Auto-Grouping Emails For Faster E-Discovery

Sachindra Joshi
IBM Research, India
jsachind@in.ibm.com

Danish Contractor
IBM Research, India
dcontrac@in.ibm.com

Kenney Ng
IBM Software Group, USA
kenney.ng@us.ibm.com

Prasad M Deshpande
IBM Research, India
prasdesh@in.ibm.com

Thomas Hampp
IBM Software Group, Germany
thomas.hampp@de.ibm.com

ABSTRACT

In this paper, we examine the application of various grouping techniques to help improve the efficiency and reduce the costs involved in an electronic discovery process. Specifically, we create coherent groups of email documents which characterize either a syntactic theme, a semantic theme or an email thread. All such grouped documents can be reviewed together leading to a faster and more consistent review of documents. Syntactic grouping of emails is based on near duplicate detection whereas semantic grouping is based on identifying concepts in the email content using information extraction. Email thread detection is achieved using a combination of segmentation and near duplicate detection. We present experimental results on the Enron corpus that suggest that these approaches can significantly reduce the review time and show that high precision and recall in identifying the groups can be achieved. We also describe how these techniques are integrated into the IBM eDiscovery Analyzer product offering.

1. INTRODUCTION

Discovery is a process in the pre-trial phase of a lawsuit in which each party involved in a dispute is required to produce relevant information, records and other evidence related to the case to the other party. In December 2006, Federal Rules for Civil Procedures (FRCP) [12] codified the requirements of producing relevant electronic information and records also referred to as electronically stored information (ESI). These amendments to FRCP gave rise to electronic discovery or e-discovery which is part of a process for providing ESI that is relevant to a case to the other party. Examples of ESI include emails, chat logs and other documents such as word documents, presentations and spreadsheets.

The rapidly increasing volume of ESI poses an enormously challenging problem of finding the information that is relevant to a case. To complicate matters further, 60% the total legal cases warrant some form of e-discovery and this number is likely to increase over the next few years according to the Socha Report [21]. These volumes contributed to the first billion dollar year for e-discovery in 2005 and it is projected to increase to $4.8 billion by 2011 [18].

Figure 1 shows the different stages involved in an e-discovery process [8][9]. The process starts with locating potential sources of ESI and determining its scope in the identification stage. This is followed by gathering ESI from heterogeneous sources for further use and ensuring that it is protected against inappropriate alteration or destruction. The collected data is then processed which includes formatting the data into a canonical form and reducing the volume of ESI data by context, keywords and patterns. The processed data is then evaluated for relevance and privilege in the review stage and is produced to the concerning party in an appropriate media form in the production stage.

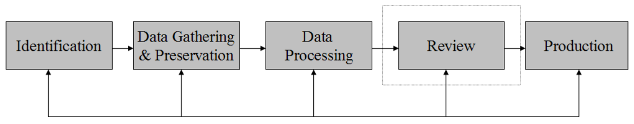


Figure 1: Different stages involved in an e-discovery process.

The huge costs involved in the e-discovery process are generally attributed to the high cost involved in the review stage. In this stage documents are categorized as responsive, non-responsive or privileged. Responsive documents are relevant to the case and need to be produced, non-responsive documents are irrelevant and can be skipped, while the privileged document need to be withheld and cannot be produced. Typically, a team of reviewers use keyword based search to first determine a set of potentially relevant documents. These documents are then manually reviewed and categorized. It is this manual processing that is the most resource consuming. As an example, in the civil lawsuit brought by the Clinton administration against tobacco companies in 1999 (U.S. Vs. Philip Morris), 32 million email records from the White House were made subject to discovery. A set of 200,000 emails along with attachments was uncovered by using an automated search engine and Boolean queries. These documents then had to be manually reviewed on a one-by-one basis to determine the responsiveness to the case. This took over six months for a team of 25 reviewers posing a huge cost [1].

In this paper, we propose techniques that have the potential to help significantly reduce the time required for manual review of documents. We focus on emails as they are one of the most important categories of ESI. Emails are ubiquitous and constitute more than 50% of the total volume of ESI [8]. Our approach is to present *groups* of documents rather than *individual* documents to an analyst at review time. The analyst can then mark the whole group as responsive, unresponsive, or privileged after reviewing some or all the documents contained in a group. Optionally, the analyst can also mark only a few documents in a group. The group level view of documents enables faster reviews because all the documents in a group have "similar" or related content. Since there are typically multiple analysts performing reviews, the grouping of documents also enables assigning entire groups of related documents to analysts, potentially making the review process more consistent.

Leveraging information retrieval technologies and techniques developed for the detection of near duplicate documents, we develop methods to create three different types of coherent email groups: (1) grouping *syntactically* similar emails (along with the detection of automated messages), (2) grouping *semantically* similar email documents, and (3) grouping emails that are part of the same *thread* or "conversation." In a group of syntactically similar documents, all documents are either exact duplicates or near duplicates of each other. Semantic grouping is achieved by associating documents with legal concepts based on their content. Email thread groups can be formed based on their reply and forwarding relationships. We evaluate the effectiveness of these grouping features in terms of the quality of grouping and show that it can lead to a significant improvement in efficiency. Many of the techniques described in this paper have also been integrated into the IBM product offering for e-discovery review: IBM eDiscovery Analyzer.

The rest of the paper is organized as follows. In Section 2, we present an architectural overview of a typical e-discovery review system. In Section 3, we present the technique that we adopt for detection of near duplicate documents. We use this technique for detecting groups of automated emails in Section 3.4. In Section 4 and Section 5 we present our methods for semantic grouping of emails and email thread detection, respectively. We present experimental evaluations of the proposed techniques in Section 6. We provide details on how these techniques are integrated into and used in the IBM eDiscovery Analyzer product in Appendix C. We conclude and discuss future work in Section 7.

2. E-DISCOVERY ARCHITECTURE

The process of e-discovery starts with locating potential sources of relevant emails. These emails are collected from multiple sources and vendor repositories and may be archived in a content management repository. An e-discovery review system takes the archived emails, processes them and provides an interface where a user can query, manually review and categorize the emails. The gathering and archiving of emails is typically done by a separate system.

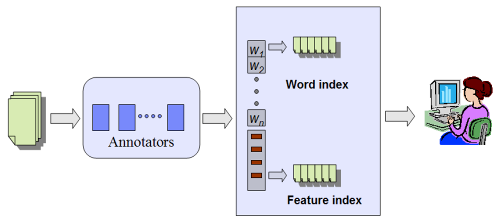


Figure 2: Architecture of a typical e-discovery review system.

Figure 2 shows the architecture of a typical e-discovery review system. Archived emails are passed through a pipeline of annotators. Each annotator in the pipeline is a piece of software that processes an email and discovers new features associated with it. Examples of annotators are *named entity annotators* which detect person names, company names, and location names that appear in the email content, *meta fields annotators* which identify meta fields such as sender, recipient, and subject associated with an email, and *language annotators* that recognize the languages in which an email is written. Once an email is processed through these annotators, its content and the newly generated features are indexed. The index provides the capability of efficiently searching emails based on words and other associated features such as named entities, language, sender, recipient, and date information. An e-discovery review system provides a user interface where such queries can be formed and the returned set of emails can be reviewed. Emails can be manually tagged with a predefined set of categories during the review process. These tags can be used, for example, to indicate whether an email is responsive, privileged, or unresponsive.

The techniques proposed in this paper are used to provide additional features in an e-discovery review system. These grouping features have the potential to significantly expedite the manual review process.

3. SYNTACTIC SIMILARITY

The problem of detection of near duplicates or syntactically similar groups has been an active area of research in the database and in the Web search communities. In this section, we present a brief overview of related work and discuss the method adopted for near duplicate detection of emails in our solution.

3.1 Related Work

Most of the existing duplicate detection techniques can be categorized into two classes: (1) *full content based* and (2) *signature based*.

Full content based approaches use a vector space model which treats a document as a "bag-of-words." One full content based approach uses the identity measure that is based on the intuition that similar documents should contain similar numbers of occurrences of words [14]. Therefore, differences in the number of occurrences are used as a penalty in computing the identity measure for a pair of documents. Computing the similarity measure using the entire document is time consuming and therefore a method based on random projection is proposed by Charikar [5]. This method has been found to be very efficient in a large scale evaluation [13].

Signature based techniques take a sequence of contiguous tokens or *substrings* of a document. A popular similarity measure used is *resemblance* that compare the number of matching substrings [3, 4]. Two documents are said to be near duplicates of each other if they share several substrings.

Signature based algorithms differ in their choice of substring size, hashing function used, and substring selection strategy. Substring size refer to the number of tokens used in each substring. A large value of substring size would yield several *false negatives* while a small value would result into many *false positives*. Prior work has suggested that substrings of 3-5 words are good [14]. The popular hashing functions are SHA1 [4] and Rabin fingerprinting [11] as they are efficient to compute and have a low rate of hash collision.

Substring selection strategy is used pick a subset of all the substrings present in a document that is used for hashing and matching. Various strategies have been proposed in the past which can be categorized as *frequency based*, *anchor based*, *hash value based*, and *position based*. The frequency based strategies select substrings based on *term frequency* (tf) and *inverse document frequency* (idf) [6][7]. Methods based on idf are known to be over sensitive to small changes in the document content and therefore can easily lead to high false negatives. The anchor based strategies pick substrings that start with some special character sequences and have been shown to work effectively [14] [22]. However these approaches require some manual supervision for identifying the special character sequences. Hash value based strategies select substrings based on their hash values. The popular DSC approach [4] selects substrings with the minimum hash value and uses them to compute the *resemblance* measure. They show that by using a min-wise independent family of permutation a fixed size sketch can be generated which preserves the resemblance measure for a pair of documents.

```
Algorithm ComputeSignatures (Document d)

    % Initialize all $n$ signatures to the maximum value
    For $(i = 1; i \leq n; i++)$
        $signature_i = MAX$

    For each $k$ length character sequence $seq$ in doc $d$
        For $(i = 1; i \leq n; i++)$
            Let $checksum_i = f_i(seq)$
            If $checksum_i < signature_i$ then
                $signature_i = checksum_i$

    % Keep only the $j$ lowest significant bits
    For $(i = 1; i \leq n; i++)$
        $signature_i = lsb_j(signature_i)$
```

Figure 3: Signature computation algorithm.

3.2 Our Approach

For detecting almost identical content, we adopt a signature based technique. Signature based techniques are not only faster to compute but can also be efficiently stored in an index structure and can thus easily be used for detecting near duplicates with varying degrees of similarity thresholds at runtime. Signature based techniques generate a set of signatures for each document and use them to compute a *similarity* measure between a pair of documents. We specifically use the resemblance measure [4] to compute the similarity of a pair of documents which is defined as follows:

$$Sim(d_1, d_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2} \tag{1}$$

where, d_1 and d_2 are two given documents and S_1 and S_2 are the set of all the substrings of length k that appear in d_1 and d_2 respectively. A high value of resemblance signifies a high similarity in the content of the given documents.

Figure 3 describes a method *ComputeSignatures* that generates n signatures for each document. The method uses n different one-to-one functions referred to as f_i in the Figure 3 that are kept the same for all documents. Each function is a 64 bit Rabin fingerprinting function. For a given document d, all the k length character sequences are considered. The value of k is empirially determined and in our experiments, we use a value of $k = 20$. Note, that if the document d contains p characters then there will be $p - k + 1$ such sequences. For each f_i all the $p - k + 1$ values are computed and the smallest of them is kept as the i^{th} signature. For efficiency reasons we keep only the j lowest significant bits for each of the smallest signatures which is computed by the function lsb_j in the Figure 3. Thus the value of signatures range from 0 to $2^j - 1$. This method has some similarities to the method proposed in [10]. It cannot be proved that these functions are min-wise independent [4] but they are found to be effective in practice [10] [13].

Assuming a uniform probability distribution on the values, the probability of two documents having the same signature is $\frac{1}{2^j}$, where each signature is a j bit value. This is the probability that two documents will have the same signature by chance. The probability that two documents will have the same m signatures by chance can be given by $(\frac{1}{2^j})^m$. This value could be very high for low value of j and m. A high value of this probability will increase the occurrence of false positives. We investigate appropriate values for these parameters in Appendix A.

3.3 Groups of Near Duplicate Emails

Each signature value can be used as a dimension and an email can be represented as a feature vector in this space. Traditional clustering algorithms such as K-means or hierarchical agglomerative clustering (HAC) [2] can then be used to create groups of near duplicate emails. However, the traditional clustering algorithms are in-memory algorithms and cannot be easily used for huge datasets with millions of documents. We therefore develop a clustering method that works out of an index.

```
Procedure CreateSyntacticGroups(Index I)

    For each document $d$ in Index I do
        If $d$ is not covered
            Let $S = \{s_1, s_2, \ldots, s_n\}$ be its signatures
            D = Query(I, atleast($S$, $k$))

        % make all the documents in D covered
        For each document $d$ in $D$
            $d$ is covered
```

Figure 4: Creating groups of syntactically similar documents.

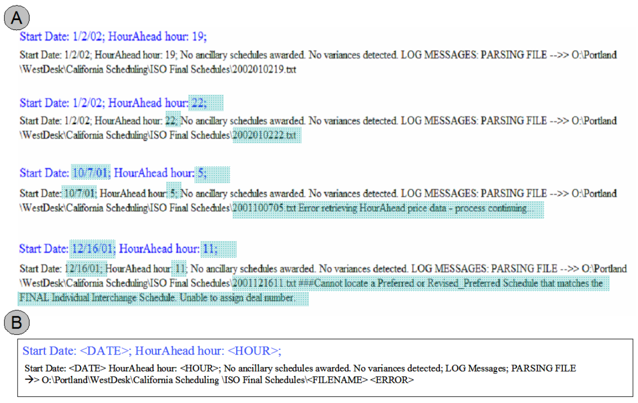
For each email n signatures are computed and indexed along with the content and other meta fields associated with the document. We then use the procedure *CreateSyntacticGroups* outlined in Figure 4 to create groups of near duplicate emails. It uses a parameter referred to as k that denotes the minimum number of signatures that needs to be the same for a pair of documents to belong to the same group.

The procedure finds groups of near duplicate emails by first picking up an uncovered email d from the index. It then searches for all the emails that have at least k same signatures as the n signatures associated with the email d. This set can easily be identified using a single query on the index using the posting list associated with each signature value. In the Figure 4, the function *Query*(I, *atleast*(S, k)) returns the set of emails that have at least k signatures from the set S. Other kinds of constraints concerning the hash value of the associated attachments can also be included in the query. These kinds of queries are supported in several off-the-shelf indexing frameworks such as Lucene[1]. The returned set forms a set of near duplicate emails as all of them have at least k same signature values. The procedure continues with the next uncovered email to find other groups of near duplicate emails. Note that while reviewing a particular group at review time, an analyst can change the value of k to obtain more or less fine grained groups. Creating the group with a new value of k involves running a single query against the index and therefore can be done in an on-line fashion.

3.4 Groups of Automated Emails

We observe that there are several types of automated messages in an enterprise collection of emails. Automated messages are frequently used for various purposes such as *automatic reminders*, *confirmations*, *out of office messages*, and *claims processing*. These messages are typically generated based on a template which consists of some textual content along with a set of fields that are filled in during message generation. All the messages that are generated from a single template have *almost* identical content and are therefore near duplicates of each other. Figure 5 shows four example emails from our large email testing set that are generated automatically along with a possible template (in the rectangular box at the bottom) that could be used to generate these messages. For each email, only the subject and content fields are shown. The subject and content of all the emails are identical except at a few places that are highlighted in the figure. In our analysis, we found as many as 3,121 emails in the large 500K email collection that are generated using a similar template and are near duplicates of the examples given in the Figure 5.

[1]http://lucene.apache.org


A

```
Start Date: 1/2/02; HourAhead hour: 19;
Start Date: 1/2/02; HourAhead hour: 19; No ancillary schedules awarded. No variances detected. LOG MESSAGES: PARSING FILE -->> O:\Portland
\WestDesk\California Scheduling\ISO Final Schedules\2002010219.txt

Start Date: 1/2/02; HourAhead hour: 22;
Start Date: 1/2/02; HourAhead hour: 22; No ancillary schedules awarded. No variances detected. LOG MESSAGES: PARSING FILE -->> O:\Portland
\WestDesk\California Scheduling\ISO Final Schedules\2002010222.txt

Start Date: 10/7/01; HourAhead hour: 5;
Start Date: 10/7/01; HourAhead hour: 5; No ancillary schedules awarded. No variances detected. LOG MESSAGES: PARSING FILE -->> O:\Portland
\WestDesk\California Scheduling\ISO Final Schedules\2001100705.txt Error retrieving HourAhead price data - process continuing...

Start Date: 12/16/01; HourAhead hour: 11;
Start Date: 12/16/01; HourAhead hour: 11; No ancillary schedules awarded. No variances detected. LOG MESSAGES: PARSING FILE -->> O:\Portland
\WestDesk\California Scheduling\ISO Final Schedules\2001121611.txt ###Cannot locate a Preferred or Revised_Preferred Schedule that matches the
FINAL Individual Interchange Schedule. Unable to assign deal number.
```

B

```
Start Date: <DATE>; HourAhead hour: <HOUR>;
Start Date: <DATE> HourAhead hour: <HOUR>; No ancillary schedules awarded. No variances detected; LOG Messages; PARSING FILE
→> O:\Portland\WestDesk\California Scheduling \ISO Final Schedules\<FILENAME> <ERROR>
```

Figure 5: (A) Examples of automated messages from an email collection. (B) A possible template for generating these messages is shown in the rectangular box at the bottom.

To discover groups of automated emails from the groups of near duplicate emails we use a simple heuristic. We observe that each group of near duplicate emails that we discover falls in one of the following three categories:

- Group of automated messages: These emails are generated using the same template and are near duplicates of each other.
- Group of bulk emails: A bulk email is an email that is sent to several people. Therefore there are several copies of the same email in the collection. By definition these emails are exactly similar to each other and therefore also appear as a group in the groups of near duplicate emails.
- Group of forwarded emails: When an email is forwarded its content may differ from the content of the original email due to formatting and boundary markers inserted by the email client or additional text inserted by the user. Thus these emails are near duplicates of each other and appear in the same group. These groups also contain emails that are replied to with extremely short messages and therefore have significant overlap with the content of the original email.

To distinguish *automated message* groups from *bulk email* groups, we select only those groups which do not have exactly the same content for all of its emails. This can be checked by using the hash-value of all the emails in a group. To distinguish *automated message* groups from *forwarded email* groups, we select only those groups in which all the emails have only one segment. When an email is forwarded or replied the original content of the email is copied as a separate segment and thus these emails have more than one segment in them. Identification of segments in an email is described in Section 4.3.1 and Section 5.

4. SEMANTIC SIMILARITY

Semantically similar emails are identified by associating them to a set of concepts that are helpful in identifying responsive, privileged and unresponsive documents. The legal concepts can be broadly divided into two categories: focus and filter.

4.1 Focus Categories

As the name suggests, focus categories help to reduce the area of interest from a large pool of information. These categories identify mails that are relevant for a case. Focus category annotators identify emails that fall under this category. It is important to achieve high recall for focus category annotators since we would not want to miss out on relevant emails. Focus categories include the following: Legal Content, Financial Communication, Intellectual Property, Audit Info, Confidential Communication, and Privileged Communication.

4.2 Filter Categories

Filter categories identify emails that are irrelevant and that can be filtered out to reduce the effort of the manual review process. Filter category annotators identify emails matching the filter categories so that they can be filtered out. It is important to achieve high precision for filter category annotators to ensure that relevant emails are not filtered out. The filter categories includes following: Private Communication and Bulk email.

4.3 Implementation

We develop a set of annotators that automatically identify the concepts present in a document. The concept annotators can be built using either machine learning or rule based approaches. The advantage of the rule based approach is that the rules are human comprehensible and can be tweaked to get the desired results. To build rules for the concept annotators, we first had to come to a common understanding of what each concept means. This required consulting legal experts to understand what they mean by each of the concepts. We looked up the standard definitions of these concepts from the sources pointed to by the legal experts. The next step is to codify the definitions of the legal concepts into System T rules (System T is an information extraction system which uses AQL, declarative rule language with an SQL-like syntax)[19]. The main issues that we had to address were as follows:

4.3.1 Email Segmentation

Emails are divided into meta-data like sender and recipient information, and content fields like subject and body. The body of an email can be divided into different parts. For example, a typical email body contains a greeting (such as "Dear XXX", "Hi" or "Hello"), the main content, salutation at the end ("Thanks", "Regards", etc followed by a name). It can optionally include the signature of the sender and a footnote text. The footnote can include standard disclaimers such as "The content of this email is confidential and subject to attorney-client privilege". Additionally, many emails are either replies to previous email threads or contain forwarded emails. In this case, the email body includes the content of the email thread being replied to or forwarded. While identifying the concepts in an email, it is important to first segment the email body since the rules may be applicable to specific parts of the email. For example, to identify Privileged email, we need to identify the footnote and check if the footnote contains the privileged declaration. Segmenting the email body consists of two phases: 1. Splitting the email containing a thread of emails into individual email blocks, each corresponding to a single email. 2. For each block, identifying the various parts of the email such as the greeting, content, signature and footer. Splitting the email into blocks is done by identifying how most of the common email clients include the content of the email being forwarded or replied to into a new email. These patterns are then encoded using regular expressions in the rules. Identifying various parts of a block is done similarly by identifying patterns that are typically used in emails. For example, the footer is most often separated from the rest of the email by separator line such as "———" or "*********".

4.3.2 Identifying patterns for concepts

For each concept, we identified a set of keywords and phrases that are indicative of that concept. For example, keywords such as "patent", "copyright", "NDA", "tradesecret", "IP", "trademark" indicate that the email may be discussing about intellectual property.

We create a dictionary of such keywords and match it against the text using rules. Regular expressions are also used to identify patterns that are indicative of certain concepts. For example, mention of currency figures in an email can be used to identify Financial Communication. We used regular expressions to encode such expressions to identify a currency amount in the email content. For each concept, we wrote multiple rules to identify the basic building blocks and the relationships between them.

4.3.3 *Consolidation*

The rules developed for each concept were independent of each other. Thus, the same email can match rules for different concepts and can be tagged with multiple concepts. In general, such a situation can happen in reality and is not a problem. For example, an email could contain mentions of Financial information as well as Legal content. However, for specific categories, we may have some constraints that preclude the same email from belonging to multiple categories. This could be either due to a "implies" or a "contradicts" relationship between the categories. The relationships for the categories we developed are listed below:

Category 1	Relationship	Category 2
Privileged	Implies	Legal Content
Intellectual Property	Implies	Legal Content
Privileged	Implies	Confidential
Bulk Email	Contradicts	Confidential

To handle these constraints, we make a pass after the concepts have been identified and eliminate redundant (implied) or contradicting concepts.

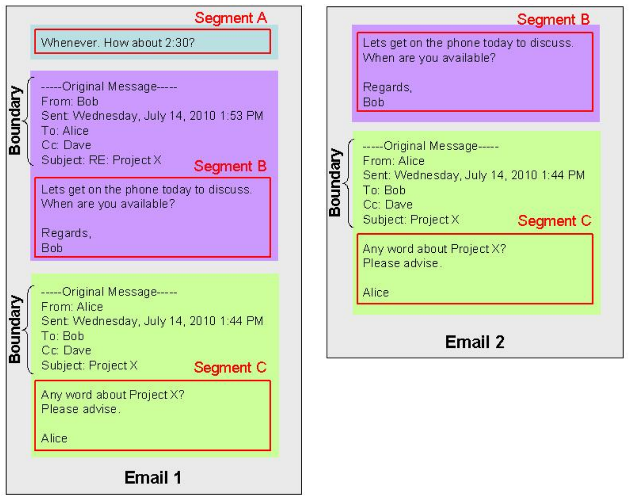
5. EMAIL THREAD DETECTION

As mentioned in Section 1, the goal of email thread detection is to find and organize messages that should be grouped together based on reply and forwarding relationships. In this section, we present a brief overview of related work and then discuss our approach to email thread detection.

5.1 Related Work

Existing methods for detecting email threads can be separated into two groups: 1) *metadata based*, and 2) *content based*. Metadata based approaches make use of email header fields, such as the *In-Reply-To* and *References* fields defined in RFC 2822 [20] to reconstruct email threads [25]. When an email client creates a new reply message, the *In-Reply-To* field should be populated with the *Message-ID* of the parent message and the *References* field should be populated with the contents of the parent's *References* field followed by the parent's *Message-ID*. As a result, the *In-Reply-To* field of an email could be used to determine its parent-child relationship and the *References* field could be used to determine all the emails in the same thread. However, since these header fields are optional for email clients, they are not always available. This is especially true in large archived email collections that span a variety of email clients. In general, we cannot rely on the availability and consistency of these header fields for email thread detection.

There are a number of approaches to email thread detection that make use of the actual email content. Some of these content based approaches look only at specific fields. For example, Wu and Oard ([23]) links emails together that have the same normalized subject field values. Klimt and Yang ([15]) additionally examines the sender and recipient fields and only groups messages into the same thread if they have the same normalized subjects and have at least one sender/recipient in common. Other approaches look at the main content of the emails. For example, Lewis and Knowles ([16]) determines if a message is a response to another message by searching for the quoted part of the message in the unquoted parts of the target



Whenever. How about 2:30?

-----Original Message-----
From: Bob
Sent: Wednesday, July 14, 2010 1:53 PM
To: Alice
Cc: Dave
Subject: RE: Project X

Lets get on the phone today to discuss.
When are you available?

Regards,
Bob

-----Original Message-----
From: Alice
Sent: Wednesday, July 14, 2010 1:44 PM
To: Bob
Cc: Dave
Subject: Project X

Any word about Project X?
Please advise.

Alice

Lets get on the phone today to discuss.
When are you available?

Regards,
Bob

-----Original Message-----
From: Alice
Sent: Wednesday, July 14, 2010 1:44 PM
To: Bob
Cc: Dave
Subject: Project X

Any word about Project X?
Please advise.

Alice

Figure 6: Example emails with segmentation boundaries and email segments.

messages. To efficiently perform the search, they use an information retrieval approach by indexing the messages and then creating a query using the quoted part of the message to search the index. Yeh and Harnly ([24]) use a similar technique but add additional constraints on the subject, date, and sender/recipient fields. They preprocess the email using a set of regular expressions to separate the reply and quotation parts of the email and use a unigram overlap metric to measure similarity between any two messages.

5.2 Our Approach

Typically when replying to an email, the contents of the old email exist in the newly composed reply email as a separate segment. Thus a segment in an email captures one communication turn and an email may contain several segments if it is part of a long running email conversation. The segments associated with an email can be used to compute a *containment* relationship between pairs of emails. An email e_i is said to contain another email e_j, if e_i *approximately* contains all the segments of e_j. Figure 6 shows two example emails where the first email contains the second email. This is true because all the segments that appear in the second email are also present in the first email. These segments may appear with slight variations across different emails. The variations may be caused by different email clients inserting some formatting syntax. The segments may also change because a user might insert some additional text in the old segments as she responds to an email. A thread can be constructed by first discovering an email that contains only a single segment. Other emails in a thread can then be determined based on containment relationships with the existing emails in the thread.

Our approach to email thread detection is a content based approach that relies on the availability of the email history and leverages near duplicate detection and information retrieval techniques. It consists of four stages: 1) email segmentation, 2) near duplicate signature computation of the email segments, 3) segment signature indexing, and 4) runtime thread group identification and thread structure generation.

First, we attempt to identify all the segments present in an email. To do this, we manually defined rules to identify email segment boundaries using the IBM System T information extraction system [19]. We used a set of several hundred manually marked reference emails spanning a variety of email clients and languages to guide our rule writing efforts. We use SystemT because it allows us to write complex rule sets that can be efficiently executed as an annotator within the context of a UIMA processing pipeline[2]. Once the boundaries in an email are identified, we take the content in between the boundaries as the segments of the email. Figure 6 illustrates the email segments generated by this first processing stage.

Next, we generate a *segment signature* for each segment in an email using the near duplicate detection technique described in Section 3. The segment signatures represent the content of the email segments for the purpose of email threading and are used to determine the *containment* relationship between a pair of emails. The use of near duplicate signatures allows us to handle small variations in the segment content.

To support efficient runtime email threading, we adopt an information retrieval approach and index the segment signatures into the search index similar to what is done for near duplicate document detection. This indexing allows fast retrieval of the segment signatures associated with a specific email message and also allows fast identification of the email messages that have a specific segment signature.

Finally, at runtime, we make use of the email segment signature index to perform thread group identification and thread structure generation. To identify the thread group for a selected email message, we first retrieve all the segment signatures associated with that email. Next, we determine the *root segment* which should correspond to the initial email in the thread group. If the email has only one segment, then that segment becomes the root segment. If the email has multiple segments, we select the last non-empty email segment as the root segment. We then create a query consisting of the root segment signature and search for all messages that contain the same root segment signature. The resulting message set should contain all the emails in the same thread. At this point we have enough information to get a count of the number of messages in the thread and a list of the emails in the thread. Additional processing can then be performed to organize the emails in the thread into a hierarchical tree structure based on containment relationships. To do this, we first find an email in the group that contains only a single segment and assign that email as the root node of the thread hierarchy. Next, we find emails in the group with two segments that contain the first segment and make them child nodes of the root node. Then we find emails in the group with three segments and make each a child of a two segment node that is contained by the three segment email. We continue this processing until all emails in the thread group have been linked into the thread hierarchy.

This runtime processing is illustrated in Figure 7. Let's assume that the selected email is *Email1*. This email has segment signatures *A*, *B*, and *C* with the root segment signature being identified as *A*. Searching for emails with root segment signature *A* returns: *Email1*, *Email2*, *Email3*, *Email4*, and *Email5*. This set of five emails belongs to the same thread group. *Email4* is identified as the root node since it has only one segment and the constructed thread hierarchy based on containment relationships is shown in Figure 7.

6. EXPERIMENTS

In this section, we present experimental evaluations of the proposed techniques and show their effectiveness for the e-discovery review process. We investigate the performance of automated message detection, semantic grouping, and email thread detection. Experiments evaluating the performance of the underlying near duplicate document detection techniques are described in Appendix A. For most of our experiments, we use a large real-world email corpus that contains over 500,000 emails from about 150 users [15]. We also present the results of a small user evaluation on reviewing documents with and without syntactic, semantic, and thread grouping in Appendix B.

[2]http://incubator.apache.org/uima

Email Id	Segment Signatures
Email1	A, B, C
Email2	A, B, C, D
Email3	A, D, E
Email4	A
Email5	A, B
Email6	X, Y
Email7	X, A

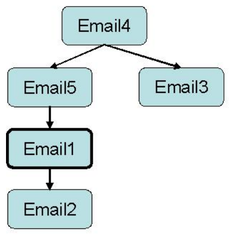


Figure 7: Example emails and the resulting thread hierarchy.

6.1 Automated Message Detection

To evaluate our method for detecting groups of automated messages, we manually selected a set of 641 emails from the large email set which are generated using 6 different templates. Therefore, these emails can be grouped into 6 clusters corresponding to the different templates. We refer to this clustering as the *reference clustering*.

We generate 10 near duplicate signatures for each email and index them as described in Figure 3. We then create groups of automated messages by clustering this dataset with varying similarity threshold values using the procedure described in Figure 4.

For each cluster, we calculate its precision by looking at each pair of documents that appears in it and checking whether the pair appears in the same cluster or not in the *reference clustering*. If a cluster contains only a single document then its precision is defined to be 1. We present the results in Table 1. For each similarity threshold value, we give the number of clusters obtained and the average precision.

Threshold (%)	No. of Clusters	Avg. Precision
90	110	1.0
70	18	1.0
50	6	1.0
30	6	0.9
10	3	0.4

Table 1: Automated message detection precision.

We observe that on this data set, our proposed method of clustering is able to achieve 100% precision when using a similarity threshold of 50% or higher. The precision drops off quickly after that. With a similarity value of 50%, we are able to discover the *reference clustering* exactly. For higher values of the similarity threshold, we observe that a single cluster in the *reference clustering* can get fragmented into many clusters. Therefore, we get increasing numbers of clusters with higher values of the similarity threshold. As a result, using a high similarity threshold value may not be advantageous for the manual review process as an analyst would need to review a larger number of clusters. From this experiment it seems that it is better to use a medium similarity threshold value in order to get a smaller number of clusters while maintaining a high level of precision. We should note that 100% precision may not be possible in all cases and results would vary based on the type and actual content of the data.

6.2 Semantic Grouping

For evaluating the quality of semantic groups, we sampled 2200 emails from the large email collection using a set of generic keywords and then manually labeled them in one or more of the following five categories: (1) Confidential, (2) Financial Communication, (3) Intellectual Property, (4) Legal Content, and (5) Privileged Communication. The manual creation of this dataset provided us insights into the concepts and helped us in building the rules. We therefore refer to this dataset as "train dataset" in this section. In order to check completeness of our rules we also asked two other candidates who were not involved in the building of rule based annotators to create a dataset which we refer to as "test dataset". Table 2 provides details of "train" and "test" datasets.

Class	Train	Test
Confidential	406	21
Finance	1215	54
Intellectual Property	153	22
Legal Content	553	41
Privileged	433	34

Table 2: Data set used for semantic grouping experiments.

In order to see how our rule based approach compares with the machine learning based approach, we pose the annotation problem as a classification problem. We use naïve Bayes classifier and build a two class classifier for each class. We use the rainbow toolkit implementation for naïve Bayes [17]. For Rule based annotators we evaluate on "train dataset" as well as for "test dataset". For naïve Bayes classifier we use "train dataset" for training the classifier and "test dataset" for testing the classifier. For each class we build a two class classifier by creating a class called "Others" by combining emails from all the other classes. Table 3 presents the precision and recall numbers for rule based (RB) approach as well as for the naïve Bayes classifier.

Class	RB (Train)		RB (Test)		Naïve Bayes	
	P	R	P	R	P	R
Confidential	77	87	100	57	22	71
Finance	91	92	52	92	63	94
Intellectual Property	83	92	75	100	79	46
Legal Content	83	82	58	95	34	51
Privileged	87	97	89	94	52	91

Table 3: Semantic grouping precision and recall.

The performance of rule based annotators on the "train dataset" is better than the performance on the "test dataset." The table also illustrates that the performance of rule based annotators is always better than the one achieved by naïve Bayes classifier. Naïve Bayes performs particularly badly for Confidential and Legal Content classes. This is due to the consolidation rules. Many emails that have terms indicating confidentiality are also privileged. Since the Privileged category is given a higher priority, these emails are not included in the Confidential class and are included in the Other class of Confidential. This leads to a poor classifier for Confidential since the discriminating terms for the Confidential class occur in both the Confidential and the Other class used while training the classifier. Similar reasoning holds for the Legal Content class since it is also given a lower priority by the consolidation rules.

6.3 Email Thread Detection

Email thread detection performance is quantified in terms of precision (P) and recall (R) which are defined as follows: $P = \frac{C}{N}$ and $R = \frac{C}{D}$ where C is the number of correctly identified near duplicate documents, N is the total number of near duplicates identified, and D is the actual number of near duplicates. A single number performance measure that is also commonly used is the F1-measure which is defined as the harmonic mean of precision and recall:

$$F1 = \frac{2 * P * R}{P + R} \qquad (2)$$

In order to compute the precision, recall, and F1 performance measures, we need to have a collection with labeled email thread information. We need to know which thread group each email belongs to. Unfortunately, the large email data set we had did not have this thread information available. But we could generate an approximation to this "ground truth" thread information using the following procedure.

One simple way to identify emails that are likely part of the same conversation or email thread is to compare the subject field in the emails. In a typical email conversation, when a user replies or forwards emails, email systems prefix the original subject with "Re:", "Fwd:" etc. If we normalize the content of the subject field to remove such prefixes, and then retrieve a set of emails that have the same normalized subject field, we get a set of emails that are potentially part of the same "thread." Of course, it is possible for a sender to change the subject entry. And of course, for very generic subjects such as "Hello" or "Lunch?" the approximation does not work very well - but we think that for most cases it is useful.

For a given email, we queried the collection and retrieved email threads using our method described in Section 5 as well as the heuristic approach based on the normalized subject field. Let the set of emails returned, as part of the thread, using our method be T_1 and those returned by the heuristic normalized subject approach be T_2. We then manually verified the emails contained in both email thread sets T_1 and T_2. Let the set of correct emails (emails that are actually part of the thread) in T_1 be C_1 and let C_2 be the set of correct emails in T_2. Then the set of correct emails T is :

$$T = C_1 \cup C_2 \qquad (3)$$

Thus, T approximates the ground truth for email threads - i.e., the correct set of emails that should be part of an email thread for a given email. This T can then be used to determine the *number of correctly identified emails in the thread* and the *actual total number of emails in the thread* values needed for computing the precision and recall performance measures.

To evaluate the performance of our email thread detection method, we randomly selected 100 emails from the large email set and used our system to retrieve email threads for each of the 100 emails. We also retrieved email threads using the normalized subject method. The results are shown in Figure 8. The average thread depth, i.e., number emails in a given thread, is 3.13 .

In order to study any impact of our approximation of ground truth of email threads, we used a separate small email data set which had thread information available. This data set has 1000 emails and contains approximately 100 email threads with an average thread depth of 6.23 . We randomly selected 50 emails and used

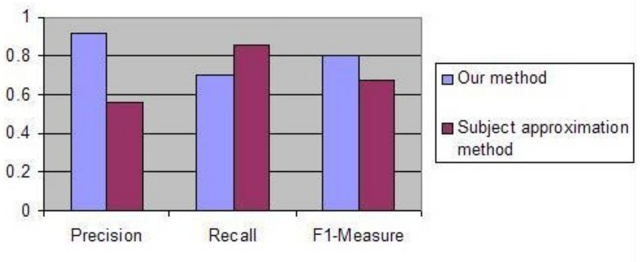


Figure 8: Results using the large email collection.

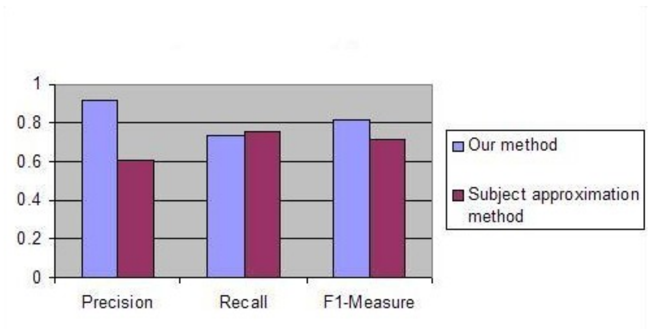


Figure 9: Results using the small email collection.

our system to retrieve email threads for each of the 50 emails. We also retrieved email threads using the normalized subject method. The results are shown in Figure 9.

We notice that the results are comparable for both data sets. In both cases we find that, while the subject normalization approach shows slightly higher recall, our method shows much higher precision. The F1-measure for our email thread detection approach is approximately 0.8 which is higher than the 0.7 value for the subject normalization approach. The precision for our approach is 0.91 which is significantly higher than the precision of 0.62 value for the subject normalization approach.

7. CONCLUSIONS AND FUTURE WORK

We presented approaches to detect syntactic groups, semantic groups and email threads that leverage near duplicate detection and information retrieval techniques. The detection of automated messages (including near-duplicates) and email threads allows multiple emails to be reviewed, categorized, and processed together in the appropriate context which can help lead to more efficient and consistent reviews. We also showed the potential of large savings in the review time by evaluating our methods on a large email collection. Many of these approaches have been integrated into the IBM eDiscovery Analyzer product offering and is described in Appendix C.

There are a number of possible areas for future work. One area is in developing improved user interactions. For example, providing an easy way for the user to see the differences between emails that have been grouped together as automated messages, near duplicates, or email threads could be a very useful and time-saving feature. A second area of potential work is to productize some of the advanced capabilities such as categorizing near duplicate message groups (synactic groups) into *automated messages*, *bulk emails*, and *forwarded emails* and organizing the emails in a thread group into a tree structure.

Acknowledgement

The authors would like to thank Sandeep Singh for some useful early discussions and helping in the implementation of certain aspects of the near duplicate detection technique during his tenure at IBM India Pvt. Ltd.

8. REFERENCES

[1] J. R. Baron. Toward a federal benchmarking standard for evaluating information retrieval products used in e-discovery. *6th Sedona Conference Journal*, pages 237–246, 2005.

[2] Pavel Berkhin. Survey of clustering data mining techniques. 2002.

[3] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. *Proceedings of the Special Interest Group on Management of Data*, pages 298–309, May 1995.

[4] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[5] M. S. Charikar. Similarity estimation techniques from rounding algorithms. *34th Annual ACM Symposium on Theory of Computing*, pages 380–388, May 2002.

[6] A. Chowdhury., O. Frieder, D. Grossman, and M. McCabe. Collection statistics for fast duplicate document detection. *ACM Transactions on Information Systems (TOIS)*, 20:171–191, 2002.

[7] J. Conrad and C. P. Schriber. Online duplicate document detection: Signature reliability in a dynamic retrieval environment. *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, pages 443–452, 2003.

[8] J.G. Conrad. E-discovery revisited: A broader perspective for ir researchers. *DESI: Workshop on Supporting Search and Sensemaking for Electronically Stored Information in Discovery Proceedings*, pages 237–246, 2007.

[9] *EDRM - The Electronic Discovery Reference Model*, 2009. http://edrm.net/.

[10] D. Fetterly, M. Manasse, and M. Najork. On the evolution of clusters of near-duplicate web pages. 1^{st} *Latin American Web Congress*, pages 37–45, 2003.

[11] FIPS. Secure hash standard. Technical Report FIPS PUB 180-1, Federal Information Processing Standards Publication, 1995.

[12] *FRCP - Federal Rules of Civil Procedure*, 2007. http://www.law.cornell.edu/rules/frcp.

[13] Monika Henzinger. Finding near-duplicate web pages: A large-scale evaluation of algorithms. *Proceedings of SIGIR*, pages 284–291, August 2006.

[14] T. Hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. *Journal of the American Society or Information Science and Technology*, 54:203–215, 2003.

[15] Bryan Klimt and Yiming Yang. The Enron Corpus:. *Proceedings of ECML*, pages 217–226, 2004.

[16] D.D. Lewis and K.A. Knowles. Threading electronic mail: A preliminary study. *Information Processing and Management*, 33(2):209–217, 1997.

[17] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[18] B. Murphy. *Special Issue: Readings on Case-based reasoning: Believe It - eDiscovery Technology Spending To Top $4.8 Billion By 2011*. http://forrester.com/Research/Document/Excerpt/ 0,7211,40619,00.html.

[19] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. An algebraic approach to rule-based information extraction. *Proceedings of ICDE*, pages 933–942, 2008.

[20] *RFC 2822 Internet Message Format.*, 2001. http://www.faqs.org/rfcs/rfc2822.html.

[21] George Scoha and Tom Gelbmann. The 2006 Scoha-Gelbmann electronic discovery survey report. *Socha Consulting*, 2007.

[22] Martin Theobald, Jonathan Siddharth, and Andreas Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. *Proceedings of SIGIR*, pages 563–570, 2008.

[23] Y. Wu and D. Oard. Indexing emails and email threads for retrieval. *Proceedings of SIGIR*, pages 665 – 666, 2005.

[24] Jen Yuan Yeh and Aaron Harnly. Email thread reassembly using similarity matching. *Proceedings of CEAS*, 2006.

[25] J. Zawinski. *Message Threading*, 2002. http://www.jwz.org/doc/threading.htm.

APPENDIX

Appendix A describes the experiments for the evaluation of the near duplicate detection technique used in the paper for grouping emails. Appendix B presents a small user study that shows how auto-grouping of emails can significantly reduce the time needed for document reviews." Finally in Appendix C, we describe the integration of many of these features into IBMs eDiscovery Analyzer product offering.

A. NEAR DUPLICATE DETECTION

In this experiment, we measure the quality of the near duplicate emails detected by our proposed method. Similar to email thread detection, near duplicate document detection performance is quantified in terms of precision and recall. Since we did not have a list of all the near duplicate documents for a given document in the large email set we used, it will have been very difficult to quantify the recall of our method. As a result, in this experiment we only focused on the precision of the returned set of near duplicates.

For quantifying precision, we manually reviewed sets of near duplicate emails returned in response to 500 queries. The queries were constructed using a randomly selected set of 15 emails. We measured precision as a function of two variables: 1) the number of bits used to store the signatures and 2) the level of similarity threshold used for search. We experimented with a minimum of 6 bits to a maximum of 14 bits for storing signatures. In all cases, 10 signatures are generated for each email. For the similarity threshold, we experimented with a maximum of 100% similarity to a minimum of 50% similarity. Note that the similarity threshold values are translated into a minimum number of signatures that needs to be shared by a document for it to be considered a near duplicate of the given document. Note that using a similarity threshold level of 100% is not the same as finding exact duplicates using a hashing function such as MD5. Emails may have exactly the same signatures (e.g., 100% similarity) even if their contents differ slightly whereas their MD5 values will be completely different. Figure 10 shows the precision for various configurations. We observe that the precision drops drastically when we use only 6 bits for each signature. We observe, in this particular case, a precision of 100% even with a low similarity threshold of 50% when we use more bits (10, 12 and 14 bits) for the signatures. It is, of course, not clear that 100% precision would be possible in all cases. Results would vary based on the type and actual content of emails. These results support the observation made in Section 3 that lower values for bits per signature and lower levels for similarity threshold have a higher probability of false positives.

Since it is important to use more bits to represent the signatures, we next examined the impact of the signature bit size on the size of the index. For this experiment, we generated ten signatures for each email and varied the range of each signature from 6 bits to 14 bits with an interval of 2. In our index, we do not have a provision for storing integers and rather store them as strings. Figure 11 shows the increase in index size as a function of the number of bits used for each signature. Overall, the storage requirement for signatures is very modest. Even after storing ten signatures of 14 bits for every document, the index size increases by less than 1.9%.

In our next experiment, we investigated how much near duplicate document detection can help in the review process. As pointed out

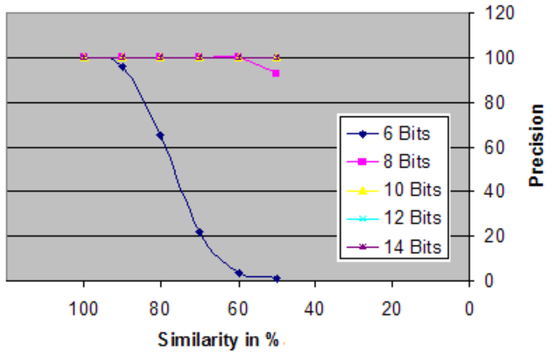


Figure 10: Precision of results obtained for several emails with different size signatures and similarity threshold.

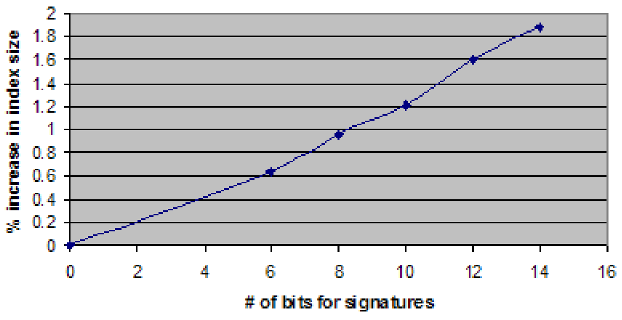


Figure 11: Increase in index size for storing signatures.

earlier, since we cannot measure the recall of our method (due to unavailable reference assessments), we instead computed the number of near duplicate emails that can potentially be determined using our method with a precision of 100%. We are able to do this since we manually reviewed the near duplicate email sets returned by the system. We create sets of documents such that all the near duplicate emails are grouped together. In this test, we used 14 bits per signature and generated near duplicate sets with varying levels of similarity threshold ranging from 100% to 50%. Since most of the content of the emails in a set is the same, all emails in the set can be reviewed together which can potentially save a lot of review time. Figure 12 shows the number of near duplicate sets obtained in the large email collection with varying levels of the similarity threshold. The graph shows that only 33.4% of the emails are unique at a similarity threshold of 50%. Therefore, more than 66% of the emails are either exact or near duplicate messages. The results suggest that identifying exact or near duplicate emails or messages and reviewing them together as a group can lead to significantly reduced review times.

The large email set contains many exact duplicates. In our next experiment, we quantified the number of exact duplicate documents so we can get a clearer picture of how many additional documents are detected by the near duplicate method. To detect exact duplicates, we used the MD5 hash algorithm on the email contents. Figure 13 shows the number of emails with a certain number of copies in the large email set. Note that the vertical axis in the graph is in Log scale. This is a power distribution showing a large number of emails with a small number of copies and a small number of emails with very large numbers of copies. We found 116,728 emails that had no copies. At the other extreme, we found one email that has as many as 112 exact duplicates in the set. Overall we found 51.8%

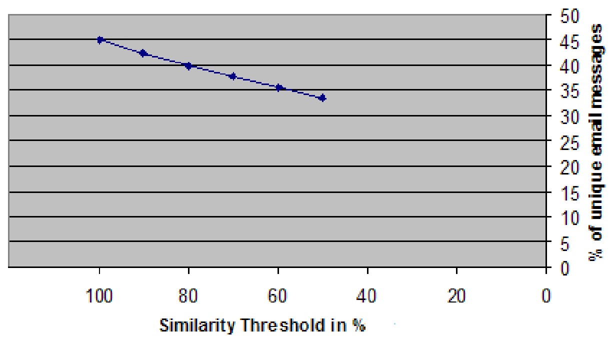


Figure 12: Decrease in number of sets that need to be reviewed using near duplicate detection.

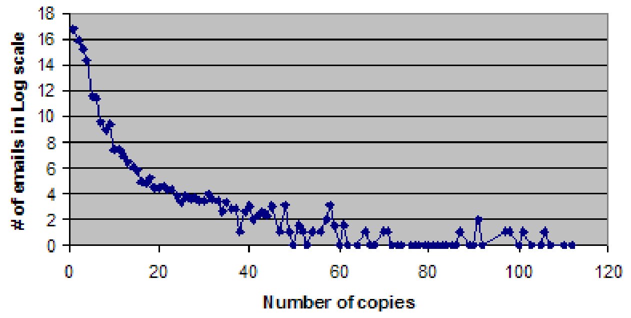


Figure 13: Distribution of exact duplicate emails.

of the email messages to be unique. Since the near duplicate detection processing found only 33.4% of the emails to be unique (at a similarity threshold of 50%), this means that the near duplicate processing is able to account for an additional 18% of the documents over exact duplicate processing. Even with a similarity threshold of 100%, the near duplicate processing is able to account for an additional 6.9% (51.8 - 44.9) of the documents over exact duplicate processing.

B. USER EVALUATION

In order to understand the benefits obtained by the group level view of documents for presentation in the e-discovery review process, we created a dataset consisting of around 150 emails. Of these 150 emails, 32 emails were picked up from the subset[3] of Enron corpus which was manually annotated by a group of people at university of California, Berkley and are related to company policies. We mixed around 120 emails with them which were not related to company policies. To showcase the benefits obtained form the group level review, we purposefully picked up emails that could be clustered in a few number of groups. We then asked 3 analysts to independently review the dataset and find all the emails related to company policies. All the analysts were asked to review the dataset twice, once without the group level view and again with group level view of the dataset. The group level view had 28 different groups to be reviewed. Table 4 presents the time taken and accuracies obtained for both the cases.

As expected, there is a significant decrease in the time taken to review the dataset with the group level view. This is true for this dataset as there were several emails that could be grouped. The

[3]http://sgi.nu/enron/use.php

User	Time (no groups)	Accuracy	Time (groups)	Accuracy
User 1	33 mins	87.5%	6 mins	100%
User 2	45 mins	93.7%	8 mins	100%
User 3	42 mins	100%	8 mins	100%

Table 4: Results of user study on grouping effectiveness.

real gains obtained from the grouping will heavily depend on the existence of such emails in a dataset. More interestingly the table shows that 2 of the 3 analysts miss to flag some emails when they review emails without group level view leading to a lower accuracy of their review. This signifies that we get improved consistency in the review process with the group level treatment of documents.

C. INTEGRATION INTO THE IBM EDISCOVERY ANALYZER

The IBM eDiscovery suite is a set of tools that can be used to collect, organize, manage, and retrieve relevant information from multiple sources and vendor repositories as an aid during litigation. The IBM eDiscovery suite [4] consists of:

- IBM ContentCollector (ICC) for collecting enterprise ESI from multiple sources and vendor repositories
- IBM P8 and CM8 content management repositories for managing and archiving the collected data,
- IBM eDiscovery Manager (eDM) for searching, culling, holding, and exporting case-relevant ESI , and
- IBM eDiscovery Analyzer (eDA) for providing conceptual search, analysis, and organization of ESI data

IBM eDiscovery Analyzer is the e-discovery review tool that can be used by legal professionals and litigation support specialists to help them quickly to refine, analyze and prioritize case-related ESI after it has been collected in content management repositories using ICC and processed and culled using eDM. The advanced search, navigation, and reporting features in eDA can be used to identify, flag, and organize the documents. These tools can help the users to identify relevant (non-responsive and responsive) documents. Documents can be flagged for further review.

The eDA product provides a good functional framework into which the near duplicate, automated message, semantic grouping, and email thread detection technologies can be integrated to provide additional and relevant value.

C.1 Syntactic Groups

There are two primary scenarios related to this goal that we attempt to address with our eDA product. In the first scenario, the user would like to have the system detect near-duplicates automatically and alert him about the existence and number of such documents. In the second scenario, the user is looking at a specific document (via a document preview functionality) and would like to run a search to help her identify any near-duplicates of that document. To support the first scenario, a link is placed in the search results row for documents where the system was able to automatically detect one or more near-duplicates with a very high similarity threshold as shown in Figure 14-A.

The link shows the number of such near-duplicates and says "View all N near duplicates of this document" where N is the number of near-duplicates. Results without any detected near-duplicates will not have this link. Clicking the link will run a search for

[4]http://www-01.ibm.com/software/data/content-management/ediscovery.html

the near-duplicates and show them in-lined in an expanded result row as illustrated in Figure 14-B. At this point, the link changes to “Hide all N near duplicates of this document”. Clicking the link again will collapse the result row and revert the query. To flag all of the identified near-duplicate documents, the user can check the “Select all’ box (Figure 14-C) and then click the flag button (Figure 14-D) to bring up a flagging dialog to select the flags to apply to the set of near-duplicate documents.

To support the second scenario, a new button has been added to the document preview screen as shown in Figure 15-A. When clicked, it brings up the near-duplicate search dialog shown in Figure 15-B. This dialog allows the user to specify the near-duplicate match threshold (to return more or less documents) and issue a new search to look for identified near-duplicates of the document currently being viewed.

C.2 Semantic Groups

The semantic groups discovered are shown in the IBM eDiscovery Analyzer UI as a panel on the screen (shown in Figure 16). The groups names can be clicked to filter the current search result set to only show the emails belonging to that group. Multiple groups can be selected to perform the results filtering.

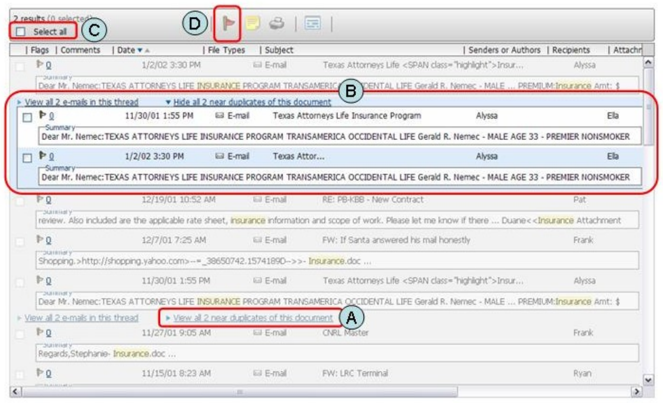
C.3 Email Thread Detection

The contents of a case will typically contain email messages that belong to the same thread. Similar to the case of near-duplicate documents, it would be more efficient for the analyst to handle, view, and flag these threaded email messages together as a group in the appropriate context.

The primary use case that we attempt to address in eDA is to allow the user to use eDA to detect threaded email messages and alert him about the existence and number of such messages. In addition, eDA will allow the user to look at that subset of messages.

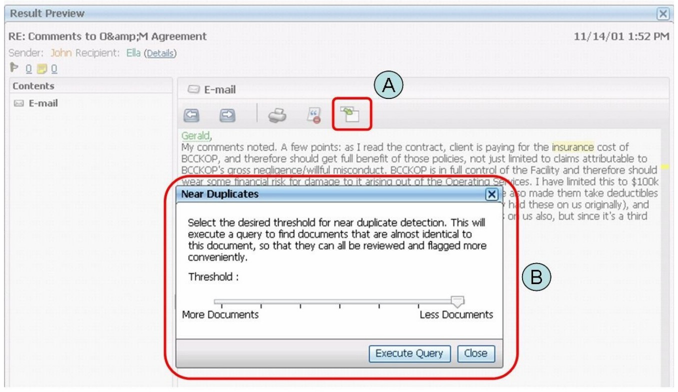
A link is placed in the search results row for documents where the system was able to automatically detect one or more email messages in the same thread as shown in Figure 17. The link shows the number of threaded messages and says “View all N e-mails in this thread” where N is the number of messages. Results not belonging to any detected email threads will not have this link.

Clicking the link will run a search for the messages in the thread and show them in-lined in an expanded result row as illustrated in Figure 17-A. The emails are sorted by date so that the initial email in the thread is listed first.



| Flags | Comments | Date | File Types | Subject | Senders or Authors | Recipients | Attachr
0 1/2/02 3:30 PM E-mail Texas Attorneys Life <SPAN class="highlight">Insur... Alyssa
Summary: Dear Mr. Nemec:TEXAS ATTORNEYS LIFE INSURANCE PROGRAM TRANSAMERICA OCCIDENTAL LIFE Gerald R. Nemec - MALE ... PREMIUM:Insurance Amt: $
View all 2 e-mails in this thread | Hide all 2 near duplicates of this document
0 11/30/01 1:55 PM E-mail Texas Attorneys Life Insurance Program Alyssa Ella
Summary: Dear Mr. Nemec:TEXAS ATTORNEYS LIFE INSURANCE PROGRAM TRANSAMERICA OCCIDENTAL LIFE Gerald R. Nemec - MALE AGE 33 - PREMIER NONSMOKER
0 1/2/02 3:30 PM E-mail Texas Attor... Alyssa Ella
Summary: Dear Mr. Nemec:TEXAS ATTORNEYS LIFE INSURANCE PROGRAM TRANSAMERICA OCCIDENTAL LIFE Gerald R. Nemec - MALE AGE 33 - PREMIER NONSMOKER
0 12/19/01 10:52 AM E-mail RE: PB-KBB - New Contract Pat
Summary: review. Also included are the applicable rate sheet, insurance information and scope of work. Please let me know if there ... Duane<<Insurance Attachment
0 12/7/01 7:25 AM E-mail FW: If Santa answered his mail honestly Frank
Summary: Shopping.>http://shopping.yahoo.com>--=_38650742.1574189D-->>- Insurance.doc ...
0 11/30/01 1:55 PM E-mail Texas Attorneys Life <SPAN class="highlight">Insur... Alyssa
Summary: Dear Mr. Nemec:TEXAS ATTORNEYS LIFE INSURANCE PROGRAM TRANSAMERICA OCCIDENTAL LIFE Gerald R. Nemec - MALE ... PREMIUM:Insurance Amt: $
View all 2 e-mails in this thread | View all 2 near duplicates of this document
0 11/27/01 9:05 AM E-mail CNRL Master Frank
Summary: Regards,Stephanie- Insurance.doc ...
0 11/15/01 8:23 AM E-mail FW: LRC Terminal Ryan

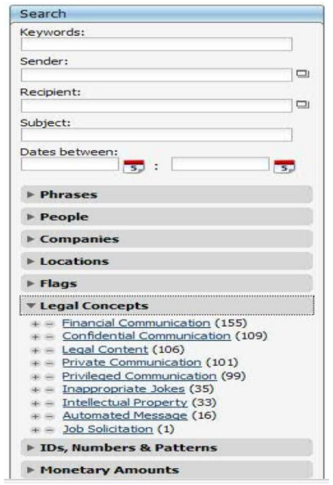
Figure 14: (A) Near-duplicate document link. (B) Expanded result row showing the near-duplicates for this document. (C) “Select all” documents check box. (D) Flag documents button.



RE: Comments to O&amp;M Agreement 11/14/01 1:52 PM
Sender: John Recipient: Ella (Details)
E-mail
Gerald,
My comments noted. A few points: as I read the contract, client is paying for the insurance cost of BCCKOP, and therefore should get full benefit of those policies, not just limited to claims attributable to BCCKOP's gross negligence/willful misconduct. BCCKOP is in full control of the Facility and therefore should wear some financial risk for damage to it arising out of the Operating Services. I have limited this to $100k

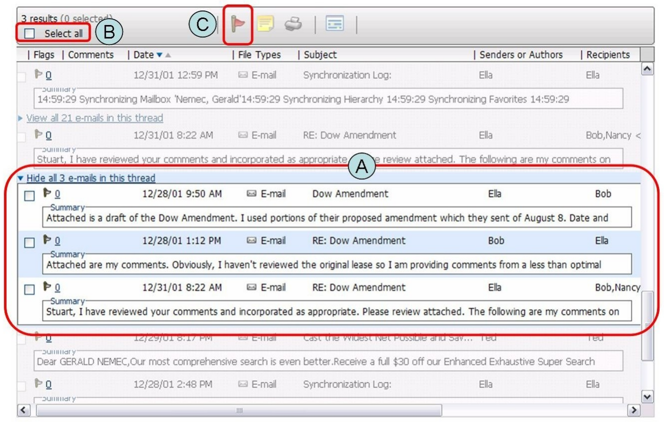
Near Duplicates
Select the desired threshold for near duplicate detection. This will execute a query to find documents that are almost identical to this document, so that they can all be reviewed and flagged more conveniently.
Threshold :

Figure 15: (A) Near-duplicate search button in the document preview screen. (B) Near-duplicate search dialog.



Financial Communication (155)
Confidential Communication (109)
Legal Content (106)
Private Communication (101)
Privileged Communication (99)
Inappropriate Jokes (35)
Intellectual Property (33)
Automated Message (16)
Job Solicitation (1)

Figure 16: Panel showing semantic groups.



| Flags | Comments | Date | File Types | Subject | Senders or Authors | Recipients
0 12/31/01 12:59 PM E-mail Synchronization Log: Ella Ella
Summary: 14:59:29 Synchronizing Mailbox 'Nemec, Gerald'14:59:29 Synchronizing Hierarchy 14:59:29 Synchronizing Favorites 14:59:29
View all 21 e-mails in this thread
0 12/31/01 8:22 AM E-mail RE: Dow Amendment Ella Bob,Nancy <
Summary: Stuart, I have reviewed your comments and incorporated as appropriate. Please review attached. The following are my comments on
Hide all 3 e-mails in this thread
0 12/28/01 9:50 AM E-mail Dow Amendment Ella Bob
Summary: Attached is a draft of the Dow Amendment. I used portions of their proposed amendment which they sent of August 8. Date and
0 12/28/01 1:12 PM E-mail RE: Dow Amendment Bob Ella
Summary: Attached are my comments. Obviously, I haven't reviewed the original lease so I am providing comments from a less than optimal
0 12/31/01 8:22 AM E-mail RE: Dow Amendment Ella Bob,Nancy
Summary: Stuart, I have reviewed your comments and incorporated as appropriate. Please review attached. The following are my comments on
0 12/29/01 8:17 PM E-mail Cast the Widest Net Possible and Sav... Ted Ted
Summary: Dear GERALD NEMEC,Our most comprehensive search is even better.Receive a full $30 off our Enhanced Exhaustive Super Search
0 12/28/01 2:48 PM E-mail Synchronization Log: Ella Ella

Figure 17: (A) Expanded result row showing the messages from the same thread as this document. To flag all of the threaded messages, the user can check the “Select all’ box (B) and then click the flag button (C) to bring up a document flagging dialog.