An Intelligent Approach to E-discovery

Steve Akers

CTO/Founder Digital Reef Inc.

Boxborough, Ma

Jennifer Keadle Mason, Esq.

Partner Mintzer, Sarowitz, Zeris, Ledva & Meyers, LLP

Pittsburgh, Pa

Peter L. Mansmann

CEO Precise Litigation Inc.

Pittsburgh, PA

Introduction

Legal Discovery and assessment is an expensive proposition for corporations and organizations of all types. Last year (2010) it is estimated that $1 billion – $3 billion was spent on legal discovery processing alone[1.] This cost is large and growing; finding more intelligent methods to assess Electronically Stored Information (ESI) and understand what is contained within it is a goal of not just corporate personnel but also lawyers and legal service providers (companies providing legal discovery services). This paper outlines a proposed "standard" methodology and defines "ideal" tools and technology methods that combined are suggested as a "standard" for search. It focuses on 1. a standard methodology to identify potentially relevant data, and 2. tools and technology that can aid this process. It discusses important technological aspects of Ediscovery and how products either address or fall short of perfection in certain areas. Using the best process of identification in combination with the proper technologies for specific data types in order to have resulting cost-effective Ediscovery is the focus of this paper.

One of the quandaries facing attorneys is how best to approach any particular data set to identify potentially relevant information either for their own use or use in responding to discovery. Increasing variety of data types and sources along with expanding volumes of unstructured data has made the decision of how to search the data more imperative than ever. Analytic search tools have blossomed in this environment and certainly provide some of the best options for searching. However analytic searching has many flavors in and of itself. Understanding the pros and cons to each approach is important in deciding which route to go. In addition attorneys cannot ignore "traditional" search methods as they can be an effective supplement to analytic searching or in some cases may be the best primary method for running a search. The decisions about which route to take is largely driven by the types of data being searched, the relative organization of the data being searched, the particularity of the case facts, and the attorneys familiarity with the case facts and client.

The application of keywords has long been the standard for searching data sets. Keyword searching in its basic form is identifying any documents that contain particular terms. Ideally the parties discuss the keywords to be run, review a report of the initial search results to discuss any necessary adjustments, apply a privilege filter, review, and produce. These steps may be repeated numerous times to allow the parties to apply new search terms based upon the knowledge gained in reviewing the records. The problems with keyword searching are several and include: The parties must have sufficient knowledge of the case facts and industry/party parlance; straight keyword searching will not find misspellings; natural language usage has the problem of synonymy (multiple words with the same meaning – kitten, cat, feline) and polysemy (same word having different meanings – strike); finding variations of people's names can be difficult (Dr. Jones, Indiana Jones, Indiana J.).

Because of these difficulties in running straight keyword searches, variants on the searching were developed to work around some of the deficiencies. Attorneys began running keyword searches in conjunction with metadata searches. Star searching allows the user to find root words to account for variations (interp* - would find interpret & interpretation). Fuzzy searching allowed users to find words within a certain percentage similarity of the word being searched. Proximity searching allowed

[1] Source: <Marketing to supply reference to report by Gartner or Forrester>

users to search for words within a certain distance of other words of each other. These variants on the keyword search alleviated some of the issues discussed above, but still didn't overcome the obstacles of synonymy and polysemy. This is where analytic searching has come to the forefront.

Analytic searching in, its most rudimentary explanation, is a method of finding or grouping documents based upon the content of the documents themselves not solely on a keyword(s) being used. This is commonly employed by internet search engines that allow the user to type in a basic subject inquiry and retrieve search results that aren't solely driven by the words entered into the search box. The basis for this search technology is the conversion of a document's contents into numeric values that allows the computer to compare differing document's values in order to determine similarity of content. By approaching document comparison in this way, specific terms (or even language) of a record becomes irrelevant to the determination of similarity.

Alex Thomo an Associate Professor in the Department of Computer Science at the University of Victoria offers the following example to explain the basis for how analytic searching (and in particular a Latent Semantic Analysis) operates in determining documents responsive to a search request:

> Suppose there is a set of five documents containing the following language:
>
> > Document 1: “Romeo and Juliet”
> >
> > Document 2: “Juliet: O happy dagger!”
> >
> > Document 3: ”Romeo died by dagger.”
> >
> > Document 4: “Live free or die - New Hampshire's motto”
> >
> > Document 5: “Did you know, New Hampshire is in New England?”
>
> A search is conducted for: ***dies, dagger***
>
> A classical IR system *(for our purposes keyword searching)* would rank d3 to be the top of the list since it contains both *dies*, *dagger*. Then, d2 and d4 would follow, each containing a word of the query.
>
> However, what about d1 and d5? Should they be returned as possibly interesting results to this query? A classical IR system will not return them at all. However (as humans) we know that d1 is quite related to the query. On the other hand, d5 is not so much related to the query. Thus, we would like d1 but not d5, or differently said, we want d1 to be ranked higher than d5.
>
> The question is: Can the machine deduce this? The answer is yes, LSA does exactly that. In this example, LSA will be able to see that term *dagger* is related to d1 because it occurs together with the d1's terms Romeo and Juliet, in d2 and d3, respectively.
>
> Also, term *dies* is related to d1 and d5 because it occurs together with the d1's term Romeo and

> d5’s term New-Hampshire in d3 and d4, respectively.
>
> LSA will also weigh properly the discovered connections; d1 more is related to the query than d5 since d1 is “doubly” connected to *dagger* through Romeo and Juliet, and also connected to *die* through Romeo, whereas d5 has only a single connection to the query through New-Hampshire.

Using the above as an example, its apparent that analytic search engines can have a significant role in searching by obviating some of the problems of straight keyword searching. However, this does not mean that analytic searching alone will always be the most defensible method of searching. In addition when running analytic searching, its important to understand the different analytic engines and the limitations of each.

With all that said, in an attempt to identify a standard search process, this paper will first identify the problem, that is to determine what data you have and who has it. Next, the paper will elicit standard characteristics of a proposed standard search process. These will include identification methods and search and process methodologies to identify potentially relevant data in an effective, efficient and repeatable manner. Finally, the paper will discuss why those search and process methodologies are suggested for the search standardization model proffered. (The reasons for the identification methods proffered have been discussed in many articles, blogs and cases. Therefore, they are not discussed herein.)

Deciding What You Have (where you have it and how much)

The first problem with Ediscovery projects is assessing the magnitude and characteristics of the data in common knowledge repositories (email archives, SharePoint repositories, etc.). IT or Litigation Support professionals know they have a lot of data but are not sure where they have it and what these repositories contain. Not understanding the locations of data in an organization may seem like an odd statement but, for example, departments put SharePoint servers into production and users copy data to shared drives on networked file systems without knowing what they have copied. In other circumstances, administrators may not have knowledge of what systems are used for what data. These types of problems are ubiquitous and growing. The first step to effective assessment of a potential legal matter is to know what exists in various repositories within an organization. This is often the biggest problem in Ediscovery; identifying how much data exists and where it exists.

Legal Problem: Identification of Potentially Relevant Data

When litigation is filed and/or is reasonably anticipated, parties and counsel are required to identify and preserve data that is potentially relevant to the claims or defenses of the parties. In order to meet this obligation, parties have utilized numerous methodologies with varying levels of success. Success is often dependant upon the participants knowledge of the location of data as well as their understanding of the legal requirements and the technology involved. However, there has been no standard method to accomplish the task of searching for and reliably and efficiently locating that data.

Deciding who has it

Another big problem is in knowing who owns (or is responsible for) the information that is stored in various repositories. When a custodian (or potential custodian) is identified, it is important to know where their data might reside. Many organizations don't have any idea who owns what data and how often the data is accessed (if ever).

Technological Problem: Lack of Ownership insight

Historically data indexing and search solutions have not provided support for just a quick scan of file ownership in a short period of time to show what data requires further deeper analysis. Historically data has required full content indexing and analysis to provide insight into what it contains. Often the first level of analysis should be just a "who owns what" look at available data. In this case not all content needs full indexing. An intelligent approach is to perform a first-level analysis with just Meta data indexing and to then identify what content needs full content indexing. These are different "representations" of the data; one in Meta data form and one with all the content in the documents represented within the index. Systems with the ability to "represent" data in various ways let (users) reviewers decide what to deeply analyze. This saves time, storage space and lots of money.

Deciding What to Look For

A legal complaint will contain key facts about the case that will get the lawyers started on what they should ask the legal counsel representing an organization to identify and produce. A set of analytics that can assess the main complaint language or other "known key terms" and use these data to help build a set of "similar" documents would be very valuable to legal staff working on a case. Analytic processes that can expose "terms of interest" within a document to help lawyers involved with the case decide what to look for in other documents would be of great assistance to legal reviewers. Analytics to identify content that is "similar" to known example content is also very valuable.

Legal Problem: Identification of Potentially Relevant Claims/Defenses/Data

Upon identification of potential litigation and/or receipt of an action that has been filed, counsel must identify potentially relevant data and preserve it. How to most efficiently and effectively accomplish this goal is the problem faced by counsel and vendors alike. The proposed standard search methodology would begin with a litigation hold which involves identification of the "relevant topics" for the case. Relevant topics include but are not limited to the claims or defenses. Relevant topics might also include particular areas of interest for the litigation including, for example, profits/losses, prior claims, prior knowledge, investigations, testing, etc. in products liability cases. Once the relevant topics are known, the next area of inquiry is to identify the key players who might have possession of and/or who have created potentially relevant data. Key player questionnaires should be sent to these individuals. The questionnaire seeks information from the key player about "basic" data of which they are aware, why they were named, what position they hold, time frames of relevance, what documents they create in that position that might be relevant to the known relevant topics and where they store that data. It also should contain basic questions about the media on which they store information and where it is mapped and/or backed up. After this information is identified, a data map for the litigation should be drafted and key player interviews held. The interviews are usually more productive in person where information sources located in the office, but often forgotten, can be identified. The interviews should be a much more detailed analysis of the way the corporation works, where data is stored, to whom it is copied, the purpose for which it is created, etc. The locations of the known potentially relevant data should also be discussed and, if possible, the path followed to locate specific server, drive, file names, etc. The data map for the litigation should be updated with this information and the client should verify the information contained therein by signature. Once the specific known locations are identified and all relevant topics have been discussed, known relevant documents can be pulled for use in creating better search parameters for further collection of data. In addition, once additional known relevant documents are located through the analytical search processes, the information from those documents can be utilized to search for other potentially relevant documents. Further, the terms/phrases from these new documents can be compared to the search results, i.e. clustering, to more efficiently identify potentially relevant data. In other words, the process should be iterative. In the meantime, an IT key player questionnaire should be sent to the person responsible for IT to determine the data architecture of the entity and backup/legacy information. The identification of mapping should also be sought along with information as to third party entities who maintain data and/or website information. Finally, IT and all key players should be asked to discontinue any document destruction, turn off auto delete and auto archive, and identify backup rotation. Potentially relevant data should be properly preserved depending upon data type and business capability until further decisions are made.

Technological Problem: lack of an "analytics toolkit" and Lack of Flexibility and Scale

Vendors have historically pushed one approach or solution on customers for Ediscovery. Every solution requires a search capability; when the solutions begin to contain analytics the vendor approach has been to offer a single type of analysis. One type of analysis does not always product the best results with all data sets. Sometimes email is the only source of data pertinent to a matter. One set of tools for email analysis may work fine for such a case. With other data pertinent to the same case, key evidence may exist in MS Word documents and email analysis techniques are not appropriate. This fact of life in Ediscovery has caused legal reviewers to turn to multiple solutions that are stand-alone applications. Moving data into and out of these applications introduces complexity and potential for error (human and otherwise). One platform providing a number of analytic tools that are appropriate at various times throughout the lifecycle of a case would be the most efficient approach to take for legal discovery. In addition, historically data indexing and search solutions lack the flexibility and scale to analyze the amount of data that may exist within a typical organization. A platform that could analyze large volumes of data efficiently would be helpful.

Deciding who shared what (and with whom)

Conversational analytics are very important to an Ediscovery solution. Knowing who spoke with whom about certain topics is often the cornerstone to legal analysis.

Technological Problem: lack of capability or full-featured capability for conversations

Some solutions use email header analysis, others use Meta data analysis and header analysis, others rely on message content. A solution that can identify content and header similarity is often the best solution. Providing this capability at scale is a challenge in many solutions.

Solution: A “Standard” Ediscovery Process

The solution to many of these problems with Ediscovery would be contained within an “standard e-discovery system” that connects to many sources of local data (behind the corporate firewall), to help litigation support personnel generate reports about data that may prove relevant to a case matter. This software would also interface with collection tools for desktop or laptop collection and process data at great scale in large data center environments. The ideal discovery system would also perform a number of functions that would allow collection processing and analysis of data regardless of file format. The system would support a system of “describing” data without moving it into a separate repository; reducing the required storage space to use the system and making collection efforts more targeted and specific; let alone more cost effective (take just what you need for legal hold for example). This “system” would be coupled with additional standard processes and best practices to form the “standard” Ediscovery process.

Such a system would also provide specific access to certain data items but not others based on user credentials and group membership of users (multi-tenancy; or the ability of multiple groups to use the system but only see specific documents depending on their role in the organization or on the review team). Please see Figure One and Figure Two (below) for an illustration of these concepts and how the system is deployed within an organization.

At the present time, it is not believed that any one platform on the market has all of the capabilities mentioned herein and certainly does not account for capabilities not yet developed. Counsel should always keep abreast of technological advances and incorporate the same into any standard process. Depending upon the case and the data set, you may want to consider one or more platforms that best fit your needs. The choice of platform may be driven by which of the following options, beyond standard keyword-Boolean options, are available and/or needed for your data set:

1. Platform capability to allow unprecedented scale of indexing, search and analytics
    a. OCR conversion to text capabilities to ensure that content is captured even in image files
    b. Exception processing of certain file types that may need special processing like forensic video or audio analysis
    c. Processing with “flexible attribute selection”
        i. Indexing with Regular Expression matching turned “on” or “off”
        ii. Numerical content turned “on” or “off”
2. Multiple representations of data corpora
    a. File system- level Meta data only
    b. Application-level Meta data
    c. Full content
    d. Analytic attribute structures for semantic analysis
    e. Analytic Meta data structures for user-supplied attributes “tagging”
    f. Analytic Meta data structures for machine generated attributes
        i. Cluster associations for similar documents

ii. Near-duplicate associations for similar documents
iii. Group views for search associations

3. Analysis Capabilities
   a. To help identify keyword criteria – figure out which words are contained within the data universe and subsequently determine which are most relevant
   b. To identify relationships in the content that is in need of scrutiny or discovery (clustering)
   c. To organize documents and relate keyword searches to content that is in an analytic folder
   d. To remove duplicate content from responsive document sets
   e. To identify versions of content within sets of documents (versions of contracts or emails)
   f. To identify language characteristics of documents (language identification)
   g. To identify email conversations and conversation "groups"
   h. Linguistic analysis (categorization in terms of meaning)
   i. Sampling to pull data from known locations to use for additional searching
   j. Supervised classification or categorization (using known relevant documents to form search queries to find other potentially relevant documents
   k. Lexical analysis (entity extraction or analysis)
4. Validation Capabilities (Whether in the platform or extraneous)
   a. To validate the search (pulling random sample of all documents to validate search methodology
   b. To validate the review for:
      i. Privilege
      ii. Confidential Information (i.e. other products, social security numbers)
      iii. Tagged/relevant topics (pulling random sample of reviewed data to validate the review process)

Definitions of Key Terms

Key terms relevant to understanding an ideal Ediscovery system are:

Representation of Data

In the ideal system, it is important to represent documents so that they can be identified, retrieved, analyzed and produced for attorney review. Documents can be represented within the system by some sort of index or by certain kinds of data structures (covered in detail in a later section of this document). Different types of analysis require different types of indices or data structures. It is ideal to build the appropriate data structures to support the kind of data analysis that is required at a certain stage of the ediscovery process.

In an ideal system document representations can be constructed to include certain kinds of information but not other types. This is valuable as it keeps the space required for an index as small as possible and maximizes the speed of indexing or other data representation.

Meta data Categories and Use Cases

There are three main types of Meta data that are important in electronic discovery. The first two are attributes of file systems and applications and help identify who created, copied or modified documents. This capability helps to identify custody or ownership criteria for documents important to a case. The third type of Meta data is supplied by either human reviewers or analytic software processes.

File System or Repository Meta data

For example, the file system where documents are found has Meta data about who copied a file to the file system or when a file was created on a specific file repository. This category would include SharePoint Meta data, NTFS (Windows) file system Meta data and any kind of Meta data that is relevant to the repository storing a data item (when it was placed into the repository, how large it is, what Access Control Lists (ACLs) apply to control the viewing of the item, etc.). If a litigation support person was looking for files that were created on a file system during a specific time period, they would be interested in file-level Meta data. An ideal discovery solution always indexes the import path of any document it represents along with as many file system attribute fields as possible.

Application-level Meta data

The application (MS Word for example) that creates a document stores certain Meta data fields inside any documents it creates. This presents an additional type of Meta data that can be indexed and analyzed to identify documents with certain characteristics. Application Meta data contains fields like who the author of a document may be, when they created the file with the application (MS Word in this instance) or when the file was modified (inside the application). The ideal discovery solution would capture as many of these document-specific Meta data fields as possible to determine everything from authorship of the document to when it was last printed (depending on what application created the document).

User-supplied or "Analytic" Meta data

The last type of Meta data that the system can store for a user is "Analytic" Meta data. This is user or machine supplied Meta data. Even though final document tagging is done by an attorney within the final review stage of a legal discovery operation, other support personnel will mark or tag documents to indicate their status. Legal support personnel may need to mark or "tag" documents with labels identifying certain documents as "important" for some specific reason (the documents may qualify for "expert" review by a professional in a certain field of expertise for example). They may want to tag them so that a supervisor can review their work and decide that they meet certain criteria that qualify them to "move along" in the discovery process.

In addition to human review, a software analytic process can be run against a document collection and identify documents that are duplicate copies of one another in a large collection. An automatic process could generate tags (within the Analytic Meta data) indicating that certain documents are duplicates of a "master" document. If the master document was described as document "DOC000002345" then a tag such as "DUP_DOC1000002345" could describe all the documents that are duplicates of the master. These documents could then be identified quickly as redundant and they would not be passed along to attorneys for review. The system could retain the original copy of a duplicate document and mark or remove the others so that attorneys would not have to read duplicates unnecessarily. The ideal discovery solution can run near-duplicate analysis and determine that certain documents meet a threshold of “similarity” to other documents, qualifying them as “versions” of an original document. Tags can then be automatically applied to the documents exhibiting these relationships so that they are identified for in-house counsel who may want to pass them along as data that outside counsel should review.

Analytic Meta data is the repository where an ideal platform can conveniently place both human and machine-assisted codes or tags that will streamline or aid review of documents in a later part of the process. Given that human review is very expensive machine-assisted “culling” of information can reduce costs dramatically. Many experts in the industry term this process as part of “assisted coding” or “predictive coding” of documents.

Analytic Processes

For purposes of this paper, “analytic processes” will refer to the following main functions within the ideal discovery solution:

1. Unsupervised Classification – some refer to this as “clustering” where documents are organized together into lists or folders with members exhibiting some level of semantic similarity to one another. The term unsupervised refers to the technique’s ability to perform this semantic matching with no human supervision.
2. Supervised Classification – this refers to a capability where the product can take example content from a user and organize documents into lists using these examples as starting points or “seed” documents. The “best matches” are taken from among the candidate population of documents that are to be classified. The user can assign meaning to the seed clusters as they see fit; assign labels, etc. In the ideal solution a user can pick a number of documents as seeds, and specify an ordinal indicator of similarity that is a number between 0-1 that indicates a “threshold” of similarity that must be met for the candidate document to be placed on a seed list. Another form of the supervised classification is “search by document” where a user can select a single document as a “seed” and have it attract the most likely matches from the candidate list.
3. Near-duplicate analysis – this is very similar to supervised classification except that the system can take one “pivot” (example) document and compute all others within a relative “similarity distance” of it. Instead of organizing the document into a list of other semantically similar documents; candidates are marked as “near-duplicate” neighbors of a pivot should they fall within a range of similarity specified by a user. The documents are marked with “near-duplicate association” markers in the analytic Meta data repository as indicated above.
4. Email conversation analysis – this is where the ideal system identifies the email and instant messaging conversations that occur between parties. The parties and who sees a message is discernible through this type of analysis.
5. Different types of searching – simple keyword search, Boolean search, fuzzy search, proximate search are other types of search that are sometimes referred to as analytics within a product. An emerging technology that is more and more important to legal discovery is conceptual

searching, where concepts are computed among the members of documents and presented with the keyword results. Often conceptual searching is referred to in the context of conceptual mining which means a process that identifies concepts in documents that transcend keywords. Conceptual mining is often used to identify "latent" or immediately "unseen" words that are significant among a population of documents. These can often help a human reviewer identify what keywords should be included in a case and also to identify documents that the initial keyword searches did not include.

Virtual Index

For legal discovery purposes, a system needs to support building and searching the aforementioned three types of Meta data and must include support for analyzing and searching full document content as well. For analytics of certain kinds documents must be represented by special data structures that allow analysis (duplicate analysis, near-duplicate analysis, similarity comparisons to example content, etc.) to be undertaken. The system has to account for these at great scale.

This entire set of capabilities should appear (to a user of the system) to be possible across one "index". In the ideal system, these capabilities are encapsulated in one entity that will be referred to as: "the virtual index". It is referred to in this way because it supports various operations on multiple data representations and encapsulates these operations transparently to the user. The user should not know or care about the different repository or representations of the documents within the ideal system. The user should simply issue searches or ask for "similar documents" and get the results. The virtual index will abstract all of these details for a user.

Multi-site Support

The ideal system should support use cases "behind the corporate firewall" for analyzing and collecting data within local enterprise or client environments, and also support large data center deployments. The indices built within the enterprise environment should be "portable" so that they can be built in the enterprise environment and then be transported to the larger data center environment where all aspects of the case can be evaluated in one "virtual place". The idea of a virtual index supports this vision, as it allows local data sources to be analyzed at various remote locations and then any relevant files moved to a legal hold location at a central data center. The indices can be added to the central location along with any data that is copied for legal hold purposes.

In all cases it is ideal to have a platform that "connects" to data sources, reads in a copy of the documents stored within them, but leaves the original in place at its source location. Instead of moving the original document into the ideal system and duplicating the document and the storage required to maintain or analyze it, the documents can be represented by an index or some data structure that is generally more compact. The original documents do not have to be resident within the ideal system to be analyzed and referenced. Please see Figure Two (below) for an illustration of the ideal system in relation to data sources it represents.

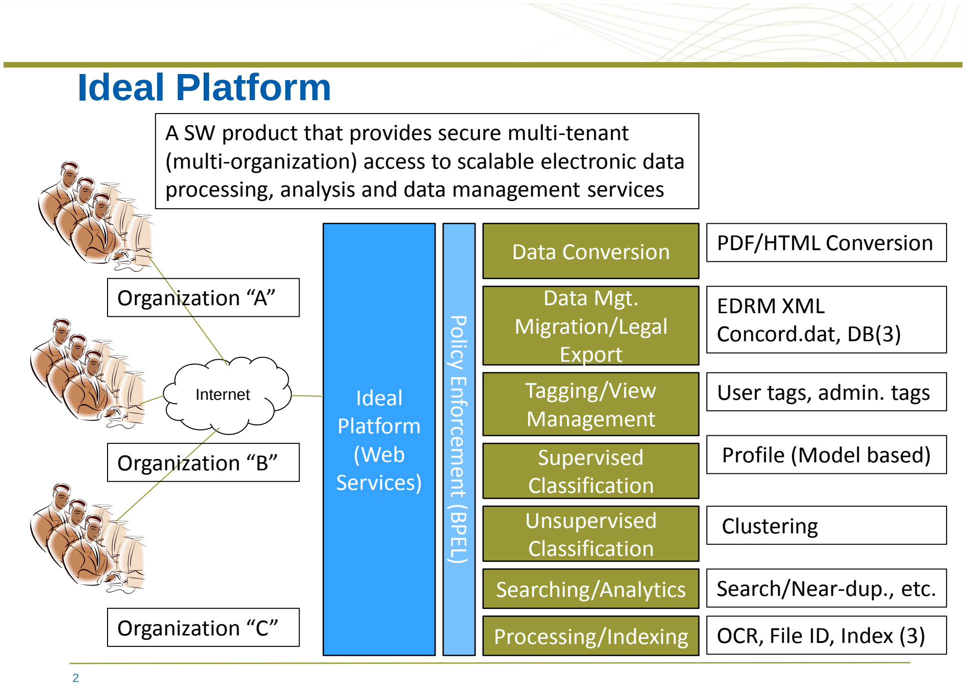
It is important that documents do not have to be loaded and analyzed in "batches" and that the ideal system has the scale to represent vast numbers of documents within one single system. A system that supports a set of analytic operations and scalable search is also an important feature of such a discovery platform. Having the ability to analyze new documents by comparing them analytically with examples already represented within the ideal discovery system is extremely important to solid ediscovery practices.

Key Architectural Attributes of an All-Inclusive Platform

An all-inclusive platform approach presents all of the capabilities shown above to the IT or legal review professional. The user can index data from locations within their data center or from sources as diverse as their SharePoint server "farm" their Exchange email server, NT file servers or large-scale NAS devices. The user can pick various levels of data representation based on the level of insight required for the task and the computational and storage burden acceptable to the reviewers. The user can then search for data that is relevant, select those results to "pass on" to other analytic processes (such as de-duplication and near-duplicate identification or email analysis) and then tag or otherwise mark the results.

All of these capabilities should be available from a single console without the need for moving the data from one tool to another. Once the data is in the platform it can be identified, analyzed and marked according to the needs of the case. The important thing is that it can be managed with these processes at unprecedented scale. Please see Figure One (below) for an illustration of the ideal platform. The reader can quickly recognize that this is a product with a full suite of analytic and legal production capabilities. It is far beyond a single function product like a search engine. Please see Figure Two below for an illustration of how this platform could operate in the IT infrastructure among various repositories of data.

Figure One: Ideal Discovery Platform

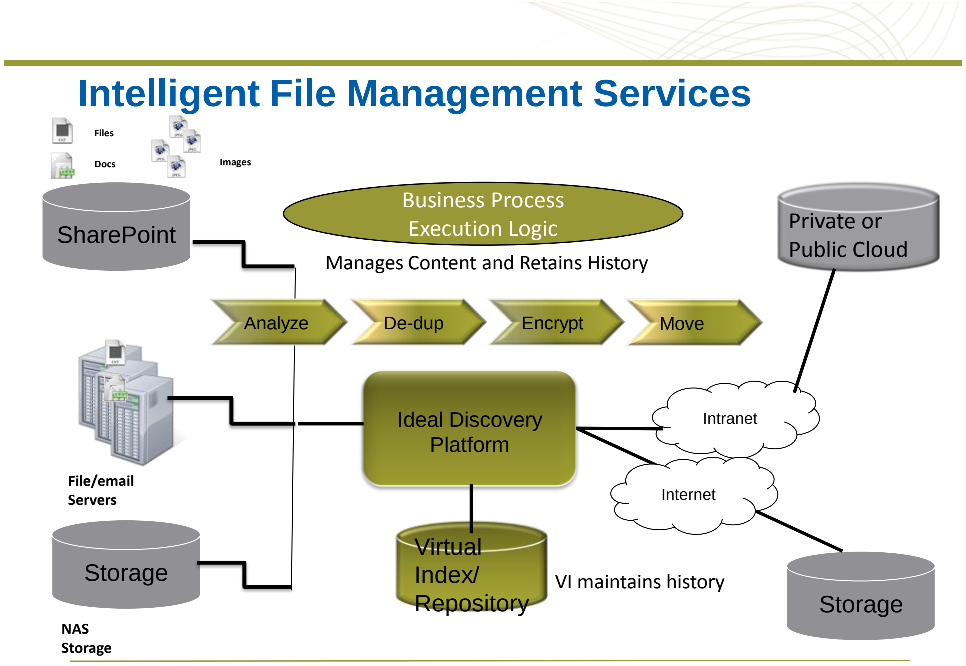


Ideal Platform

A SW product that provides secure multi-tenant (multi-organization) access to scalable electronic data processing, analysis and data management services

The ideal discovery platform will perform all of the functions in the illustration above. The power of having all these capabilities in one platform is undeniable. Being able to process content (OCR, REGEX), index it, “cull” it down to a smaller size and then analyze it (remove duplicate material, perform NIST analysis, identify near-duplicate content, calculate email conversation “threads”) all in one platform without having to move the content from one system to another eliminates labor and potential human error. Promoting efficiency in electronic discovery is a key component to success in legal review matters.

Figure Two: Intelligent File Analysis and Management



Scale of Indexing and Representation

An ideal discovery solution must have unprecedented scale. Scale is provided through superior use of physical computing resources but also through the segmenting of the various data resources into the virtual index components described previously.

Scale of Indexing, Search and Analytics [List of All Unique Terms in a Collection]

With the correct architecture hundreds of millions to billions of documents can be indexed and managed in a fraction of the time required for other solutions, and with a fraction of the hardware they require. One vendor, utilizing a unique grid-based (multi-server) architecture has demonstrated the indexing and preparation of a given 17.3 TB data set in less than a twenty-four hour period. This is possible due to two factors:

1. The platform’s unique “Grid” architecture (see Figure Three)
2. The platform’s unique “Virtual Indexing” architecture and technology (see Figure Five)

This platform can be deployed as a single server solution or in the large data center configurations shown in figure three below. The ability to expand as the customer needs to index and analyze more data in a given amount of time is made possible by the architecture. Certain software components of the architecture schedule activities on the analytic engine components shown in the diagram. These analytic engines "perform intensive work" (indexing, searching) and the controlling software requests them to perform the work to produce results for users. The controlling software is the "intelligence or brains" of the system and the analytic engines are the "brawn" of the system. As the user needs more processing power, more analytic engines can be employed within the "grid" to provide more processing and analytic power (the user is again referred to Figure Three)

Scale of Representation

This architecture also supports the representation of content in multiple ways so that the search, classification and other analytic operations available from the analytic engines can "work on" the data that has been processed. This means that the index is really a set of "managed components" which include:

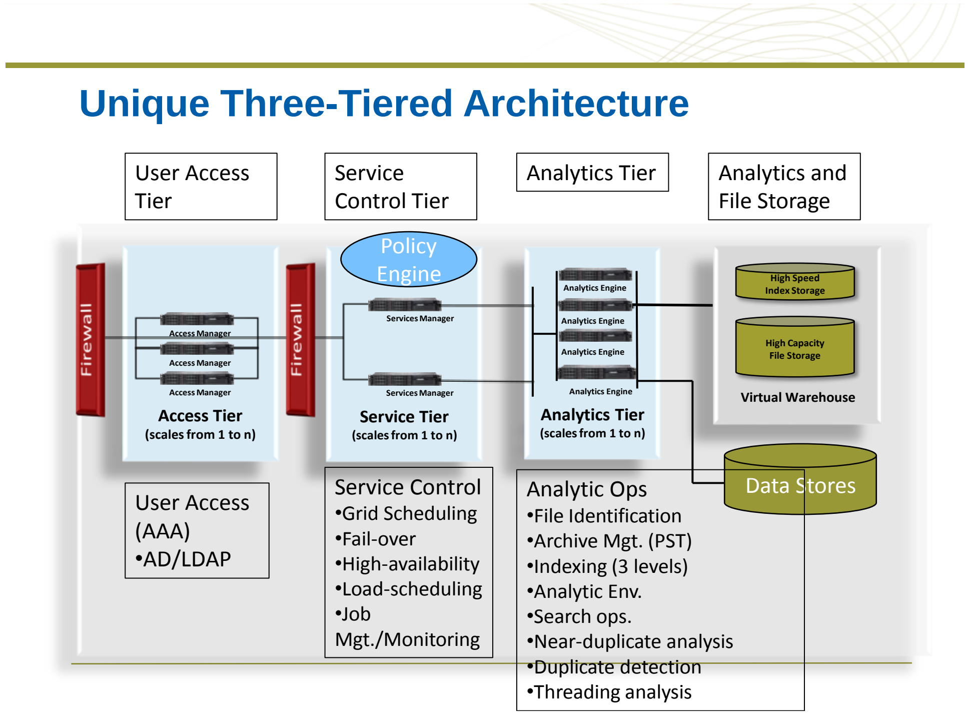
1. Meta data indices
2. Content indices
3. Analytic data structures
4. Analytic Meta data (tags, cluster groups, other associations)

All of these things are what is meant by "scale of representation"; the platform can represent content in multiple ways so that the appropriate level of search or analytics can be undertaken on the documents that are within a collection or corpus.

Speed and Scale of Indexing

A second aspect of scale is the speed with which data can be processed and made available for assessment. With a superior architecture an index can be presented for searching within hours. Other solutions require days if not months to build the content for a case into a searchable representation. The ability to build an index and get results in one or two days and have it done reliably allows case matters to be investigated rapidly and with fewer errors. The sooner a reviewer can determine what is relevant within the scope of discovery the sooner lawyers can begin making intelligent decisions about the case. This leads to better outcomes because the reviewers are not as rushed and because they have better analysis options than they would have with traditional methods. With the data prepared faster, organizations have time to perform search operations and then perform more complex analysis of data that will aid the reviewer later in the case.

Figure Three: Scalable Grid Architecture



Unique Three-Tiered Architecture

User Access Tier

Service Control Tier

Analytics Tier

Analytics and File Storage

User Access (AAA)
- AD/LDAP

Service Control
- Grid Scheduling
- Fail-over
- High-availability
- Load-scheduling
- Job Mgt./Monitoring

Analytic Ops
- File Identification
- Archive Mgt. (PST)
- Indexing (3 levels)
- Analytic Env.
- Search ops.
- Near-duplicate analysis
- Duplicate detection
- Threading analysis

As one can see from the illustration above, the data processing and search workload can be distributed over various machines in the "grid". The customer simply has to install and provision more "engines" to exist in the grid and the intelligent management layer of software will use these resources for processing, indexing and search operations. This allows the product to scale to handle unprecedented levels of documents and to process them in unprecedented timeframes. In Figure Eight (below) the search operation is illustrated as being distributed over the available grid processing power.

Virtual Indexing: a Key to Large Corpus Management

In addition to a distributed "grid-like" architecture, another key to managing large data sets is using the proper constructs to represent the data. As mentioned, this platform builds different representations of the data based on the needs of the analysis tasks that will be required for specific discovery activities. It ties them together in a logical set of components that is referred to as a "Virtual Index". This is necessary because the Meta data from files, the user-supplied Meta data from other reviewers, and analytically generated Meta data all must be searched as a single logical entity to make decisions about a given case.

A virtual index stores the various pieces of the logical index separately so that the Meta data can be built and searched separately for efficiency reasons, but also for scale purposes. A virtual index can be grown to an unprecedented size because it is "built up" from smaller more efficient components. Further, it can be transported from one location to another, and then "added in" to a matter as the user sees fit. Earlier in this document the example of remote office local collection with the data being transported with the appropriate indices to a data center. This is possible because of the virtual index. Such an index can also grow arbitrarily large. The virtual index component of software can "open" the parts of a virtual index that matter to a case at a certain point in time. This makes searching more efficient and also it allows the virtual index to grow or shrink as necessary. Also, the “pieces” of a virtual index can be repaired if they become corrupt for some reason. The ideal system retains “manifests” of documents that comprise a given portion of the virtual index and from these the component indices can be rebuilt if necessary.

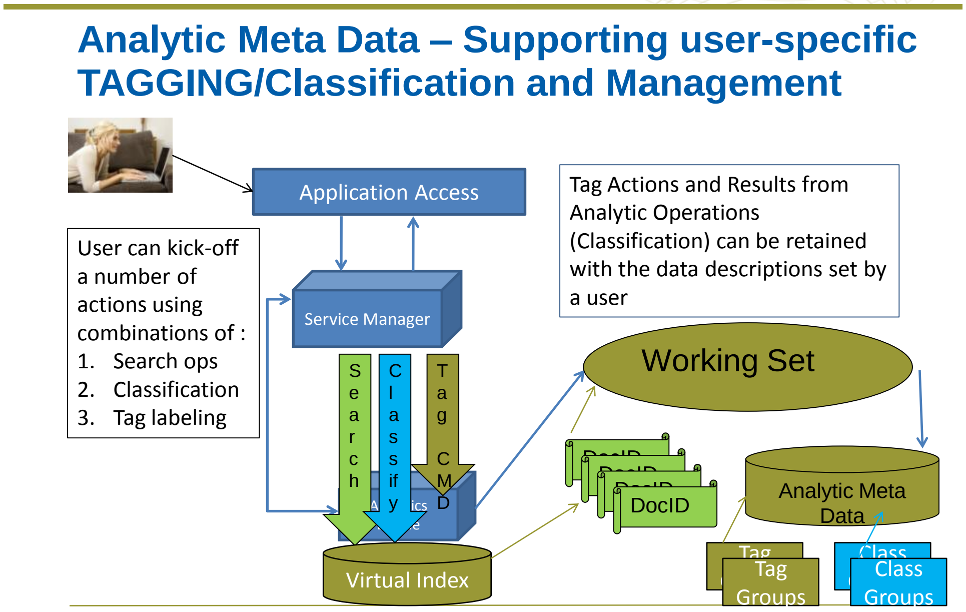
The user may want to just look at file system Meta data and characteristics of content stored within an enterprise. For that a straight forward file system Meta data index (basically POSIX-level Meta data) will satisfy the need. This type of index only requires about 4% of the original data size for storage. A full content index (on average) consumes between 25-30% of the original data size. The full-content index will require more storage than the Meta data variety of index, and it will take longer to build.

If the user needs to understand the application (MS Word, PDF) Meta data or that and the full content of documents for keyword search, they will be willing to wait for the extra processing (full content indexing) to complete and are likely willing to consume extra storage. If the user is not sure if all the available content meets the criteria that their search may require, they may want to use the POSIX Meta data indexing technique initially to identify what content should be fully indexed (before committing to extra time and storage resources).

One key aspect of the ideal system is that the Meta data index is separate and stands alone from the content index that supports it. The system presents one index for a given corpus of documents, but beneath this construct is at least a Meta data index. If a corpus is represented as a full content index, the corpus has a Meta data and a full content index component. The two indices are logically connected but physically separate. The virtual index software layer “binds” them together. Please see Figure Five for an illustration of a virtual index and its components. This virtual index approach makes the index more scalable; it allows it to be searched more rapidly and makes it resilient against potential corruption.

In addition to the full content inverted index construct, the corpus of documents can be further represented by analytic feature descriptors (where each “word” or “token” is represented as a feature of the document). These feature descriptors for single documents can be combined as “models” where complex relationships between the words or tokens are stored. These analytic descriptors are separate data structures that support document similarity operations, clustering and near-duplicate analysis. They do not depend upon the inverted index that is used for keyword searching; they are separate data structures and are used independently of the index.

Figure Four: Analytic Meta Data



Analytic Meta Data – Supporting user-specific TAGGING/Classification and Management

Application Access

User can kick-off a number of actions using combinations of :

1. Search ops
2. Classification
3. Tag labeling

Service Manager

Search

Classify

Tag CMD

Tag Actions and Results from Analytic Operations (Classification) can be retained with the data descriptions set by a user

Working Set

DocID

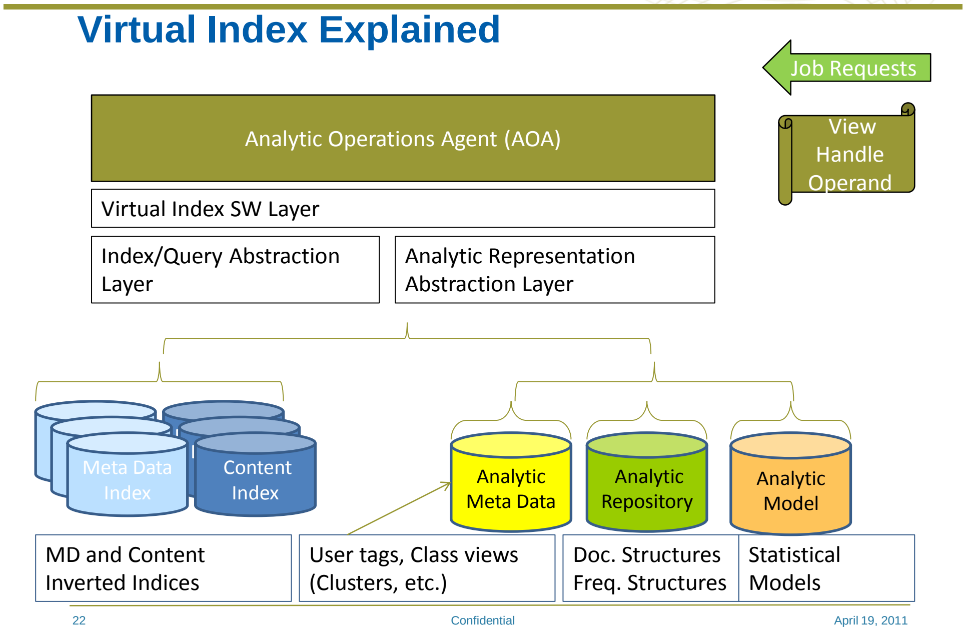
Analytic Meta Data

Tag Groups

Class Groups

Virtual Index

Figure Five: Virtual Index Illustration



Virtual Index Explained

Job Requests

View Handle Operand

Analytic Operations Agent (AOA)

Virtual Index SW Layer

Index/Query Abstraction Layer

Analytic Representation Abstraction Layer

Meta Data Index

Content Index

Analytic Meta Data

Analytic Repository

Analytic Model

MD and Content Inverted Indices

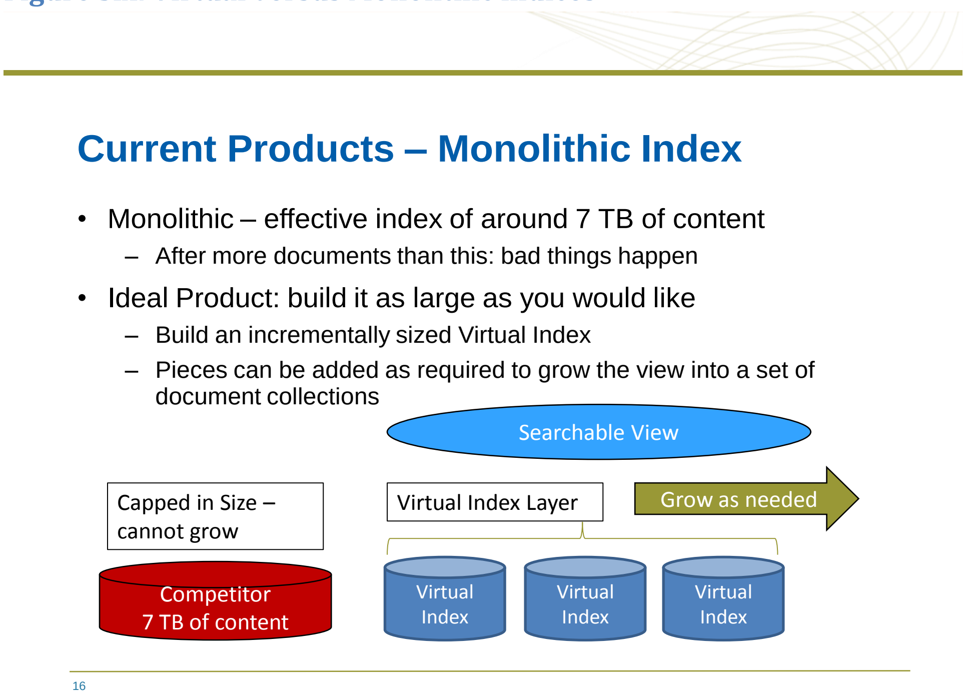
User tags, Class views (Clusters, etc.)

Doc. Structures Freq. Structures

Statistical Models

22 Confidential April 19, 2011

Figure Six: Virtual versus Monolithic indices



Current Products – Monolithic Index

- Monolithic – effective index of around 7 TB of content
  - After more documents than this: bad things happen
- Ideal Product: build it as large as you would like
  - Build an incrementally sized Virtual Index
  - Pieces can be added as required to grow the view into a set of document collections

Searchable View

Capped in Size – cannot grow

Competitor
7 TB of content

Virtual Index Layer

Grow as needed

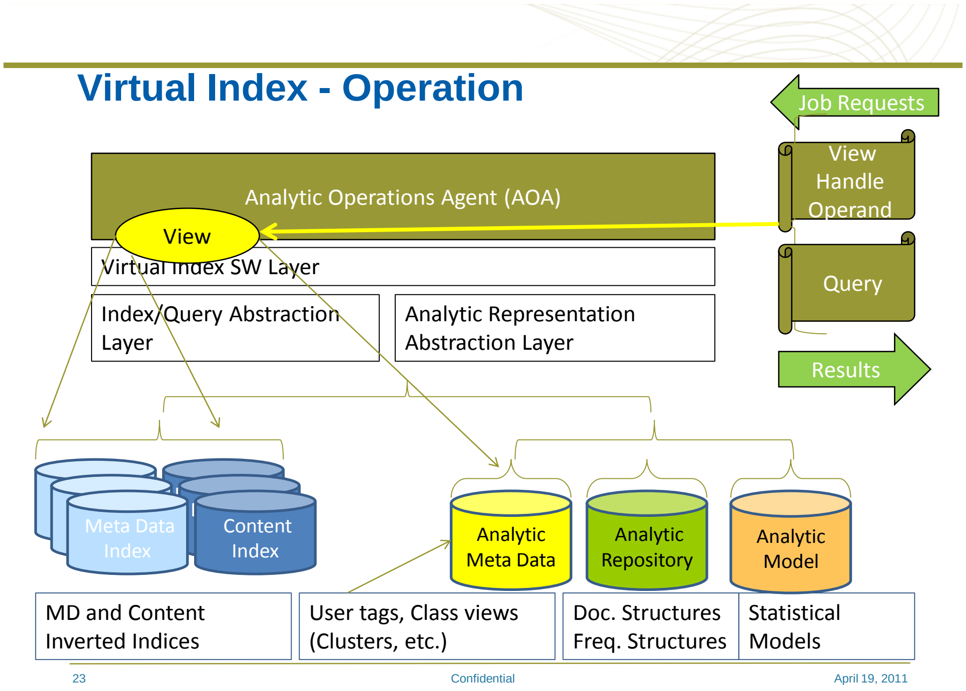
Virtual Index

Virtual Index

Virtual Index

16

Figure Seven: Virtual Indexing in Action



Virtual Index - Operation

Job Requests

View Handle Operand

Query

Results

Analytic Operations Agent (AOA)

View

Virtual Index SW Layer

Index/Query Abstraction Layer

Analytic Representation Abstraction Layer

Meta Data Index

Content Index

Analytic Meta Data

Analytic Repository

Analytic Model

MD and Content Inverted Indices

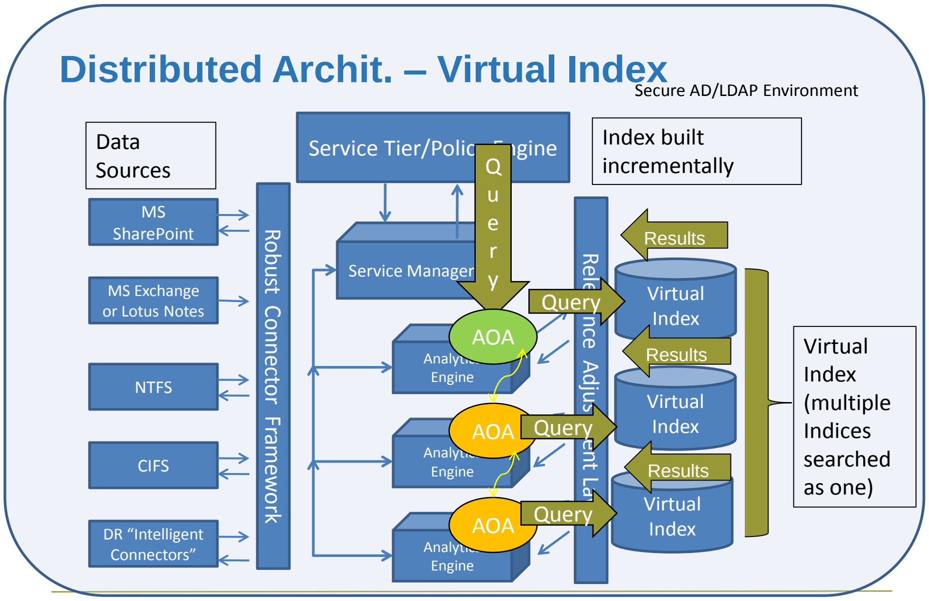
User tags, Class views (Clusters, etc.)

Doc. Structures Freq. Structures

Statistical Models

23 Confidential April 19, 2011

Figure Eight: Virtual Indexing plus Grid Architecture


Distributed Archit. – Virtual Index
Secure AD/LDAP Environment
Data Sources
MS SharePoint
MS Exchange or Lotus Notes
NTFS
CIFS
DR "Intelligent Connectors"
Robust Connector Framework
Service Tier/Polic Engine
Service Manager
Query
AOA
Analytic Engine
Query
Index built incrementally
Results
Virtual Index
Results
Virtual Index
Results
Virtual Index
Relevance Adjustment La
Virtual Index (multiple Indices searched as one)

Summary of Architectural Concepts

Now that we understand how the unique architecture of the ideal system solves several issues around Discovery, we can talk about some important types of processes which we will refer to as "analytics" and their importance to the overall process. The prior sections of this document explained how:

1. The grid architecture allows very large collections to be represented as indices in record time. Before this architecture became available, the Ediscovery process could not analyze the extremely large collections of documents that have become common in legal matters. These collections were either not analyzed, or they were analyzed in pieces, leading to human error and inconsistency of results.
2. The virtual index constructs let the user select various levels of index representation
    a. File Meta data only
    b. Application Meta data
    c. Full Content
    d. Analytic Descriptions (document feature attributes)
    e. Models and Profiles (example content in feature-attribute form)
3. The virtual index also lets the document collections that are represented grow to unprecedented size and still remain usable and efficient
    a. The virtual index lets the user add to the collection at any time as the virtual index is really a multi-index construct within the product
4. Monolithic Indices can be problematic
    a. Monolithic indices can grow in size and be very inefficient to search and manage

    b. Monolithic indices can become corrupt at a certain size and become unusable
    c. Monolithic indices can take long periods of time to construct in the first place

5. Virtual Indices Supply Several Key Advantages
    a. In a virtual index Meta data only searches work on smaller absolute indices and complete more rapidly
    b. In a virtual index Meta data and full content searches actually execute in parallel increasing efficiency and scale while providing results more rapidly than from classical monolithic indices
    c. Virtual indices can support similarity operations like “search by document” that expose relevant documents that are meaningful to a human reviewer
    d. Virtual Indices can be repaired efficiently without requiring entire document collections to be re-processed and re-represented.

Analytics

Analytic processes in EDiscovery present distinct advantages to human reviewers. In this section, analytic processes are described that can aid the legal review process. Discussion is presented about why they can be of help during that process. In addition, aspects of these approaches are presented and compared and advantages and disadvantages of each are explored. This is to give the reader a sense of how competing products in the discovery space compare. This is intended to help the reader value each and determine when one technique needs to be applied with others to be effective in a legal review process.

Major Categories of Analytic Processing

It is easy to become confused by all of the techniques that are available to aid human reviewers of electronic documents. These techniques fall into three main categories:

1. Unsupervised classification or categorization (often referred to as "clustering")
2. Supervised classification or categorization
3. Specific types of analysis
    a. Near-duplicate analysis
    b. Duplicate identification or analysis
    c. Conversational analysis (who spoke to whom)

Unsupervised Classification

This is often referred to as "clustering" because one wants to form document groups that "belong together" because they "mean the same things". The main idea behind this kind of analysis is that the human reviewer does not have to know anything about the data in advance. The reviewer can just "push the button" and find out what belongs where in a dataset and see folders or ordered lists of documents that are related somehow.

See Figure Twenty Two (below) for a screenshot of a product that identifies documents according to their similarity to one another. This particular system uses a series of algorithms to perform its work; but the end result is folders of documents that are related to one another. Close inspection of the diagram will show that the foreign language documents end up in the same containers or folders. The predominantly foreign language documents get grouped together in a folder that is labeled with foreign language "concepts" to make the review process more efficient. Other advantages of this technique will be explained in later sections of this document. This is an example of a multi-level algorithm that accounts for language differences. Some unsupervised classification algorithms do not account for the language differences of documents and they can produce results that appear "confusing" in some circumstances. This phenomenon will be discussed later in the document.

Most of these unsupervised techniques culminate in some kind of "conceptual" clustering or conceptual mining and analysis of the data they represent. As each specific technique is described in later sections of this document the reader will be informed about how the technique relates to conceptual analysis of documents being analyzed.

Supervised Classification

Supervised classification means that a user “supervises” the process by providing at least some examples of documents that are similar to what they want the algorithm to find for them in a larger population of documents. These documents are usually put into what is called a “model” and they “attract” other documents that belong “closely” to them. Please see Figure Twenty Four for an illustration of supervised classification.

Examples of supervised approaches:

1. Seed-model “nearest neighbor example” type clustering.
2. Support Vector Machines (see reference [6]) – the user must supply “known positive” and “known negative” examples of documents that the system can use to “compute the differences” between for purposes of classifying new documents.
3. Bayesian Classifiers (see section below and reference [3]) – the user must supply “good” and “bad” examples so that the algorithm can compute a “prior” distribution that allows it to mark documents one way or the other.
4. Statistical Concept Identifiers that arrange documents based on the characteristics of words and topics in a set of “training data” (documents that have been selected from a larger population of documents but that have not been reviewed by a user)
5. Linguistic Part of Speech (POS) models where certain patterns of a specific language are noted within a linear classification model and new documents are “matched” against it based on their linguistic characteristics.

Specialized Analysis

There are specialized analytic techniques such as:

1. Near-duplicate detection (finding things that should be consider versions of other documents)
2. Email conversational analysis (threads of conversations between specific parties)

Mathematical Framework for Data Analysis

In the preceding discussion of the ideal architecture the concept of representing data as an index for searching or as a mathematical model for analysis was presented. This section contains a description of how data is represented mathematically. There are many ways to represent data for mathematical analysis; the technique being described below is one way. This is not intended to be an exhaustive review of all available text representation techniques; it is offered to help the reader visualize methods that are not the same as an inverted index that can be used as a basis for document analysis.

Basic Overview of Document Analysis Techniques

The basics of how documents are compared to one another relies on representing them as “units of information” with associated “information unit counts”. This is intended to give the reader context to understand some of the terminology that follows. The goal of this is to support a mathematical process that can analyze document contents: the “vector space model” [7]. The vector space model was developed by a team at Cornell University in the 1960’s [8] and implemented as a system for information retrieval (an early search engine). The “pros” and “cons” of the vector space model are discussed in the references, but since it is a good way to understand how to think about documents in an abstract and mathematical way it is explained initially. When we refer to this in general it will refer to the document-term representation model where documents can be thought of as vectors.

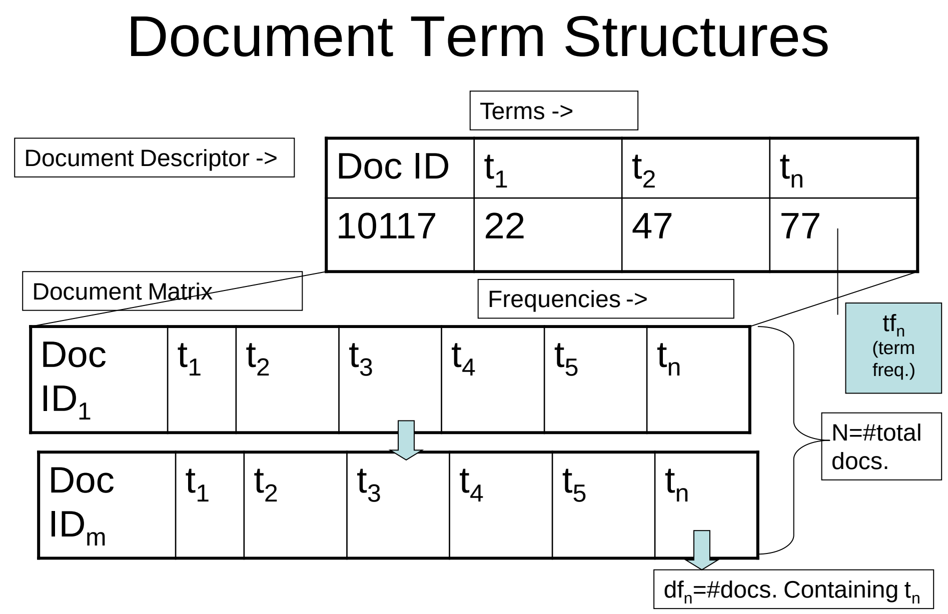
The term Vector Space Model would imply that in all cases we mean that the vectors are compared with cosine angular measurements after their “term frequency-inverse document frequency” attributes are computed. In the context below I discuss how that is possible but I don’t explain “tf-idf” in detail. The reader can consult [7] and [8] for information on computing similarity with tf-idf techniques. Furthermore, I am offering the model as an example of how documents can be represented for comparison. Other techniques than tf-idf used for cosine similarity comparisons use the vector concept so I want to make sure the reader understands the context in which this discussion is offered.

The Vector Space Model (VSM) is often referred to not just as a data representation technique but as a method of analysis. Some of the techniques mentioned in the sections that follow utilize this vector type model in some way (for representation; but their mathematical approaches are different). Not all of the techniques discussed use the vector space model, but it is presented to give the reader a grasp on how a document can be analyzed mathematically. Some form of vector is used in many cases to describe the document content. Some of the techniques just need some data structure that represents the words in a document and how often they occur. This is often constructed as a vector even if the vector space calculations are not used to analyze the data the vector represents.

Representing Text Documents for Mathematical Analysis – Vector Space Model

The “Vector Space Model” is a very well known structure within the field of information theory and analysis. It allows documents and their words or “tokens” to be represented along with their frequencies of occurrence. Documents are represented by a “document identifier” that the system uses to refer to it during analytic operations or so that it can be retrieved for a user. The overall combination of document identifier and the token frequency information is referred to as a “document descriptor” because it represents the information with the document and provides a “handle” to use to grab the document when necessary.

Figure Nine: Vector Document term Frequency Structures



Document Term Structures

Terms ->

Document Descriptor ->

Doc ID	t_1	t_2	t_n
10117	22	47	77

Document Matrix

Frequencies ->

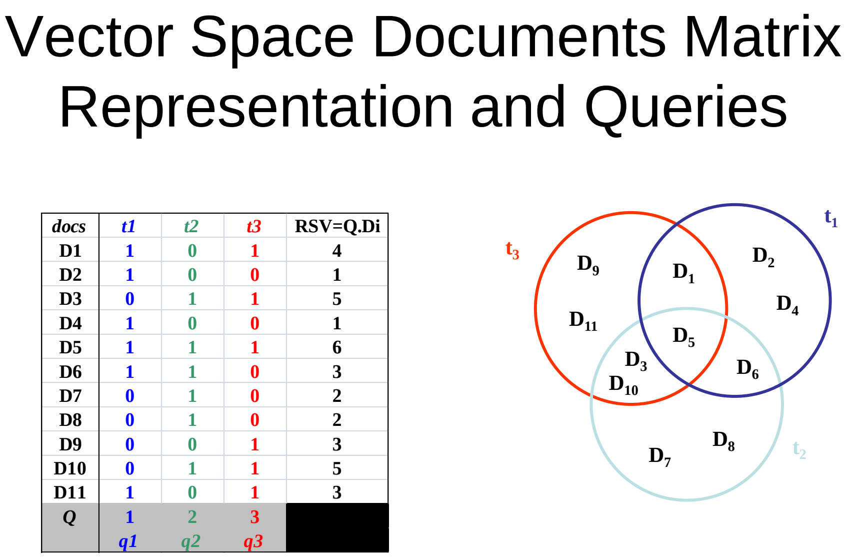
Doc ID$_1$	t_1	t_2	t_3	t_4	t_5	t_n
Doc ID$_m$	t_1	t_2	t_3	t_4	t_5	t_n

tf_n (term freq.)

N=#total docs.

df_n=#docs. Containing t_n

Figure Ten: Documents in a Matrix



Vector Space Documents Matrix Representation and Queries

docs	*t1*	*t2*	*t3*	RSV=Q.Di
D1	1	0	1	4
D2	1	0	0	1
D3	0	1	1	5
D4	1	0	0	1
D5	1	1	1	6
D6	1	1	0	3
D7	0	1	0	2
D8	0	1	0	2
D9	0	0	1	3
D10	0	1	1	5
D11	1	0	1	3
Q	1	2	3	
	q1	*q2*	*q3*	

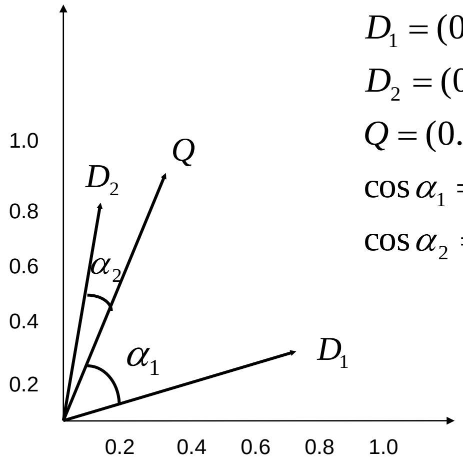
Notice that when documents are represented as document-term structures the documents are like "rows" in a matrix. The "columns" of the matrix are the terms, and the columns can be the frequencies of the given terms that are found to occur in the documents. The term positions can be fixed (per term) and labeled with some integer with the actual string of the word/token being kept in a separate dictionary or some other means can be used to "keep track" of what the terms mean. The representation of a document will be the document being an entire row of terms with the columns representing the frequency of occurrence of any given term within a specific row or document.

Comparing Documents to One Another or Queries

With the vector-space model of vector comparison, each document is treated as a “vector” in a dimensional space. The number of dimensions equals the number of terms in the largest document. If a document has a given term in a row of the matrix, the value in the matrix is equal to that document’s frequency for the given term. If the document does not have that given term, the column in the row of a given document is zero. To compare two documents, a “similarity calculation” is undertaken and a “score” is computed between the documents. The score represents the cosine of the angle between the two documents, or their “distance apart” in the “n-dimensional vector space”. This can be visualized in two-dimensions below. A query can be represented as a document so a query entered by a user can be compared to documents and the closest ones can be retrieved as results.

Figure Eleven: Cosine Similarity

Computing Similarity Scores



$$D_1 = (0.8, 0.3)$$

$$D_2 = (0.2, 0.7)$$

$$Q = (0.4, 0.8)$$

$$\cos \alpha_1 = 0.74$$

$$\cos \alpha_2 = 0.98$$

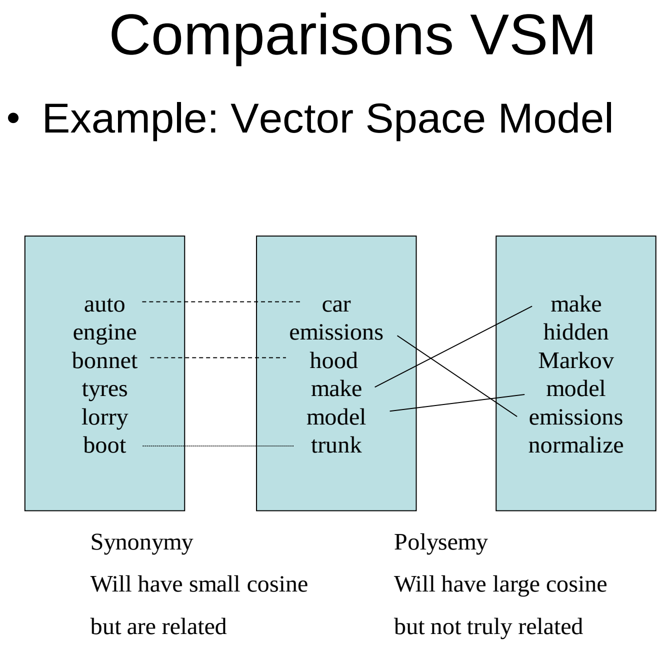
The basics of the vector space model are that the cosine angle can be computed between any two vectors in the document-term matrix [7]. This number is guaranteed to be between zero and one and it shows that a document is identical to another document (score equals one) or the document has a zero score (nothing in common with the reference document) or something in between. The closer that the score is to the number one, the more similar two documents are in the “vector space” or “semantic space” of the documents. This model gives the reviewer some idea of how similar two documents are. It is useful in this respect; but it has some limitations. The reader can review the references for more detail on the mathematics, but the basic idea is that documents are: 1) the same; 2) totally unrelated; 3) somewhere in between.

Problems with Vector Space Model (VSM)

The problems that arose using the vector space model included:

- synonymy: many ways to refer to the same object, e.g. car and automobile
- polysemy: most words have more than one distinct meaning, e.g. model, python, chip
- Vectors are still sparse and there is a lot of extra computation involved with analyzing them

Figure Twelve – Illustrative Behavior Vector Space Model



Comparisons VSM

- Example: Vector Space Model

auto
engine
bonnet
tyres
lorry
boot

car
emissions
hood
make
model
trunk

make
hidden
Markov
model
emissions
normalize

Synonymy
Will have small cosine
but are related

Polysemy
Will have large cosine
but not truly related

As can be seen above, the VSM puts things together that have the same (literal) words. It is efficient to compute and gives an intuitive basis for understanding what is literally similar. What it does not help with is in finding documents that “mean” the same things. In the example above, one document with “car” and another with “auto” would not be grouped together using this technique. This is because the technique cannot account for synonyms or polyesters (these are explained in [5] within the references); polyesters are words that have more than one meaning (“Java meaning coffee” and “Java meaning the island of Java”, or “Java the programming language”).

There are ways to improve the behavior of the vector-space model and it is still used and proves very useful for many operations in data analysis. For conceptual analysis of data however, there are other methods that can be used that don’t suffer from these drawbacks.

VSM: Historical Significance

The VSM as one of the first techniques to model documents in a mathematical context and present them in a fashion that is conducive to analytic review. It is not perfect, but it represents a lot of value to folks trying to find similar documents and it paved the way for researchers to use a common model for thinking about document analysis. The techniques that were subsequently put forward used this representation method but applied very different mathematical techniques to the matrix model of viewing document collections.

There are ways to improve the behavior of the vector-space model and it is still used and proves very useful for many operations in data analysis. For conceptual analysis of data however, there are other methods that can be used that don't suffer from the drawbacks of VSM. These techniques perform dimensionality reduction on the data sets at hand, and expose "latent relationships" in data that are hard to find otherwise. Before moving on to techniques, some discussion of reducing the dimensionality of data sets is presented.

Some More Basics

Before we discuss the various techniques at our disposal for legal discovery analytics; we have to discuss a general concept in language theory: "dimensionality". Dimensionality is the number of words or the number of things that we have to account for in a language model.

Analyzing Text Documents – "The Curse of Dimensionality"

The problem with text is that with most languages there are so many words to choose from. Documents vary in their vocabulary so much that it is hard to build one mathematical model that contains all the possibilities of what a document might "mean". Language researchers and computer scientists refer to this problem as "the curse of dimensionality" and many information analysis approaches seek to reduce the number of dimensions (words) that their models have to contain.

In the matrix above, if this represented a "real" collection of documents, the columns of the matrix would be much more numerous and for many of the documents the columns would not have an entry. This is what is meant by "sparse data" within a "document-term matrix". Many approaches are aimed at identifying what words in a document or set of documents represent "enough" of the total so that others can be ignored. Information theorists refer to this as removing "noisy data" from the document model. This concept revolves around choosing enough of the attributes (words) within the documents to yield a meaningful representation of their content. Other techniques are used to actually remove words from the documents before they are analyzed.

Stop Word Removal

Certain words (such as "a", "and", "of") that are not deemed "descriptive" in the English language can be removed from a document to eliminate the number of dimensions that an algorithm needs to consider. This may be helpful in some contexts and with some algorithms; it does reduce the numbers of dimensions that need to be considered by an analysis algorithm. When these are removed it can be hard to determine specific phrases that might carry meaning to a legal reviewer however. A search engine that can find phrases may not consider the difference between two documents with similar phrases:

Document #1:“We agree on the specific language outlined below…”

And:

Document #2: “We agree to pursue a process where we agree on a specific language to describe….”

In these two documents many search engines would produce both documents; each with clearly different meanings in response to a phrase search of: “agree on the specific language”. This may not be a problem to a reviewer because both documents are likely to be returned; but the reviewer will have to read both documents and discard the one that is not specific enough for the case at hand. In this instance, stop word removal would yield results that are less specific than a reviewer might want. The searches with this type of index may produce more documents, but the cost will be that they may not be as specific to the topic at hand.

Stemming of Language

With most languages, there are ways to find the “stems” or “root meanings” of many words through an algorithm pioneered by Martin Porter [9] that has been named: “Porter Stemming”. This technique has been used widely and is often referred to simply as: “stemming”. Any serious language theorist recognizes the term “Porter Stemming”. The algorithms were initially released for English language analysis but have been extended for many other languages. See the reference (again [9]) for more discussion of the techniques and the languages supported.

The idea with porter stemming is to reduce words with suffix morphologies to their “root” meaning. The root of “choosing” is “choose” and would show up in some stemmers as: “choos”. The roots of many common words can change after stemming to common “roots”:

```
alter
alteration
altered

become:

alter
alter
alter
```

This reduces the number of tokens that a document has to account for and the argument for using this technique is that the meaning of the words is “about the same” so the corresponding behavior this induces in the mathematics will not be deleterious to any given analysis technique.

The theory behind using stemming for search is that more documents of “about the same meaning” will be produced for a given query. In a legal review context documents that are not specific to a query could be returned with stemmed collections. This is similar to the situation that could exist when stop-words are removed from a collection. For analytic approaches, the same thing can occur. The algorithms that group documents together could produce results that are less specific than might be desired by a human reviewer.

For analytic approaches, the designer of an algorithm must consider this trade-off between precision and recall. The benefits of having fewer things to keep track of in the model may outweigh any lack of clarity around usage that the token suffixes may have conveyed. In a legal discovery "clustering" context this may not be true (as we will discuss), but stemming is an important attribute of a collection of documents that should be considered when preparing documents for legal review purposes. It can help immensely and it can make other things less specific (which it was designed to do) than one might want for a legal discovery application. The designer of the analysis system should consider how the documents need to be prepared for the optimal performance inside the algorithms the system will implement.

Higher-Order Mathematical Techniques

To this point, we have seen how a legal discovery platform must include many different pieces of functionality and respect both Meta data and full content search at great scale. We have also seen how analytics can be important to legal discovery professionals. We have set the framework for how to represent documents in a way that allows mathematical operations to be defined on abstract representations of their content.

We reviewed the vector space model and how document matrices have many "dimensions" that impact analytical performance for legal review purposes. We have discussed how to reduce dimensions by removing certain words from a collection or by reducing certain words to their "root forms" so that they can be considered more generally with fewer burdens being placed on the modeling technique. These techniques can reduce the specificity of results returned by the system. This may or may not be acceptable for a legal review application. For conceptual analysis of data, there are other methods that can be used that perform dimensionality reduction on the data sets at hand in a different manner.

One of the first solutions to this problem that was proposed was Latent Semantic Indexing (or Analysis) by a team or researchers at Bell Laboratories in 1988. Before these are explored, some of the commonly used techniques are listed and discussed. This is not intended to be an exhaustive review of every technique available for language analysis. It is not a critique of any technique or vendor implementation. This is a discussion of some common techniques that have been used within legal discovery products and presents a "pro" and "con" set of considerations for the reader.

Commonly Used NLP Techniques

Within legal discovery, there are some NLP techniques that have become commonly known within the industry. These have been championed by vendors who have had success in providing them as pieces of various edicovery products. These are:

1. Latent Semantic Analysis/ Indexing (LSA/LSI) – this is an unsupervised classification technique used in a popular review platform and some other products
2. Probabilistic Latent Semantic Indexing or Analysis (PLSI; sometimes referred to as PLSA) – this is a supervised learning technique that has been implemented within search engine products
3. Bayesian Modeling (this is described below; the term "Bayesian" is commonly understood for SPAM filtering and other knowledge based products)

4. Discrete Finite Language Models (companies with these technologies have used linguists to build a “rules based” engine of some sort based on the “Parts of Speech” found in a text collection) these are included as “linguistic models and algorithms” that they use to help find keywords for search and to “understand“ collections. These probably are useful in some contexts; generally these are specific to a given language and will not provide much value to other languages without tuning by the authors of the model.

Techniques Discussed/Analyzed

Each of these techniques will be discussed briefly in the context of their use within legal review. All of these are of course useful in the appropriate context. Their usefulness in certain situations and what needs to be added to them to make them an integral part of the legal review process is noted below. Their behavior at a certain scale can become problematic for each technique; this will be discussed below.

Latent Semantic Analysis

Latent Semantic Analysis (sometimes referred to as Latent Semantic Indexing or “LSI”) was invented by a team of researchers at Bell Laboratories in the late 1980’s. It uses principles of linear algebra to find the “Singular Value Decomposition” (see reference [10]) of a matrix which represents the sets of independent vectors within the matrix that exhibit the best correlations between term members of the documents it represents. Notice that documents represented as “vectors” in a matrix make this technique available in the same way that vector space calculations are (as described earlier) available for VSM similarity operations. With LSI/LSA the terms that emerge from the document-term matrix are considered “topics” that relate to the documents within the matrix. These topics are referred to as the “k” most prevalent “topics” or words in the matrix of document terms.

LSA – “The Math”

From reference [11] (Wikipedia page on LSI):

“A rank-reduced, Singular Value Decomposition is performed on the matrix to determine patterns in the relationships between the terms and concepts contained in the text. The SVD forms the foundation for LSI.[15] It computes the term and document vector spaces by transforming the single term-frequency matrix, *A*, into three other matrices— a term-concept vector matrix, *T*, a singular values matrix, *S*, and a concept-document vector matrix, *D*, which satisfy the following relations:

$$A = TSD^T$$

$$T^T T = D^T \quad D = I_r \quad TT^T = I_m \quad DD^T = I_n$$

$$S_{1,1} \geq S_{2,2} \geq \ldots \geq S_{r,r} > 0 \quad S_{i,j} = 0 \text{ where } i \neq j$$

In the formula, **A**, is the supplied ***m*** by ***n*** weighted matrix of term frequencies in a collection of text where ***m*** is the number of unique terms, and ***n*** is the number of documents. **T** is a computed ***m*** by ***r*** matrix of term vectors where ***r*** is the rank of **A**—a measure of its unique dimensions ≤

min(*m,n*). **S** is a computed ***r*** by ***r*** diagonal matrix of decreasing singular values, and **D** is a computed ***n*** by ***r*** matrix of document vectors.

The LSI modification to a standard SVD is to reduce the rank or truncate the singular value matrix **S** to size ***k*** « ***r***, typically on the order of a ***k*** in the range of 100 to 300 dimensions, effectively reducing the term and document vector matrix sizes to ***m*** by ***k*** and ***n*** by ***k*** respectively. The SVD operation, along with this reduction, has the effect of preserving the most important semantic information in the text while reducing noise and other undesirable artifacts of the original space of **A**. This reduced set of matrices is often denoted with a modified formula such as:

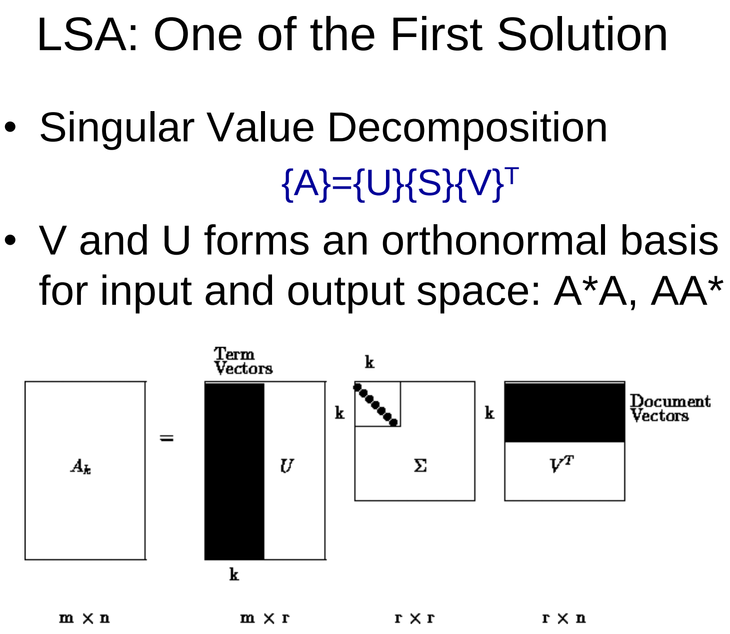
$$\mathbf{A} \approx \mathbf{A}_k = \mathbf{T}_k\, \mathbf{S}_k\, \mathbf{D}_k^{\mathbf{T}}$$

Efficient LSI algorithms only compute the first ***k*** singular values and term and document vectors as opposed to computing a full SVD and then truncating it.”

This technique lets the algorithm designer select a default number of topics which will be “of interest” to them (the default value is usually between 150-300 topics). There is research to indicate that around 100-150 topics is the “best” or “optimum” value to configure when using LSA. This sparks debate among language theorists but has been discussed in other documents (see reference [4] and [11]).

The topics generated via LSA SVD decomposition are referred to as the “k-dimensional topic space” within the new matrix. This is because there are k (100-150-300) topics or terms that now “matter” (instead of the thousands of individual terms in a set of documents before the dimensionality reduction has occurred). So the original matrix that could have contained thousands of unique terms is now represented by a much smaller matrix with terms that are highly correlated with one another. Figure Thirteen outlines some of the mathematical concepts that apply with Latent Semantic Analysis.

Figure Thirteen: LSA Matrix Illustrations



LSA: One of the First Solution

- Singular Value Decomposition

$\{A\}=\{U\}\{S\}\{V\}^T$

- V and U forms an orthonormal basis for input and output space: A*A, AA*

In the diagram it can be seen that the large matrix has been “reduced” to the smaller dimensional area and “important” terms are represented in the matrix. What the “mathematics removed” were topics or terms that did not appear strongly in relation to the terms that “survived” the operations that reduced the larger matrix. So it seems like this is a great idea (it was; it just is not perfect).

LSA Practical Benefits

To help the reader see the benefits of LSI, and how it can find correlations in data, an actual example is provided from a blog maintained by Alex Thomo [mailto:thomo@cs.uvic.ca]. This example was used in an earlier section of the paper to illustrate how LSA as a technique is very valuable. Here we delve into it a bit more and explain its “pros” and “cons”:

>>>

> An Example
>
> Suppose we have the following set of five documents
>
> d1 : Romeo and Juliet.
> d2 : Juliet: O happy dagger!
> d3 : Romeo died by dagger.

> d4 : "Live free or die", that's the New-Hampshire's motto.
> d5 : Did you know, New-Hampshire is in New-England.
>
> and search query: dies, dagger.
>
> A classical IR system would rank d3 to be the top of the list since it
> contains both dies, dagger. Then, d2 and d4 would follow, each containing
> a word of the query.
>
> However, what about d1 and d5? Should they be returned as possibly
> interesting results to this query? A classical IR system will not return
> them at all. However (as humans) we know that d1 is quite related to the
> query. On the other hand, d5 is not so much related to the query. Thus, we
> would like d1 but not d5, or differently said, we want d1 to be ranked
> higher than d5.
>
> The question is: Can the machine deduce this? The answer is yes, LSA does
> exactly that. In this example, LSA will be able to see that term dagger is
> related to d1 because it occurs together with the d1's terms Romeo and
> Juliet, in d2 and d3, respectively.
>
> Also, term dies is related to d1 and d5 because it occurs together with
> the d1's term Romeo and d5's term New-Hampshire in d3 and d4,
> respectively.
>
> LSA will also weigh properly the discovered connections; d1 more is
> related to the query than d5 since d1 is "doubly" connected to dagger
> through Romeo and Juliet, and also connected to die through Romeo, whereas
> d5 has only a single connection to the query through New-Hampshire.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

>End Example from Blog

The example shows how the LSA technique can “link” together the concepts across the document collection. Even though the query has nothing to do with New Hampshire; the state motto: “live free or die” associates the query with the state. The power of the latent technique is obvious; it also can lead to obfuscation as we will also see in the next section.

The Power

LSA provides a set of concepts that were “latent” or unobserved in the data from the original document matrix. Due to the mathematical technique of computing linearly independent vectors that have pronounced term correlations; things that “belong together” show up because they relate to common linking words in certain ways. In a document collection, terms such as “astronaut” will be paired with “rocket” and “space” and “travel” or “expeditions”. Terms such as “cosmonaut” will be related to the same terms. The astronaut and cosmonaut term ascendancies are good examples of the benefits with

LSA. A human reviewer may not have thought about cosmonaut as a possibility for a keyword search along with astronaut but after LSA reveals it as a latent concept the reviewer can include it in the keyword list of a given matter.

For legal reviewers it is valuable to find latent terms that are non-obvious that can be included in with obvious keyword selections. Other relationships in the data can be seen so that the legal reviewer can consider unseen aspects of document collections for building their legal strategies. If the document set is appropriately sized, the lawyer can receive “good ideas” from LSA operations that are run on data.

Another benefit to the technique is that new documents that arrive after LSA has been performed can by “folded in” to an existing reduced matrix (with a vector multiplication operation). This is often necessary as documents for a given case are often found as part of an iterative process where prior review leads to a widening scope of collection. The technique can also have applicability across languages as it may identify correlations in documents that are similar regardless of language [11]. This is not universally true for all languages, but it can be seen in some instances. There can be drawbacks to the LSA approach however.

The Problems: LSA Limitations

For large document collections, the computational time to reduce the matrix to its k important dimensions is very great. On large document collections (100,000 – 200,000 documents) computing SVD’s can take a month or more; even with large capable servers equipped with large memory configurations. The technique does not distribute well over a large number of computers to allow the computational burden to be shared.

Even if the computational burden is acceptable, the technique is difficult to use after a certain number of documents because the data seems to put too many things in too few buckets. The terms that it seems to correlate don’t always seem to make sense to human reviewers. This is due to a problem statisticians call “over-fitting”. Too many wide-ranging topics begin to show up in the reduced matrix. They are related somehow, but it is not clear why.

Some good examples: good correlations occur where the terms “astronaut” and “cosmonaut” are paired with “rocket” and “space” and “travel”. This all makes sense, cosmonauts and astronauts engage in space travel. But also included in the matrix are documents containing travel documentary reviews of the Sahara desert, “camels”, “Bedouins” and “Lawrence” and “Arabia”. These don’t seem at all related to the documents about space travel. This occurs because the correlation of these topics with the ones about space travel relates to long journeys over harsh dry regions with little water, harsh temperatures and environments forbidding or deadly to humans. This over-fitting occurs as more documents with more topics exhibit these correlative effects. Soon there are too many associations to be crisp and logical to a human reviewer. Even in the example with just a few documents, it does not make sense that “New Hampshire” was introduced into the search results as “relevant” when the terms of the query were: “dies” and “dagger”. If this were a murder case it would not have made sense to drag in documents that are about the state of New Hampshire.

So the dimensionality of the matrix was reduced with LSA and there are fewer things for a human to consider, but its discerning power was reduced as well. The results that emerge from the technique are confusing and do not lead to crisp conclusions around what the document population “represents”. The technique is a purely mathematical one; there is no syntactic knowledge imparted to the model to check for consistencies with language usage. So it is clear that LSA is important, helpful under certain circumstances and that also it can be a bit confusing. Across a large population of documents it can take a long time to compute the relationships between documents and the terms they contain and the results of all that computation can end up being confusing.

Probabilistic Latent Semantic Analysis

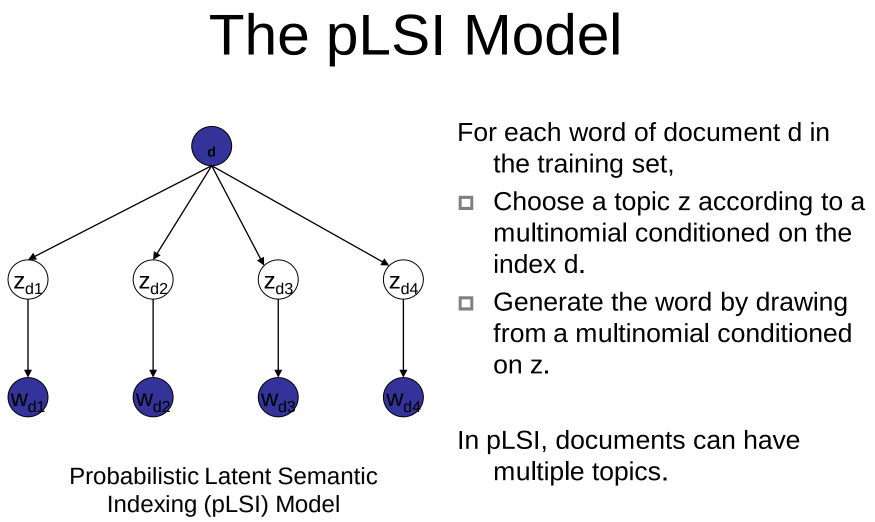
Because of the discernment issue with LSA and as a result of other researchers looking at the problem of conceptual mining in a new way, Probabilistic Latent Semantic Analysis, or PLSA was invented. The reader is referred to [2] in the references section for a full discussion of the technique, but it is built on a foundation from statistics where co-occurrences of words and documents are modeled as a mixture of conditionally independent multinomial distributions.

Instead of using linear algebra to reduce a matrix, the PLSA technique looks at how often a certain topic occurs along with a certain word. The following formula is from [2] and it basically says that for a given class of documents, the word “w” occurs at a certain frequency or with a certain probability along with the topic “z”. This is found by iterating over a training set of documents and finding the highest correlations in the documents that contain both w and z. This is what they mean by a “multinomial distribution” within the documents of the collection. This technique was invented by Thomas Hoffman at Brown University (and others) and is referenced in [13].

$$P(w,d) = \sum P(c)P(d \mid c)P(w \mid c) = P(d) \sum P(c \mid d)P(w \mid c)$$

This may be easier to visualize with an illustration. It can be seen that the user picks a representative set of documents and then the PLSA software finds the highest valued topics for each word. This is accomplished with what is called an “Expectation-Maximization” algorithm that maximizes the “logarithmic likelihood” that topic z occurs with word w for a given word document combination.

Figure Fourteen: PLSA Illustrated



The pLSI Model

For each word of document d in the training set,

- □ Choose a topic z according to a multinomial conditioned on the index d.
- □ Generate the word by drawing from a multinomial conditioned on z.

In pLSI, documents can have multiple topics.

Probabilistic Latent Semantic Indexing (pLSI) Model

Benefits to PLSA

The benefits that PLSA has are that correlations can be found with a statistical basis from a document population. One can say that there is a definite and certain “likelihood” that certain documents contain certain optics. Each document can contain multiple topics as well.

Drawbacks to PLSA

This technique represents the topics among a set of training documents, but unlike LSA it does not have a natural way of fitting new documents into an existing set of documents. It also is computationally intensive and must be run on a population of documents selected by the user. Unlike LSA, it is a supervised classification method; it relies on a set of documents identified by a user. If the population of documents selected by the user is not representative of the entire collection of documents, then comparisons to documents that have been analyzed previously are not necessarily valid. One cannot take into account any prior knowledge of the statistics underlying a new unclassified data set.

PLSA (like LSA) also suffers from over-fitting. With PLSA several hand-selected parameters have to be configured to allow it to perform acceptably. If these “tempering factors” are not set correctly, the same issues with ambiguous topic identification can be seen with PLSA (as with LSA). Given that most users of data analysis products don’t understand the impacts of hand-tuning parameters, let alone the techniques being used (the mathematics involved) this concept of setting parameters in a product is impractical at best. Therefore, PLSA is a statistically based and mathematically defensible solution for concept discovery and search within a legal discovery product, but it can be quite complex to tune and maintain. It is likely a very difficult technique to explain to a judge when a lawyer has to explain how PLSA might have been used to select terms for search purposes.

Problems with Both LSA and PLSA

With both PLSA and LSA, a product incorporating these must still provide an inverted index and basic keyword search capability. Therefore, if one just implemented PLSA or LSA, the problem of providing a scalable keyword indexing and search capability would still exist for legal discovery users. All of the problems that were presented in the first part of this document still exist with platforms supporting these two analytic techniques.

Bayesian Classifiers

Bayesian Classifiers are well known for their work in the area of SPAM detection and elimination. Basically they find "good" examples and "bad" examples of data and compare new messages to these to determine if a new message should be classified as one or the other. The mathematics behind this kind of technology is discussed in [3] and is based on "Bayes Theorem" of conditional probability:

"Bayes Theorem basically says that a document probability of belonging to a certain class ("C" in the equation below) is conditional on certain features within the document. Bayesian theory relies on the fact that these are all independent of one another. This "prior" probability is learned from training data.

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

From [3]: "in plain English the above equation can be written as":

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

These can be used in a legal discovery context and some companies employ these types of technologies in their products. These kinds of classifiers are useful, they just have to be trained to accomplish their work, and this requires a human to perform this prior classification.

Bayesian Benefits

When properly trained, they work quite well. They are surprisingly efficient in certain cases. Like all tools, they are good at certain "jobs".

Bayesian Classifier Drawbacks/Limitations

They sometimes have no idea what to do with unseen data; if there is no example to guide them, they can make "bad choices". They can take skilled humans to collect the data for the "models" that they need to be effective. A lot of times this is not possible and can lead to unproductive behavior.

Natural Language Models: AKA Language Modeling

There are products that claim to have “proprietary algorithms” where “linguists” construct classifiers based on part of speech tagging (POS tagging), or specific dictionary based approaches that they feel “model” language. These often require professional services from the same companies that sell the software implementing the models. This is because the linguist who constructed the model often has to explain it to users. In a legal setting these approaches often require the linguist to become an expert witness if the model results come under scrutiny. These are not stand-alone software tools that one can run at the outset of a legal matter to “get some ideas” about the electronic information available for a case.

These models often require hand-tuning of the models given an initial keyword set produced by attorneys who have some initial knowledge about a case and are therefore not tools to expose meaning in language innately. They are more like “downstream” language classifiers that help identify documents in a large collection that meet some well understood semantic criteria established by the keyword analysis.

There are other products that use a combination of dictionaries and language heuristics to suggest synonyms and polyesters [5] that an attorney could use for keyword searches given a well-understood topic or initial list of keywords. These also may require that an expert explain some of the results if there is a dispute over the keywords it may suggest.

Drawbacks to Linguistic Language Models

The drawbacks to these methods include:

1. They often require hand-tuning and are not general software packages that can organize and classify data
2. They often require professional services and expert witness defense
3. They are very language specific (English, French, etc.) and do not scale across multi-lingual data sets

Other Specialized Techniques

Near-duplicate Analysis

Often versions of documents that are measured to be within “some similarity measure” of reference or example documents are very useful to identify. Knowing that a certain document has been edited in a certain place and in a certain way can be very useful to a legal reviewer. Knowing when this is done on a timeline basis is again a very crucial piece of many legal cases. Near-duplicate data identification products perform this kind of analysis.

For two documents:

- A near-duplicate identification product would build efficient data structures to compare two documents:
  - “Mary had a little lamb” – document #1

  - “Mary had a little white lamb” – document #2

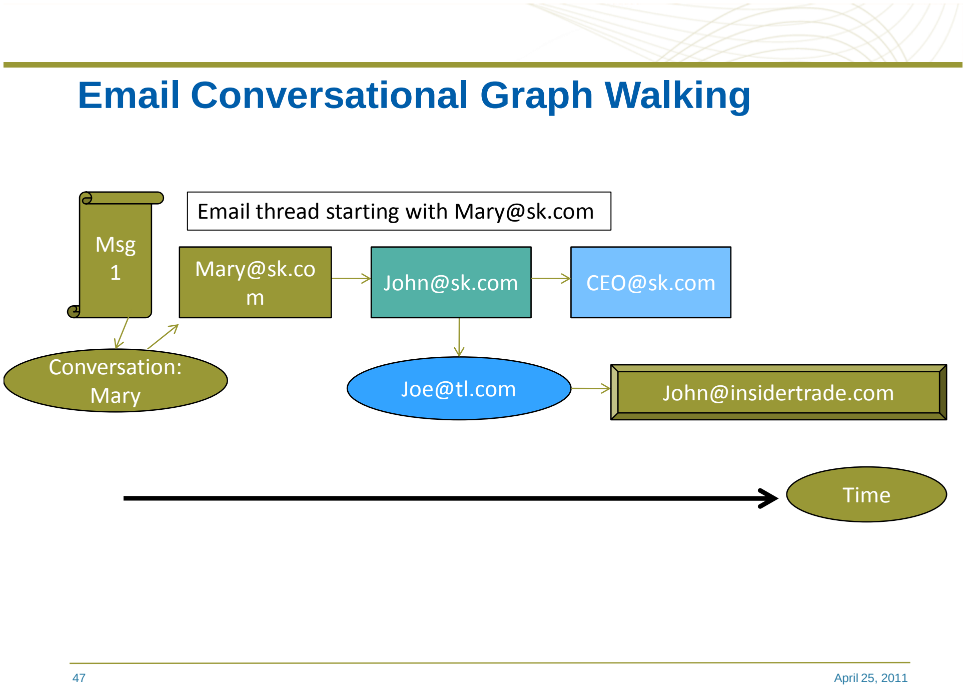
- Yielding a “difference” of one word between the two documents; after “little” and before “lamb”. So the difference would be defined as an “insertion” between “little” and “lamb”.

Mixing this kind of capability with the timeline that could be derived for when the edits occurred (by analyzing the Meta data that should be stored with the documents) both can be used to expose evidentiary facts about a case.

Email Conversational Analysis

Analyzing conversations within email and other message types is a very important part of legal discovery. A full email conversational analysis tool must include the ability to see what individuals sent messages on certain topics to others. In addition, it is important to have a tool that can display the full list of email “domains” that a person used in a given time period. This explains the main sites or companies contacted by a given individual over a specific period of time.

Figure Fifteen: Email Analysis



Email Conversational Graph Walking

47

April 25, 2011

There are several approaches to email conversational analysis. The important aspect of this is allowing the correct attributes (both Meta data (header) information and content) to be included in the algorithm that constructs or follows the conversations.

Legal Discovery Processing Versus “Pure” Natural Language Processing

As we saw in the previous sections, there are a number of techniques that one can use to find conceptual relationships across document collections. In a desire to use the latest computer science techniques to discover information users of legal review technology have turned to Natural Language Processing (NLP) and information analysis approaches that have been used in search engines and e-commerce applications.

Unfortunately legal review professionals have often turned to vendors who confuse general NLP techniques with sound legal discovery practice. The notion that a single mathematical technique will identify everything within a data set that is relevant to a case and help produce all relevant documents as a result is incomplete thinking. NLP techniques when applied correctly can be very helpful and powerful; but like any tool they can only be used in the correct circumstances. Also, at certain magnitudes of scale, the techniques break down, or experience limitations in their feasibility to produce relevant results. Over-fitting can be a problem that obfuscates results and makes the burden of computation a luxury for what the techniques provide in benefits to the review process.

This is why this paper started off with the full explanation of what a platform for legal discovery needs to contain. If the user understands that multiple operations need to be supported to find all aspects of the information relevant to a case (keyword data, Meta data, user-supplied Meta data, analytic associations) then NLP techniques can be one of those operations and the user will have great results. If the system a user selects relies on some specific NLP technique alone, the results it produces will not be complete enough for legal review purposes.

Data Preparation is Important to Obtaining Appropriate Results

We saw in previous sections that the way documents are prepared and represented within a legal discovery system is very important to obtaining good results. If data is not prepared correctly, certain techniques will break down (such as phrase searching performance). With analytics, stemming may reduce the dimensions an algorithm must analyze, but that may yield less specific results than one would envision.

In legal discovery, it can be very important to find documents that say: “by taking the following action; the party is ***choosing*** to violate the contract”. If the documents in a collection are prepared for “NLP” approaches, more documents than one really wants will be returned when looking for the phrase shown above or documents may be missed in the review phase. The near-duplicate mechanisms shown can find too many items that are not true “near-duplicates” if stemming is utilized. So one-step approaches must be carefully scrutinized if the most relevant results are to be obtained.

This can require extra human review and perhaps lead to human error during review. Many products prepare their collections of documents one way (to support both NLP and keyword search approaches). It is important to prepare documents specifically for the type of analysis (NLP or straight keyword-phrase searching) they will undergo. For legal discovery, it is important to prepare collections that can return results specific enough to save time in the initial collection reduction and the eventual legal review portions of the process.

The total platform approach (with the virtual index) lets one prepare data for the analytic operations that are important to each stage of a legal discovery process. This is possible because the virtual index can represent the same data in multiple ways. Along with this, it is important to realize the benefits that analytics can provide.

Aspects of Analysis Algorithms for Legal Discovery

Another aspect of processing data for legal discovery and utilizing NLP techniques is that language characteristics of documents must be taken into account. Most NLP techniques use the statistical nature of data (via token frequency or occurrence) to derive some sort of model that describes data of certain types. If documents containing multi-lingual characteristics are combined with English-only documents, the predictive power of the model will decrease.

If language is not properly accounted for, the predictive power can become even less precise than it would be otherwise. Data preparation is very important to analytic performance in these types of systems. Legal review requires more precision than other applications so it is especially important to be precise with the preparation of data sets.

Benefits of Analytic Methods in Ediscovery

In an Ediscovery context, analytics are very important. They help the reviewer in several ways:

1. They can expose information about a collection of documents that is non-obvious but that can help one understand the meaning of the information they contain. This can help a lawyer understand what keywords would be relevant to a matter, and to select the ones that ultimately get used to discover information about a legal matter.
2. They can identify relationships in data that reveal what information was available to the parties to a lawsuit at certain points in time.
3. They can identify versions of documents and relate these to a timeline to make a reviewer aware of how knowledge related to a lawsuit or regulatory matter has evolved over time.
4. They can be used to find the documents that relate to known example documents within a collection. This helps a reviewer find all documents that are relevant and can also help a reviewer find other relevant concepts that may not have been in an initial keyword search list.

Problems with Analytic Procedures in Ediscovery

As stated above in the introduction to this section of the document, a major problem with analytic procedures in Ediscovery is that one technique is not appropriate in all circumstances. As vendors have tended to champion one technology for their analytics, they tend to promote the over-use of one technique that is available through the use of their particular technology. In their desire to find “the holy grail” or identify the “magic bullet” for legal review users often grab on to technology pushed forward from a certain vendor and then find that it is not the panacea that it was supposed to be.

Once they realize that this is an issue, some customers buy what they perceive as best of breed products. For legal discovery this has historically meant multiple ones; some for analytics and others for keyword search; perhaps a third or fourth for legal processing. Users typically try to use them separately. Outside of a single platform these technologies lose some of their value because loading data into and unloading data from various products introduces the chances of human and other error.

The introduction of one platform that can handle multiple analytic approaches is how one confronts the fact that there is no single analytic technique that masters all problems with electronic discovery.

Related to this issue of multiple products is that the products on the market do not run at the scale necessary to add value in even a medium sized legal matter. Because of this, analytic procedures are (practically) run after a data set has been reduced in size. This can be appropriate, but it can also reduce the useful scope and overall usefulness of the analytic technique in question. If some analytic techniques are run on a very large data set they can take an inordinately long time to run, making their value questionable. In addition, some techniques "break-down" after a certain scale and their results become less useful than they are at lower document counts.

An Ideal Platform Approach

To combat these issues with analytics, the correct platform with a scalable architecture and the appropriate "mix" of analytics is proposed as the answer. In the following sections a set of techniques that have been developed to ameliorate most of the issues with well-known analytic approaches will be shown.

The platform approach includes a "two-tier" ordering algorithm that first "sorts" data into related categories so that deeper analysis can be undertaken on groups of documents that belong together (at least in some sense). This helps the second-level algorithm run at appropriate scale and even avoid "bad choices" when sampling documents for running analysis that can identify conceptual information within documents. This is possible because of the grid architecture explained above and the correct mix of analytic techniques.

Analytic Techniques in Context of a Legal Discovery "Ideal Platform"

So given the assertion that no single analytic technique is adequate on its own to provide legal discovery analysis, this section discusses how a single platform using a combination of different analytic techniques could be valuable. In addition, it shows how a platform implementing several techniques allows the overall system to provide better results than if it had been implemented with one single analytic technique.

The ideal discovery platform:

1. Uses a specific and powerful initial unsupervised classification (clustering) technique to organize data into meaningful groups, and identifies key terms within the data groups to aid the human reviewer. Other analytic processes can take advantage of this first order classification of documents as appropriate
2. Uses a powerful multi-step algorithm and the grid architecture to organize data which has semantic similarity; conceptual cluster groups are formed after accounting for language differences in documents
3. Allows the user to select other analytic operations to run on the classification groups (folders) built in the first unsupervised classification step. This allows other analytic algorithms to be run at appropriate scale and with appropriate precision within the previously classified data folders

(LSA or PLSA for example) the benefit would be that the ideal platform could break the collection down and then allow PLSA or LSA to run at an appropriate scale if a judge ordered such an action

4. Allows the user to select documents from within the folders that have been created
    a. Using keyword search
    b. Using visual inspection of automatically applied document tags
    c. Via the unsupervised conceptual clustering techniques
5. Allows the user to select documents from folders and use them as example documents
    a. "Search by document" examples where the entire document is used as a "model" and compared to other documents
    b. Examples that can be used as "seed" examples for further supervised classification operations
6. Allows the user to tag and otherwise classify documents identified from the stage one classification or from separate search operations
7. Allows the user to identify predominant "language groups" within large collections of documents so that they can be addressed appropriately and cost effectively (translation, etc.)

Conceptual Classification in the Ideal Platform

As we learned in an earlier section of this document, this is an analytic technique that answers the question: "what is in my data"? It is designed to help a human reviewer see the key aspects of a large document collection without having to read all the documents individually and rank them. In some sense it also helps a reviewer deduce what the data "means". In the context of this discussion, it should be noted that the user of this functionality does not have any idea about what the data set contains and does not have to supply any example documents or "training sets".

Conceptual classification supports a number of uses within the ideal discovery product. These include:

1. Organizing the data into "folders" of related material so that a user can see what documents are semantically related; also it builds a set of statistically relevant terms that describe the topics in the documents
2. Presenting these folders so that search results can be "tracked back" to them. This allows a user to use keyword search and then select a document in the user interface and subsequently see how that document relates to other documents the unsupervised classification algorithm placed with the one found from keyword search. This is possible because the virtual index contains the document identifiers and the classification tags that show what related information exists for a given document.
3. Allows other "learning algorithms" to use the classification folders to identify where to "sample" documents for conceptually relevant information (explained below). This means that a first-order unsupervised classification algorithm orders the data so that other analytic processes can select documents for further levels of analysis from the most fruitful places in the document group. This allows higher-order language models (LSA, PLSA or n-gram analysis) to be run on them with a finer-grained knowledge of what the data set contains and to avoid sampling documents and adding their content to a model of the data that might make it less powerful or

predictive. This allows the system to identify the best examples of information where higher-level analysis can reveal more meaningful relationships within document content. Building models of similar documents from a previously unseen set of data is a powerful function of a system that contains analysis tools.

Unsupervised Conceptual Classification Explained

This technique solves the problems that were seen above with the single-technique approach (LSA/PLSA) where over-fitting can become an issue and specificity of results is lost. This technique:

1. Orders the data initially into folders of related material using a linearly interpolated statistical co-occurrence calculation which considers:
    a. Semantic relationships of absolute co-occurrence
    b. Language set occurrence and frequency
    c. This stage of the algorithm does NOT attempt to consider polysemy or synonymy relationships in documents; this is considered in the second stage of the algorithm
2. Performs a second-level conceptual "clustering" on the data where concepts are identified within the scope of the first-level "Clusterings". A latent generative technique is used to calculate the concepts that occur in the first-level cluster groups. This portion of the algorithm is where synonymy and polysemy are introduced to the analysis; "lists" of concepts are computed per each first-level or first-order cluster group; these may be left alone or "merged" depending on the results of the stage three of the algorithm
3. The "lists" of second-level conceptual cluster groups are compared; concepts from one folder are compared to those computed from another. If they are conceptually similar (in cross-entropy terms) they are combined into a "super-cluster". If they are not similar, the cluster groups are left separate and they represent different cluster groups within the product
4. The algorithm completes when all first-order clusters have been compared and all possible super-clusters have been formed

Algorithm Justification

This algorithm allows the data to be fairly well organized into cluster groups after the first-level organization. More importantly, it removes documents from clusters where they have nothing in common, such as documents primarily formed from foreign language (different character set) data. This is important because most latent semantic algorithms will consider information that can be irrelevant (on a language basis) thus obfuscating the results of the concept calculations. This also localizes the analysis of the conceptual computations. Secondly, the over-fitting problem is reduced because the latent concept calculations are undertaken on smaller groups of documents. Since conceptual relationships can exist across the first-level folder groups, the concept lists can be similar; denoting the information in two folders is conceptually related. The third step of the algorithm allows these similarities to be identified and the folders "merged" into a "super-folder" or "super-cluster" as appropriate. Therefore the result is a set of data that is conceptually organized without undue over-fitting and dilution of conceptual meaning.

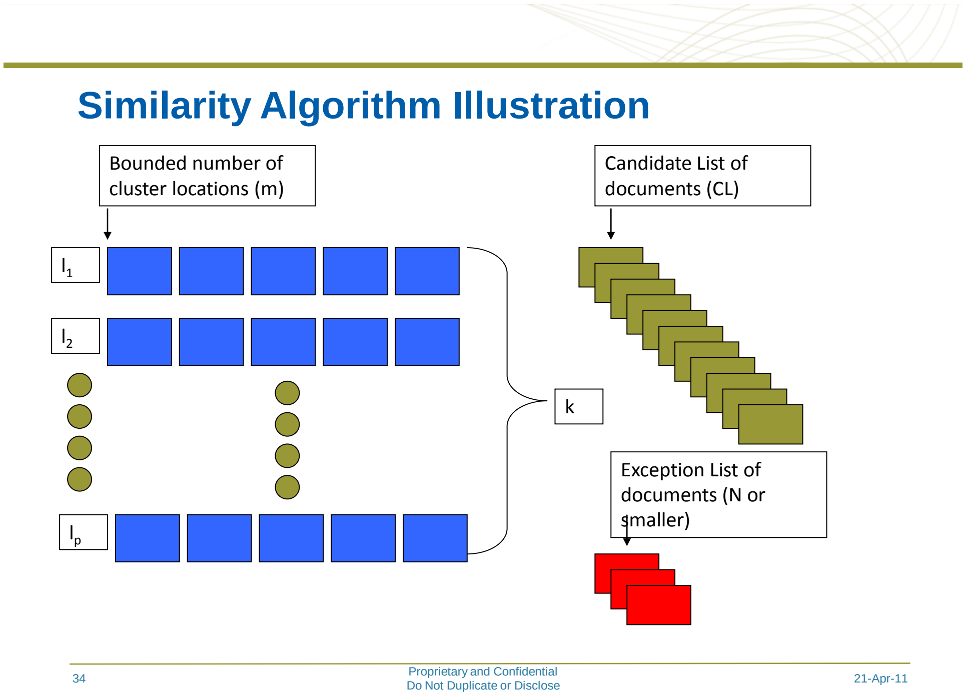
The trick to unsupervised learning or classification is in knowing where to start. The algorithm for unsupervised classification automatically finds the documents that belong together. The algorithm starts by electing a given document from the corpus as the "master" seed. All subsequent seeds are selected relative to this one. This saves vast amounts of processing time as the technique builds lists that belong together and "elects" the next list's seed automatically as part of the data ordering process.

Other algorithms randomly pick seeds and then try to fit data items to the best seed. Poor seed selection can lead to laborious optimization times and computational complexity. With the ideal platform's linear ordering algorithm, the seeds are selected as a natural course of selecting similar members of the data set for group membership with the current seed. There will naturally be a next seed of another list which will form (until all documents have been ordered).

First-Order Classification

This seed-selection happens during the first-order organization of the data. The aim of this part of the algorithm is to build "lists" of documents that belong together. These lists are presented to the users as "folders" that contain "similar" items. The system sets a default "length" of each list to 50 documents per list. The lists of documents may grow or shrink or disappear altogether (the list can "lose" all of its members in an optimization pass); initially the list is started with 50 members however. Please see Figure Sixteen for an illustration of the clustering technique.

Figure Sixteen: Unsupervised Classification



Similarity Algorithm Illustration

Bounded number of cluster locations (m)

Candidate List of documents (CL)

l_1

l_2

l_p

k

Exception List of documents (N or smaller)

34 Proprietary and Confidential Do Not Duplicate or Disclose 21-Apr-11

The algorithm starts with a random document; this becomes the first example document or "seed" to which other documents are compared. The documents are picked from the candidate documents in the collection being classified (candidate list; or every document in the corpus initially). There are "N" of these documents in the collection. The lists are of length "m" as shown in the illustration (as stated the default value of m is 50); the number of lists is initially estimated at k=N/m. The first document is a seed and all other documents are compared to this document; the similarity calculation determines what documents are ordered into the initial list (l_1).

One key aspect of this technique is that initially all documents are available for selection for the initial list. Each subsequent list only selects documents that remain on the candidate list however. This is a "linear reduction" algorithm where the list selections take decreasing amounts of time as each list is built. There is a second optimization step to allow seeds that did not have a chance to select items that were put on preceding lists to select items that "belong" (are more similar to) them and their members.

Each document is compared to the initial seed. The similarity algorithm returns a value between 0 and 1; a document with exact similarity to a seed will have a value of 1, a document with no similarity (nothing in common) will have a value of zero. The candidate document with the highest similarity to the seed is chosen as the next list member in l_1. When the list has grown to "m" members the next document found to be most similar to the seed S_1 is chosen as the seed for the next list (l_2). The list l_2 is then built from the remaining documents in the candidate list. Note that there are N-m members of the candidate list after l_1 has been constructed. This causes (under ideal conditions) a set of lists, with members that are related to the seeds that represent each list, and with seeds that have something in common with one another.

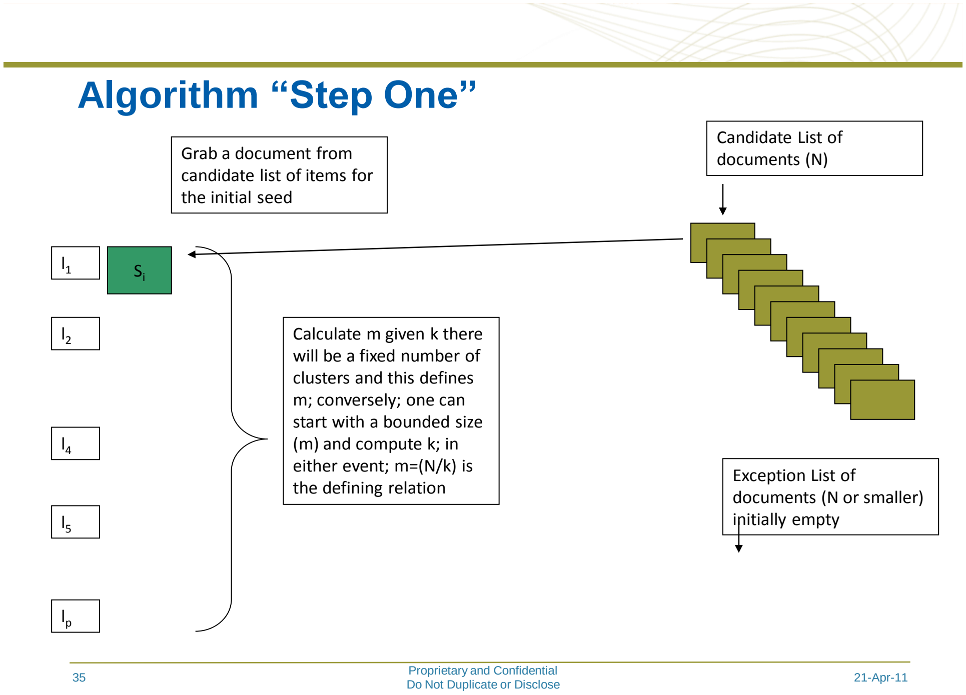
Similarity Calculation

This similarity calculation takes into account how often tokens in one document occur in another and account for language type (documents that contain English and Chinese are "scored" differently than documents that contain only English text). This is a linearly interpolated similarity model involving both the distance calculation between data items and the fixed factors that denote language type. A document that would have a similarity score of "0.8" relative to its seed (which has only English text), based on its English text content alone, but that has a combination of English, Chinese and Russian text will have a score that is "lower" than 0.8 because the semantic similarity score will be reduced by the added attributes of the document having all three languages. This way the system can discern documents that have very similar semantics but that have different languages represented within them. The three language types are viewed as three independent statistical "events" (in addition to the language co-occurrence events of the tokens in the two documents). All events that occur within the document influence its overall probability of similarity with the seed document.

With some "language-blind" statistical scoring algorithms it is possible to have document scores which represent a lot of commonality in one language (English) and where the presence of Chinese text does not influence this much at all. If specific language types are not added into the calculation of similarity, documents with three different languages will appear to be as significantly similar to a seed as those which have only one language represented within their content.

Please see Figure Seventeen for an illustration of the first stage of the algorithm. Please see Figures Seventeen through Figure Twenty Two for other aspects of the algorithm.

Figure Seventeen: “Normal Operation” (Step One)



Algorithm “Step One”

Grab a document from candidate list of items for the initial seed

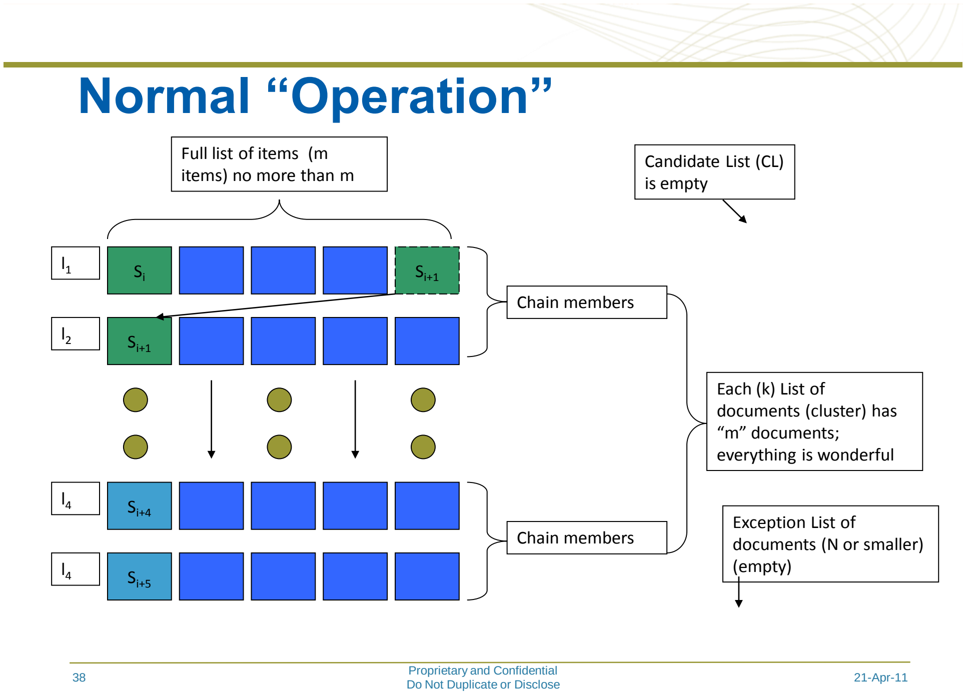
Candidate List of documents (N)

Calculate m given k there will be a fixed number of clusters and this defines m; conversely; one can start with a bounded size (m) and compute k; in either event; m=(N/k) is the defining relation

Exception List of documents (N or smaller) initially empty

35 Proprietary and Confidential Do Not Duplicate or Disclose 21-Apr-11

Figure Eighteen: Normal Operation



Normal “Operation”

Full list of items (m items) no more than m

Candidate List (CL) is empty

Chain members

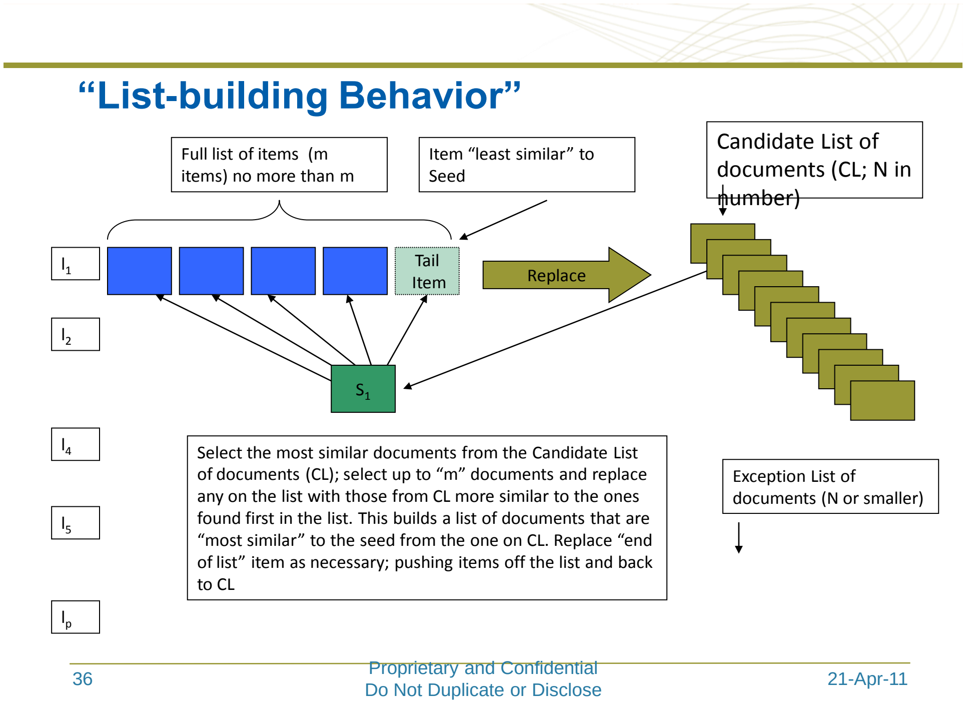
Each (k) List of documents (cluster) has “m” documents; everything is wonderful

Chain members

Exception List of documents (N or smaller) (empty)

38 Proprietary and Confidential Do Not Duplicate or Disclose 21-Apr-11

Figure Nineteen: List Formation



“List-building Behavior”

Full list of items (m items) no more than m

Item “least similar” to Seed

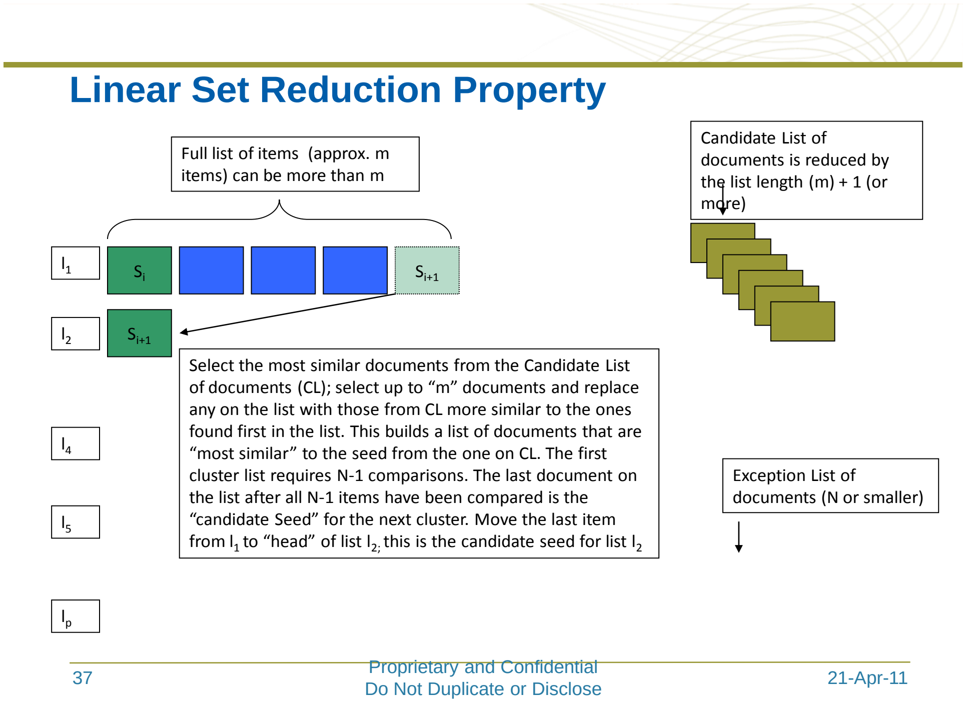
Candidate List of documents (CL; N in number)

Select the most similar documents from the Candidate List of documents (CL); select up to “m” documents and replace any on the list with those from CL more similar to the ones found first in the list. This builds a list of documents that are “most similar” to the seed from the one on CL. Replace “end of list” item as necessary; pushing items off the list and back to CL

Exception List of documents (N or smaller)

36 Proprietary and Confidential Do Not Duplicate or Disclose 21-Apr-11

Figure Twenty: Linear Set Reduction



Linear Set Reduction Property

Full list of items (approx. m items) can be more than m

Candidate List of documents is reduced by the list length (m) + 1 (or more)

Select the most similar documents from the Candidate List of documents (CL); select up to “m” documents and replace any on the list with those from CL more similar to the ones found first in the list. This builds a list of documents that are “most similar” to the seed from the one on CL. The first cluster list requires N-1 comparisons. The last document on the list after all N-1 items have been compared is the “candidate Seed” for the next cluster. Move the last item from l_1 to “head” of list $l_{2;}$ this is the candidate seed for list l_2

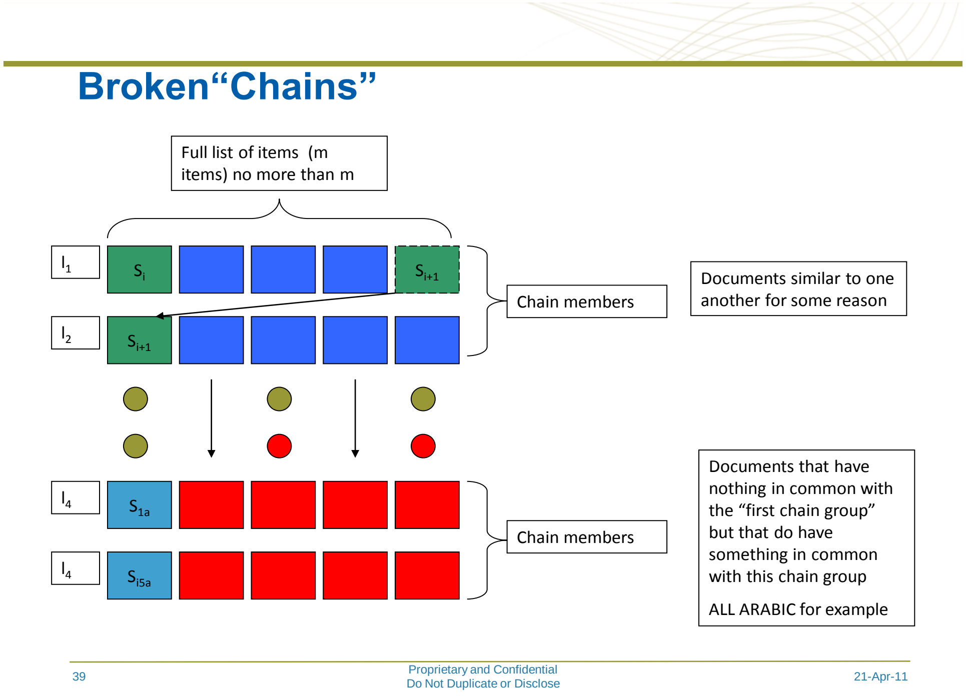
Exception List of documents (N or smaller)

37 Proprietary and Confidential Do Not Duplicate or Disclose 21-Apr-11

The lists which form under normal operation where there are documents with some similarity to a given seed for a given list are as shown in Figure Eighteen. The seed of each new list is related to the seed of a prior list and therefore has a transitive similarity relationship with prior seeds. In this respect each list that has a seed related to a prior seed forms a "chain" of similarity within the corpus. The interesting thing that occurs is when a seed cannot find any documents that are similar to it. This occurs when a chain of similarity "breaks" and in many cases, a "new chain" forms. This is when a seed cannot find a relationship in common with any remaining item on the candidate list. The document similarity of all remaining members of the candidate list, relative to the current seed is zero. This causes the chain to break and the seed selection process to begin again. Please see Figure Twenty One for an illustration of this behavior.

When a seed in a prior list does not find any items on the candidate list which is "similar" to it the algorithm selects an item from the candidate list as the next seed and the process starts over again. If items that are similar to this newly selected seed document exist on the candidate list, they are selected for membership in a new list (headed up by the newly selected seed) and the new list forms the head of a new chain.

Figure Twenty One: Broken Chains



Broken"Chains"

Full list of items (m items) no more than m

Chain members

Documents similar to one another for some reason

Chain members

Documents that have nothing in common with the "first chain group" but that do have something in common with this chain group

ALL ARABIC for example

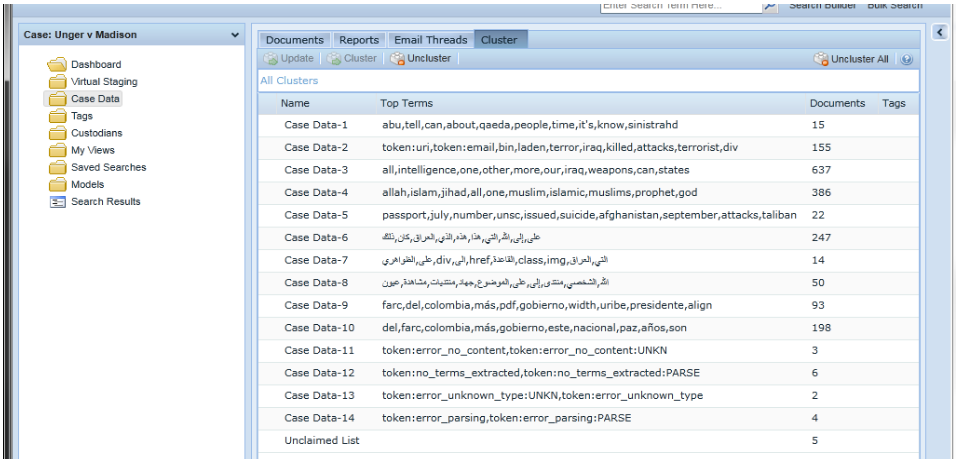
39

Proprietary and Confidential
Do Not Duplicate or Disclose

21-Apr-11

In Figure Twenty One it is shown that the documents in the first chain group have no commonality with documents in the second chain group. The documents in the second chain group do have some commonality among themselves however. The second chain group is formed by selecting a new seed at random from the candidate list when a seed from the first chain group finds no documents in the candidate list with which it has attributes in common. This phenomenon indicates a major change in the nature of the data set. This major change is often related to the document corpus having a set of documents from a totally different language group than that represented in the first chain of lists and documents. This occurs when the language of the documents in a given chain group are English and the next chain group is Arabic for example. Figure Twenty Two is an actual screen shot from a product that shows a “cluster” of documents that are composed of Arabic text. This same behavior can occur within the same language group, but this is the most common reason that it occurs.

Figure Twenty Two: Chain Behavior Displayed in Classification Groups



Name	Top Terms	Documents	Tags
Case Data-1	abu,tell,can,about,qaeda,people,time,it's,know,sinistrahd	15	
Case Data-2	token:uri,token:email,bin,laden,terror,iraq,killed,attacks,terrorist,div	155	
Case Data-3	all,intelligence,one,other,more,our,iraq,weapons,can,states	637	
Case Data-4	allah,islam,jihad,all,one,muslim,islamic,muslims,prophet,god	386	
Case Data-5	passport,july,number,unsc,issued,suicide,afghanistan,september,attacks,taliban	22	
Case Data-6	على,إلى,الله,التي,هذا,هذه,الذي,العراق,كان,ذلك	247	
Case Data-7	التي,العراق,img,class,القاعدة,href,الى,div,على,الظواهري	14	
Case Data-8	الله,الشخصي,منتدى,إلى,على,الموضوع,جهاد,منتديات,مشاهدة,عيون	50	
Case Data-9	farc,del,colombia,más,pdf,gobierno,width,uribe,presidente,align	93	
Case Data-10	del,farc,colombia,más,gobierno,este,nacional,paz,años,son	198	
Case Data-11	token:error_no_content,token:error_no_content:UNKN	3	
Case Data-12	token:no_terms_extracted,token:no_terms_extracted:PARSE	6	
Case Data-13	token:error_unknown_type:UNKN,token:error_unknown_type	2	
Case Data-14	token:error_parsing,token:error_parsing:PARSE	4	
Unclaimed List		5	

In this example the folders labeled: “Case-Data 6”, “Case-Data 7” and “Case Data 8” contain Arabic text documents. This occurred because the textual similarity of the documents had little to do with English and were very much in common because of the Arabic text they contain. The similarity algorithm put the Arabic documents together because the product evaluates each “token” of text as it is interpreted in Arabic. The frequency of Arabic tokens in the documents compared with “English seeds” showed no similarity with a given English document seed. An Arabic “chain” formed and attracted Arabic documents to these particular folders. Similar assignments happened in these data for Spanish and French documents.

Second-Order Classification

It was explained above, but with the benefit of the illustration it is clear that the conceptual calculations are undertaken on the folders consecutively. They benefit from the fact that the overall calculation has been broken into groups that bear some relationship to a seed document that leads the cluster. Even if conceptual similarity spans two clusters, the third and final stage of the algorithm will “re-arrange” the cluster membership to order the documents conceptually. Computing concepts on each individual cluster from the first stage of the algorithm reduces the number of documents in the calculation and thus over-fitting.

The technique used in the ideal platform at this stage is reviewing conditional probabilities with prior statistical distributions. It is drawing an initial “guess” of how the terms in the documents are distributed statistically by gathering information about the document classifications found in the first-order classification step. It computes the likelihood of certain terms being “topics” within certain documents and within the overall collection of documents. It orders topics and documents so that they can be regarded as “topic labels” for the documents that are contained within the folders.

Third-Order Classification

This stage of the algorithm will re-order any documents into the final folder groups according to the conceptual similarity of the concept lists computed in stage two of the algorithm. Documents which “belong” to a “super cluster” are merged to be with the documents that are most similar to the concept list computed for some number of second stage clusters. Concept lists for each second stage cluster are compared and if their members are similar, the documents forming the lists are merged into a final super cluster; otherwise the documents are left in the cluster they inhabit. This allows conceptually similar folders to be merged together; the documents comprising folders with similar concept lists will be re-organized into a super-cluster. The documents the folders represent “belong together” so the folders are “merged”.

First-Order Classification Importance

As stated previously, when documents contain different languages, the algorithms that compute the labels for document groups can lose precision. These algorithms look at the probabilities of certain terms occurring with other terms and when multiple languages are involved their results can become skewed. The first-order classification algorithm puts the documents with similar first-order language characteristics together, which aids the performance of the second-order topic generation algorithm. The two algorithms together are more powerful than either one is together. This also helps the third-order classification algorithm as the folders that have similar characteristics tend to be “near” one another in the folder list. Even if they are not, the concept clustering algorithm will find the conceptually related information that “goes together” but merging is often possible early on in the “walking” of the folder concept lists because of the first-order classification operation.

Second-Order and Final Classification Importance

With this technique, the topics that are latent are generated by the algorithm running on the folders of pre-ordered documents themselves. This provides the benefits of LSA or PLSA on the pre-ordered sets of data but with much less computation (by taking advantage of the classification from the first pass of the algorithm). Without the first-order technique, more computation would be required to arrive at the optimal generation of the topics. This would place a computational burden on the system unnecessarily. The first-order classification of the documents assists the second algorithm and makes further analysis much more “clear” as well. When the final check is done on the semantic label lists of the folders in the collection, they allow for the documents that belong together conceptually to be re-clustered as necessary.

Multi-lingual Documents

It is important to note that the algorithm still handles multi-language documents, and the concept generation algorithm can find cross-correlations of terms in multiple languages. Terms that occur in a document containing English and Arabic text will have conceptual lists that contain both Arabic and English members. The first-order grouping of primarily Arabic or English documents together will still allow for single language correlations to predominate but will not prohibit the generation of concepts from documents containing both Arabic and English text.

Value of Classification

The value of classification like this is that a human reviewer can quickly identify document groups that may be of interest. The “top reasons” or “top terms” that a folder contains (the predominant terms in the documents it contains) is shown at the top of the folder in the screen shot contained in Figure Twenty Two. The user of the product can determine if the documents are of any interest to him/her quickly by reading the labels on each folder. Further, the reviewer does not have to open and read documents that may be Arabic or Chinese (unless they want to read them). This folder based ordering of documents allows a reviewer to avoid obviously irrelevant information such as documents in a language that is of no interest to them. More importantly however, this pre-ordering first-stage classification technique makes higher-order analysis of the data in the folders accurate and predictable and more computationally efficient. The end-stage classification yields strong language semantics for members of final classification groups.

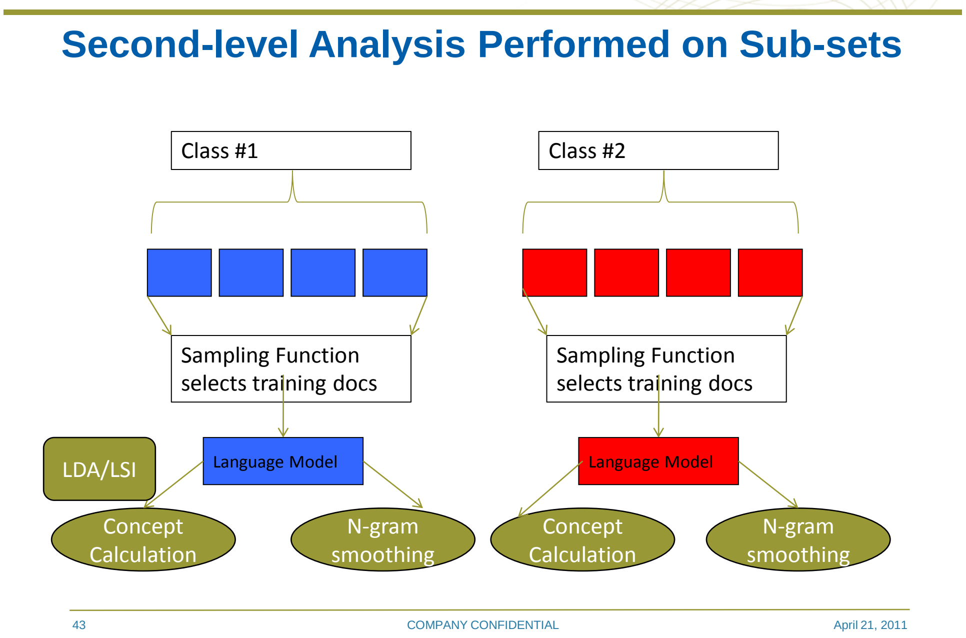
Other Benefits of Document Classification

Other benefits are that any document found with a keyword search can be related back to the classification groups. Since the documents in these classification folders are related to their seed, they are very likely to be related to one another. This allows a reviewer to see what other documents are similar to a given document found via keyword search where the reviewer knows that the reference document contains at least agreed upon key terms. The conceptual organization past this step allows for the final clusters to include semantically conceptual clustering relationships that can be related back to keyword searches. The ideal system also allows one to use the pre-ordering classification for other analysis techniques such as PLSA or LSA. Either would benefit from operating on pre-ordered data that is smaller than the entire collection.

N-gram or Other Statistical Learning Models

For building language specific tools on top of the base classification engine, the pre-ordering technique is especially useful. After the classification algorithm has ordered a collection, higher-order language models can be built from samples within the larger collection. This may be beneficial for building n-gram models for language specific functions like part of Speech (POS) tagging. Other tools that could benefit from this would be learning models that support functions such as sentence completion for search query operations. In these cases, knowing that a cluster group contains primarily Arabic textual information would allow the n-gram model to select samples from an appropriate set of documents. This can be important if one is building a model to handle specific functions such as these. The first-order algorithm will “mark” the analytic Meta data for a certain cluster group to show that it represents a predominant language. For a POS tagger, this would be important as many of these are highly sensitive to the input model data and training them with appropriate samples is important. It would be counter-productive to train an English language POS tagger with German training data for example. The first-order algorithm allows one to select documents from the appropriate places within the larger collection of documents for specific purposes. See Figure Twenty-Three for an illustration of this behavior.

Figure Twenty Three: Selecting from Pre-Classified Data for Higher-Order Models



Second-level Analysis Performed on Sub-sets

43 COMPANY CONFIDENTIAL April 21, 2011

How Pre-Ordering Can Make Other Techniques Better

This first-order classification capability can reduce the amount of documents that any second order algorithm has to consider. This can help other less-efficient algorithms run more effectively. As with the concept calculation example (above) it may be desirable to run LSA or other tools on data that the platform has processed. By utilizing the folder-building classification algorithm within the platform, the large population of documents for a given case could be reduced to more manageable sized increments that an algorithm like LSA can handle.

If opposing counsel were to insist on running LSA or PLSA or some other tool from a select vendor on a data set, the ideal platform could order and organize smaller folders of data that LSA or PLSA could handle. This technique of pre-ordering the data will generate smaller sized related folders that these other techniques could process at their more limited scale. The reduced size of the data set would help focus the results of LSA because it would have fewer documents and find fewer concepts to fit into the "buckets". Therefore the platform could help reduce the LSA over-fitting issue. This would help PLSA in this regard as well.

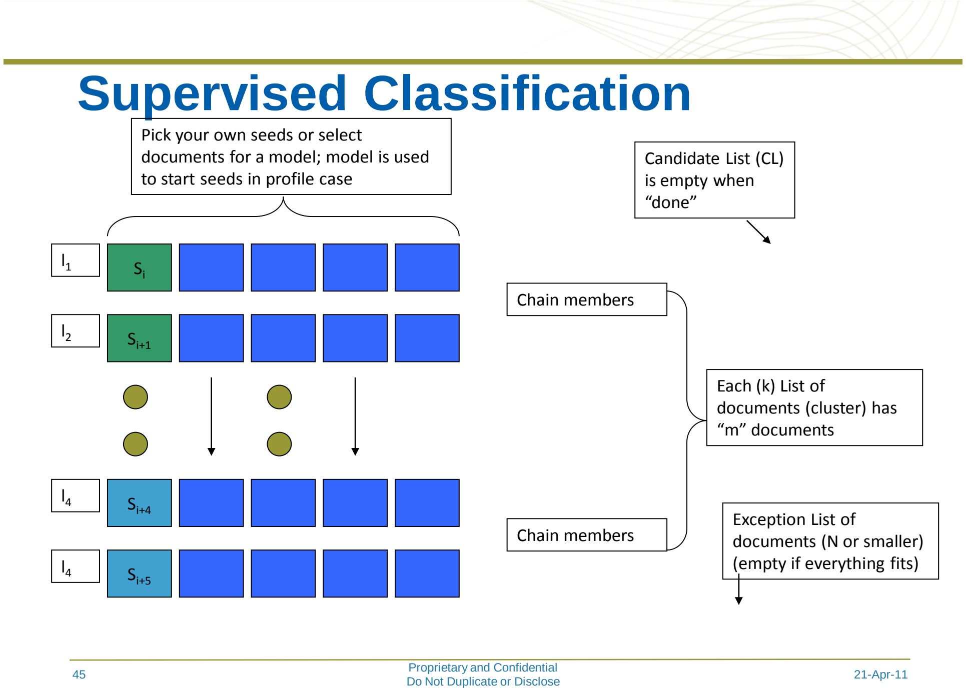
As mentioned in other sections of this document, LSA was envisioned and works best on collections of documents that are "smaller" than those that are routinely found during current legal discovery matters. In [4] it is stated by the authors of the LSA algorithm that they envision "reasonable sized" data sets of "five thousand or so" documents. In today's cases, it is routine to see 100,000 to 200,000 or more documents (millions are not uncommon).

With the combined platform approach a technique like LSA could be run on the documents from the most likely "clusters" from the ideal platform so that the computation would be tractable. If a judge felt comfortable with LSA as a technique due to prior experience with the algorithm he or she could see better results by reducing the amount of documents that the algorithm has to address at any one time. The use of the ideal platform would benefit a legal review team by making a previously used product implementing LSA more effective at what it does.

Supervised Classification

Supervised Classification requires the user to provide some example data to which the system can compare unclassified documents. If a user has some documents that they know belong to a certain classification group, a system can compare documents to the examples and build folders of documents that lie within a certain "distance" from the seeds (in terms of similarity).

Figure Twenty Four: Supervised (Example Based) Classification



Supervised Classification

Pick your own seeds or select documents for a model; model is used to start seeds in profile case

Candidate List (CL) is empty when "done"

Chain members

Each (k) List of documents (cluster) has "m" documents

Chain members

Exception List of documents (N or smaller) (empty if everything fits)

45 Proprietary and Confidential Do Not Duplicate or Disclose 21-Apr-11

If the system has pre-ordered a lot of the data (using unsupervised techniques as before), then finding examples is simplified for the user. They have some idea where to look for examples that they can use to classify documents that will enter the case as new documents. Secondly, search results can be related to document clusters that exist, then examples of the "strongest most similar" documents to the search results can be located within a cluster folder, and then the supervised classification technique can identify other documents that belong with pre-selected "seeds". Again, documents added to a case can be classified with examples using the supervised technique shown above. This continuous classification of documents can help reviewers find the most relevant documents rapidly with more or less automated means.

Email Conversational Analysis

Email conversational analysis is an important aspect of any legal review platform. Seeing what conversations transpired between parties is important. This was discussed previously. With the ideal platform approach of providing classification along with the email threading, these two techniques can be used simultaneously to identify documents that are in a conversation thread, and that have similar documents which may exist outside that thread. The existence of documents that are similar to those in a thread will lead to the identification of email addresses that perhaps were outside the custodian list but that should be included. Having the conversation analysis and the classification capability all within one platform makes this "analytic cross-check" capability possible.

Near-Duplicate or Version Analysis

The version analysis mentioned above, combined with supervised clustering and Meta data search can identify what documents were edited at certain times and by whom. Using near duplicate analysis can allow the system to "tag" all members of a "near-dupe group" within a collection of documents (auto-tagging of analytic Meta data). Using supervised clustering with a known seed (from the near dupe group) a user can identify other versions within the collection of documents comprising a case. Using the Meta data attributes to identify owners and import locations of documents that have been collected exposes information about who owned or copied version of files at any time during the life cycle of the case. This is a very powerful attribute of a platform that handles these combined sets of analytic processes at large scale.

Summary

This paper attempted to display the value behind a comprehensive platform that handles various levels of indexing for Meta data content and analytic structures. It exposed new concepts behind analysis and storage of these constructs that implement high-speed indexing and analysis of data items within the context of legal discovery. Further, it explored and discussed several aspects of large scale legal discovery processing and analysis and how the correct architecture combined with indexing and search capabilities can make legal discovery effective and productive relative to current single product approaches.

The "ideal platform" approach (as it was called) presented both architecture and a set of capabilities that remove risk of error from legal discovery projects. Examples of how this combination would reduce cost and risk of error in legal discovery engagements were presented.

Finally, analytic approaches that are available from electronic discovery products today, how they work, where they are effective and where they are not effective were presented. These were compared and contrasted with one another and were discussed in relation to the ideal platform approach. The ideal platform and its ability to pre-order and classify data at great scale, and then perform generative concept and label generation to identify the "meaning" of content and assign it to "folders" of documents within large cases was discussed. It was shown how the platform approach of pre-classifying data and using a hierarchical model of classification algorithms could aid other products and techniques such as those that utilize LSA and PLSA.

References

[1] Wikipedia description of Latent Semantic Analysis: http://en.wikipedia.org/wiki/Latent_semantic_analysis

[2] Wikipedia description of Probabilistic Latent Semantic Analysis: http://en.wikipedia.org/wiki/PLSA

[3] Wikipedia description of Bayesian Classifiers: http://en.wikipedia.org/wiki/Bayesian_classification

[4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman (1990). "Indexing by Latent Semantic Analysis"

[5] Wikipedia page on Polysemy: http://en.wikipedia.org/wiki/Polysemy

[6] Wikipedia page on Support Vector Machines: http://en.wikipedia.org/wiki/Support_vector_machine

[7] Wikipedia page on Vector Space Model: http://en.wikipedia.org/wiki/Vector_space_model

[8] Wikipedia page on SMART system: http://en.wikipedia.org/wiki/SMART_Information_Retrieval_System

[9] Web Page of Martin Porter: http://tartarus.org/~martin/PorterStemmer/

[10] Wiki entry on mathematical treatment of SVD: http://en.wikipedia.org/wiki/Singular_value_decomposition

[11] Wiki entry on LSA/LSI: http://en.wikipedia.org/wiki/Latent_semantic_indexing

[12] Adam Thomo Blog: mailto:thomo@cs.uvic.ca] Blog entry for LSI example

[13] ^ Thomas Hofmann, *Probabilistic Latent Semantic Indexing*, Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), 1999