Query Syntax

This page gives an overview of the available query syntax, going from “basic” to “advanced”. The query syntax is provided by Apache Lucene, DocFetcher Pro’s underlying search engine, and it’s described in a more technical manner on Lucene’s query syntax page.

Boolean Operators

DocFetcher Pro supports the boolean operators `OR`, `AND` and `NOT`. If words are concatenated *without* boolean operators, DocFetcher Pro will by default treat them as if they were concatenated with `OR`. In the Preferences, you can change this default to `AND`.

Instead of `OR`, `AND` and `NOT`, you can also use `||`, `&&` and `–` (minus symbol), respectively. You can use *parentheses* to group certain expressions. Here are some examples:

Query	**Resulting documents contain...**
`dog OR cat`	either `dog`, or `cat`, or both
`dog AND cat`	both `dog` and `cat`
`dog cat`	(by default equivalent to the query `dog OR cat`)
`dog NOT cat`	`dog`, but not `cat`
`–dog cat`	`cat`, but not `dog`
`(dog OR cat) AND mouse`	`mouse`, and either `dog`, or `cat`, or both

Case-insensitivity

DocFetcher Pro does not distinguish between lowercase and uppercase characters, so it doesn’t matter if the words you enter are completely lowercase, or completely uppercase, or a mix of both. The only exceptions are the keywords `OR`, `AND`, `NOT` and `TO`, which must always be entered in uppercase. (For the `TO` keyword, see the ‘Range Searches’ section below.)

Phrase Searches and Required Terms

To search for a phrase (i.e., a sequence of words), put the phrase in double quotes. To indicate that the documents to search for must contain a particular word, put a `+` in front of the word. Of course you can combine these constructs with boolean operators and parentheses. Again, some examples:

Query	**Resulting documents contain...**
`"dog cat mouse"`	the exact phrase `dog cat mouse`
`+dog cat`	definitely `dog`, and maybe also `cat`
`"dog cat" AND mouse`	the phrase `dog cat`, and the word `mouse`
`+dog +cat`	(equivalent to the query `dog AND cat`: both words are required)

Wildcards

DocFetcher Pro supports the two wildcards `*` and `?`. These are placeholder characters to match *zero or more* characters and *exactly one* character, respectively. Examples:

Query	**Resulting documents contain...**
`luc?`	`luca`, `luck`, `lucy`, ...
`luc*`	`luc`, `luca`, `luck`, `lucy`, `lucene`, ...
`*ene*`	`lucene`, `energy`, `generator`, ...

Note: If wildcards are used as the first character of a word, the search tends to be slower on average. This is due to how the index is structured: It's as if you tried to look up someone's phone number, and you only know that person's first name. So, in the example above, searching for `*ene*` will probably be slower than the other searches because `*ene*` starts with a wildcard.

Fuzzy Searches

Fuzzy searches allow you to search for words *similar* to a given word. For example, if you search for `roam~`, DocFetcher Pro will find documents containing words like `foam` and `roams`.

Additionally, you can append a similarity threshold between 0 and 1, like so: `roam~0.8`. The higher the threshold, the higher the similarity of the returned matches. Leaving out the threshold is the same as using the default value of 0.5.

Proximity Searches

Proximity searches allow you to find words that are within a specific distance away from each other. To do a proximity search, put a tilde ~ at the end of a phrase construct, followed by a distance value. — Note that this is syntactically similar to fuzzy searches. For example, to search for documents containing `wikipedia` and `lucene` within 10 words of each other, type in: `"wikipedia lucene"~10`

Boosting Terms

You can influence the relevance sorting of the results by assigning custom weights to words. Example: If you enter `dog^4 cat` instead of just `dog cat`, documents containing `dog` will receive a higher score and therefore move closer to the top of the results.

Although the boost factor must be positive, it can be less than 1 (e.g., 0.2). If no boost factor is specified, the default value 1 is used.

Field Searches

By default, DocFetcher Pro will search all the textual data it is capable of extracting, i.e., the contents, the filenames and the metadata of the documents. However, you can also restrict your searches to the filename and/or certain metadata fields. For example, to search for documents whose titles contain `wikipedia`, enter: `title:wikipedia`.

Field search can be combined with other search constructs, e.g.:

- Phrase search using quotes, e.g., `title:"dog cat"`, or using parentheses, e.g., `title:(dog cat)`. If you omit the quotes and parentheses, only `dog` will be matched against the title, not `cat`.
- Wildcards: `title:dog*` will, for instance, match occurrences of `doggy` in the title.

Which fields are available generally depends on the document type, but you can use this as a rule of thumb:

- *Files*: filename, title, authors
- *Emails*: subject, sender, recipients

Range Searches

DocFetcher Pro allows searching for words that are lexicographically *between* two other words. For example, the word `beta` lies between `alpha` and `gamma`. So, if you want to find all documents that contain words between `alpha` and `gamma`, type in: `[alpha TO gamma]`.

When using the square brackets, the range query is *inclusive*, i.e., `alpha` and `gamma` are included in the results. To do an *exclusive* range search, use curly brackets instead: `{alpha TO gamma}`

You may combine range searches and field searches as follows: `title:{alpha TO gamma}`. This will restrict the range search to the title field.

Note: The preview pane currently does not support highlighting matches from a range search.