v.3, 01.12.2016

Helpfile for Norecopa's search engine

The search engine has been constructed using Apache Solr (http://lucene.apache.org/solr) and this helpfile is based on documentation from the developers (https://cwiki.apache.org/confluence/display/solr/The+Standard+Query+Parser).

For a quick overview of the whole of the website, put an asterisk (*) in the search field and press ENTER (or click on the magnifying glass to the right of the search field). You can then reduce the number of hits by using the filters on the right-hand side of the search results page.

For example, you can limit the search to one or more of Norecopa's 4 databases (NORINA, TextBase, 3R Guide and Classic AVs) and/or the textpages on the website, and/or to one or more of a variety of search fields and scientific categories.

These restrictions can be increased or decreased "on the fly" (live) and the hit list will immediately respond by decreasing or increasing in size.

The search engine takes into consideration several factors when returning hits:

- the words which in fact have been entered in the search field
- a list of synonyms which has been constructed by Norecopa specifically for these databases
- an "auto-complete" function which suggests search terms based on the engine's own dictionary (this is best visualised by typing slowly and watching what happens in the search field) and an index of all the words in the databases
- algorithms which prioritise or suppress certain words depending on their likely relevance and also on Boolean operators which the user may choose to add
- "fuzzy logic" (the search engine suggests words which resemble terms entered by the user)

The user can construct specific search commands which will have priority over an automated search, by using the following methods:

Wildcard searches:

The search engine supports single and multiple character wildcard searches within single words (e.g. **rat**), but not within search phrases (e.g. **"albino rat"**).

? matches a single character.
The search **te?t** would match both test and text

* matches zero or more sequential characters
The search **tes*** would match test, testing, and tester.

Wildcard characters can also be used in the middle of a word:
te*t would match test and text
***est** would match pest and test

Fuzzy searches

Fuzzy searches discover terms that are similar to a specified term without necessarily being an exact match. To perform a fuzzy search, use the tilde ~ symbol at the end of a single word.
For example, to search for a term similar in spelling to "**Karina**," use: **Kar~**
This search will match terms like **Karina**, **Karl**, and **Kay**. It will also match the word **Karina** itself.
An optional distance parameter specifies the maximum number of edits allowed, between 0 and 2, defaulting to 2. For example: **roam~1**
This will match terms like **roams** & **foam** (but not **foams** since it has an edit distance of "2").

Stemming (truncating)

In many cases, stemming (=truncating, i.e. reducing terms to a common stem, e.g. **chem**) can produce similar effects to fuzzy searches and wildcard searches.

Proximity searches

A proximity search looks for terms that are within a specific distance from one another. To perform a proximity search, add the tilde character ~ and a numeric value to the end of a search phrase.
For example, to search for **animal** and **research** within 10 words of each other in a document, use the search: **"animal research"~10**
The distance referred to is the number of term movements needed to match the specified phrase.

In the example above, if **animal** and **research** were 10 spaces apart in a field, but **animal** appeared before **research**, more than 10 term movements would be required to move the terms together and position **animal** to the right of **research**.

Boolean Operators

OR Requires that either term (or both terms) be present for a match.
The OR operator is the default conjunction operator.

AND (or &&) Requires both terms on either side of the Boolean operator to be present for a match

NOT (or - or !) Requires the following term not to be present

+ Requires that the following term be present.

Double quotation marks (**"rat"**) can be used to construct a search string.

Examples

To search for documents that contain either **albino rats** or just **albino**:

" albino rats " albino *or*
" albino rats " OR albino

1. To search for documents that must contain **albino** and that may or may not contain **rat**:
   +albino rat
2. To search for documents that must contain both the phrase **albino rat** *and* the phrase **Swiss mouse**:
   "albino rat " AND "Swiss mouse" or **"albino rat " && "Swiss mouse"**
3. To search for documents that contain the phrase **albino rat** but do *not* contain the phrase **Swiss mouse**:
   "albino rat" NOT "Swiss mouse" *or*
   "albino rat" ! " Swiss mouse " *or*
   " albino rat " -" Swiss mouse"

Use of brackets to group searches

To search for either **rat** *or* **mouse** *and* **albino**:
(rat OR mouse) AND albino

Ignore special characters

The search engine gives the following characters special meaning when they appear in a query:

+ - && || ! ( ) { } [ ] ^ " ~ * ? : /

To make the search engine interpret any of these characters literally, precede the character with a backslash character \.
For example, to search for **(1+1):2** without having the search engine interpret the plus sign and parentheses as special characters:

\(1\+1\)\:2

Boosting a word with ^

Search terms can be allocated a boost factor to increase their relevance in matching documents. Use the caret symbol ^ followed by a number. The higher the boost factor, the more relevant the term will be.
For example, if you are searching for **albino rat** and you want the term **rat** to be more relevant, you can boost it by adding the ^ symbol along with the boost factor immediately after the term: **albino rat^4**
You can also boost one or more phrases: **"albino rat"^4 "brown mice"**
By default, the boost factor is 1. Although the boost factor must be positive, it can be less than 1 (e.g. it could be 0.2).