Next Steps in Near-Duplicate Detection for eRulemaking

Hui Yang
Language Technology Institute
School of Computer Science
Carnegie Mellon University
+1-412-268-4083
huiyang@cs.cmu.edu

Jamie Callan
Language Technology Institute
School of Computer Science
Carnegie Mellon University
+1-412-268-4525
callan@cs.cmu.edu

Stuart Shulman
Library and Information Science
School of Information Sciences
University of Pittsburgh
+1-412-624-3776
shulman@pitt.edu

ABSTRACT
Large volume public comment campaigns and web portals that encourage the public to customize form letters produce many near-duplicate documents, which increases processing and storage costs, but is rarely a serious problem. A more serious concern is that form letter customizations can include substantive issues that agencies are likely to overlook. The identification of exact- and near-duplicate texts, and recognition of unique text within near-duplicate documents, is an important component of data cleaning and integration processes for eRulemaking.

This paper presents DURIAN (**DU**plicate **R**emoval **I**n l**A**rge collectio**N**), a refinement of a prior near-duplicate detection algorithm DURIAN uses a traditional bag-of-words document representation, document attributes ("metadata"), and document content structure to identify form letters and their edited copies in public comment collections. Experimental results demonstrate that DURIAN is about as effective as human assessors. The paper concludes by discussing challenges to moving near-duplicate detection into operational rulemaking environments.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**] Clustering, Query formulation, Retrieval models, Search process

General Terms
Algorithms, Performance, Experimentation.

Keywords
Duplicate detection, clustering, eRulemaking, public comments, information retrieval, text analysis

1. INTRODUCTION
U.S. law and standard regulatory practice requires U.S. regulatory agencies to give notice of a proposed rule and then respond to substantive comments from lobbyists, companies, trade organizations, special interest groups, and the general public before issuing a final regulation or rule [5][12][18]. When the comment volume is low, as is usually the case, this task is a minor burden. However, a small number of high profile regulations can attract hundreds of thousands of comments, most of which are exact or near duplicate form letters [7][21]. Near duplicates are generated in large volume when an organization posts a form letter on a Web page that allows or encourages customization before submission. For example, letters and email generated by MoveOn.org for a recent rule were characterized by two form letter sentences followed by text ranging in length from one sentence to, on a few rare occasions, many paragraphs.

Currently, simple heuristics are used to manually identify the many copies of form letters. Often this work is conducted by consulting firms, rather than agency personnel. Most form letter customizations express largely the same opinion in slightly different language [20], but occasionally customizations include substantive issues, which the agencies or their contractors are likely to overlook [19]. The automatic sorting of documents into exact and near duplicate categories, as well as the automatic identification of modified passages, could significantly lower the costs and risks involved in processing large volumes of public comments.

One novel characteristic of near-duplicate detection for notice and comment rulemaking is that two comments may be considered near-duplicates even if they share a relatively small amount of text. A public interest group may encourage people to personalize a form letter by appending their own text to text written by the interest group. A regulatory agency would likely want these comments grouped together even though the amount of modified text varied greatly.

Early research on duplicate detection was done mostly in the areas of databases, digital libraries, and electronic publishing. Recently, duplicate detection has been studied for web search tasks, for example, to give more effective and efficient web-crawling, document ranking, and document archiving. Duplicate detection techniques have been proposed that range from manually coded rules to applications of the latest machine learning techniques [1][2][6][10][14][15][17][22]. Their focus varies from providing high detection rates to minimizing the computational and storage resources. Accuracy varies as well. For large collections, some techniques are too expensive computationally to be deployed in their full capacity. Some algorithms are very efficient yet very brittle and sensitive to even small changes of the text.

This paper proposes a similarity measure for duplicate detection that is based on Kullback-Leibler (KL) distance. It also investigates the use of clustering techniques to find near-duplicate documents. Data clustering is a popular approach for automatically grouping similar objects. In practice this discovery process should avoid redundancies with existing knowledge about groupings, and reveal novel, previously unknown aspects of the data. The technique proposed below uses instance-level clustering

constraints based on document attributes and the editing styles of near-duplicates: *Block Edit* (add or delete several paragraphs), *Key Block* (contains one or more well-known paragraphs), *Minor Change* (small editing changes), *Minor Change & Block Edit Combination,* and *Block Reordering* (reorder known paragraphs). We show that an existing clustering algorithm can be modified to enforce these constraints.

Our goal in this work is to greatly improve near-duplicate detection accuracy for notice and comment rulemaking as well as to maintain efficiency. Our methods are evaluated in experiments with subsets of a public comment database collected for a recent U.S. Environmental Protection Agency (EPA) rulemaking. The experimental results show that system-human intercoder agreement is comparable to human-human intercoder agreement.

The rest of the paper is organized as follows. Section 2 details our algorithm. Section 3 describes our evaluation methodology. Section 4 presents experimental results. Section 5 discusses the next steps along this research path, and concludes.

2. ALGORITHM OVERVIEW

Our goal is a software system that assists rule writers and other interested parties in understanding comments that U.S. regulatory agencies receive as part of notice and comment rulemaking. The system should identify reference copies of form letters, modified copies of form letters (near-duplicates) and how they were modified, and unique comments. The system should make decisions that are consistent with human assessments in this domain, which means that documents might be considered near-duplicates even if they have only relatively small passages in common. Duplicate and near-duplicate detection for notice and comment rulemaking in the era of pervasive email and the Web has several characteristics that help define or constrain the problem. The sections below describe an algorithm that meets these requirements.

2.1 Previous Algorithm

An earlier version of our research [22] uses a two stage duplicate detection algorithm. The first stage involves feature-based retrieval. The second stage uses a single-pass clustering algorithm to group the near duplicates.

The system starts with lexical preprocessing intended to normalize the representation of documents. Document mark-up such as HTML tags and email headers are automatically removed. Metadata (email senders, relayers, and recipients; timestamps), address blocks, and signature blocks are recognized and tagged automatically, using simple, rule-based heuristics.

Any comment that has more than 5 exact duplicates (after lexical preprocessing) is considered an instance of a form letter. The copy with the earliest timestamp is the *reference copy* of the form letter. Each reference copy becomes a *seed document* for the clustering algorithm. Some documents that are not reference copies may be also become seeds later in the clustering process.

The body text of each seed document is broken into chunks. Chunks are constrained to not cross paragraph boundaries. The number of chunks created from a document is determined by a heuristic step function. The number of words in each chunk is:

$$\begin{array}{ll} \text{if document length} > N: & m \\ \text{otherwise}: & \text{document length} / n \end{array} \quad (1)$$

If a document contains more than N words, the size of each chunk is set to m words; if it contains fewer words, there will be at most n chunks within it. Thus the compression ratio is higher for longer documents and lower for short ones. In our system, we set N=200, m=40, n=8 empirically.

The text chunks for a document are combined with metadata to form a Boolean query. A text search engine (the Lemur Toolkit[1]) is used to find candidate near-duplicate efficiently. We call this process *feature-based document retrieval*, because it combines substrings and features extracted during preprocessing (e.g., email senders, receivers, signatures, docket IDs, delivered dates, and email relayers). A query to Lemur looks like: *"#AND (docket.oar20020056 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"))*"

After a set of candidate near-duplicates is retrieved, the similarity of each candidate to the seed document is measured, using KL-divergence. Similar documents are placed into the seed document's cluster. A preliminary evaluation suggested that the algorithm was generally very effective [22].

2.2 Algorithm Refinement

This section describes refinements to the algorithm described above. The refinements include efficient detection of exact duplicates, a modified similarity measure, and a new clustering algorithm.

2.2.1 Exact Duplicate Removal

Exact duplicates are either unmodified copies of form letters or comments that someone accidentally submitted multiple times (a fairly common event). They are usually a large proportion of most large public comment datasets. The first refinement is to identify exact duplicates very efficiently.

To identity exact duplicates, all white space is removed from the document, and all words in the document are converted into a long string of characters (a *document string*). A hash function is applied to the document string to create a (nearly) unique identifier for this particular document. This process is applied to all documents in the collection. All documents resulting in the same hash value are considered exact duplicates.

The hash function is the security hash function, SHA1 [16], suggested by NIST. It makes sure that the chance of hash value collision is very low and the whole process is secure. It is also designed to be very fast and is good for messages of any length. It is designed for text processing and is known for its even distribution of hash values. SHA1 produces a 20-byte (160-bit) hash value. By using a secure digest algorithm, it reduces the probability of two different token streams creating the same hash value to $p(2^{160})$.

After hashing, there is a <hash-value, doccument id> tuple for each document. Tuples are sorted by their hash values, then a

[1] http://www-2.cs.cmu.edu/~lemur/

simple linear scan of the list is sufficient to identify documents that have identical hash values.

Given a set of exact duplicates, an arbitrary choice determines which one to consider the reference copy – the seed document. Our system selects the document with the earliest timestamp to be the reference copy, annotates it with the number of exact duplicates, and retains it as a candidate for further study. The rest are marked as exact duplicates and are eliminated from further consideration, although they remain available for reference purposes.

2.2.2 Document Similarity Measure

After a set of candidate near-duplicates is retrieved (Section 2.1), the similarity of each candidate to the cluster seed is measured. Our earlier work [22] used a modified version of KL divergence (relative entropy) to measure the similarity of near-duplicate documents. It first selects a seed document, and then measures the similarity between it and every document in its candidate set. For any two documents d_a and d_b with word probability distributions p_a and p_b respectively, the KL divergence measure of the difference between the two probability distributions is:

$$KL(p_a \| p_b) = \sum_{w_j \in d_a} p_a(w_j) \log \frac{p_a(w_j)}{p_b(w_j)} \quad (2)$$

Since KL-divergence is non-negative and non-symmetric, [22] defines and uses the minimum value of two KL-divergences as the distance measure between two documents d_a and d_b:

$$dist(d_a, d_b) = \min(KL(p_a \| p_b), KL(p_b \| p_a)) \quad (3)$$

One flaw in the prior algorithm is that words that appear in p_a but not in p_b are ignored in the calculation. *Block Edits or Key Blocks*, in which a large block of text is added to or deleted from a form letter, are common in this domain, so it is not unusual for a near-duplicate and its reference copy to have unaligned vocabularies. A modification to the similarity measure solves this problem. Instead of assigning zero weights to an unseen word, it is more effective to give the word a probability proportional to its overall probability in a background language model. The KL distance in Equation (2) becomes:

$$\begin{aligned} KL(p_a \| p_b) &= \sum_w p_a(w) \log \frac{p_a(w)}{p_b(w)} \\ &= \sum_w p_a(w) \log p_a(w) - \sum_w p_a(w) \log p_b(w) \end{aligned} \quad (4)$$

The first term in Equation 4 depends on document distribution p_a and hence is irrelevant to ranking other documents. It can be dropped and the KL divergence becomes:

$$KL(p_a \| p_b) \propto -\sum_w p_a(w) \log p_b(w) \quad (5)$$

$$\text{where } p_b(w) = \begin{cases} p_s(w|d_b) & \text{if w is seen} \\ \alpha_d p(w|C) & \text{otherwise} \end{cases} \quad (6)$$

α_d is a coefficient for each unseen word's probability (and also insures that all of the probabilities sum to one). Hence the KL divergence becomes:

$$-\sum_{w \in d_a} p_a(w) \log \frac{p_s(w|d_b)}{\alpha_d p(w|C)} + \log \alpha_d \quad (7)$$

By Dirichlet prior smoothing [1], we have:

$$p_s(w|d_b) = \frac{tf(w,d_b) + \mu p(w|C)}{\mu + |d_b|}, \quad (8)$$

$$\alpha_d = \frac{\mu}{\mu + |d_b|}, \quad (9)$$

$p_a(w)$ and $p(w|C)$ are estimated by maximum likelihood and given by

$$p_a(w) = p(w_j|d_a) = \frac{tf(w_j, d_a)}{\sum_{w_i \in d_a} tf(w_i, d_a)} \quad (10)$$

$$p(w|C) = \frac{\sum_{d_k \in C} tf(w, d_k)}{\sum_{d_j \in C} \sum_{w_i \in d_j} tf(w_i, d_j)} \quad (11)$$

μ is a parameter in Dirichlet smoothing and is set to 1 in this work.

2.2.3 Incorporating Instance-level Constraints

Near-duplicate detection proceeds more smoothly and efficiently when there are clues about which documents are duplicates. In some duplicate-detection scenarios, files that have identical metadata, such as size, date, and base filename are likely to be copies kept on different directories or on different servers.

Clustering algorithms seek to automatically discover underlying patterns in a dataset. Usually, a search is conducted through the space of possible organizations of the data, preferring those that group similar instances and keep dissimilar instances apart. If additional knowledge about the clustering is known beforehand, the clustering algorithm could be more effective and efficient since we can have pruning at the earlier stage. For example, a user may indicate that a certain pair of documents in the dataset is judged to be similar and a certain other pair of documents is judged to arise from separate clusters. Techniques for introducing additional knowledge to perform constrained clustering have primarily focused on formulating and expressing the knowledge by instance-level constraints [23]. As described in [23] these constraints typically take the form of relations such as *must-link* and *cannot-link* that are enforced between pairs of instances.

In a clustering approach to the problem of near duplicate detection, knowledge about the collection characteristics and document attributes can be used to compute instance-level constraints indicating that certain pairs of documents either must be, or cannot be in the same duplicate cluster.

The must-link conditions include the complete containment of the seed document (*key block*), and minor change < 5% word coverage (*minor change*). The cannot-link condition is only includes for documents that have different email relayers or

A) Initialize the Duplicate Cluster Collection N: $N \leftarrow \square$ and document collection B.

B) Get the initial seed documents and pick one seed d_i Note that the non-seed document will be examined in this process and if there is no cluster for it in the first pass, it will be used as seed in the next pass.

C) Retrieve candidate set S_i for seed document d_i, For each document $s_{ij} \square S_i$,

a) if $(s_{ij},d_i) \square Must$,
add s_{ij} into duplicate cluster n_{dk}: $n_{dk} \leftarrow n_{dk} \quad \{s_{ij}\}$

b) $\forall$ cluster centroid d_k in N, if $(s_{ij},d_k) \square Must$
add s_{ij} into duplicate cluster n_{dk}: $n_{dk} \leftarrow n_{dk} \quad \{s_{ij}\}$

c) if $dist(s_{ij},d_i) < \theta_i$

$\forall$ cluster centroid d_k in N,
if $dist(s_{ij},d_i) > dist(s_{ij},d_k)$,
add s_{ij} into duplicate cluster n_{dk}: $n_{dk} \leftarrow n_{dk} \quad \{s_{ij}\}$
unless $(s_{ij},d_k) \square Cannot$
else if $dist(s_{ij},d_i) <= \min_k (dist(s_{ij},d_k))$ and $d_i \notin N$,
create a new cluster n_{di}, add it into N: $N \leftarrow N \quad \{n_{di}\}$,
add s_{ij} into n_{di}: $n_{di} \leftarrow n_{di} \quad \{s_{ij}\}$ unless $\exists\, d_l \square n_{di}$, $(s_{ij},d_l) \square Must$
eliminate s_{ij} from n_{dk}: $n_{dk} \leftarrow n_{dk} - \{s_{ij}\}$, unless $\exists\, d_l \square n_{di}$, $(s_{ij},d_l) \square$ Cannot

E) If $B = \square$, output N as the final set of duplicate clusters.

Figure 1: Algorithm to form duplicate clusters

docket ids. Note that instance-level constrained clustering is very flexible; as more background knowledge about the dataset is acquired, it can be added as new constraints.

3. EVALUATION METHEDOLOGY

Clustering and near-duplicate detection algorithms are difficult to evaluate because "ground truth" information is rarely available. This section describes the datasets used in our experiments, our evaluation metrics, and in particular our methodology for acquiring "ground truth" assessments.

3.1 Data Sets

Our research was conducted with a public comment dataset for the U.S. Environmental Protection Agency's (EPA) proposed National Emission Standards For Hazardous Air Pollutants For Utility Air Toxics rule (USEPA-OAR-2002-0056, "Mercury rule"). The dataset contains 536,975 email messages. The algorithm successfully ran on the entire dataset.

However, it is impractical to have human assessment on the entire dataset. To make the human assessment doable, two random samples of size 1.000 were generated as evaluation set. The exact duplicates were also removed and left 275 and 270 documents in what became known as the *NTF (Name That Form)* and *NTF2* subsets. Table 1 provides statistics about these samples. Section 3.3 provides additional description of these datasets.

3.2 Evaluation Metrics

The accuracy of near-duplicate detection was measured using well-known evaluation metrics such as Precision, Recall and F1-measure [22], and intercoder agreement metrics such as Cohen's Kappa and AC1. The experiments were designed to determine whether the system's agreement with human assessors was comparable to agreement between two or more human assessors.

Table 1: Sample Dataset Statistics

Sample Set Name	NTF	NTF2
Source	USEPA-OAR-2002-0056, "Mercury rule"	USEPA-OAR-2002-0056, "Mercury rule"
# of documents originally	1000	1000
# of documents after exact duplicates removal	275	270
# of reference copies (document with > 5 exact duplicates)	28	26
average document length before removing header/signature lines	220	213
average document length after removing header/signature lines	156	152
# unique terms	3330	3437

Cohen's Kappa: Cohen's kappa statistic [4] is often used to evaluate clustering effectiveness. It assesses agreement between two sets of results or in another word, two coders. Therefore in our experiments, the intercoder agreements are always measured between two coders. The kappa coefficient is defined is:

$$\kappa = \frac{p(A) - p(E)}{1 - p(E)} \qquad (12)$$

where $p(A)$ is the observed agreement between the two assessments, 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 different in the in the clustering (false negative), c is the number of pairs in the different groups in the ground truth but 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). $p(A)$ can be calculated as $(a+d)/m$ and $m=a+b+c+d$. $p(E)$ is the agreement expected by chance, and is calculated as:

$$p(E) = (a+b)(a+c)/m^2 + (b+c)(c+d)/m^2 \qquad (13)$$

AC1: Cohen's kappa suffers from problems of bias and prevalence. If the agreement between assessors is high but skewed to a few categories, as is common in public comment datasets, the resulting kappa value cannot truly represent the degree of agreement [9]. AC1 corrects this problem and calculates the chance agreement in another way:

$$p(E) = 2P1(1-P1) \text{ where } P1 = ((a+b)+(a+c))/2m. \qquad (14)$$

Although kappa is not appropriate for our task, it is used often in prior research, so we report both kappa and AC1 values. We

P59: 029932.txt - 59:1 [The missuse of power here is incredible!!!! Doesn'..] (5:5) (Pitt2)

Codes: [Disappointment] [Economic] [Public Health & Safety] [Social Values] [Strength=High] [Unique Text in a Form Letter]

No memos

The missuse of power here is incredible!!!! Doesn't the welfare of our children come first? Enact more regulations for big business and keep our children safe!!!!! There is no excuse for such blatant mishandling of this situation.. Act Now!!!!!

Figure 2: Human Annotation Example

believe that AC1 is a reliable measure for this task, thus we mainly rely on it to draw conclusions.

3.3 Human Annotation Methodology

The authors developed a manual annotation (*coding*) methodology to guide student annotators in identifying near-duplicate public comments and unique texts. The coding system was developed and deployed through an iterative process using coders at the University of Pittsburgh's Qualitative Data Analysis Program (QDAP). The QDAP coders used ATLAS.ti, a commercial off-the-shelf qualitative data analysis application, to capture and report their annotations.

3.3.1 Evolution of the Coding Scheme

In its earliest iteration, during the spring of 2005, the current coding scheme was designed primarily to serve social science goals related to the project. It contained 28 distinct sub-topic and discourse style codes, such as "Legal," "Economic," "Public Health & Safety," "Personal Experience," and "Strength-High," as well as a code for "Unique Text in a Form Letter." Five coders were trained to identify sub-paragraph and paragraph level instances of these codes in a random sample of 1,000 e-mails from a different dataset. The sample was divided to ensure a unique 2-coder overlap on 320 documents. In addition, at the document level, the coders were expected to identify whether the document was an exact duplicate or a near duplicate. An example of this earliest coding scheme is shown in Figure 2.

Analysis of the first round of coding revealed that many of the sub-topic and document-level codes were applied inconsistently. The overall measure of inter-rater reliability was extremely low. This necessitated further training sessions and a more thorough clarification of the coding heuristics. After the retraining of the coders and returning them to review and correct their annotations, the F-measure of inter-rater reliability for all codes combined (including overlapping spans of text) remained low (0.53). While the coders were still struggling to consistently apply some of the amorphous and insufficiently defined subtopic codes, they showed promise identifying stakeholders (0.77) and unique text added to a form letter (0.89). The 28 subtopic codes required multiple passes by the coders on codes and clusters to correct the many errors of the first unwieldy round of coding. Altogether they looked at the initial set of texts 3 times and still produced generally unreliable coding.

For the purpose of preparing this paper, a new coding approach was developed in mid-August 2005. Using lessons from the first round, a second coding was specifically tailored to the needs of researchers developing and evaluating near-duplicate detection tools and federal agencies trying to manage large public comment campaigns. In the modified coding scheme, a new sample of 1,000 e-mails was selected and the exact duplicates were removed using the exact-duplicate detection algorithm described above (Section 2.2.1), leaving 275 documents in the NTF pool.

In the first NTF round, coders were provided with a set of 28 "known form letters", defined as comments that had 5 or more exact duplicates in the dataset. Two coders were trained to apply a single code (Unique Text). The NTF experiment also required the coders to associate each comment with one of the 28 known form letters, or to identify it as unique. Three further document-level labels were also used ("Block Edit," "Minor Change," and "Singleton"[2]). The initial NTF round, with its narrow focus on coding for near-duplicate detection, predictably produced a high measure of inter-rater reliability (0.82) for the unique text code on the first pass, with no need to retrain the coders or redo the coding. It also highlighted the need to better specify a precise rule set that more effectively defined the proper spans of text to be annotated. In the NTF round, one coder annotated almost 50% more text spans than the other. Note that the NTF rounds were completed with a single pass on clusters and a single pass on codes. Different from the previous 28 sub-topic coding scheme, which requires multiple annotation passes to reach a reasonable intercoder agreement, NTF round coding resulted in higher intercoder agreement at the first time.

A review of the NTF coding raised the possibility that introducing a second code ("Signature") might reduce the variance between the coders and produce a more accurate annotation for the code "Unique Text." The NTF2 round did result in high measures of reliability for both "Signature" (0.95) and "Unique Text" (0.87).

By early November, a decision was made to repeat the coding of the NTF and NTF2 samples using a new set of six coders who had no prior NTF experience. Three of the six were new to coding in QDAP, having just completed their ATLAS.ti training, while the remaining three were relatively experienced and acted as mentors for the new coders. A more precisely specified rule set was generated and relayed to the experienced coders. For both the new NTF and NTF2 rounds, two and four coders were used respectively. Each of these experiments required coders to use a total of four codes, with the addition of "Header" and "Stakeholder" to "Unique Text" and "Signature." Both rounds offered unique insights into the techniques for training coders to produce reliable annotations. Moreover, as mentioned in the previous sections, to group duplicates into subcategories by the editing styles is a common strategy. In the two datasets NTF and NTF2, human assessors identified the block added, block deleted, minor change, block rearrange and singleton document and "repeated copies". Note that "repeated copies" refers to a special situation where the form letters are repeated several times in the submitted public comments and the public comments contain only the repeated paragraphs.

[2] We call it *singleton* since it forms a cluster by itself alone in the clustering algorithm.

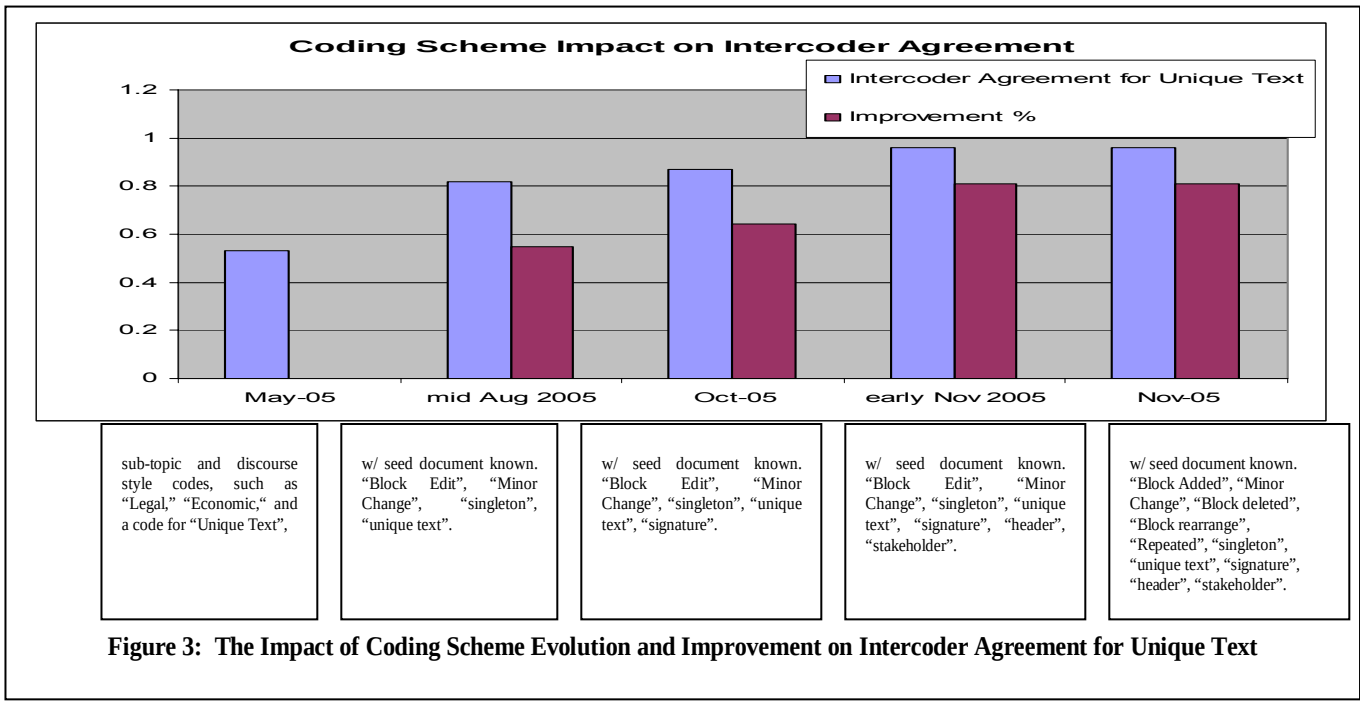

Figure 3: The Impact of Coding Scheme Evolution and Improvement on Intercoder Agreement for Unique Text

Figure 3 illustrates the impact of the coding scheme evolution and the improvement gained in the intercoder agreement, in particular, for the code of "unique text" only. The more precise coding scheme produces much higher and more intercoder agreements between human annotators.

The latest NTF round produced the most reliable coding to date of any pair of QDAP coders working on any project. As reported in Table 2 the F-measure for exact matches (0.91) and matches including overlapping text spans (0.98), suggest this is a gold standard for what two coders with a consistent, narrow and very precise rule set can accomplish. The NTF2 round (Table 3) also resulted in solid inter-rater-reliability coefficients, with an average F-measure of 0.90 across all four codes and coders.

3.3.2 Training Effects

Repeated rounds of coding for near-duplicates taught us something important about training effects in manual annotation. In new NTF2, experienced coders UCSUR 8 & 9 received instructions directly from the QDAP Director and then independently relayed them to UCSUR 15 & 17. This to say, UCSUR 8 trained 17 and 9 trained 15.

As Table 3 shows, there is a pattern perhaps reflecting differences in how 8 trained 17 and 9 trained 15 to code "Unique Text" since the number of "unique text" found by coder 8 is close to 17's while 9's is closer to 15's. It is interesting to study more the training effects among the coders. We conducted 4 intercoder agreements for each pair of the coders. They are: kappa (w/o overlap), which is the cohen's kappa agreement of two coders for the texts that exact match; kappa, which is cohen's kappa for the texts including partial match (a more Lenin evaluation), F-measure (w/o overlap), which is the F-measure of two coders for the texts that exact match by using one coder as the ground truth and calculate the other one using traditional F-measure in information retrieval; F-measure, which is F-measure for the texts including partial.

Table 2: NTF Coding Results

Code	# found by 13	# found by 16	Kappa (w/o overlap)	Kappa (w/ overlap)	F-measure (w/o overlap)	F-measure (w/overlap)
Header	266	266	0.93	0.99	0.94	0.99
Signature	281	283	0.94	0.98	0.95	0.98
Unique Text	218	220	0.86	0.96	0.86	0.96
Total	798	799	0.92	0.98	0.91	0.98

Table 3: NTF2 Coding Results with Training Effects

Code	# found by 8	# found by 9	# found by 15	# found by 17	Avg F-measure (w/o overlap)	Avg F-measure (w/overlap)
Header	264	262	264	265	0.93	0.99
Signature	276	281	275	277	0.91	0.96
Unique Text	**213**	**146**	**171**	**214**	0.61	0.72
Total	781	715	723	778	0.83	0.90

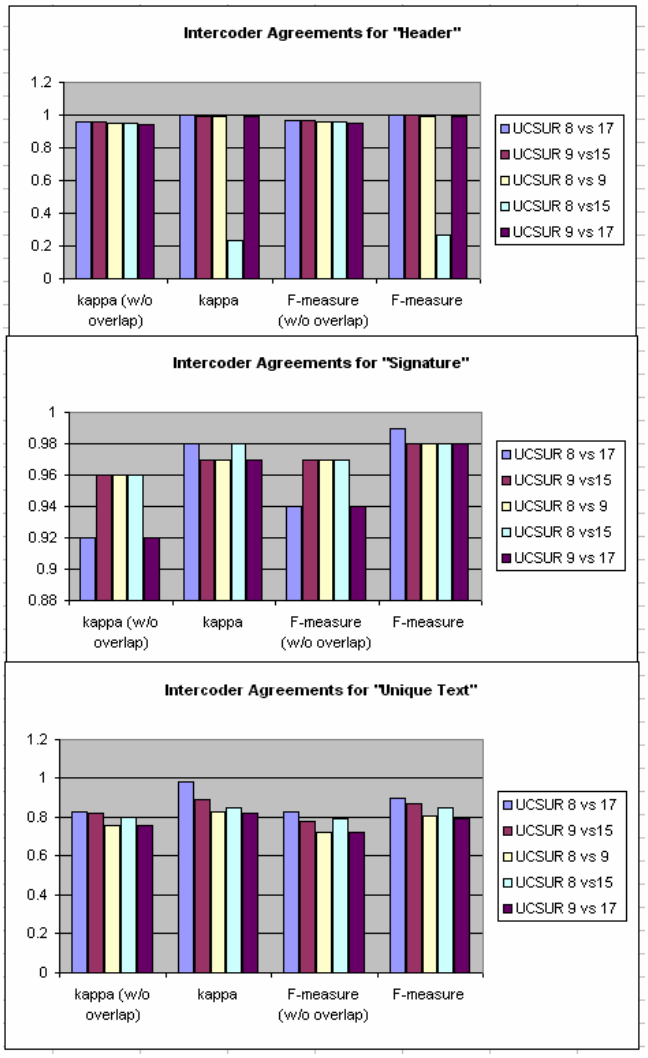

Figure 4: Training Effects on Intercoder Agreements

Figure 4 illustrates the possible presence of training effects on the coding of "header", "signature" and "unique text". We can see the most obvious effects on the "unique text", where the pairings of coder 8 vs. 17 and coder 9 vs. 15 give the top two intercoder agreements among all four kinds of agreements. Interesting enough to notice that the intercoder agreements of coder 8 and 15 are also higher than that of the two senior coders (8 and 9) who are directly trained by the QDAP Director. We are not sure whether it means that coder 15 actually understands the coding scheme best. However, we do use the annotation from coder 15 as a gold standard for the later duplicate detection tests.

3.3.3 Duplicate Editing Styles

In the latest runs, the subcategories of near-duplicates, or in other words, the editing styles of near-duplicates are considered as part of the coding scheme. It is interesting to see what makes up the near-duplicates. It is also very important for us to study the effectiveness of the automatic near-duplicate detection algorithm on each subcategory. Figure 5 and Figure 6 show the duplicate editing style distribution for the NTF and NTF2 datasets. The

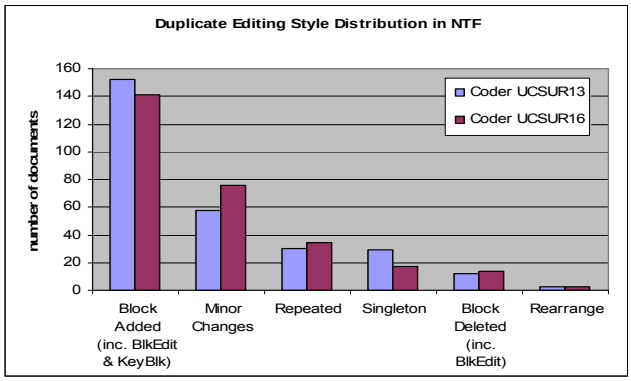

Figure 5: Duplicate Editing Style Distribution in NTF

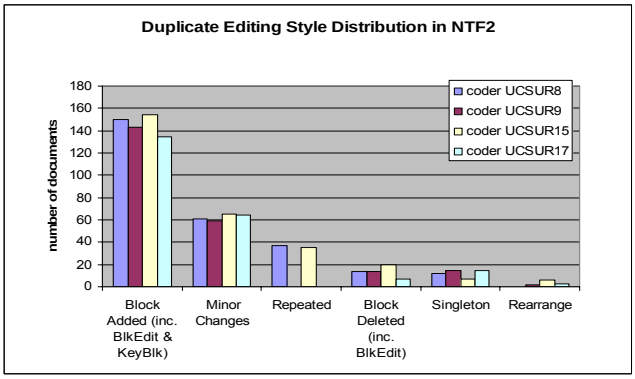

Figure 6: Duplicate Editing Style Distribution in NTF2

annotations from coders UCSUR13 and UCSUR16 are used for NTF and annotations from coders UCSUR8, UCSUR9, UCSUR15 and UCSUR17 are used for NTF2. Based on the two figures, we have the following observations.

- Block Added is the dominant editing style for people submitting public comments. Block Edit and Key Block both belong to this category.
- Minor Change is the next major editing style for near-duplicates in the public comment domain.
- Singleton documents are unique comments not based on a form letter. They could contain unique insights and opinions, derivative opinions, spam or viruses. The amount of singleton documents in both datasets is reasonably large. These are of potentially of interest for social science research. Another important source of possibly-unique opinions added to form letters (Block Added).
- Note that for NTF2, two coders, UCSUR9 and UCSUR17 considered that no documents belong to the "exact copy" category while the other two coders considered that there are at least 30+ documents in this category. This is an interesting disagreement that we cannot yet explain.

4. EXPERIMENTAL RESULTS

4.1 System-to-Human Intercoder Agreement

The first set of experiments explored the agreement between DURIAN and human assessors at identifying near-duplicate comments, unique passages in near-duplicate comments, and unique comments. In these experiments DURIAN was considered another coder, and accuracy is measured using the intercoder agreement metrics discussed previously. Experiments were conducted on the NTF and NTF2 datasets. NTF has 28 form letters while NTF2 has 26.

Usually intercoder agreement is calculated across all pairs of documents (*micro-averaging*). However, it is also useful to measure intercoder agreement for each cluster and then to average over the number of clusters (*macro-averaging*). Macro-averaging focuses on agreement on the large letter-writing campaigns, whereas micro-averaging gives a better measurement over the set of letter-writing campaigns.

In the following tables, we use Coder A to represent UCSUR 13 for NTF and UCSUR8 for NTF2; and Coder B to represent UCSUR 16 for NTF and UCSUR9 for NTF2. We pair them this way because these are the best combinations of assessors in terms of intercoder agreement between humans. They are the gold standard.

Table 4: Duplicate Detection Intercoder Agreement

	Averaged across all clusters				Averaged across all pairs			
	Kappa(Cohen)		AC1		Kappa (Cohen)		AC1	
	NTF	NTF2	NTF	NTF2	NTF	NTF2	NTF	NTF2
Coder A/ Coder B	0.53	0.19	0.93	0.90	0.99	0.97	0.99	0.95
Coder A/Program	0.52	0.15	0.92	0.80	0.90	0.89	0.93	0.90
Coder B/Program	0.53	0.17	0.90	0.82	0.92	0.90	0.91	0.91

Table 4 summarizes the first set of experimental results. The macro-averaged Cohen's kappa values are surprisingly low for both the human-to-human and program-to-human intercoder agreements. This is surprising because the agreement between coder A and B about whether a document belongs to a certain cluster is actually high in raw numbers, and there is prevalence in the agreement categories. The agreement about the number of documents in the same cluster is much higher than the agreement about the number of document not in the same cluster (i.e., a>>d in the kappa value calculation). This unbalanced distribution causes Cohen's kappa to fail to represent the true agreement.

The macro-averaged AC1 values are high. Given that AC1 is a stable agreement measure even when prevalence and bias problems are present, we believe that this metric accurately reflects the degree of agreement between humans and DURIAN.

In the pairwise comparison, Cohen's kappa and AC1 both show high intercoder agreements. This is very desirable, however the result might be misleading. There are some very large letter writing campaigns in these datasets, which creates huge duplicate clusters. Huge clusters tend to dominate the final result of pairwise comparison since they have more pairs. Thus macro-averaged AC1 is probably the best measure to use for evaluating the near-duplicate clustering. DURIAN's agreement with human coders as about good as the agreement between pairs of humans.

In addition to identifying near-duplicates and forming duplicate clusters, DURIAN is able to recognize header blocks, signature lines, and unique text in a public comments. We again use both Cohen's kappa and AC1 as the evaluation metrics. Tables 5-7 summarize the results.

Table 5: Unique Text Intercoder Agreement

	Kappa (Cohen)		AC1	
	NTF	NTF2	NTF	NTF2
Coder A/ Coder B	0.96	0.83	0.98	0.82
Coder A/Program	0.80	0.76	0.86	0.74
Coder B/Program	0.78	0.75	0.84	0.74

Table 6: Header Detection Intercoder Agreement

	Kappa (Cohen)		AC1	
	NTF	NTF2	NTF	NTF2
Coder A/ Coder B	0.99	0.99	0.99	0.99
Coder A/Program	0.93	0.92	0.93	0.91
Coder B/Program	0.93	0.92	0.93	0.91

Table 7: Signature Line Detection Intercoder Agreement

	Kappa (Cohen)		AC1	
	NTF	NTF2	NTF	NTF2
Coder A/ Coder B	0.98	0.97	0.99	0.97
Coder A/Program	0.90	0.92	0.90	0.91
Coder B/Program	0.90	0.90	0.90	0.89

Unique text is the text a person added to a form letter; it might raise a substantive issue, so it requires agency review. The human assessors have a high agreement in NTF and a lower but still very good agreement on NTF2. However, the agreements of the program with both human assessors are lower, perhaps just above what we would consider acceptable. The human assessors only consider a major addition of text to the original form letter to be "unique text". The program, however, is sensitive to changes varying from the large block changes to even small punctuation changes. Table 5 suggests that DURIAN might benefit from some tuning to be less sensitive to small changes that human coders consider trivial.

As a preprocessing step, DURIAN's header and signature detection algorithms were very effective. Although pleasing, this result may simply indicate that that public comments based on form letters mainly follow a formal letter-writing style that makes it easy to detect the header and signatures lines.

Cohen's Kappa and AC1 tend to agree about unique text, header and signature line detection accuracy. Pairwise comparisons were used for Table 5-7 because cluster-based comparisons are less meaningful when evaluating header, signature, and unique text capabilities. The skewed distribution problem does not happen in this case, thus Cohen's kappa is reliable.

4.2 Duplicate Detection Algorithms

This paper presents DURIAN, a new near-duplicate detection algorithm designed for public comment datasets and notice and comment rulemaking tasks. However, other algorithms also detect duplicate documents and might be applied to this task. A set of experiments was conducted to evaluate several well-known duplicate-detection algorithms on the NTF and NTF2 datasets. The contenders are described below.

Full fingerprinting (full): Every substring of size *s* in the documents are selected and hashed. *s* is set to 3 in our experiments. Every hash value (a fingerprint) is stored for the document in a form of <fingerprint, document id> tuple. Every substring will contributes one such tuple since every substring resulting one fingerprint. Therefore we have a huge list. The duplicate detection is performed by 1) sorting the tuples <fingerprint, document id>; 2) generating overlapping fingerprint records, if a document with id =567 contains a fingerprint in the form letter, a tuple <567, 1> is generated; 3) counting the overlap fingerprint records for all documents, we get <document id, count>. If the count of the overlap fingerprints in a document to the form letter is above 80%, it is considered as a duplicate.

Shingling (DSC) [2]***:*** Every 5 overlapping substring of size *s* in the documents is selected and hashed. *s* is set to 3 again in our experiments. The hash values are stored for each document. Duplicate detection is performed in the same way as described in full fingerprinting. If the count of the overlap fingerprints in a document to form letter is above 80%, it is considered as a duplicate to form letter.

I-Match [3]***:*** The *N* words with the highest idf values in a document are selected, *N* is set to 30 in our experiments. Note that the top 5 idf words are ignored here since they might be some random mistakes such as misspellings. A single fingerprint is generated for each document. Duplicate detection is performed by sorting all <fingerprint, document id> tuples. Those documents agree with the fingerprint of the form letter, are selected as the (near) duplicates.

Durian: The algorithm proposed in this paper.

The parameters used in all the experiments for competing methods were tuned using parameter sweeps and/or the best values reported in other researcher's work [10][6].

In this experiment we assume that the form letters are known, and that the task is to identify the near-duplicates identified by the human coders UCSUR16 (NTF) and UCSUR15's (NTF2). In order to study the effectiveness of duplicate detection techniques on different duplicate categories, the detection results are further distributed into different duplicate categories. The average precision, average recall, and average F1-measure for each category averaged for each form letter are reported here.

Full fingerprinting was the most simple substring selection technique. It gave the largest possible set of fingerprints for a document. Not surprisingly, it either gives the best or the second best F1 value in every category. Full finger printing is very effective, however it is also the most computationally expensive method. Since every substring is stored as a hash number, the effort of sorting a huge list of tuples is unavoidable. Both the storage and execution time is very costly.

Durian consistently performs well on all the categories, occasionally beating the full fingerprint approach. However, the retrieval and detection time is much lower than for full fingerprinting.

Two other techniques DSC and I-Match were not as effective as full fingerprinting and DURIAN. In general, DSC outperforms I-Match. I-Match is very sensitive to both Block Added and Block Deleted. It is also very sensitive to Text Minor Change. When the changed words are critical, i.e., appear in the fingerprint that I-Match selected, the algorithm fails to detect the near-duplicates. In general, I-Match produced fairly low Precision and the Recall.

Table 8: Comparison of duplicate detection technologies

Duplicate category	Algorithm	Avg Precision		Avg Recall		Avg F1	
		NTF	NTF2	NTF	NTF2	NTF	NTF2
Exact	Full	1	1	1	1	**1**	**1**
	DSC	0.97	0.98	0.98	0.98	0.97	0.98
	I-Match	0.91	0.9	0.8	0.75	0.85	0.82
	DURIAN	1	1	1	1	**1**	**1**
Minor change	Full	0.95	0.95	0.95	0.95	0.95	0.95
	DSC	0.9	0.9	0.9	0.9	0.9	0.9
	I-Match	0.79	0.8	0.76	0.78	0.77	0.79
	DURIAN	0.95	0.95	1	1	**0.98**	**0.97**
Block Added	Full	0.97	0.98	0.98	0.98	**0.98**	**0.98**
	DSC	0.73	0.7	0.74	0.78	0.73	0.74
	I-Match	0.32	0.35	0.4	0.42	0.36	0.38
	DURIAN	0.98	0.98	0.98	0.98	**0.98**	**0.98**
Block Deleted	Full	0.9	0.9	0.9	1	**0.98**	**0.95**
	DSC	0.72	0.75	0.74	0.78	0.73	0.76
	I-Match	0.3	0.33	0.4	0.4	0.36	0.36
	DURIAN	0.98	0.98	0.98	0.98	**0.98**	**0.98**
Singleton	Full	0.9	0.9	0.9	0.9	0.93	0.9
	DSC	0.72	0.74	0.8	0.8	0.76	0.77
	I-Match	0.84	0.88	0.78	0.8	0.81	0.84
	DURIAN	0.94	0.94	0.94	0.94	**0.94**	**0.94**
Rearrange	Full	1	1	1	1	**1**	**1**
	DSC	0.67	0.83	0.67	0.83	0.67	0.83
	I-Match	1	1	1	1	**1**	**1**
	DURIAN	1	1	1	1	**1**	**1**

5. CONCLUSION AND NEXT STEPS

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 research focused on the process of identifying near-duplicates of form letters, and unique passages added to form letters. An extensive study of human intercoder agreement on public comments provided by the Environmental Protection Agency set a baseline against which to evaluate automated techniques. This paper demonstrates that statistical similarity measures and instance-level constrained clustering can be quite

effective for efficiently identifying near-duplicates. In some tasks the algorithm is comparable to our best human assessors; in other tasks it is slightly less effective than our best assessors, but perhaps sufficiently effective for production use. When the algorithm "fails", it tends to do so by classifying a modified form letter as a unique comment, thus referring it for human review.

The current study reports results against two samples of one corpus of public comments to the EPA. We are currently at work on evaluation using another corpus provided by another agency. Additional experiments will provide greater insight into the strengths, weaknesses, and generality of the algorithm.

Whether, how or when this type of technology will emerge as a factor in regulatory rulemaking is beyond the scope of this paper. Optimists may read this report as a bellwether signaling the imminent end of a temporarily vexing problem. Mass comment campaigns will be more manageable when tools such as ours make a successful technology transfer into the hands of agencies receiving the comments. Furthermore, if agency personnel are so inclined, the addition of unique stakeholder views to mass comment campaigns will come into much greater focus with much less effort and expense.

ACKNOWLEDGMENTS

We are grateful to the USDA, US DOT, and US EPA for providing the public comment data that made this research possible. 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. We are also grateful to invaluable comments from the anonymous reviewers.

6. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] J. Cohen. A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20, 37-46, 1960.

[5] C. Coglianese, E-Rulemaking: Information Technology and the Regulatory Process. Administrative Law Review 56(2): 353-402. 2004.

[6] 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 New Orleans, LA, USA, 2003.

[7] F. Emery and A. Emery, A Modest Proposal: Improve E-Rulemaking by Improving Comments. Administrative and Regulatory Law News, 31(1): 8-9. 2005.

[8] Government Accountability Office. Electronic Rulemaking: Progress Made in Developing a Centralized E-Rulemaking System. GAO-05-777, 2005.

[9] K. Gwet. Kappa Statistic is not Satisfactory for Assessing the Extent of Agreement between Raters. Statistical Methods for Inter-rater Reliability Assessment, No.1, April 2002.

[10] T. Hoad and J. Zobel. Methods for identifying versioned and plagiarized documents. In Journal of the American Society or Information Science and Technology, Volume 54, Issue 3, 2003.

[11] C.M. Kerwin, Rulemaking: How Government Agencies Write Law and Make Policy 3rd Ed. CQ Press, Washington, DC, 2003.

[12] G.T. Lau, K.H. Law, and G. Wiederhold,. A Relatedness Analysis Tool for Comparing Drafted Regulations and Associated Public Comments. I/S 1(1): 95-110. 2005.

[13] J.S. Lubbers, A Guide to Federal Agency Rulemaking. Third Edition. Chicago, ABA, 1998.

[14] U. Manber. Finding similar files in a large file system. In 1994 Winter USENIX Technical Conference, pages 1-10, San Francisco, CA, January 1994.

[15] D. Metzler, Y. Bernstein and W. Bruce Croft. Similarity Measures for Tracking Information Flow, Proceedings of the fourteenth international conference on Information and knowledge management, CIKM'05, October 31.November 5, 2005, Bremen, Germany.

[16] NIST, "Secure Hash Standard", Federal Information Processing Standards Publication 180-1, 1995.

[17] N. Shivakumar and H. Garcia-Molina. SCAM: a copy detection mechanism for digital documents. In Proc. International Conference on Theory and Practice of Digital Libraries, Austin, Texas, June 1995.

[18] S. Shulman, L. Thrane, and M.C. Shelley. eRulemaking, in G. David Garson (Ed.) The Handbook of Public Information Systems 2nd Ed. CRC Press, Boca Raton, FL, 2005, 237-254.

[19] S.W. Shulman, E-Rulemaking: Issues in Current Research and Practice. International Journal of Public Administration 28: 621-641. 2005.

[20] S.W. Shulman, The Internet Still Might (But Probably Won't) Change Everything. I/S 1(1): 111-145. 2005.

[21] S.W. Shulman, An Experiment in Digital Government at the U.S. National Organic Program. Agriculture and Human Values 20(3): 253-265, 2003.

[22] H. Yang and J. Callan. Near-Duplicate Detection for eRulemaking. In Proceedings of the 5th National Conference on Digital Government Research (DG.O2005), Atlanta, GA, USA, 15-18 May 2005.

[23] K, Wagstaff and C, Cardie, 2000. Clustering with instance-level constraints. In Proceedings of ICML-2000. pp. 1103–1110. Palo Alto, CA.

[24] C. Zhai and Lafferty, J. (2001b). A study of smoothing methods for language models applied to ad hoc information retrieval. In Proceedings of SIGIR 2001, pages 334-342.