Thinking about reorganizing your data so that you spend less time looking for what you need? ... dtSearch has a simple hack for your problem: download a search engine.

Article contributed by dtSearch®

Thinking about reorganizing your data so that you spend less time looking for what you need? Elizabeth Thede of dtSearch has a simple hack for your problem: download a search engine.

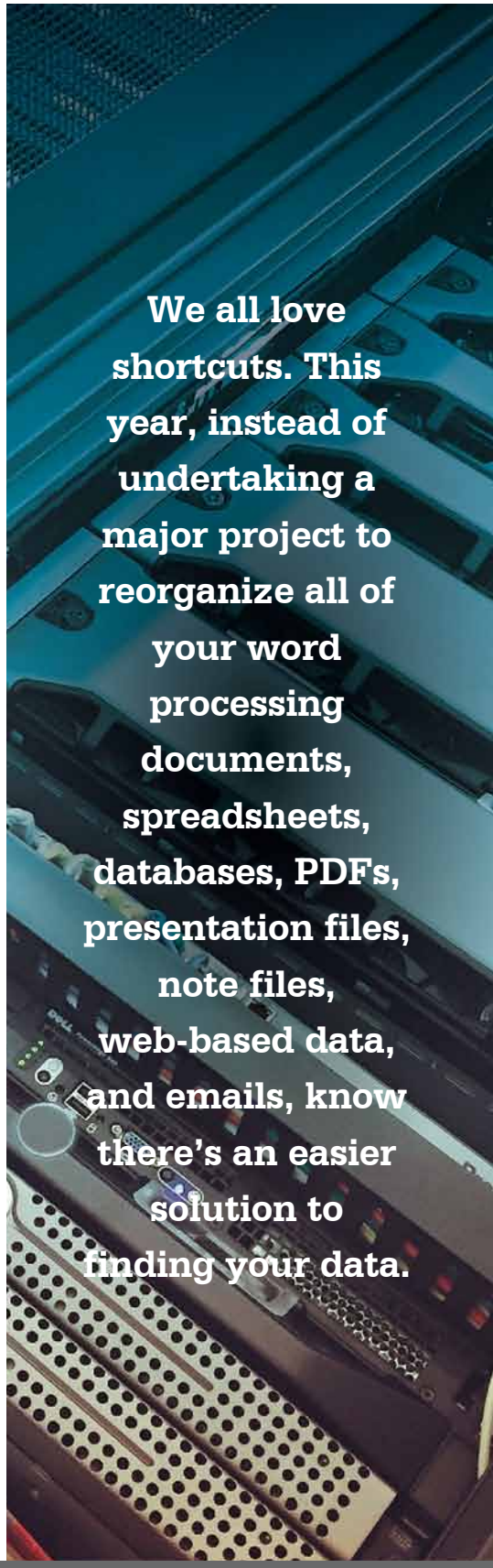
We all love shortcuts. This year, instead of undertaking a major project to reorganize all of your word processing documents, spreadsheets, databases, PDFs, presentation files, note files, web-based data, and emails, know there's an easier solution to finding your data. A search engine can instantly search terabytes without any requirement that you personally organize anything.

Behind the scenes, there are two different technological components at work in a search engine. The first component is data recognition and parsing. When you view a Word file in Microsoft Word, the Word program pulls up a specialized view of the file that you would typically think of as "your document." But to pull up each Word file in Microsoft Word to search across all of them would be painfully slow. By contrast, a search engine approaches files in their binary format.

The binary format view of a file can look vastly different from the application view of that same document. In fact, in many binary files, it is hard to recognize any document text at all through the sea of binary coding. For that reason, the first step a search engine takes is sifting through that binary format to recognize all of the text and metadata. The component that does this is called the document filter.

But what if you have a mix of Microsoft Word files, Excel spreadsheets, Access databases, OneNote files, PowerPoints, PDFs, HTML files, XML files, etc.? And what if some of these files are in compression archives like ZIP or RAR? And what about emails that can have any of these as attachments?

Document filters can automatically recognize different formats. In doing so, smarter document filters will look at the binary file to recognize the file type, not the file extension. That way, if you have a Word file accidentally saved with a PDF extension, the document filters can still correctly handle the file. Modern file formats can also be multilayer. If you have an email with a ZIP attachment containing a PDF and an MS Access database, and inside the MS Access database is an Excel spreadsheet, the document filters can work through all of that.



Reprinted with permission of Toolbox, toolbox.com | Original article link (1/6/2021)

The second component, after recognizing all of the text and metadata, is building an index. What the index does is record each unique word or number and its location in the data. A single index can span multiple different data repositories at once, and even different types of data repositories like network data and web-residing data.

But doesn't building an index take work? Yes, for the search engine, but not for you. Just point to any number of directories, email archives, online data repositories, etc., that you want the search engine to index, and the application will automatically index all that data.

A search engine like dtSearch can hold up to a terabyte in a single index, and there are no limits on the number of indexes that you can generate and simultaneously search. If you need to update an index to accommodate additions, deletions, or edits to your data, you can specify that the indexes automatically refresh at specific times. The index update process will not lock out searching.

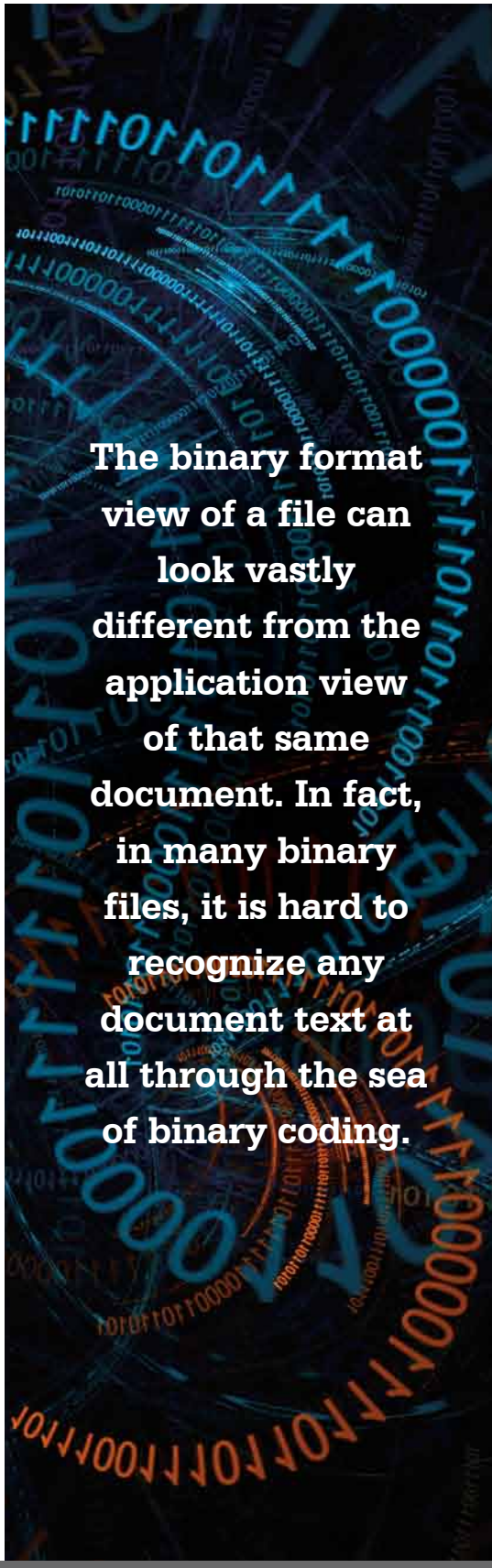
Indexed search can operate in different ways. You could individually search your own PC or laptop data, or multiple users can concurrently search network data. An online search can run in a stateless manner for multithreaded operation. That way, multiple users can concurrently search across a locally-hosted website or on a remote platform like Microsoft Azure or AWS without affecting search performance.

After a search, the search engine can display a copy of the file with highlighted hits for easy browsing. In doing so, the search engine returns to its document filters as it goes back to the original file for display with highlighted hits. Alternatively, a caching index option can store the full-text and metadata of everything inside the index itself, avoiding the need for the search engine to return to the original files at all.

Best of all, an index can support over 25 different search options. Federated search automatically lets you search across multiple different data repositories in the same index or in different indexes, including integrated relevancy ranking. Natural language search lets you enter a "plain English" search query, similar to how a Boolean "or"/ any words search works.

The application will then automatically rank retrieved files by hit term density and rarity, or vector-space ranking so that you get the most relevant documents first. That way, if marzipan is in a ton of files, but licorice is just in a few files, licorice files (and particularly densely packed licorice files) would get a more prominent ranking.

Article contributed
by dtSearch®



Along with Boolean "or" / any words search comes Boolean "and" / all words search plus Boolean "and not" search. The Boolean search expression can include individual words or specific phrases. A proximity search looks for a word or phrase within X words of another word or phrase. And the X words can be in either direction of each other or with one word only before the second word or phrase.

Other search options include phonic for sound-alikes, wildcards, thesaurus or concept search expansion, specific metadata search, regular expression search, number matching and numeric range search, and date or date range search.

Using any of these search features, the product can rank retrieved files via a vector space ranking system. You can also override default ranking by adding a positive or negative ranking to certain keywords. You can even provide a positional higher or lower ranking to keywords appearing in certain metadata or near the top of a file.

The application can also locate credit cards that may appear in files. More advanced forensics-oriented search options include the ability to generate a hash value for each file and then search on that hash value. Unicode support works automatically with all Unicode text. That covers pretty much any international language text, even double-byte Chinese/Japanese/Korean text and right-to-left languages like Hebrew and Arabic.

For developers, an SDK can offer even more search options like the ability to classify search results so that not everyone can access all of the same search results for security or other reasons. In that case, the developer can filter search results across a group of users or even on an individual-by-individual user basis.

Filtering can leverage any of the following: existing metadata inside of files; metadata added "on the fly" while indexing; full-text keyword presence; or metadata in a separate database (SQL, NoSQL, SharePoint, etc.) with files residing as referenced data or stored internally as BLOB data. And the same metadata, etc., can also form the basis of faceted or "drill down" user interface searching.

Bottom line: resolve to use a search engine to quickly sift through your data – no reorganization necessary!

Article contributed by dtSearch®