

# Near-Duplicate Detection for eRulemaking

Hui Yang, Jamie Callan  
Language Technology Institute  
Carnegie Mellon University  
Pittsburgh, PA, 15213, USA  
{huiyang, callan}@cs.cmu.edu

## ABSTRACT

U.S. regulatory agencies are required to solicit, consider, and respond to public comments before issuing regulations. In recent years, agencies have begun to accept comments via both email and Web forms. The transition from paper to electronic comments makes it much easier for individuals to customize “form” letters, which they do, creating “near-duplicate” comments that express the same viewpoint in slightly different languages. This paper explores the use of simple text clustering and retrieval algorithms for identifying near-duplicate public comments. Experiments with public comments about a recent regulation proposed by the Environmental Protection Agency (EPA) demonstrate the effectiveness of the algorithms.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval] Clustering, Query formulation, Retrieval models, Search process

## General Terms

Management, Measurement, Performance, Experimentation

## Keywords

Near Duplicate Detection, eRulemaking, Public Comments, Information Retrieval

## 1. INTRODUCTION

Section 553 of the Administrative Procedures Act (APA), which was passed in 1946, requires U.S. regulatory agencies to publish proposed regulations in draft form and to solicit, consider, and respond to public comments before issuing the final regulation or rule. Although most proposed regulations attract few comments, some popular ones attract hundreds of thousands of comments from the general public. For instance, for the USDA’s national organic standard in late 1990s, a small team of rule writers manually sorted over 250,000 public comments [15]. In 2004 the EPA’s proposed National Emission Standards For Hazardous Air Pollutants For Utility Air Toxics rule (USEPA-OAR-2002-0056, “Mercury rule”) attracted over 530,000 email messages. When the comment volume becomes large, it usually includes many comments that are created based on *form letters* written by special interest groups, such as BushGreenWatch<sup>1</sup>, which claims to “track the Bush administration’s environmental misdeeds”.

When comments were submitted only on paper, modifying a form letter was time-consuming and hence few people did that. At that time form letters tended to be exact duplicates of one another

(except for address lines and signatures), thus identifying them was relatively simple, and considering them was also straightforward since they contained just one viewpoint which was expressed repeatedly. In recent years, however, regulatory agencies have begun to accept comments via both email and Web forms. Modifying an electronic form letter is very easy, and hence many people have begun doing so to better represent their opinions or stylistic preferences. As a result, regulatory agencies now face a huge volume of *near-duplicate* comments that are similar, but not exactly identical, to a (usually unknown) initial form letter, and to one another. Recognizing, organizing, and considering near-duplicates are significant challenges for the agencies because near-duplicates increase the likelihood of overlooking *substantive information* (information that by law the agency *must* consider) that an individual adds to a form letter. Our focus in this paper is on recognizing and organizing near-duplicates in a way that makes them easier for rule-writers and analysts to consider.

Duplicate and near-duplicate detection is not required only in the eRulemaking domain but also in many other domains. It is important in database research and electronic publishing. Moreover, search engines, such as Google, also find enormous duplicate and near duplicate pages when crawling the Web, but try to avoid returning them in search results in order to give users distinct and useful information on the first page. Nevertheless, near-duplicate detection for eRulemaking has unique characteristics that both simplify and complicate the problem.

Our goal in this work is to determine the extent and types of duplication existing in large public comment collections. To the best of our knowledge, this is the first work to study duplicate and near-duplicate detection for public comments. We focus on automating the process of near-duplicate detection, especially form letter detection, in this domain. We give a clear near-duplicate definition and explore simple and efficient methods of using feature-based document retrieval and similarity-based clustering to discover near-duplicates. Our methods are evaluated in experiments with a subset of a large public comment database collected for a recently proposed EPA rule. The experimental results show that our methods provide reasonable performance on detecting duplicates and near-duplicates in public comment.

The rest of the paper is organized as follows. Section 2 describes related research in other domains. Section 3 defines exact-duplicate and near-duplicate requirements for eRulemaking. Section 4 sketches our system architecture, and Section 5 describes the implementation. Section 6 discusses experimental methodology while Section 7 presents experimental results. Section 8 concludes the paper.

---

<sup>1</sup> <http://www.bushgreenwatch.org>

## 2. RELATED WORK

Early research on duplicate detection was done mostly in the areas of databases [1][17] and electronic publishing [2][12]. In database research, the goal of duplicate detection is to find records referring to the same entity but possibly in different representations. In electronic publishing, duplicate detection is used to detect plagiarism or to identify different versions of the same document. Most recently, duplicate detection is extensively studied for web search tasks, for example, to give more efficient web-crawling, effective search results ranking, and easy web documents archiving.

A widespread duplicate detection technique is to generate a document *fingerprint*, which is a compact description of the document, and then to do pair-wise comparisons of document fingerprints. The assumption is that fingerprints can be compared much more quickly than complete documents. A common method of generating fingerprints is to select a set of character sequences from a document, and to generate a fingerprint based on the hash values of these sequences. Similarity between two documents is measured by counting the number of common fingerprints. Different algorithms are characterized, and their computational costs are determined, by the hash functions and how character sequences are selected.

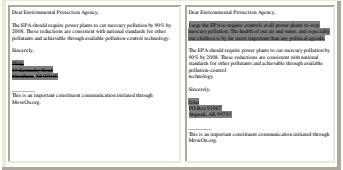
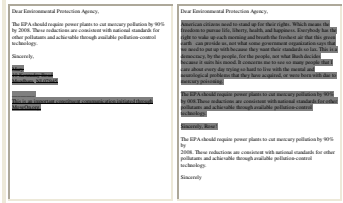

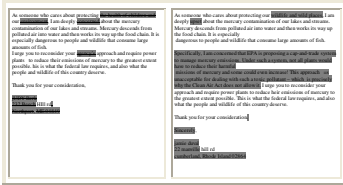
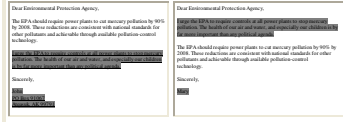
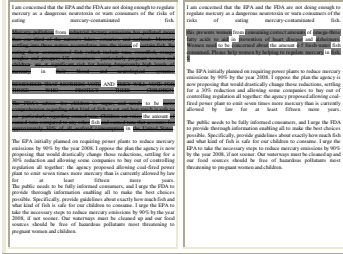

The dominant approach in duplicate detection is “shingling”, which was proposed by Broder, et al. [3]. It represents a document as a series of numeric encodings for an n-term text span, i.e. a shingle. A document sketch is created by keeping every *m*th shingle or the shingle with the smallest hash value. The authors also presented a super-shingling technique that creates meta-sketches to further reduce the computational complexity. Pairs of documents that share a large amount of common shingles were considered to be near-duplicates to each other.

Heintze studied selective document fingerprinting [11]. To reduce the size of the fingerprints, he selected a subset of the character sequences to generate fingerprints. He demonstrated that selective fingerprints usually perform within a factor of two of complete fingerprints. It can identify similarities between documents which have as little as 5% or less in common. Most importantly, selective fingerprinting is able to scale to large environments.

Pugh, worked at Google, claimed that duplicate and near-duplicate detection techniques may assign a number of fingerprints to a given document by “(i) extracting parts from the document, (ii) assigning the extracted parts to one or more of a predetermined number of lists, and (iii) generating a fingerprint from each of the populated lists”. Two documents may be considered to be near-duplicates if any one of their fingerprints matches [14].

Recent duplicate detection research in the Web environment has focused on issues of computational efficiency and detection effectiveness. “Collection statistics” is used to consistently recognize document replicas in full-text collections [6][15]. Since a term’s inverse document frequency (idf) is a measurement of a word’s rareness across a given collection, its value measures how well a term discriminates one document from others. Chowdhury, et al., selected terms best characterizing a document, i.e., high idf words. Moreover, a term is not considered unless it appears across the collection at least five times. This is to filter out possible misspelling tokens and typos. They claimed that in addition to

**Table 1: Duplicate Category**

Category	Definition	Example (text difference in gray)
Block Edit	Add or remove one or more paragraphs (<200 words) to another document	
Key Block	Add one or more paragraphs (200-500 words) to typical form letter paragraphs	
Minor Change	Alter words within a paragraph (<5% or 15 words in a paragraph)	
Minor Change + Block Edit	A combination of minor word change and block edit	
Block Reordering	Reorder the same set of paragraphs	
Bag-of-word similar	>80% word overlap (not in above categories)	
Exact	100% word overlap	

improving accuracy over shingling, it execute in one-fifth of the time. They showed that a statistically based approach (i) is more efficient than a fingerprint-based approach, (ii) scales in the number of documents and (iii) works well for documents of diverse sizes.

## 3. PROBLEM DEFINITION

The task of identifying and evaluating near duplicates manually is usually vague and subjective due to the lack of a clear definition of duplicates. Careful evaluation requires a clear definition of

exact- and near-duplicates. We begin with definitions from prior work, and then present the near-duplicate definition used in this research.

Pugh declared that “two documents are considered near duplicates if they have more than  $r$  features in common” [14]. Conrad, et al., stated that two documents are near duplicates if they share more than 80% terminology and their length difference is not more than  $\pm 20\%$  [8]. This is a necessary condition for most duplicates. However, simple percentage-based definitions neglect the structural differences among duplicates. Conrad, et al., also tried to define five categories of near-duplicates for Web documents: “excerpt”, “elaboration”, “insertion”, “focus” and “revision” in another paper [9]. They assumed that the original document can be identified easily. For example, “excerpt” means “one document takes first section from another article” and “elaboration” means “one document adds one or more paragraphs to another article”. However, in domains such as eRulemaking it may not be possible to tell by just looking at the documents which document is the source and which is the modified copy.

We agree with previous researchers that it is helpful to have a clear definition of what it means for two documents to be near-duplicates. In the eRulemaking domain there are two broad categories of documents: Comments that are written more-or-less from scratch (unique), and comments that are based on a form letter (exact-duplicates and near-duplicates).<sup>2</sup> Our experience with public comments leads us to define two documents to be near-duplicates if one of seven relationships holds between the two documents: *Block Edit*, *Key Block*, *Minor Change*, *Minor Change & Block Edit Combination*, *Block Reordering*, and *Bag-of-word Similar* (illustrated in Table 1: Duplicate Category). Our subcategories are similar in spirit to those of Conrad, et al., but reflect typical patterns appearing in public comment datasets.

The definitions given in Table 1: Duplicate Category can be used to explain why two documents are considered near-duplicates. The definitions also can be viewed as defining structure-based near-duplicate classifiers, which can be used either as alternatives to bag-of-words clustering/classification, or as a device for automatic assessment of bag-of-words clustering/classification. For example, structure-based definitions might help identify the causes of mistakes in bag-of-words clustering/classification. We return to this point in Sections 6 and 7, when we discuss large-scale evaluation.

#### 4. SYSTEM ARCHITECTURE

Our duplicate and near-duplicate detection system includes three modules: Text preprocessing, feature-based document retrieval and similarity-based document clustering (Figure 1).

The text preprocessing module consists of information extraction (IE), binning by length, and seed document selection. The IE module identifies the email sender, the receiver, address, salutation and signature lines; docket IDs mentioned in the text; delivery dates and the email-relaying organizations identified in

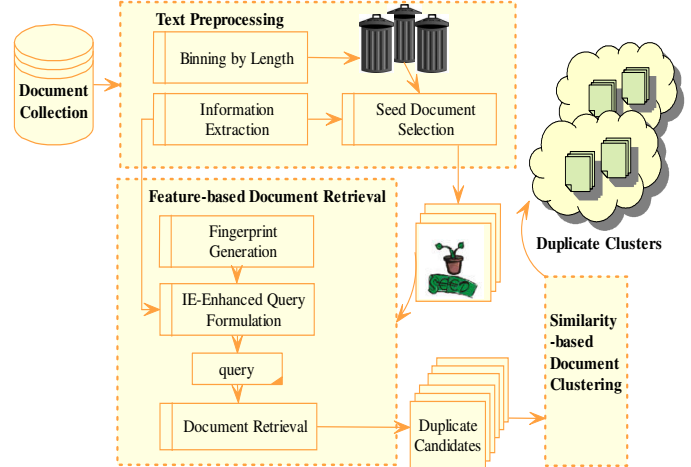


Figure 1: System Architecture

email headers. These features are used later in duplicate detection. Documents are binned based on their word-based content lengths (i.e., after removal of headers, address, salutation, and signature lines are removed). Binning insures that exact-duplicates are grouped together. Moreover, the system will have a rough guess about what are the large duplicate sets before the clustering starts. These sets are handled first, to improve efficiency. Finally, to identify the dominating document within a bin, the document that repeats most (i.e., has the most exact duplicates, using word-based comparison) is selected to be the initial seed in any bin whose size is greater than 100.

In the feature-based document retrieval module, the system begins near-duplicate detection by going through the size-based bins in a round robin manner. On each pass a document is selected from a bin, and a feature-based query is generated based on a fingerprint and metadata. The system then performs probabilistic document retrieval, using a Boolean query, to get candidate duplicates for the document.

The similarity-based clustering module measures the similarity of each document in the candidate set to the current seed; similar documents are grouped together. The next section describes the algorithms for feature-based document retrieval and similarity-based clustering.

#### 5. NEAR-DUPLCATE DETECTION

Analysis of public comment databases reveals that they contain a wide range of near-duplicate documents. Email programs frequently adjust layout, for example spacing and formatting, but these are of little concern to word-based matching algorithms, so we do not consider them further. It is surprisingly common for people to make minor adjustments to punctuation or wording. It is also common for people to make larger changes, such as adding or deleting sentence or paragraphs. Hence it is not enough to know that two documents differ; human analysts need similar documents grouped together, simple measures of how similar two documents are, and differences highlighted.

We consider the problem from Information Retrieval (IR) point of view. Documents are treated as unordered ‘bags-of-words’, and each document is represented by a unigram word distribution. Exact-duplicates and near-duplicates can be detected by

<sup>2</sup> Regulators are also interested in identifying letters that express the same viewpoint, whether or not the letters are based on the same form letter, but that is a different, and more complex, problem.

identifying documents with identical or similar unigram language models. Kullback-Leibler (KL) divergence, a distributional similarity measure, is one way to measure the similarity of one document given another. Our duplicate detection system exploits an unsupervised clustering algorithm based on KL divergence.

Pair-wise KL-divergence comparisons on large datasets are time-consuming. To avoid clustering the entire dataset, given a cluster seed, the system performs feature-based document retrieval to quickly create a set of potential exact- and near-duplicate documents. A Boolean query is created based on the information extraction results in preprocessing and representative text spans extracted from the documents. This query could be considered a combination of metadata and document fingerprints. In addition to the simplicity of using standard IR system capabilities, the complexity and computational costs of creating and using this type of fingerprints are much lower than using those produced by traditional fingerprinting techniques, such as shingling.

### 5.1 IE-enhanced Feature-based Retrieval

An important issue that forms the foundation for all duplicate detection work is the feature set selected to characterize a given document or to generate its “fingerprint.” In our system, we extract fingerprints from text chunks and combine them with other metadata to form a Boolean query. The system then uses the Lemur Toolkit<sup>3</sup> as the document retrieval engine to get duplicate candidates. We call this process *feature-based document retrieval*, since it combines information from both fingerprints and a group of features extracted during the preprocessing stage. The system starts with the seed documents selected during preprocessing for the first round robin runs. Each seed document is broken into chunks. The size of each chunk depends on the length of a document. In our system, how many chunks selected from a given document is determined by a normal step function. If a document contains more than 200 words, the size of each chunk is set to 40 words; if it contains fewer words, there will be at most 8 chunks within it. Thus the compression ratio of a chunk to a fingerprint is higher for longer documents and lower for short ones. Note that chunks should not cross paragraph boundary and terms under consideration exclude headers, address lines, etc.

Among terms  $t_j$  in a chunk  $C_i$ , the fingerprint is constructed as a text span around term  $t^*$  which is the term with minimal document frequency in the chunk and is defined as:

$$t^* = \arg \min_j df(t_j) \tag{1}$$

where  $df(t_j)$  is the number of documents containing  $t_j$ , and  $\arg \min$  is the term with the minimal  $df()$  value.

One text span (of length 3 for the sample document in Figure 2), centered around  $t^*$ , is selected from each chunk. Note that the text span could be of any length; it is not necessary to be a trigram. A database-specific text span size parameter is used; usually we set it to 3 to 5. For the sample document, the fingerprints are “standards proposed by” “will harm thousands” “unborn children

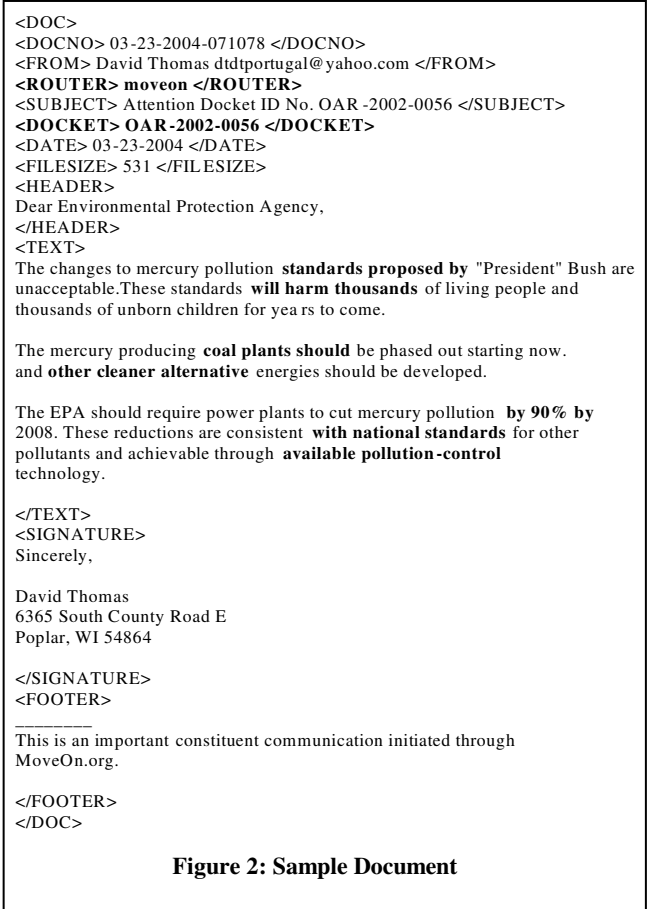


Figure 2: Sample Document

for” “coal plants should” “other cleaner alternative” “by 90 by” “with national standards” and “available pollution control”.

The information extractor identifies email senders, receivers, signatures, docket IDs, delivered dates, and email relayers during preprocessing; the retrieval system indexes them, and supports their use in queries. Since many near duplicates are based on form letters created by special interest groups (“mass mailers”) or are commercial advertising (“spam”), duplicate documents are likely to share the same features as the seed document.

Finally, the feature-based query is formed based on fingerprints and metadata. For instance, for the seed document shown in Figure 2, the query is:

```
#AND ( docketoar.20020056 router.moveon #OR(“standards proposed by” “will harm thousands” “unborn children for” “coal plants should” “other cleaner alternative” “by 90 by” “with national standards” “available pollution control”) )
```

The system uses Lemur to retrieve all matching documents, which collectively form the duplicate candidate set for the seed document.

### 5.2 Similarity-based Duplicate Clustering

Once the duplicate candidate set  $S_i$  for a seed document  $d_i$  is formed, the system measures the similarity between  $d_i$  and every document  $s_{ij} \in S_i$ . In language model approach, a document  $d_i$  is

<sup>3</sup> <http://www-2.cs.cmu.edu/~lemur/>

- A) Initialize the Duplicate Cluster Collection  $N$ :  $N \leftarrow \emptyset$ .
- B) Sort the Bin Collection  $B$  in decreasing order.
- C) Get the initial seed documents by seed selection and move the initial seed documents to the front of the corresponding bins.
- D) From each histogram bin  $b_l \in B$ , pick one document  $d_i$  as the seed document. Note that the initial seed document will be the first one to start with within a bin.
- a) Generate a feature-based query  $q_i$  and retrieve candidate set  $S_i$  for  $d_i$ ,
  - b) For each document  $s_{ij} \in S_i$ , if  $\text{dist}(s_{ij}, d_i) < \theta_i$ ,
- $\forall$  cluster centroid  $d_k \in N$ ,
- if  $\text{dist}(s_{ij}, d_i) > \text{dist}(s_{ij}, d_k)$ ,
  - add  $s_{ij}$  into duplicate cluster  $n_{dk}$ :  $n_{dk} \leftarrow n_{dk} \cup \{s_{ij}\}$
  - otherwise if  $\text{dist}(s_{ij}, d_i) \leq \min_k (\text{dist}(s_{ij}, d_k))$  and  $d_i \notin N$ ,
  - create a new cluster  $n_{di}$ , add it into  $N$ :  $N \leftarrow N \cup \{n_{di}\}$ ,
  - add  $s_{ij}$  into  $n_{di}$ :  $n_{di} \leftarrow n_{di} \cup \{s_{ij}\}$
  - eliminate  $s_{ij}$  from  $n_{dk}$  if necessary:  $n_{dk} \leftarrow n_{dk} - \{s_{ij}\}$
- Eliminate  $s_{ij}$  from bin  $b_l$ :  $b_l \leftarrow b_l - \{s_{ij}\}$
- c) If  $b_l = \emptyset$ , remove  $b_l$  from  $B$
- E) If  $B = \emptyset$ , output  $N$  as the final set of duplicate clusters.

**Figure 3: Algorithm to form duplicate clusters**

represented by a unigram word probability distribution  $p_i$  over the vocabulary. For any given two documents  $d_a$  and  $d_b$  with their word probability distributions  $p_a$  and  $p_b$  respectively, the KL divergence, also known as the relative entropy, a measure of how different two probability distributions are, is given as:

$$\begin{aligned} KL(p_a \parallel p_b) &= \sum_{w_j \in d_a} p_a(w_j) \log \frac{p_a(w_j)}{p_b(w_j)} \\ &= \sum_{w_j \in d_a} p(w_j | d_a) \log \frac{p(w_j | d_a)}{p(w_j | d_b)} \end{aligned} \quad (2)$$

where  $w_j$  is a word appearing in document  $d_a$ , and

$$p_a = p(w_j | d_a) = \frac{\text{tf}(w_j, d_a)}{\sum_{w_i \in d_a} \text{tf}(w_i, d_a)} \quad (3)$$

Since KL-divergence is non-negative and non-symmetric, we define and use the minimal value of two KL-divergences as the distance measure between two documents  $d_a$  and  $d_b$ :

$$\text{dist}(d_a, d_b) = \min(KL(p_a \parallel p_b), KL(p_b \parallel p_a)) \quad (4)$$

If a word never occurs in document  $d_a$ , maximum likelihood estimation (Equation 3) gives it zero probability, which is clearly too strict. Laplace smoothing (or add-one smoothing) [13] is used to adjust the maximum likelihood estimation.

Given this model, duplicate clusters are formed using the steps shown in Figure 3. Several issues are worth noting. First, step D.a performs a feature-based Boolean retrieval to get a set of candidate documents as the basis to start clustering. It works fairly efficient for large clusters since it usually cuts the number of documents needed to be clustered from the size of the entire dataset to a reasonable number. One of our test collections contains 536,975 public comments; the average size of candidate duplicate sets is 10,995. This reduction is extremely significant

for the big clusters. However, for small clusters, especially for those only containing a single unique document, it is inefficient to retrieve and look through the candidate set. Therefore, after most of the big clusters have been found, if there are 5 clusters in a row only have a single document in them, the feature-based retrieval module is disabled based on the assumption that most of the remaining unclustered documents are unique. Only similarity-based clustering is used on them.

Second, the cut-off threshold  $\theta_i$  for different clusters should be different. Documents in a cluster are sorted by their document-centroid similarity scores. The sorted scores are sampled at 10 document intervals. If there is a large value change, i.e. greater than 5% of the initial cut-off threshold within an interval, a new cut-off threshold is set at the beginning of the interval. Note that this requires an extra step to first store the similarity scores above a relatively loose initial cut-off threshold.

## 6. EVALUATION METHODOLOGY

It is usually difficult to evaluate clustering performance, including duplicate clustering. Ideally there would be a ‘‘ground truth’’ identification of which documents are near-duplicates and which are not. Our initial research was conducted with public comment datasets provided by the USDA, US DOT, and US EPA [4]. None of these datasets have ‘‘ground truth’’ assignments. The largest, a public comment dataset for the EPA’s proposed National Emission Standards For Hazardous Air Pollutants For Utility Air Toxics rule (USEPA-OAR-2002-0056, ‘‘Mercury rule’’) contains 536,975 email messages. Comprehensive manual assessment on this dataset is a challenge.

Hence, two subsets of 1,000 email messages each were randomly selected from the Mercury dataset for experiments described in Section 7. Each subset was given to two graduate research assistants, who manually organized the documents into clusters of documents that they felt were near-duplicates. Because both students did both datasets, it is possible to measure inter-rater agreement between the students, as well as to measure agreement between the duplicate detection system and the students. The experiments were performed after the system was developed, i.e., the system was not tuned to mimic the behavior of the students.

The effectiveness of clustering algorithms is often measured with a variety of metrics that assess inherent cluster quality or consistency, as well as agreement between the clustering algorithm and a ‘‘ground truth’’. In the context of clustering, the quality of clustering with respect to the ground truth is assessed in terms of precision ( $p$ ) and recall ( $r$ ). The precision and recall for each cluster  $i$  with respect to each class  $j$  in the ground truth are  $p_{ij}$  and  $r_{ij}$  and are defined as:

$$p_{ij} = n_{ij} / n_i \quad (5)$$

$$r_{ij} = n_{ij} / n_j \quad (6)$$

where  $n_{ij}$  is the number of documents from class  $j$  in cluster  $i$ ,  $n_i$  is the number of documents in cluster  $i$  and  $n_j$  is the number of documents in class  $j$ . In the following discussions, we will use ‘‘cluster’’ to refer to a duplicate set generated by our algorithm while ‘‘class’’ generated manually by human assessors.

The metrics used in our experiments are F-measure, purity, pairwise measure and kappa inter-rater agreement. The first three

measures are commonly used in the IR community for evaluating clustering performance while the last measure is widely used in assessing inter-rater agreement when human assessments are the main resource that can be relied on. The detailed definition of each measure is given below.

## 6.1 F-Measure

The first metric is F-measure, which is widely used in the IR community and tries to capture how well the groups of the investigated clustering at the best match the groups of the ground truth. F-measure for cluster  $i$  and class  $j$  is defined as:

$$F_{ij} = 2 \cdot r_{ij} / (p_{ij} + r_{ij}) \quad (7)$$

F-measure for the entire set of clusters is:

$$F = \sum_j \frac{n_j}{n} F_j \quad (8)$$

where  $F_j = \max_i \{F_{ij}\}$ ,  $n_j$  is the number of documents in class  $j$ ,  $n$  is the total number of documents.

## 6.2 Purity

The second metric is purity. The purity of each cluster tries to capture how well the groups of the first on average cluster match those of the ground truth. The weighted average purity  $\rho$  over all clusters measures the quality of the whole clustering. It is defined as:

$$\rho = \sum \frac{n_i}{n} \rho_i \quad (9)$$

where  $\rho_i = \max_j \{p_{ij}\}$ ,  $n_i$  is the number of documents in class  $j$ ,  $n$  is the total number of document and  $p_{ij}$  is precision of cluster  $i$  with reference to class  $j$ .

## 6.3 Pair-wise Measure

The third metric is based on the distribution of pairs of documents. Pair measures reflect degrees of overlapping between clusters and classes. Folkes and Mallows index [10] is one of such pair measures and is defined as:

$$FM = \frac{a}{\sqrt{(a+b)(a+c)}} \quad (10)$$

where  $a$  is the number of pairs in the same group in the ground truth and in the clustering (agreement),  $b$  is the number of pairs in the same group in the ground truth but in the different in the clustering (false negative),  $c$  is the number of pairs in the different groups in the ground truth but in the same in the clustering (false positive),  $d$  is the number of pairs in the different groups in the ground truth and in the clustering (agreement).

## 6.4 Kappa

Finally, we use the Cohen's kappa statistics [7] to evaluate both clustering effectiveness and structure-based duplicate categories. For the first task, kappa assesses agreement between clustering results relative to two ground truth given by different human assessors. We consider the investigated clustering algorithm as another "assessor". The kappa coefficient is defined is:

$$\kappa = \frac{p(A) - p(E)}{1 - p(E)} \quad (11)$$

**Table 2: Clustering Performance**

		$F$	$\rho$	$FM$	$\kappa$ ( $\kappa_{BC} = .74$ )	
					$\kappa_{AB}$	$\kappa_{AC}$
1	Baseline1 filesize	0.14	0.12	0.24	0.12	0.08
2	Baseline2 router	0.62	0.28	0.34	0.31	0.45
3	Metadata-based (router+filesize+rank+e+docket)	0.68	0.34	0.42	0.42	0.56
4	fingerprint-based	0.68	0.74	0.34	0.64	0.64
5	Feature-based (3+4)	0.70	0.74	0.40	0.64	0.68
6	Similarity-based clustering	0.62	0.72	0.32	0.56	0.58
7	Current approach (5+6)	0.74	0.81	0.55	0.66	0.71

where  $p(A)$  is the observed agreement between the two assessments. It can be calculated as  $(a+d)/m$  by using  $a$ ,  $b$ ,  $c$  and  $d$  given in Equation 10 and  $m=a+b+c+d$ .  $p(E)$  is the agreement expected by chance. It can be calculated as  $(a+b)(a+c)/m^2 + (b+c)(c+d)/m^2$ .

Our idea is that if the concordance of the clustering results given by our system and one of the assessors are as good as or even better than that of two human assessors, the algorithm could be considered as effective.

For the second task, kappa assesses agreement between judgments on a certain clustering run given by structure-based assessor and human assessors. If the inter-rater agreement of the structure-based assessor and a human assessor is comparable with inter-rater agreement between two human assessors, we would claim that the structure-based assessor is effective. In computational linguistics  $\kappa > 0.67$  has often been required to draw any conclusion of agreement [5]. However, there has been a number of objections to this standard and less strict evaluations consider  $0.20 < \kappa < 0.40$  indicating fair agreement and  $0.40 < \kappa < 0.60$  indicating moderate agreement.

## 7. EXPERIMENTAL RESULTS

### 7.1 Duplicate Detection Effectiveness

The experimental results are shown in Table 2. The descriptions of experiments are given as follows:

- 1) Experiment 1 runs a baseline algorithm to partition a sample public comment dataset into clusters by document file sizes. File size, plus or minus 10%, is a simple feature that regulatory agencies often use to get an initial organization of documents.
- 2) Experiment 2 runs another baseline algorithm where datasets are partitioned by the email relayers. This partition is based on the observation that comments delivered by a particular email router, for example the mail router of a special interest group website, are more likely to be based on a form letter. Therefore, it is another way for the agencies to organize the dataset.
- 3) Experiment 3 tests an algorithm that uses a combination of several metadata to partition the dataset. Information such as email routers, file size and docket ID is joined to have a finer

partition of documents. Note that the 10% file size heuristic is used here to identify near-duplicates with slightly varying lengths.

- 4) Experiment 4 tests clustering effectiveness. It performs ranked Boolean document retrieval by generating a fingerprint for each document with substrings extracted from it. Note that the experiment uses “probabilistic OR” to score each document pair and form the clusters following a clustering algorithm similar to the one shown in Figure 3.
- 5) Experiment 5 is a combination of metadata and fingerprint extraction in experiment 3 and 4. It is the feature-based document retrieval described in Section 5.1, the first part of our duplicate detection algorithm.
- 6) Experiment 6 conducts a “bag-of-words” similarity-based document clustering as most IR systems do. It calculates KL-divergence for each document pair in the dataset and generates clusters. It is the second part of our current approach.
- 7) Experiment 7 shows the current approach, i.e., feature-based document retrieval and similarity-based clustering, which was detailed in Section 5.

Based on the above experimental results, we have the following observations:

- Clustering based on just metadata like file size and email routers (experiments 1, 2, and 3) are not as effective as other approaches. Although these may seem like simplistic approaches, they are the main methods currently used by regulatory agencies to identify near-duplicates. The weakness of these results suggests that more sophisticated approaches, such as what presented in this work, are required.
- The traditional method of fingerprint-based duplicate detection (experiment 4) provides reasonable performance. It greatly improves the quality of duplicate clusters as compared with the baselines (0.54 increments in F, 0.62 in purity, 0.10 in FM, and almost 0.6 in Kappa for baseline 1).
- The featured-based document retrieval in our system (experiment 5) improves the fingerprint approach further. Although the improvement is not so significant, it helps to distinguish the public comments from different form letters which come from different email relayers. It helps to put the comments generated based on different form letters into separate clusters. Moreover, it reduces the size of the duplicate candidates in the earlier stage, so that the algorithm runs more efficiently in the later process.
- "Bag-of-words" similarity-based clustering (experiment 6) also gives reasonable performance as compared with the baseline approaches (0.48 increments in F, 0.60 in purity, 0.12 in FM and almost 0.5 in kappa for baseline 1), however, lower than the traditional fingerprint-based approach. We still recommend this approach because of its simplicity, efficiency and comparable performance. However, it gives us a hint that assessment based on just word overlapping might not be adequate for duplicate detection.

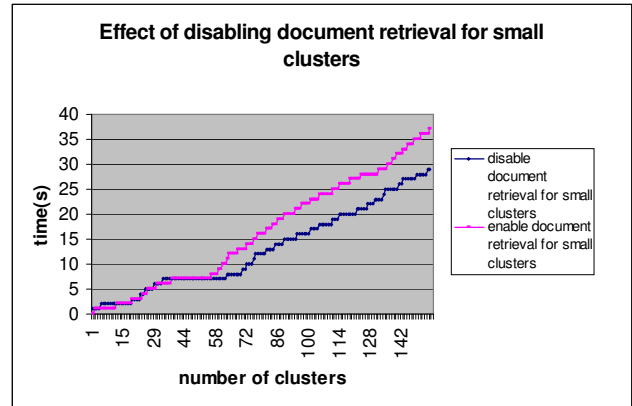


Figure 4: Effect of disabling document retrieval for small clusters

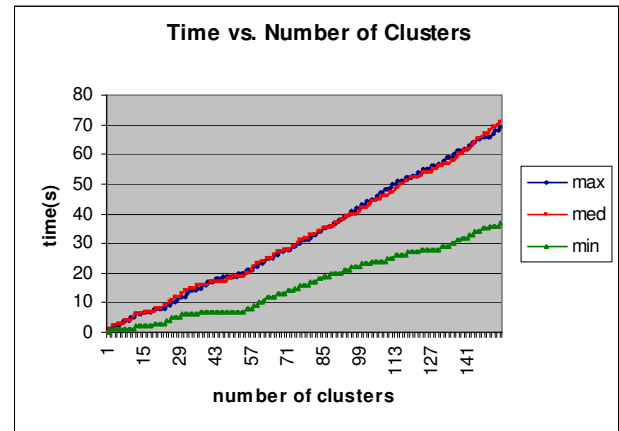


Figure 5: Time complexity of different fingerprint methods

- The combination approach (experiment 7) provides the best performance (0.74 for F, 0.81 for purity and 0.55 for FM).

Worth noting is that the kappa values of 0.66 and 0.71 are close to the level of 0.74 human inter-rater agreement for this data. Although it is too early to conclude that our approach is comparable to the quality of human-produced duplicate clusters, it is an encouraging result.

## 7.2 Duplicate Detection Efficiency

Our system generates document fingerprints by extracting a text span around a term in each chunk. Let’s call this term the seed term. The system selects the seed term based on its document frequency (*df*). Section 5.1 suggests that the seed term be the one with minimum *df* within a chunk. However this choice is not explained or justified. A series of experiments explored the effects of choosing seed terms based on minimum, median or maximum *df*. There was little difference among how the seed terms were selected in a text chunk in terms of effectiveness, i.e. they resulted in the same duplicate clusters. This is not surprising, of course, because the retrieved documents are merely candidate near-duplicates; it is the similarity-based clustering that forms the final set of near-duplicate clusters. However, different

methods of selecting the seed terms *do* affect computational efficiency. The minimum document frequency method uses half the time used by the median and maximum document frequency methods (Figure 5). This result confirms related work that favored high idf terms [6].

Feature-based Boolean retrieval aims to get a smaller set of candidate documents as the basis for clustering. It works efficient for large clusters since it usually cuts the number of documents to be clustered from the size of the entire dataset to a reasonable number. However, for small clusters, especially for those only containing a single unique document, it is inefficient to retrieve and look through the candidate set. Instead, the system could just look over the remaining unclustered documents and do pair-wise similarity-based clustering straight away. Experiments show that the clustering effectiveness remains while the time complexity drops. Figure 7 shows that after disabling document retrieval for small clusters, the time used to create new clusters becomes less. However, we need to know where a good point lies to disable the document retrieval module. It could be based on the initial binning of documents to roughly guess which clusters are the possible big ones and which are small ones. However, this heuristic is not so reliable. In our experiments, if 5 clusters in a row contain only a single document, it is a good time to disable the document retrieval module. This heuristic is extremely conservative, but we have not yet found a more effective method.

### 7.3 Evaluation of Structure-based Assessor

The previous experiments studied the accuracy of the near-duplicate detection algorithms by comparing the results with ground truth generated manually. Although this is a reasonable experimental methodology, it does not address the problem of conducting evaluations on large-scale public comment databases, where human assessments are impractical. Another approach is necessary.

Table 1 defines near-duplicate categories based on simple relationships among pairs of documents. These relationships can be easily recognized by simple algorithms, hence they provide another approach to assessment. The *structure-based assessor* defines two documents as near-duplicates if they fall into any of the categories defined in Table 1.

The structure-based assessor makes it possible to evaluate near-duplicate detection on large public comment datasets, for which human assessment is not supportable. Each experiment conducted by the near-duplicate detection system produces a set of duplicate or near-duplicate clusters. For every document in a cluster, the structure-based assessor compares it with the cluster centroid and decides whether they belong to any of the duplicate categories defined in Section 3. If so, the documents belong in the same cluster; if not, they don't.

We evaluated the accuracy of the structure-based assessor by investigating how well it agrees with human assessments on a small corpus. The two human assessors conducted a second assessment to examine the results of one of the near-duplicate detection algorithms. We selected the most accurate of the algorithms described in Section 7.1 (Table 2, experiment 7). The two human assessors gave “yes” or “no” subjective judgments about whether two clustered documents should actually be considered near-duplicates. This methodology is distinct from the

previous experiment in which the assessors clustered documents manually, without seeing the results of the clustering system; when human assessors actually saw the clustering results, their judgments usually changed a bit from the ground truth judgments they had given earlier. The purpose of this evaluation was to evaluate the correctness of our definition of duplicates (Table 1), i.e., to evaluate the effectiveness of structure-based assessment.

Table 3 summarizes the result. The assessors are labeled B and C, as before. The structure-based assessor is labeled D. The inter-assessor agreement among the humans is  $\kappa_{BC}=0.75$ , whereas the agreements between the structure-based assessor and the two humans were 0.71 and 0.75. We interpret this as meaning that the structure-based assessor is comparable to a human assessor.

**Table 3: Evaluation of Structure-based Assessor (for experiment 7)**

	$\kappa$ ( $\kappa_{BC}=0.75$ )	
	$\kappa_{DB}$	$\kappa_{DC}$
structure-based assessor	0.71	0.75

## 8. CONCLUSION AND FUTURE WORK

U.S. regulatory agencies are required to solicit, consider, and respond to public comments before issuing regulations. Recently, the shift from paper to electronic public comments makes it much easier for individuals to customize form letters while harder for agencies to identify substantive information since there are many near-duplicate comments that express the same viewpoint in slightly different language.

This is the first work done for exact- near-duplicate detection in eRulemaking domain. In this work we focus on the process of identifying near duplicates of form letters. The definitions of near- and exact-duplicates that are appropriate to eRulemaking are given. This paper explores the use of simple text clustering and retrieval algorithms for the task. Experiments in public comment domain show that our method is effective.

Our future work includes comparing its performance to other state-of-the-art techniques such as shingling, and performing large scale tests of the proposed structure-based assessment scheme described in this paper. Since the structure-based assessor works well, it might be helpful to adopt some heuristics from it into our clustering algorithm. However, system efficiency should be under close consideration.

Since the seed documents selected for each size-based bins might not be the centroid of the real near-duplicate set, the documents that are within the similarity threshold of the seed document might include false-positive documents if the similarity threshold is large. For public comments related to the proposed regulation, all the documents are focused on a relatively narrow set of topics. Therefore, in general, it is very likely that many documents would be relatively similar to each others. In our implementation, it is preferable to keep a small initial similarity threshold. However, it might tend to break up a real duplicate set into many smaller duplicate sets. Machine learning approaches might enable the algorithm to find more desirable thresholds and text chunk sizes.

## ACKNOWLEDGMENTS

We are grateful to the USDA, US DOT, and US EPA for providing the public comment data that made this research

possible. We thank Jie Lu and Pucktada Treeratpituk for their work on preliminary versions of the research described here. This research was supported by NSF grants EIA-0327979 and IIS-0429102. Any opinions, findings, conclusions, or recommendations expressed in this paper are the authors', and do not necessarily reflect those of the sponsor.

## REFERENCES

- [1] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD-2003), Washington D.C., August 2003.
- [2] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In Proceedings of the Special Interest Group on Management of Data (SIGMOD 1995), pages 398–409. ACM Press, May 1995.
- [3] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. In Proceedings of WWW6 '97, pages 391–404. Elsevier Science, April 1997.
- [4] J. Callan, eRulemaking testbed. <http://hartford.lti.cs.cmu.edu/eRulemaking/Data/>. 2004
- [5] J. Carletta. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254, 1996.
- [6] A. Chowdhury, O. Frieder, D. Grossman, and M. McCabe. Collection statistics for fast Duplicate document detection. In *ACM Transactions on Information Systems (TOIS)*, Volume 20, Issue 2, 2002.
- [7] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 37–46, 1960.
- [8] J. Conrad and C. P. Schriber. Constructing a Text Corpus for Inexact Duplicate Detection. In Proceedings of ACM SIGIR'04, Sheffield, South Yorkshire, UK. July 25–29, 2004
- [9] J. Conrad, X. S. Guo, and C. P. Schriber. Online duplicate document detection: Signature reliability in a dynamic retrieval environment. In Proceedings of CIKM'03, pages 443–452. ACM Press, Nov. 2003.
- [10] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.
- [11] N. Heintze. Scalable document fingerprinting. In Proceedings of the Second USENIX electronic Commerce Workshop, pages 191–200, Nov. 1996.
- [12] T. Hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. In *Journal of the American Society for Information Science and Technology*, Volume 54, Issue 3, 2003.
- [13] P. Laplace. *Philosophical essay on probabilistic*. New York: Springer-Verlag, 1995.
- [14] W. Pugh. US Patent 6,658,423 <http://www.cs.umd.edu/~pugh/google/Duplicates.pdf>. 2003
- [15] S. Shulman. An experiment in digital government and the United States National Organic Program. *Agriculture and Human Values*. 2003
- [16] N. Shrivakumar and H. Garcia-Molina. Finding near-replicas of documents on the Web. In Proceedings of Workshop on Web Databases (WebDB '98), pages 204–212, March 1998.
- [17] T. Yan and H. Gracia-Molina. Duplicate removal in information dissemination. In Proceedings of the 21st International Conference on Very Large Data Bases (VLDB '95), 1995.