

How to Shine a Search Light Through Terabytes of Data to Get to “Tag You Are It”

Have terabytes of data at your fingertips but no ability to find anything? This article lists hard-won tips after many years working for an enterprise and developer search engine software company. While the tips use terminology from the dtSearch® product line, these tips are generally applicable.

Build an Index

The first tip is to use the search engine to build an index instead of simply doing an unindexed search. Unindexed search is slow. Indexed search is typically instantaneous, even for multiple concurrent search requests across terabytes. (As a technical matter, concurrent indexed searches can run from different threads in an online or network environment without affecting each other.)

What is an index? An index is simply an internal tool that lets the search engine search terabytes in an instant. How do you get such an index? Just point to whatever you want to index, and the search engine will do the rest.

It is no problem if you don't have a clear idea of what is in your data. The search engine can automatically identify file formats like Microsoft Word, Access, Excel, PowerPoint, and OneNote; email files; PDFs; and web-based formats like HTML or XML. And the search engine can automatically sift through compressed archives like RAR and ZIP to index the files.

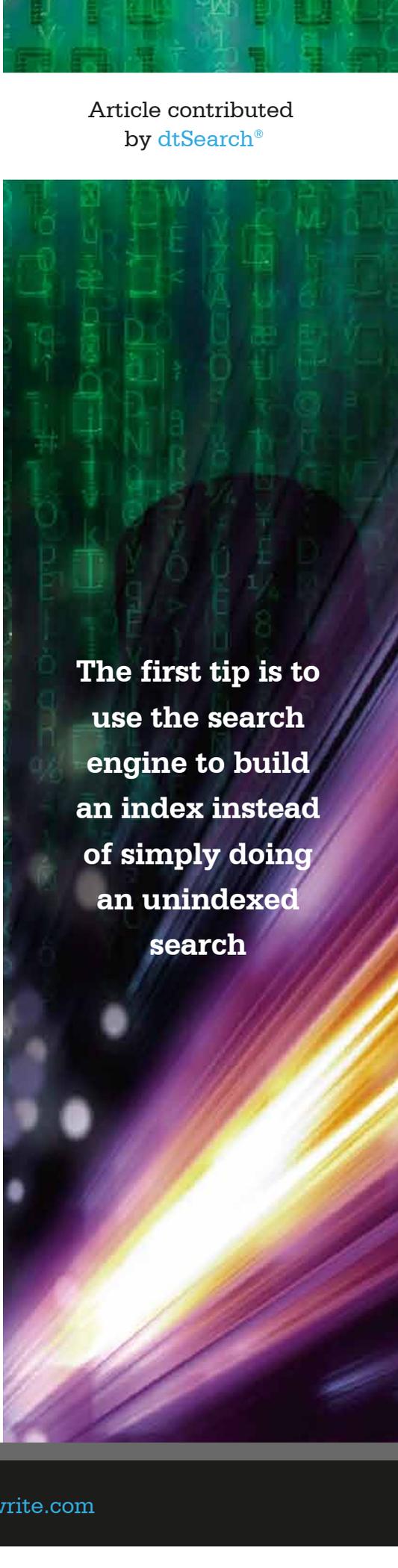
But what if some of the PDF files are saved with MS Word file extensions like .DOCX — and some Access files are saved with Excel file extensions, etc.? This situation also does not present a problem. The search engine's document filters which parse the data can look inside each file to determine the correct file type without reference to the file extension.

The document filters can also go through files looking for nested documents. If there is a ZIP or RAR file with an embedded Excel file and embedded in the Excel file is an Access database and a Word file, the document filters will find and parse the embedded documents as well.

Note that text that is black on black or white on white or red on red may be invisible when you view a file in that file's relevant application. But it is just straight-up text for a search engine.

One last pointer within the broader “build an index” tip: index email files directly, if possible as PST, OST, MSG, etc. files, without going through Outlook. The search engine can index Outlook emails through Outlook, but going through Outlook / MAPI will slow down the indexer relative to direct access to these file types.

Article contributed
by dtSearch®



**The first tip is to
use the search
engine to build
an index instead
of simply doing
an unindexed
search**

Check Index Logs

The second tip is to check the index logs. The logs can identify files that the search engine cannot index for whatever reason. A key example is “image only” PDFs.

An ordinary PDF combines text and images. You can tell that you have actual text in a PDF if you can copy and paste a selection of text into another file. But “image only” PDFs are different. If you try to copy and paste what may look like words from these, that process goes nowhere. Of course, with no actual text, just pictures, the search engine cannot index and search the contents of such files. (The search engine can still index metadata, but the main event will be missing.)

Here’s the tricky part: “image only” PDFs can occur in data collections along with ordinary PDFs with no external identifiers that these “image only” PDFs are present. But the indexing log file will flag “image only” PDFs. You can then run these “image only” PDFs through an OCR application such as Adobe Acrobat to turn them into regular PDFs and add these to your index.

Consider Document Caching

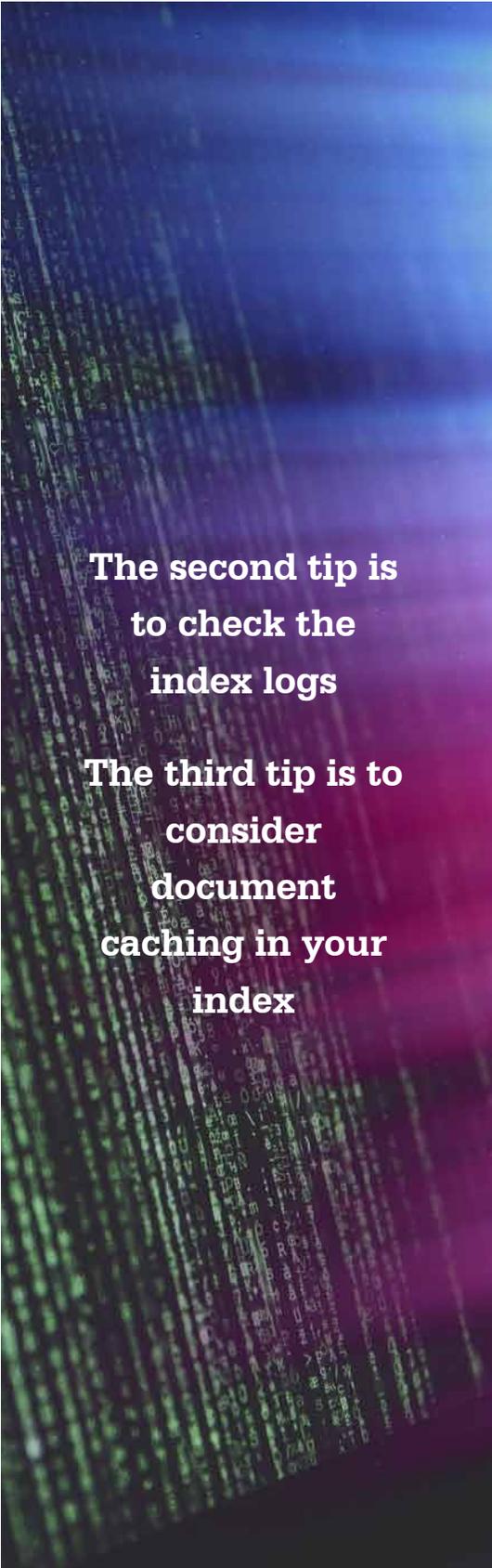
The third tip is to consider document caching in your index, where documents or other data are subject to a remote or otherwise unreliable connection or may even be completely unavailable in their original location. A quick explanation of how the search results display works helps explain this tip.

A search engine processes standalone and multithreaded search requests using data from the index itself. To display the full text with highlighted hits, the search engine goes back to the original file or other data to pull up a copy of that item. The search engine then uses the index to determine where the hits should be in that copy and marks those in the search results display.

Highlighted hits are quite literally the light that shines through your data. If the original file is easily accessible and quick to retrieve, this process is straightforward. However, if the original file is slow to retrieve or simply gone, the display process ceases to be seamless.

The answer is to cache or store a full copy of the file or other data along with the index itself. Using that cache, the display process remains smooth and instant even without access to the originals. The disadvantage to caching is that it makes the size of the index a lot bigger, as the index is now storing the complete text of all files along with the basic index itself. But when the original is slow or unavailable, caching is well worth it.

Article contributed
by [dtSearch®](#)



**The second tip is
to check the
index logs**

**The third tip is to
consider
document
caching in your
index**

Update Your Indexes

The next tip is to keep your indexes updated to reflect files that have been added, deleted, or changed. This process is easier than it may seem. To add something new doesn't require rebuilding an index from scratch. Rather, the search engine can automatically check each file and see if that file has been modified, deleted, or added since the last index build and simply index "the difference."

A compress option streamlines the extra baggage that can follow multiple index updates. You can also set automatic index updates via the Windows Task Scheduler at specific times. Importantly, searching, even concurrent searching can continue uninterrupted as an index updates.

Refine Your Search Request

The fifth tip is to pay attention to how you frame a search request. For example, natural language searching lets you enter a "plain English" search request or even copy and paste a paragraph of text and get relevancy-ranked search results.

I use the term "plain English" here to capture the essence of natural language searching. But note that a search engine can work automatically with any of the hundreds of Unicode languages, even right-to-left languages like Hebrew and Arabic, and double-byte languages like Chinese, Japanese and Korean.

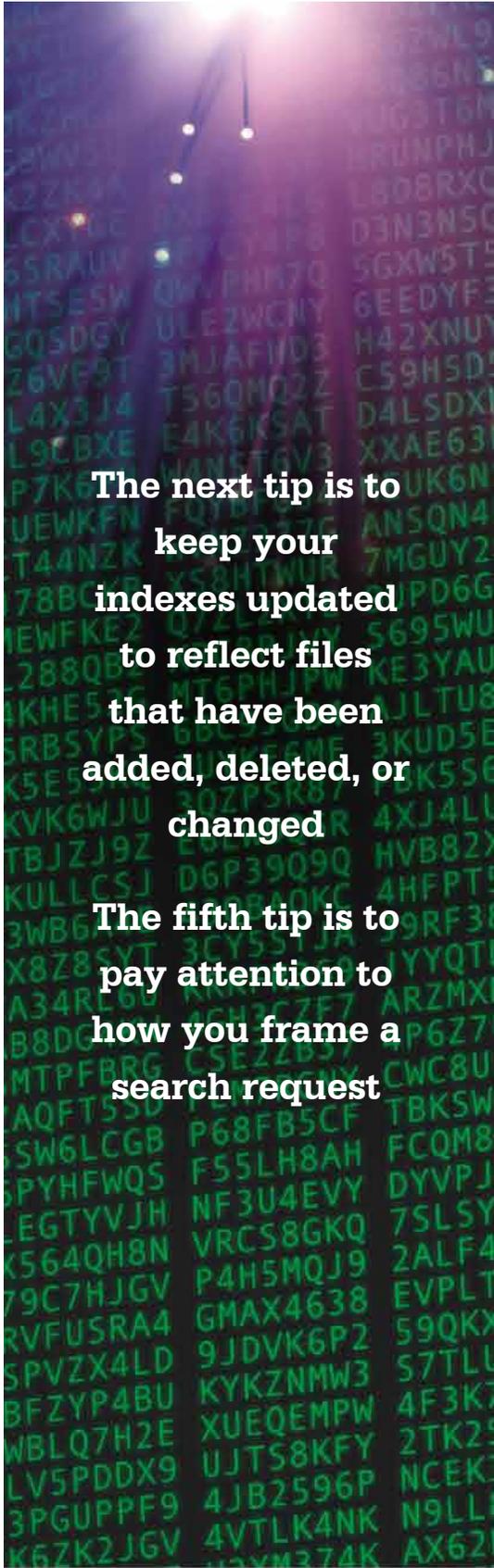
Underneath the hood, relevancy ranking works as follows. If you search for *purple* or *blue*, and *blue* is all over your indexed data, but *purple* references are much rarer, then files with *purple* will get a higher relevancy ranking. Furthermore, files with denser *purple* mentions receive an even higher relevancy ranking.

While natural language search requests require little effort to compose, it is often more fruitful to take the time to enter a precision search request instead. A search engine can also support phrase searching, Boolean and/or/not search requests, proximity searching in one direction (X before Y) or both directions (X before or after Y), concept searching, metadata-specific searching, number and numeric range searching, date and data range searching, and much more.

Use these different options to refine your search requests to get exactly what you are looking for. Also, don't forget about the more specialized search options, like the ability to identify credit card numbers in data, generating and searching for file hash values, positive and negative variable term weighting including in specific metadata, etc.

One specific search option that you may want to use as an add-on to both natural language and structured search requests is fuzzy searching. Fuzzy searching looks for minor typographical deviations that can crop up in emails and in OCR text. So, for example, a search for *purple* would also pick up *purqle* with a low-level of fuzzy search, to make sure that you find what you are looking for even with slight misspellings.

Article contributed
by [dtSearch®](#)



The next tip is to
keep your
indexes updated
to reflect files
that have been
added, deleted, or
changed

The fifth tip is to
pay attention to
how you frame a
search request

A final point regarding search requests: you are not stuck with your default sorting option. If you have natural language searching as the default sorting option, you can click to immediately change that to sort by ascending or descending file date, ascending or descending file size, the presence of keywords in specific metadata, etc. All of these options add a different window into search results and retrieved items.

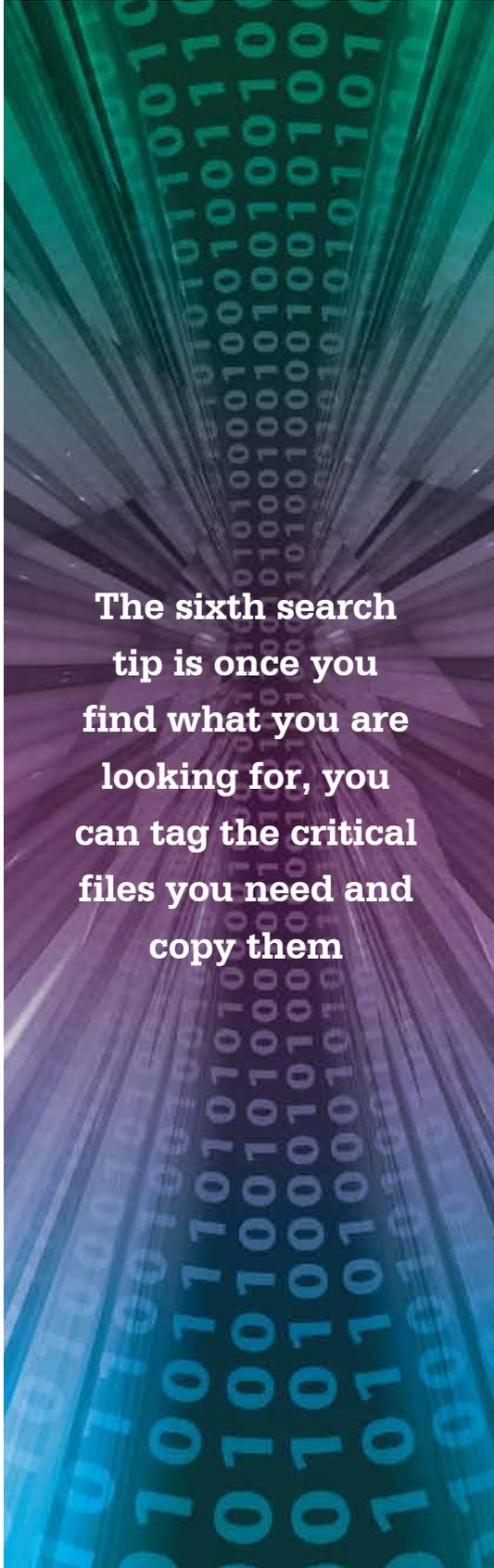
Tag Relevant Files

The sixth search tip is once you find what you are looking for, you can tag the critical files you need and copy them. You can even copy select files from inside a larger email archive or a compressed ZIP or RAR-type archive (no separate “un-ZIP” required).

You can also tell the search engine to prepare a search report showing all hits with as much context around each hit as you want. Search reports can work across all retrieved files, or you can tag the files to include in a search report and limit the search report to just these.

These tips will help shine a light through terabytes of data, whether the data you are working with is your own or from a third-party where you've never seen the dataset before.

Article contributed
by [dtSearch®](#)



**The sixth search
tip is once you
find what you are
looking for, you
can tag the critical
files you need and
copy them**