

The technology surrounding the detection of duplicate documents, emails, attachments, etc. is a very mature science at this point and can be relied upon to do a very good job at isolating duplicates. The two key areas fall into “Exact” duplicates, and “Near” duplicates.

## **EXACT duplicates:**

Detecting an “Exact” duplicate involves generating a hash or message digest from the source document, email etc. and storing this calculated # in a database. Every document has a Hash calculated for it, then that hash is compared to the other Hash values in the database. If the Hash exist more than once, it’s an exact duplicate.

earlyCASE uses the MD5 Hash algorithm to identify duplicates. At 128bits in length it can be calculated quickly and does a good job of creating reliable uniqueness to base the duplicate detection process on. When running in Professional Analysis mode you may elect to calculate and store SHA-1 hashes (160bit) and use these as the basis for duplicate detection.

Hash algorithms (SHA-224, SHA-256, SHA-384, SHA-512) using more bits are generally used for encryption purposes and are overkill for duplicate identification.

Handling duplicate documents, emails, etc. in a manner that ensures you only process and review that document one time will save you substantial time and money as you move through the processing of working with electronically stored information (ESI). **GOOD PRACTICE:** Use the duplicates reports to reach agreement among everyone involved to remove duplicates PRIOR to processing and review.

## **NEAR Duplicates:**

This is where the advanced math comes in, and a bunch of PhD mathematicians have figured out how to identify “Similar” documents. In layman’s terms, a hash value is generated for a groups of words and stored, systematically the document is decomposed into a sequence of these hashes (known as Shingles) and form a multipart fingerprint to a document. An Exact duplicate depends on a single Hash value to determine if the documents are exactly the same. Near duplicate detection looks at the number of “Exact” same Shingle Hashes that are the same between this document and ALL of the other Shingle Hashes that are in the database.

Once you know how many matches you have within a document, you also know how much of the document DOES NOT match anything you have in the collection. By setting a threshold of “Resemblance” you can control how conservative or Aggressive the Near Duplicate Detection engine classifies a document (or group of documents as a duplicate). If you are more interested in some of the science behind this, have a look at some of the published articles at Princeton.edu. This link is a good basic primer:

<http://www.cs.princeton.edu/courses/archive/spr05/cos598E/bib/Princeton.pdf>

## So how reliable are these Approaches:

Both approaches are very reliable, EXACT duplicates are low hanging savings and it should be the NORM to remove or isolate these from the collection before you do any processing or review. Near duplicates are a little tougher sell if you are negotiating with another party to treat near duplicates as exact duplicates. The primary reasoning for this is the concept of False Positives and False Negatives – in essence people are fearful that you will treat something as a duplicate that is NOT in fact a duplicate and they will not get a document that should have been treated as responsive and produced. Expect resistance in regards to eliminating documents based on “Near” duplicate algorithms for a while.

## So how do you leverage “Near” duplicate detection:

Using a reasonableness test is a good starting place for this. FIRST – remove the EXACT duplicates, get them out of the way. SECOND – tag, group, cluster (whatever you want to call it) the “Near” duplicates so you can deal with possible duplicates together as one. NEXT – start with an aggressive approach (a Lower % resemblance that causes a document to be suspected as a duplicate). Look at the documents that resemble other documents, if you think some are NOT duplicates, use a higher % resemblance (ie more conservative) setting and repeat the process of looking at the groupings. When you have reached a resemblance setting that you feel has in fact grouped just the documents that are really duplicates. You are set to use this quickly look at the near duplicate groups. In some cases once you are comfortable with the resemblance settings you have arrived at you can use this to treat “Near” duplicates with this degree of resemblance or higher as if they are Exact duplicates for the purpose of reducing the population of documents to be processed and reviewed.

**GOOD PRACTICE:** Document your decision process and tested documents / groups in case you are called upon to explain why you did what you did.

**GOOD PRACTICE:** Reach Agreement on “Near” duplicates ahead of time before you eliminate any of them from the document population. Some teams, use “Near” duplicate tagging in the document review process and NOT as a means of reducing the overall population.

## Processing Power:

Because “Near” duplicate detection involves generating exponentially more Hashes it will take much longer to process and create a larger database to house the hashes. Using a computer to help identify documents which are POSSIBLY duplicates is a very good idea. Regardless of how you leverage this technology, it will save you time and money.