Cross Organizing Your Files off Your 2022 To-do List? Download a Search Engine Instead

Article contributed by dtSearch®

Make 2022 the year of never organizing your files again. A search engine can give you all of the benefits of organizing with none of the time-consuming efforts.

Throughout history, there have been two categories of people. Type A shows up at the ancient gathering place or modern supermarket with a neat list of every ingredient to bake holiday cookies. Type B shows up with no list and always ends up forgetting something. Type A takes the holiday cookies out of the oven, leaving extra time to wrap presents. Type B has to go back to the gathering place or supermarket because Type B forgot the sugar or butter. Or worse, Type B tries to bake the cookies without sugar or butter and ends up throwing out the entire batch.

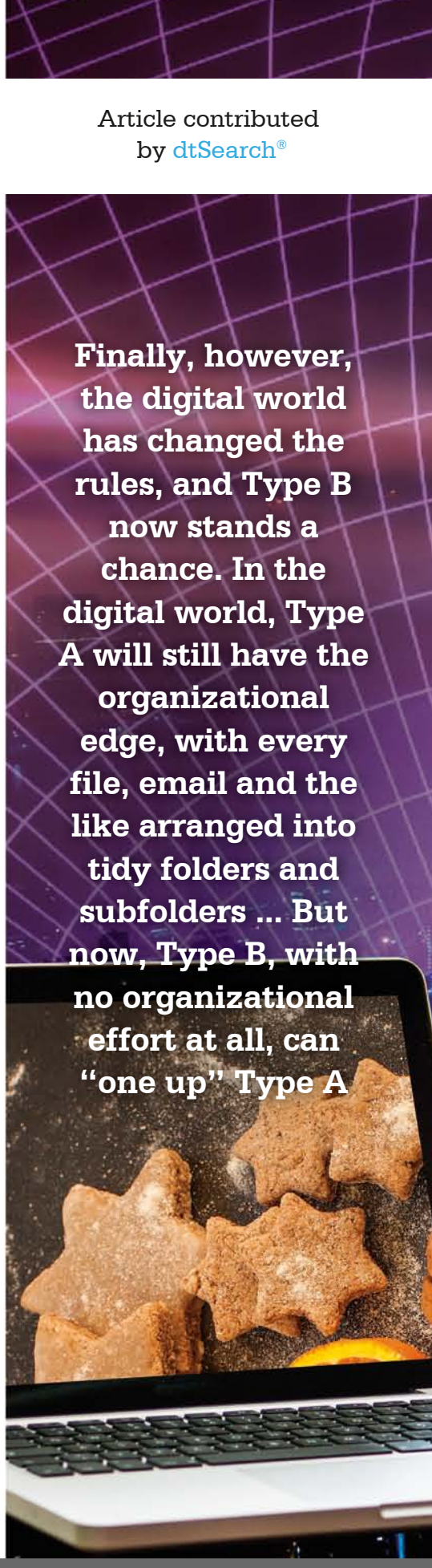
Finally, however, the digital world has changed the rules, and Type B now stands a chance. In the digital world, Type A will still have the organizational edge, with every file, email and the like arranged into tidy folders and subfolders. Following this organizational effort, Type A can navigate to the holiday cookie subfolder with just a few clicks. But now, Type B, with no organizational effort at all, can "one up" Type A, jumping immediately to the perfect recipe.

Type B's secret is a search engine. To be clear, this is not a "search the whole Internet" search engine like Google. Rather, this is a "precision search your own data" search engine like dtSearch.® This type of search engine can instantly find anything in the full text or metadata of terabytes of "Office" files, emails plus attachments, PDFs, compressed archives like ZIP or RAR, web-ready formats, etc., with no organizational requirement at all.

How does a search engine work this magic? By first building a search index across all the data. But doesn't it take a lot of work to build an index, Type B wants to know? Yes, but only for the search engine. Simply point to the folders and other data to index, and the search engine does the rest.

Let's take a look under the hood at what the search engine is actually doing. If Type A wanted to look through each item in the holiday cookie subfolder, Type A would need to go file-by-file, retrieving each item in its associated application. Type A would need to open each Microsoft Word file in Word, open each OneNote file in OneNote, etc.

Finally, however, the digital world has changed the rules, and Type B now stands a chance. In the digital world, Type A will still have the organizational edge, with every file, email and the like arranged into tidy folders and subfolders ... But now, Type B, with no organizational effort at all, can "one up" Type A



By contrast, a search engine bypasses this associated application view entirely, going straight to the binary format of each item. Parsing a binary format requires two major steps. First, the search engine has to figure out what type of file a specific item is. This is critical, as the parsing specification for an email is very different from the parsing specification for a PDF, for example. Second, the search engine needs to apply the correct parsing specification to "read" all text and metadata.

The binary format approach has definite advantages in terms of enabling comprehensive search:

- Because the search engine goes into the binary format to figure out the relevant file type without reference to the filename, mismatched file extensions do not stymie the search engine's work. A PDF can have a .DOCX extension, and a OneNote file can have a .PDF extension and none of that will impact the binary format parsing.
- Metadata that may require a great deal of "clicking around" to see in an associated application view is immediately apparent to a search engine in binary format.
- The search engine can handle multilevel nested formats inside a "single" binary item. The holiday cookie recipe can be in an Excel spreadsheet embedded in an Access database as part of an email ZIP attachment. The search engine can sift through all of that (without needing to pull up each component part in its associated application).
- Finally, black on black or white on a white text that may be invisible in an associated application is fully transparent to the search engine. In fact, such text is the same as any other type of text, making any secret notes on a recipe readily discernible in binary format.

When complete, an index includes a compilation of each unique word and each unique number in the full-text and metadata and the location of each unique word and number in the data. Using just that information, the search engine can immediately jump to just the right files. Type B can search for holiday cookies and white chocolate chips and cinnamon within 12 words of sprinkles and not pumpkin spice, and the search engine will go straight to matching items, displaying each with highlighted hits.

If Type B has made a resolution to eat more healthfully, Type B could enter the same search request adding a requirement for low-fat in recipe metadata. With a small level of fuzzy searching, if *cinnamon* is misspelled *cinnaqon*, the search engine can still find it. And suppose a family member has recipes for *holiday biscuits* instead of *holiday cookies*. Type B can make *biscuits* and *cookies* synonyms for purposes of the search.

Article contributed by dtSearch®

The search engine can look for numeric ranges, such as locating recipes that serve ten or more people. Or the search engine can search for date ranges, finding recipes that date from September 12, 1998, to 12/15/20 (correctly processing mix-and-match date formats). A search can even locate any stray credit card numbers hiding in the recipe data, so Type B can find these and safely delete them. And all of this works not only with English text but any of the hundreds of international languages Unicode supports, even Asian double-byte character languages and right-to-left languages.

If there are just a handful of files that match the search request, then Type B is all set. But what if the same search retrieves hundreds or even thousands of matching files? Where there is an abundance to choose from, the search engine lets you add different options for relevance ranking.

With default vector space relevancy ranking, a rarer word in the data (perhaps cinnamon vis-à-vis chocolate) would get a higher relevance ranking, particularly if there are comparatively denser mentions in a single file. Or Type B could apply custom positive and negative relevance ranking to each keyword. Or Type B could just instantly sort or instantly re-sort by a completely separate criterion like file date.

But what if Type B is not alone in Type B's digital universe but is instead part of a multi-user office setting? In that case, the search engine supports concurrent searching with no limit on the number of search threads that can proceed instantly and independently. Concurrent searching can operate in a classic Windows network environment, from an "on premises" web server or on a cloud platform like Azure or AWS. And even updating an index to add the latest in cooking baking technology does not affect concurrent searching.

Type B, whether using a standalone PC or running in a shared search environment, can finally beat Type A in getting to those holiday cookies.

Article contributed by dtSearch®