

Savvy searching

Query refinement by word proximity and position

Péter Jacsó

The author

Péter Jacsó is a Professor at the University of Hawaii, Manoa, Hawaii, USA.

Keywords

Searching, Internet, Information retrieval

Abstract

Traditional online information services have been offering proximity and positional operators for many years to narrow or broaden a search by specifying how far and in what order the words of the query should appear in the documents or in their surrogates to qualify for retrieval. The proximity and positional operators which define the distance and the order of the words required are not exactly intuitive and vary greatly from one online system to the other. Still, they are used by the savviest searchers intensively. This paper examines and evaluates various search engines in terms of word proximity and position.

Electronic access

The Emerald Research Register for this journal is available at
www.emeraldinsight.com/researchregister

The current issue and full text archive of this journal is available at
www.emeraldinsight.com/1468-4527.htm

Traditional online information services have been offering proximity and positional operators for many years to narrow or broaden a search by specifying how far and in what order the words of the query should appear in the documents or in their surrogates to qualify for retrieval. The proximity and positional operators which define the distance and the order of the words required are not exactly intuitive and vary greatly from one online system to the other. Still, they are used by the savviest searchers intensively. Few Web search engines offer such features for users, although many of them use word proximity and position in ranking the results. Proximity and positional operators are essential, especially when using full-text databases which are becoming more common even for end-users as publishers make available their very large full-text archives of journal articles and conference proceedings through libraries. Information professionals may have been willing to learn the many syntax variations of positional and proximity operators, but end-users need a much more user-friendly option for query refinement.

Implicit proximity and positional operators

The most important proximity and positional operator is an invisible one, the space character between query words. One of the most difficult things in teaching new students (and practitioners when switching systems) is that the space between two or more query words can have very different implications across different search systems, especially when the user can remain silent also about field suffixes and prefixes. Without the intention of sounding too blasphemous, it is as if using your right to remain silent would mean in one state to plead guilty, in another state to deny culpability, in yet another state to plead no contest, and in the rest of the states something in between.

In searching beyond entering the topical words, you have the right to remain silent about the specifics. The vast majority of users just enter two or three words separated by a space, and the consequences can be very surprising, depending on which system you use. In the Web-born systems the implications are mostly transparent, because many of the systems rank the results behind the scenes – among others – by the proximity and position of the query words in the retrieved documents. Some produce excellent results (like Google and AllTheWeb), and the latter even can convert your query into an exact phrase if the conversion option is enabled. Most

The author has used the following typography: **bold** = descriptor or other search term; *bold italic* = query/search command; *italic* = operator by itself (not as part of a query).



of them, however, do an abysmal job with ranking.

In the traditional information retrieval systems the situation is different, as you have several options to refine your query. At the most extreme is DIALOG, which has the far most sophisticated proximity and positional operators but treats the space between query words in the *select* command in an ultra-conservative and restrictive manner. For DIALOG the space implies that the search is to be restricted to the descriptor and identifier fields (if there is an identifier field in the database). In addition, the query words must exactly match the descriptor. For example, the query *select human machine systems* in PsycINFO (which I used across the online systems in all of the examples here unless otherwise noted) retrieved 1,845 records where this term is the descriptor. However, it did not retrieve 1939 records where **human machine systems design** is the exact descriptor, but not **human machine systems** because the software is looking for exact descriptor, not just part of a descriptor, unless you end the last word with a question mark.

Neither will DIALOG find any records where the exact term **human machine systems** appears in the record (say, in the title, abstract or full text), but not as an exact descriptor or identifier (see Figure 1). Luckily, DIALOG re-introduced in 1996 the *find* command that was originally introduced in 1983 and quickly withdrawn after two months. I bet it was deemed to be too easy for a professional online service. It searches for the query words next to each other in the given sequence (known as unidirectional

adjacency) in the Basic Index, typically generated from the title, abstract, descriptor, identifier and full-text fields, but not from the journal name or author affiliation fields. It is another question that this command is not discussed in the most widely used textbook for online searching, and was hardly mentioned in the professional journals.

The other extreme is represented by some of the search engines used by Web-born databases, such as FindArticles.com. Although its help file claims that it searches the query words in *AND* relationship, in reality it searches them in *OR* relationship if you use the Normal Search option. Of course, this yields absurd results that have nothing to do with the user's query. It has 121 chapters from the *Annual Review of Psychology*, and for the query **human machine systems** it finds 118 chapters matching the query. The three that were not retrieved are the ones which do not have in the full text of the chapters any of the three words. When you use the + signs before each query words the result list shrinks to 10 items, and when you put the query between quotes for exact phrase searching there is no hit. One of my favorite Web-born systems, HighWire Press also assumed *OR* relation when encountering space in the query, but recently it switched to the much more reasonable *AND* relation.

In between these two extremes are the dozens of systems which handle multiple word queries differently. CSA-IDS, for example, assumes exact phrase searching for space between the query words (which I find appropriate), and if you don't choose an option, it searches the entire record. If one of the options (title, keyword, author, source) is chosen, the phrase search is limited to that field's index, or in case of keywords, to the combined title, abstract and descriptor index.

Ovid also assumes phrase searching for space, but it does not require strict adjacency (see Figure 2). Stop word(s) may appear in records between the query words, and such records still match. This is fine, as you do not have to worry about how to handle stop words in a query, such as **bill of rights of patients**, just type in **bill rights patients**. On the negative side, however, you cannot search for **nursing at home** in Ovid, because the stop word is ignored (even when the query is enclosed between quotes) and you get the same 3,840 records as for the query about: **nursing home**. It is the confusion of the user about the results that is of concern, not the fact that a couple of records may not be retrieved.

ProQuest has a unique approach to handling space in the query. It handles two-word queries as exact phrase and ignores stop words, while in case of more than two words it uses *AND* between the words. This is a smart approach, but it backfires for queries like **nursing at home**

Figure 1 The ultraconservative SELECT command vs the progressive FIND command of DIALOG in processing a query with space (year of introduction of the descriptor is added automatically by DIALOG to the query)

Search History		
Database Details		
Set	Term Searched	Items
S1	HUMAN MACHINE SYSTEMS (1997)	1845
S2	HUMAN MACHINE SYSTEMS DESIGN (1997)	2151
S3	S1 OR S2	3784
S4	S1 AND S2	212
S5	S2 NOT S1	1939
S6	S1 NOT S2	1633
S7	HUMAN MACHINE SYSTEMS?	3784
S8	FIND - HUMAN MACHINE SYSTEMS	3816
S9	S8 NOT S7	32

Note: Year of introduction of the descriptor is added automatically by DIALOG to the query

Figure 2 Ovid's handling of space in the query and the adjacency operator with and without word distance parameter

#	Search History	Results
1	nursing at home.mp. [mp=title, abstract, heading word, table of contents, key concepts]	3840
2	"nursing at home".mp. [mp=title, abstract, heading word, table of contents, key concepts]	3840
3	nursing home.mp. [mp=title, abstract, heading word, table of contents, key concepts]	3840
4	(nursing adj home).mp. [mp=title, abstract, heading word, table of contents, key concepts]	3840
5	(home adj nursing).mp. [mp=title, abstract, heading word, table of contents, key concepts]	100
6	(nursing adj1 home).mp. [mp=title, abstract, heading word, table of contents, key concepts]	3880
7	(home adj1 nursing).mp. [mp=title, abstract, heading word, table of contents, key concepts]	3880

because it consists of three words, which will trigger an *AND* operation, followed by dropping the stop word, so it yields the same results as *home AND nursing* which obviously also retrieves nursing home. Still, it is a remarkably good idea which works well for the typical queries.

OCLC FirstSearch searches query words separated by space in *AND* relationship (in word indexed fields, including the keywords index used by default). It ignores stop words in a query like *nursing at home*. However, if there is a stop word in a query enclosed by quotes, such as "*nursing at home*", FirstSearch converts the query to *nursing w1 home*, i.e. replaces the stop word by a positional/proximity operator which specifies nursing must precede home and at most one word can intervene. This leads me to the next issue.

Explicit proximity and positional operators

If the differences in handling of space between query terms did not discombobulate you, the major differences behind the seeming similarities of explicit operators surely will. In OCLC First Search the *w1* operator specifies that the first word must be followed by the second word with maximum 1 intervening word, so *home w1 nursing* would retrieve home nursing, home health nursing, home care nursing, but not nursing home. This unidirectional proximity is just fine. The same is true for the DIALOG command if you use the *home 1w nursing* command, minding the syntactical difference

from the OCLC version. The closely resembling *home w1 nursing* command of ProQuest, however, specifies bi-directional proximity, where the two query words can be in either order, and will retrieve home nursing, nursing home, nursing staff home, because it requires only that the query words be within some undefined proximity, but does not specify order. ProQuest's command for unidirectional proximity is *home pre1 nursing*. The good news is that Lexis-Nexis uses almost identical query syntax.

CSA-IDS offers the within N option, such as *home within 2 nursing* which specifies a bi-directional proximity. It is to be noted that the number is required, and *within 1* does not mean that there may be one intervening word between the query words as in most other software, but that the query words must be next to each other in any order for the record to qualify for a match. CSA-IDS has two positional operators, *before* and *after*, but these offer no parameters and are of little use as illustrated by my intentionally weird test query *home before nursing AND nursing before home* which retrieved 6999 records, meeting both criteria, i.e. doing nothing useful.

In the WebSPIRS software the *with* operator requires that the query terms appear within the same field. Closer proximity operation is triggered by using the *near* operator. The command *home near nursing* finds all the 4450 records where (according the help file) the query words are in the same sentence in any order. Adding a number to the bi-directional *near* operator, such as *home near15 nursing* limits the maximum distance between words, although not within sentence boundary. My tests have shown that only the command *home near210 nursing* retrieves the same 4450 records as the *near* operator by itself. Not even my anaconda sentences (which make my editors cringe), or your lawyer's letter have such excessively long sentences. For unidirectional proximity WebSPIRS offers the *adj* operator. The *home adj nursing* command retrieves only records where home is followed by nursing. The distance cannot be specified, as it is absolute adjacency. As we saw, the similar operator in Ovid allows intervening stop words. Ovid offers adding a number after the *adj* operator, such as *home adj1 nursing* to allow one intervening word (beyond the stop word). However, if you add a number to the *adj* operator the command loses its unidirectional proximity requirement!

Post-query refinement

I have shown only the tip of the iceberg here. Of the largest commercial information services, Gale, H.W. Wilson and EBSCO have their own particular interpretation of the space character,

and implementation of the search syntax of proximity and positional operators. Add to these the archives of Elsevier, Kluwer, Wiley, Springer, to name a few major publishers that users will find in an academic library, and you realise the magnitude of the problem, even without throwing in the implications of the maddening variety of explicit and implicit truncation and wildcarding operations.

It is not realistic to expect users to cope with the inconsistencies and idiosyncrasies of the dozens of information retrieval programs they may encounter in the same library. Waiting for the implementation of a Common Command Language initiated in the mid-1980s is as realistic as waiting for Godot.

Expecting affordable federated search engines to translate a user query to reflect the same functionality when broadcasting it to a dozen search engines is naïve (more about this in an upcoming Savvy Searching column). The best solution could be to offer an option which I saw several years ago when using the Library of Congress' Experimental Search System (LC-ESS), which has, unfortunately, not become the standard software for the LC Catalog. However, it is still used for the relatively small and rather outdated Country Studies database of the LC Federal Research Division. It cannot do justice to the intuitive and user-friendly software, but is good enough to illustrate my point about how it analyses and groups the results into clusters, clearly explaining to what extent the items meet the query (see Figure 3).

Many of the current systems remind me of the aloof maitre d' of a posh restaurant: waiting with arms folded around the leather-bound wine list with that piercing and condescending gaze for my choice of drink – hopefully matching the food. The problem is that I fly blind, as I am not familiar with wines, let alone able to pronounce their names from the top of my head. I much prefer the user-friendly waiters who tell me my options with a mini-blurb of the wines' features and how they match the main course – after I selected the meal. Learning about the syntax and semantics of search operators, and committing to one of them a priori when formulating the query, is much less appealing to patrons than getting feedback to their queries with results in clusters by their level of matching the query in terms of the presence, proximity and order of its component words. Then again, there are gourmets who know exactly which vintage of which wine from which vineyard they want to have. They should still have the option of making

Figure 3 Post-query feedback with results labeled and clustered by proximity, position and presence of the query words

Search Results:

- 3 Sections containing the exact words *per capita spending health*.
- 2 Sections containing the words *per capita spending health* **near** each other.
- 35 Sections containing all of the words: *per capita spending health*, but **not** near each other.
- 3373 Sections containing one or more of the words: *per capita spending health*

Sections 1 through 20 of 3413

Sections containing the exact words <i>per capita spending health</i> .	
1	<u>Health Care</u> (Uganda)
2	<u>HEALTH AND WELFARE</u> (Caribbean Islands)
3	<u>HEALTH AND WELFARE</u> (Egypt)
Sections containing the words <i>per capita spending health</i> near each other.	
4	<u>Role of the Government</u> (India)
5	<u>SOCIETY</u> (Macau)
Sections containing all of the words: <i>per capita spending health</i> , but not near each other.	
6	<u>The Health Care System</u> (Brazil)

their order and query unassisted – with perfect syntax, accent, dialect and posture.

Further reading

- Hawking, D. and Thistlewaite, P. (1995), "Proximity operators – so near and yet so far", *4th Text REtrieval Conference (TREC-4)*, Gaithersburg, MD, November 1-3, National Institute of Standards and Technology (NIST), Special Publication 500-234, pp. 131-43.
- Jacsó, P. (1996), "What is in a space? Find out now", *Information Today*, Vol. 13 No. 11, pp. 9, 54.
- Jacsó, P. (1998), "Post-search aids for relevance ranking of results by the user in the experimental search system of the library of congress", *Proceedings of the 19th National Online Meeting*, New York, NY, pp. 177-86.
- Mitchell, P.C. (1973), "A note about the proximity operators in information retrieval", *Proceedings of the ACM SIGPLANSIGIR Interface Meeting*, pp. 177-80.
- Morton, D. (1993), "Refresher course: getting next to proximity operators", *Online*, Vol. 17 No. 6, pp. 56-8.
- Stojanovic, N. (2003), "On the role of query refinement in searching for information: the librarian agent query refinement process", *4th International Conference on Web Information Systems Engineering (WISE'03) December 10-12*, pp. 41-52.
- Vélez, B., Wiess, R., Sheldon, M. and Gifford, D. (1997), "Fast and effective query refinement", *Proceedings of the 20th ACM Conference on Research and Development in Information Retrieval (SIGIR 97)*, Philadelphia, PA, July, pp. 6-15.