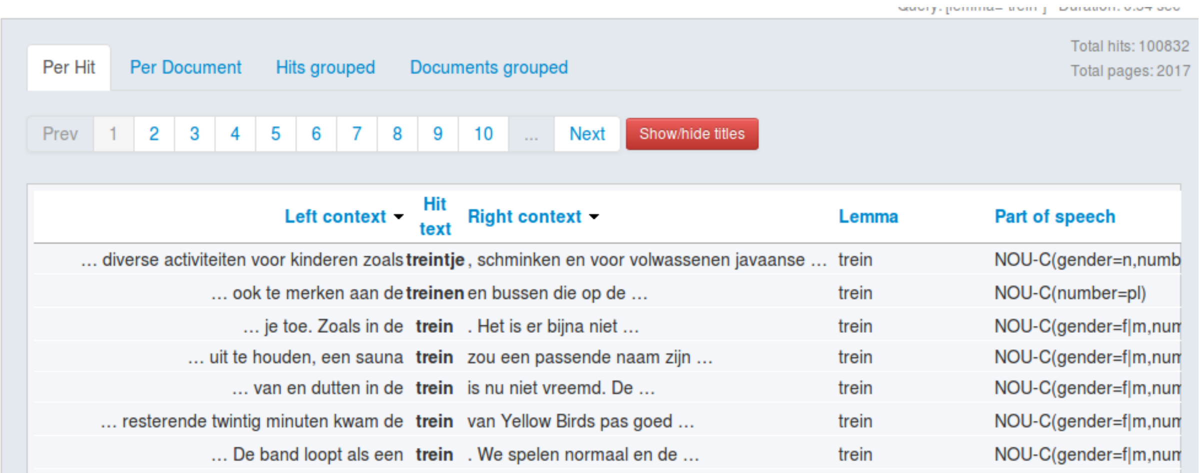
You may enter one word or multiple words in each of these fields. Multiple words are interpreted as a phrase query. For example, typing “de trein” (without the quotes!) in the word form field searches for these two consecutive words.

You can combine these fields, so for example, you could search for the word “leven” used as a noun (and not as a verb).

You can also type phrase queries in multiple fields. For example, to search for the word “de” followed by a form of the word “trein”, type “de *” into wordform and “* trein” into lemma.

Default results view (Per Hit)

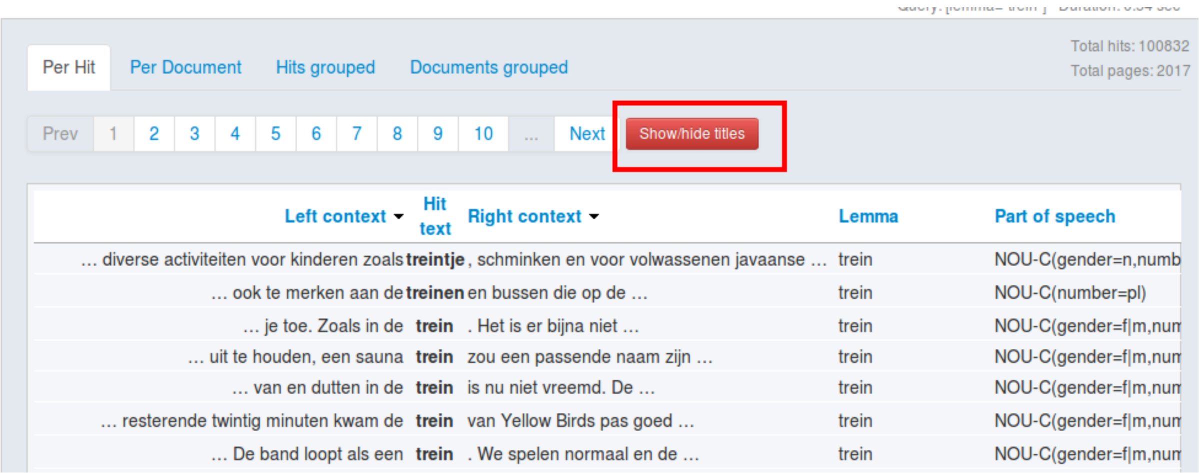
After you enter your query and click Search, you will see results, similar to these:



Left context	Hit text	Right context	Lemma	Part of speech
… diverse activiteiten voor kinderen zoals	treintje	, schminken en voor volwassenen javaanse …	trein	NOU-C(gender=n,numb
… ook te merken aan de	treinen	en bussen die op de …	trein	NOU-C(number=pl)
… je toe. Zoals in de	trein	. Het is er bijna niet …	trein	NOU-C(gender=f|m,nun
… uit te houden, een sauna	trein	zou een passende naam zijn …	trein	NOU-C(gender=f|m,nun
… van en dutten in de	trein	is nu niet vreemd. De …	trein	NOU-C(gender=f|m,nun
… resterende twintig minuten kwam de	trein	van Yellow Birds pas goed …	trein	NOU-C(gender=f|m,nun
… De band loopt als een	trein	. We spelen normaal en de …	trein	NOU-C(gender=f|m,nun

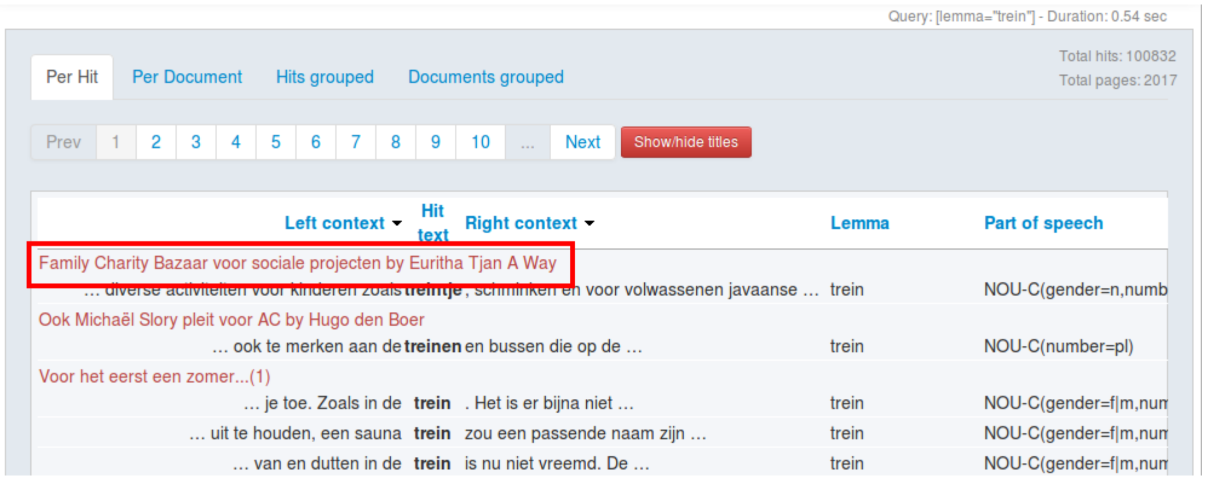
You can click on a hit to show a bit more context around the matched word(s).

You can show the document titles by clicking “Show/hide titles”:



Left context	Hit text	Right context	Lemma	Part of speech
… diverse activiteiten voor kinderen zoals	treintje	, schminken en voor volwassenen javaanse …	trein	NOU-C(gender=n,numb
… ook te merken aan de	treinen	en bussen die op de …	trein	NOU-C(number=pl)
… je toe. Zoals in de	trein	. Het is er bijna niet …	trein	NOU-C(gender=f|m,nun
… uit te houden, een sauna	trein	zou een passende naam zijn …	trein	NOU-C(gender=f|m,nun
… van en dutten in de	trein	is nu niet vreemd. De …	trein	NOU-C(gender=f|m,nun
… resterende twintig minuten kwam de	trein	van Yellow Birds pas goed …	trein	NOU-C(gender=f|m,nun
… De band loopt als een	trein	. We spelen normaal en de …	trein	NOU-C(gender=f|m,nun

After you do this, click on a document title:



Left context	Hit text	Right context	Lemma	Part of speech
Family Charity Bazaar voor sociale projecten by Euritha Tjan A Way				
… diverse activiteiten voor kinderen zoals	**treintje**	, schminken en voor volwassenen javaanse …	trein	NOU-C(gender=n,numb
Ook Michaël Slory pleit voor AC by Hugo den Boer				
… ook te merken aan de	**treinen**	en bussen die op de …	trein	NOU-C(number=pl)
Voor het eerst een zomer...(1)				
… je toe. Zoals in de	**trein**	. Het is er bijna niet …	trein	NOU-C(gender=f|m,num
… uit te houden, een sauna	**trein**	zou een passende naam zijn …	trein	NOU-C(gender=f|m,num
… van en dutten in de	**trein**	is nu niet vreemd. De …	trein	NOU-C(gender=f|m,num

Some information about the document, as well as the original contents, should now be shown.

Other results views

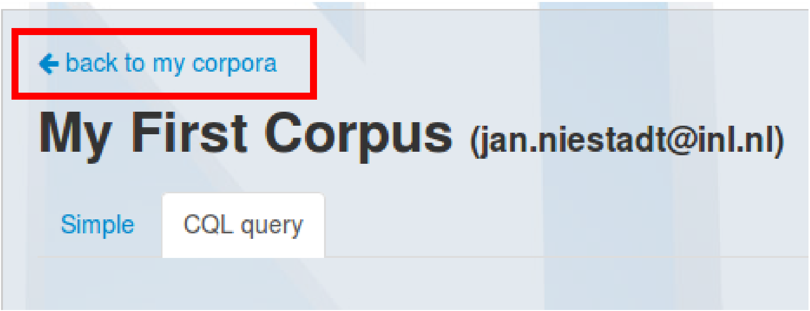
In addition to the default “Per hit” view, there are three other views: Per document, Hits Grouped and Documents Grouped. Click on the tab to switch to that view. For the “grouped” views, you will have to specify what property you wish to group on.

In any of the views, you can sort the results by clicking on the column titles (for Per hit: Left context, Hit text, Right context, Lemma and Part of speech). To switch between ascending and descending order, simply click on the chosen item again.

In the grouped views, you can show results from a group by clicking on the colored bar with the frequency number. If you wish to explore all the results in the group, click on “detailed concordances in this group”.

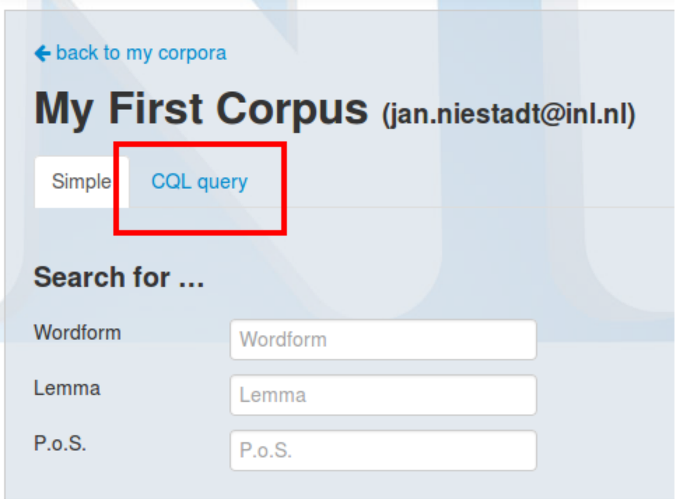
Return to corpora page

To return to your corpora page at any time, click the “back to my corpora” link at the top of the page:

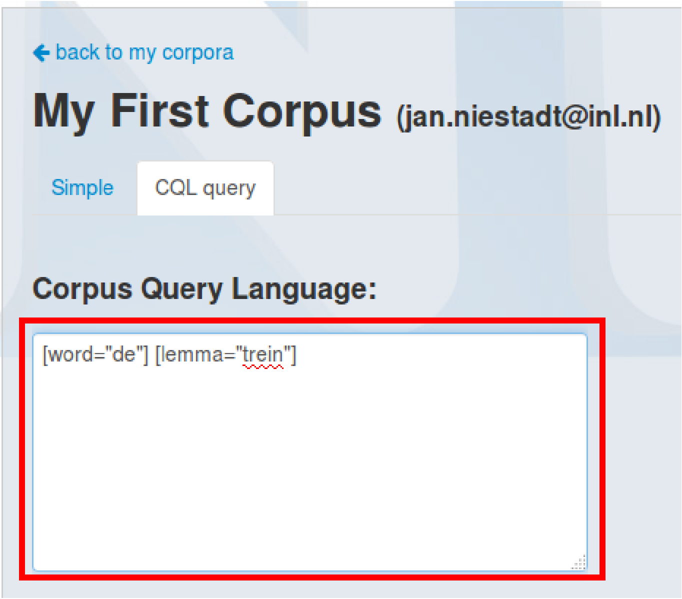


Advanced: Corpus Query Language

If you wish, you can also use the advanced query mode, which allows you to construct more complex queries. To do this, click on “CQL query” in the search interface:



A textarea appears where you can compose your query in “Corpus Query Language”:



```
[word="de"] [lemma="trein"]
```

This is an advanced query language developed at IMS, University of Stuttgart in the early 1990s, designed to support very specific retrieval of phrase types in the corpus. It expresses phrase queries as sequences of token queries. It is therefore mainly useful if you want to find specific types of phrases in a larger text. An example of a simple query (note that the quotes are required): `"grote" ".*heid"`.

This is how you search for the word "grote" followed by a word ending with "heid", using regular expressions to specify the pattern for the second word. Equivalent to the above query is: `[word="grote"] [word=".*heid"]` which has the typical form of a CQL query: a phrase query built up from token queries surrounded by square brackets.

Single token queries

These typically consist of a combination of simple attribute value queries in the form of either:

- token attribute=single token regular expression `[word=".*heid"]` surrounded by square brackets
- default token attribute regular expression `".*heid"`

The following token attributes are available for querying:

- word - The word as it was written. This is the default attribute in this corpus, so querying by only giving a word between brackets, eg. "man" means asking for `[word="man]`.
- lemma - Dictionary headword form of words
- pos - Part of speech

A typical example using all three token attributes: `[pos="AA.*"] [lemma="man" & word != "man"]`. This is how you search for adjectives followed by an occurrence of the lemma "man", which may or may not be the form 'man' itself.

Phrase queries

As you have seen, phrase queries can simply consist of a sequence of single token queries. Apart from this, regular expression notations are available to express sequences of tokens. For example: `"der.*"{2,}`. This query finds two or more successive words starting with "der". At the token level, regular expression operators such as *, + and ? are available. Another example: `[pos="AA.*"]+ "man"`. This will find the word "man" with one or more adjectives applied to it.

Summary of Corpus Query Language support

The following CQL constructs are supported:

- Token constraints of the form [word="koe"] or "koe" (default property). Constraint values may be regular expressions, e.g. [word="str.+"] . The regular expression operators are:
  - `.` (full stop) matches arbitrary character: b.k finds bok, bak, bek, bik, etc..
  - `*` matches zero or more occurrences of the preceding letter or bracketed group: .*schip finds words ending with schip (also matching "schip"), dia.* finds words beginning with dia, .*deel.* finds words with deel in the middle.
  - `+` matches 1 or more occurrences of the preceding letter or bracketed group: .+schip finds words ending with schip (not matching "schip")
  - `{n,m}` matches a sequence of n to m occurrences of preceding letter or group. Use {n} to find a sequence of length n, {n,} to find at least n, {0,n} to find at most n: `[word=".*[aeiou]{5,6}.*"]` finds words containing a group of 5 or 6 vowels
  - `?` The bracketed items are optional characters: blond(e)? finds blond and blonde
  - `|` Vertical bar (disjunction): paard|koe|schaap searches for all of these items
  - `[]` Square brackets (character groups): b[ae]k finds bak, bek
  - \ To search for a full stop, use the backslash and the full stop: Dr\. finds Dr.
- Constraints may be combined by using boolean operators, both between token specifications (e.g. "stad" | "dorp") and within token specifications (e.g.

[lemma="zijn" & pos="VRB.*"]). Supported boolean operators are ! (not), & (and) and | (or). Implication (->) is not supported yet. Parentheses may be used to group expressions.

- Phrase searches, by putting several token specifications in sequence, e.g.: "de" [lemma="koe"]
- You can apply regular expression operators (* + ? {a,b}) to token specifications, e.g.: [type="VRB.*"]+ (one or more verbs) or "k.*"{3,} (three or more words starting with k)
- You can use match all tokens ([]) to match any word, e.g. "koe" []{1,2} "schaap" to find "koe" and "schaap" in that order with 1-2 words between them.
- You can search for XML tags (in the current version the only indexed tag is <s/>, sentence) in the following ways: <s> "Gelukkig" (word at start of sentence), "gelukkig" </s> (word at end of sentence).

Appendix: TEI example

```
<!--

Example of a TEI file that AutoSearch understands.
Important details:
- root element is TEI or TEI.2 (TEI P4 or P5, respectively)
- words are surrounded by w tags
- (optional) w tags have 'lemma' attribute with each word's lemma
  (=headword)
- (optional) w tags have 'type' or 'function' attribute containing part of
  speech (PoS) for each word (N.B. specify which attribute should be used
  when creating the corpus)
- s tags indicate sentence boundaries (you can enter an advanced query
  under "CQL query", for example to search for a word at the start of a
  sentence. Example: <s> "the")

-->


<!-- TEI.2 tag: the TEI P5 root element (P4 uses TEI as its root element) -->
<TEI.2>

  <!-- teiHeader tag: metadata about this TEI file
       (not yet searchable, will be added in a future update) -->
  <teiHeader>
      <!-- ...metadata... -->
  </teiHeader>

  <!-- text tag: the text content -->
  <text>

      <!-- body tag: the main text body (text can also have front and back,
           but that's not used here) -->
      <body>

      <!-- p tag: paragraph -->
      <p>

      <!-- s tag: sentence -->
      <s>

      <!-- w and pc tags: words and punctuation -->
      <w lemma="de" type="PD(type=d-p,subtype=art-def)">De</w>
      <w lemma="centraal"
               type="AA(degree=pos,position=prenom,formal=infl-e)"
                >Centrale</w>
      <w lemma="bank" type="NOU-C(gender=f|m,number=sg)">Bank</w>
      <w lemma="van" type="ADP(type=pre)">van</w>
      <w lemma="suriname" type="NOU-P(number=sg)">Suriname</w>
      <w lemma="governor" type="RES(type=for)">governor</w>
      <w lemma="zijn"
```

```
               type="VRB(finiteness=fin,mood=imp|ind,tense=pres,number=sg)"
               >is</w>
      <w lemma="met" type="ADP(type=pre)">met</w>
      <w lemma="nieuw"
               type="AA(degree=pos,position=prenom,formal=infl-e)"
               >nieuwe</w>
      <w lemma="maatregel" type="NOU-C(number=pl)">maatregelen</w>
      <w lemma="komen" type="VRB(finiteness=part,tense=past)"
               >gekomen</w>
      <pc type="post">.</pc>

      </s>

      </p>

      </body>

  </text>

</TEI.2>
```