dtSearch Search Help

Introduction

To search, enter a search request in the space provided and click the **Search** button. A list of matching documents will appear. To view a document in the list, click on the link. After you have opened a document, you can use the **Next Hit** and **Prev Hit** buttons on the button bar to navigate from hit to hit. For PDF files, press Ctrl+Shift+Space to navigate to the next hit and Ctrl+Shift+Backspace to navigate to the previous hit.

Search Requests Overview

Search types

An **any words** search is any sequence of text, like a sentence or a question. In an **any words** search, use quotation marks around phrases, put + in front of any word or phrase that is required, and - in front of a word or phrase to exclude it. Examples:

```
banana pear "apple pie"
"apple pie" -salad +"ice cream"
```

An **all words** search is like an **any words** search, except that all of the terms have to be found in a document.

A **boolean** search request consists of a group of **words** or **phrases** linked by connectors such as *and* and *or* that indicate the relationship between them. Examples:

apple **and** *pear*	Both words must be present
apple **or** *pear*	Either word can be present
apple **w/5** *pear*	*Apple* must occur within 5 words of *pear*
apple **not** *w/5 pear*	*Apple* must not occur within 5 words of *pear*
apple **and not** *pear*	Only *apple* must be present

`author contains smith`	The field `author` must contain `smith`

If you use more than one connector, you should use parentheses to indicate precisely what you want to search for. For example, `apple and pear or orange juice` could mean `(apple and pear) or orange`, or it could mean `apple and (pear or orange)`.

Search Features

stemming

Finds grammatical variations on endings, like `applies, applied, applying` in a search for `apply`

fuzzy searching

Finds words even if they are misspelled. A search for `alphabet` with a fuzziness of 1 would also find `alphaqet.` With a fuzziness of 4, the same search would find both `alphaqet` and `alpkaqet`

phonic searching

Finds words that sound alike, like `Smythe` in a search for `Smith`

synonym expansion

Finds word synonyms using a comprehensive English language thesaurus (dtSearch Web can also support custom thesaurus terms)

Search terms may include the following special characters:

?	**Matches** any single character. Example: `appl?` matches `apply` or `apple.`
*	**Matches** any number of characters. Example: `appl*` matches `application`
~	**Stemming**. Example: `apply~` matches `apply`, `applies`, `applied`.
%	**Fuzzy search**. Example: `ba%nana` matches `banana`, `bananna`.
#	**Phonic search**. Example: `#smith` matches `smith`, `smythe`.

&	**Synonym search**. Example: *`fast&`* matches *`quick`*.
~~	**Numeric range**. Example: *`12~~24`* matches *`18`*.
:	**Variable term weighting**. Example: *`apple:4 w/5 pear:1`*

Words and Phrases

Use quotation marks to indicate a phrase. You can use a phrase anywhere in a search request. Example:

```
apple w/5 "fruit salad"
```

If a phrase contains a noise word, dtSearch will skip over the noise word when searching for it. For example, a search for *`statue of liberty`* would retrieve any document containing the word *`statue`*, any intervening word, and the word *`liberty`*.

Punctuation inside of a search word is treated as a space. Thus, *`can't`* would be treated as a phrase consisting of two words: *`can`* and *`t`*. *`1843(c)(8)(ii)`* would become *`1843 c 8 ii`* (four words).

Noise words, such as *`if`* and *`the,`* are ignored in searches.

Wildcards (* and ?)

A search word can contain the wildcard characters ***** and **?**. A **?** in a word matches any single character, and a ***** matches any number of characters. The wildcard characters can be in any position in a word. For example:

`appl*` would match apple, application, etc.
`*cipl*` would match principle, participle, etc.
`appl?` would match apply and apple but not apples.
`ap*ed` would match applied, approved, etc.

Use of the ***** wildcard character near the beginning of a word will slow searches somewhat.

Synonym Searching

Synonym searching finds synonyms of a word in a search request. For example, a search for *`fast`* would also find *`quick`*. You can enable synonym searching for all words in a request or you can enable synonym searching selectively by adding the & character after certain words in a request. Example: `fast& w/5 search.`

The effect of a synonym search depends on the type of synonym expansion requested on the search form. dtSearch can expand synonyms using only user-defined synonym sets, using synonyms from dtSearch's built-in thesaurus, or using synonyms and related words (such as antonyms, related categories, etc.) from dtSearch's built-in thesaurus.

Fuzzy Searching

Fuzzy searching will find a word even if it is misspelled. For example, a fuzzy search for `apple` will find `appple`. Fuzzy searching can be useful when you are searching text that may contain typographical errors, or for text that has been scanned using optical character recognition (OCR). There are two ways to add fuzziness to searches:

1. Check the "Fuzzy searching" box to enable fuzziness for all of the words in your search request. You can adjust the level of **fuzziness** from 1 to 10.
2. You can also add fuzziness selectively using the % character. The number of % characters you add determines the number of differences dtSearch will ignore when searching for a word. The position of the % characters determines how many letters at the start of the word have to match exactly. Examples:

    - ba%nana Word must begin with `ba` and have at most one difference between it and `banana.`
    - b%%anana Word must begin with `b` and have at most two differences between it and `banana.`

Phonic Searching

Phonic searching looks for a word that sounds like the word you are searching for and begins with the same letter. For example, a phonic search for `Smith` will also find `Smithe` and `Smythe`.

To ask dtSearch to search for a word phonically, put a # in front of the word in your search request. Examples: `#smith, #johnson`

You can also check the **Phonic searching** box in the search form to enable phonic searching for all words in your search request. Phonic searching is somewhat slower than other types of searching and tends to make searches over-inclusive, so it is usually better to use the # symbol to do phonic searches selectively.

Stemming

Stemming extends a search to cover grammatical variations on a word. For example, a search for `fish` would also find `fishing`. A search for `applied` would also find `applying`, `applies`, and `apply`. There are two ways to add stemming to your searches:

1. Check the **Stemming** box in the search form to enable stemming for all of the words in your search request. Stemming does not slow searches noticeably and is almost always helpful in making sure you find what you want.
2. If you want to add stemming selectively, add a ~ at the end of words that you want stemmed in a search. Example: `apply~`

Variable Term Weighting

When dtSearch sorts search results after a search, by default all words in a request count equally in counting hits. However, you can change this by specifying the relative weights for each term in your search request, like this:

```
apple:5 and pear:1
```

This request would retrieve the same documents as apple and pear but, dtSearch would weight apple five times as heavily as pear when sorting the results.

In a natural language search, dtSearch automatically weights terms based on an analysis of their distribution in your documents. If you provide specific term weights in a natural language search, these weights will override the weights dtSearch would otherwise assign.

Field Searching

When you index a database or other file containing fields, dtSearch saves the field information so that you can perform searches limited to a particular field. For example, suppose that you indexed an Access database with a `Name` field and a `Description` field. You could search for `apple` in the `Name` field like this:

```
name contains apple
```

Field searches can be combined using `and`, `or`, and `not`, like this:

```
(City contains (portland or Seattle)) and (Address contains
(Washington))
```

The parenthesis are necessary to ensure that dtSearch interprets the search request correctly.

Some file formats such as XML support nesting of fields. Example:

```
<record>
     <name>John Smith</name>
     <address>
           <street>123 Oak Street</street>
           <city>Middleton</city>
           ...
```

In dtSearch, a search of a field includes any fields that are nested inside of the field, so the XML file above would be retrieved in a search for any of the following:

```
record contains oak
address contains oak
street contains oak
```

To specify a specific subfield of a field, use / to separate the field names, like this:

```
record/address contains oak
address/street contains oak
record/address/street contains oak
```

Put a / at the front of the field name to specify that it cannot be a sub-field of another field:

```
/record/name contains Smith
/name contains Smith
```

The second search request above would `not` match the XML example because, while it contains a "name" field, the name field is a sub-field of the record-field. A search for /name specifies a "name" field at the top of the field hierarchy.

Finally, you can use // to specify any number of unspecified intervening fields, like this:

```
    /record//city contains Middleton
```

AND Connector

Use the AND connector in a search request to connect two expressions, both of which must be found in any document retrieved. For example:

`apple pie and poached pear` would retrieve any document that contained both phrases.

`(apple or banana) and (pear w/5 grape)` would retrieve any document that (1) contained either *`apple`* OR *`banana`*, AND (2) contained *`pear`* within 5 words of *`grape`*.

OR Connector

Use the OR connector in a search request to connect two expressions, at least one of which must be found in any document retrieved. For example, *`apple pie or poached pear`* would retrieve any document that contained *`apple pie`*, *`poached pear`*, or both.

W/N Connector

Use the W/N connector in a search request to specify that one word or phrase must occur within N words of the other. For example, *`apple w/5 pear`* would retrieve any document that contained *`apple`* within 5 words of *`pear`*. The following are examples of search requests using W/N:

```
    (apple or pear) w/5 banana
    (apple w/5 banana) w/10 pear
    (apple and banana) w/10 pear
```

Some types of complex expressions using the W/N connector will produce ambiguous results and should not be used. The following are examples of ambiguous search requests:

```
(apple and banana) w/10 (pear and grape)
(apple w/10 banana) w/10 (pear and grape)
```

In general, at least one of the two expressions connected by W/N must be a single word or phrase or a group of words and phrases connected by OR. Example:

```
(apple and banana) w/10 (pear or grape)
(apple and banana) w/10 orange tree
```

dtSearch uses two built in search words to mark the beginning and end of a file: `xfirstword` and `xlastword`. The terms are useful if you want to limit a search to the beginning or end of a file. For example, `apple w/10 xlastword` would search for `apple` within 10 words of the end of a document.

NOT and NOT W/N

Use NOT in front of any search expression to reverse its meaning. This allows you to exclude documents from a search. Example:

```
apple sauce and not pear
```

NOT standing alone can be the start of a search request. For example, `not pear` would retrieve all documents that did not contain `pear`.

If NOT is not the first connector in a request, you need to use either AND or OR with NOT:

```
apple or not pear
not (apple w/5 pear)
```

The NOT W/ ("not within") operator allows you to search for a word or phrase not in association with another word or phrase. Example:

```
apple not w/20 pear
```

Unlike the W/ operator, NOT W/ is not symmetrical. That is, `apple not w/20 pear` is not the same as pear not w/20 apple. In the `apple not w/20 pear` request, dtSearch searches for `apple` and excludes cases where `apple` is too close to `pear`. In the `pear not w/20 apple` request, dtSearch searches for `pear` and excludes cases where `pear` is too close to `apple`.

Numeric Range Searching

A numeric range search is a search for any numbers that fall within a range. To add a numeric range component to a search request, enter the upper and lower bounds of the search separated by ~~ like this:

```
    apple w/5 12~~17
```

This request would find any document containing *`apple`* within 5 words of a number between *`12`* and *`17`*.

Numeric range searches only work with positive integers. A numeric range search includes the upper and lower bounds (so *`12`* and *`17`* would be retrieved in the above example).

For purposes of numeric range searching, decimal points and commas are treated as spaces and minus signs are ignored. For example, *`-123,456.78`* would be interpreted as: *`123 456 78`* (three numbers). Using alphabet customization, the interpretation of punctuation characters can be changed. For example, if you change the comma and period from **space** to **ignore**,
then *`123,456.78`* would be interpreted as *`12345678`*.