Hierarchical Comments-Based Clustering

Chiao-Fang Hsu
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77843
drakishu@cse.tamu.edu

James Caverlee
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77843
caverlee@cse.tamu.edu

Elham Khabiri
Department of Computer Science and Engineering
Texas A&M University
College Station, TX 77843
khabiri@cse.tamu.edu

ABSTRACT

Information resources on the Web like videos, images, and documents are increasingly becoming more "social" through user engagement via commenting systems. These commenting systems provide a forum for users to discuss the resources but have the side effect of providing valuable editorial and contextual information about the resources. In this paper, we explore a comments-driven clustering framework for organizing Web resources according to this user-based perspective. Concretely, we propose a hierarchical comment clustering approach that relies on two key features: (i) comment term normalization and key term extraction for distilling noisy comments for effective clustering; and (ii) a real-time insertion component for incrementally updating the comments-based hierarchy so that resources can be efficiently placed in the hierarchy as comments arise and without the need to re-generate the (potentially) expensive hierarchy. We study the clustering approach over the popular video sharing site YouTube. YouTube is a challenging and difficult environment, notorious for its extremely short, ill-formed, and often unintelligible user-contributed comments. Through extensive experimental study, we find that the proposed approach can lead to effective and efficient comments-based video organizing even in a YouTube-like environment.

1. INTRODUCTION

One of the cornerstones of emerging participatory information environments – like Web 2.0 social news aggregators, social media sites, digital libraries incorporating social computing features, etc. – is the emphasis on *user-driven commenting and discussion*. By encouraging users to comment, resources in these systems (like videos, images, news articles) can become "social" resources that reflect the attitudes and interests of the community of users in a way that may depart from the viewpoint of system experts, editors, and the content of the underlying information resource itself. Popularized by weblogs, commenting systems are now in wide use by major media (e.g., NYTimes), social media sites (e.g., YouTube, Flickr), and other participatory environments. This rising prevalence of user-contributed comments is inspiring new approaches for enhancing how users view and access information resources in these systems. As an example, NYTimes now prominently features highly-rated comments as an added dimension to their article. From a search perspective, researchers are examining techniques for retrieving and ranking information resources via comments [23].

Since user-contributed comments provide a potentially rich source of contextual information, we are interested to study whether these comments can be used to automatically self-organize a collection of information resources. In this way, the comments themselves may provide a "semantic overlay" that groups similar resources by the collaborative user perspective encoded in the comments (rather than editorial grouping, e.g., into "News" or "Sports"). Our vision is a self-organizing collaborative information sharing space where user comments are automatically reflected in how resources are organized. Concretely, we study one popular approach for organizing resources – hierarchical clustering. Hierarchical clustering has been widely studied in the context of structuring text documents (like Web pages and email) [1, 6, 11, 19, 21] and has shown success in improving information search and browsing [4, 15, 22].

Automatically clustering resources by their comments is challenging, however. Comments are typically free-form and highly unstructured with users engaging in a wide variety of commenting purposes, including: (i) describing the underlying resource; (ii) engaging in a back-and-forth discussion with other users; (iii) expressing emotional reaction (e.g., "Awesome!"); (iv) providing new perspective and pointers (e.g., summarizing a related article and adding a hyperlink). In addition to the variation in purpose and substance, comments are often syntactically "messy" with a huge variation in quality and style. Spelling errors (both intentional and not), grammatical errors, typos, and shorthand are all typical of the comments generated by a large group of (typically) volunteer commenters. These challenges suggest that effective clustering may be dependent on high-quality comment distillation – for finding the "essence" of a community's collective comments. Additionally, since comments themselves are dynamic (with comments being written at unknown time intervals and reflecting the changing perspective of different commenters), any proposed automatic clustering approach should be designed to balance stable resource clustering (by considering all possible comments) with timely resource clustering (by immediately organizing a resource according to the first comment).

With these challenges in mind, we present a comments-based hierarchical clustering approach for organizing information resources. Two of the salient features of the proposed approach are its (i) comment term normalization and key term extraction for distilling noisy comments for effective clustering; and (ii) a real-time insertion component for incrementally updating the comments-based hierarchy so that resources can be efficiently placed in the hierar-

SAC'11 March 21-25, 2011, TaiChung, Taiwan.

chy as comments arise and without the need to re-generate the (potentially) expensive hierarchy. Concretely, we study the clustering approach over the popular video sharing site YouTube. YouTube is a challenging and difficult environment, notorious for its extremely short, ill-formed, and often unintelligible user-contributed comments. Through extensive experimental study, we find that the proposed approach can lead to effective and efficient comments-based video organizing even in a YouTube-like environment.

Our main contributions can be summarized as follows:

- We propose to automatically organize information resources in participatory information environments through comments-based hierarchical clustering.
- We introduce a comments distillation framework for extracting core comment features via comment term normalization and KL-divergence-based key term extraction for enabling effective clustering.
- We present an adaptive resource insertion component for incrementally updating the comments-based hierarchy, leading to efficient and accurate updates.
- Finally, we evaluate the proposed approach over a large YouTube video testbed and find that the proposed methods achieve high quality results while maintaining efficiency.

The rest of the paper is organized as follows. We discuss related work in Section 5. Section 2 introduces the overview of hierarchical comments-oriented clustering and outlines two key challenges to successful clustering. The following two sections – Section 3 and Section 4 – address these two key challenges by introducing and evaluating methods for distilling user-contributed comments to an informative core and for adaptively updating the resource hierarchy via a comments-driven resource insertion component. We conclude in Section 6 with some final thoughts.

2. PROBLEM STATEMENT

In this section, we present the formal problem statement for organizing information resources using user-generated comments. Our goal is to automatically construct a self-organizing hierarchy of information resources so that the community perspective of comments can drive how resources are grouped and linked. A hierarchical structure can provide users with more freedom in browsing through different level of granularity, from high-levels grouping many loosely-related resources to lower-levels with fewer and more tightly-related resources. Concretely, we approach the problem of organizing resources (like videos, images, etc.) as a *comments-based hierarchical clustering problem*.

2.1 Hierarchical Clustering

A hierarchical clustering algorithm takes as input a set of k resources (e.g., Web documents, images, videos) $O = \{o_1; o_2; ...; o_k\}$ and outputs a rooted hierarchical tree (called a dendrogram), such that each leaf of the tree represents one of the k resources; intermediate nodes in the tree represent subclusters of resources (combining all resources below that node in the tree); and the root of the tree is a super-cluster containing all k resources. The final hierarchical structure is a set of clusters $\Omega = \{\omega_1; \omega_2; ...; \omega_m\}$ where each cluster node ω has a set of up to c resources $\omega = \{o_1; o_2; ...; o_c\}$ and two children nodes $\omega \rightarrow \{\omega_{child_1}, \omega_{child_2}\}$. All leaf nodes contain only a single resource $|\omega_{leaf}| = 1$ and no children. Compared to flat clustering algorithms (like k-means [8], probabilistic models like Latent Dirichlet Allocation [2], and graph-based approaches [12]), hierarchical clustering algorithms have the advantage of revealing more structure and allowing greater freedom of navigation from cluster to cluster (by ascending and descending the hierarchy), though at the cost of less efficient computation (typically with time complexity of $O(N^3)$).

Hierarchical clustering algorithms are either bottom-up (agglomerative) or top-down (divisive). A divisive clustering algorithm begins with all resources in a single cluster and recursively divides clusters into sub-clusters until each resource belongs to a leaf node. An agglomerative algorithm begins with each resource as a leaf cluster, and subsequently merges clusters until all resources belong to a single root node. In this paper, we focus on the widely-studied agglomerative clustering approach illustrated in Algorithm 1.

Algorithm 1 Hierarchical Resource Clustering

1: Assign resource representation using term vector derived from comments associated with the resource
2: Let each resource o be in a singleton group $\omega = o$.
3: Set all Ω as available for clustering.
4: **while** $|\Omega_{available}| > 1$ **do**
5: Choose most similar ω_α , $\omega_\beta \in \Omega_{available}$ according to similarity $s(\omega_\alpha, \omega_\beta)$
6: Mark ω_α and $\omega_\beta \in \Omega_{available}$ as "not available"
7: Insert $\omega_\alpha \cup \omega_\beta$ into Ω

2.2 Comments-Based Representation

Given the hierarchical clustering algorithm, it is still an open question as to how to represent each resource and how to determine which clusters are most similar for merging. We adopt a "bag-of-words" comments-based resource representation such that each resource o_i is represented by the text of the n comments associated with it $C_i = \{c_{i_1}; c_{i_2}; ...; c_{i_n}\}$, where each comment c_{ij} has a set of m terms $T_{c_{ij}} = \{t_1; t_2; ...; t_m\}$ associated with it. Each resource is then initially represented as the feature vector $\vec{v}$ over all terms in all comments (we shall revisit this choice in the following section when we discuss comment distillation for extracting key terms). Weights can be assigned to each feature using standard measures like term frequency, TFIDF, etc. As resources merge into clusters $\omega_{new} = \omega_\alpha \cup \omega_\beta$, we view each cluster as the average of the comments-based features: $\vec{v}_{\omega_{new}} = \frac{1}{|\omega_{new}|} \sum_{o \in \omega_{new}} \vec{v}$.

To determine which clusters are most similar, we measure the group-average cosine similarity between resources of two clusters. Cosine similarity is a similarity measurement between two vectors – in this case, the vectors associated with clusters $\vec{v}_{\omega_\alpha}$ and $\vec{v}_{\omega_\beta}$:

$$cos(\vec{v}_{\omega_\alpha}, \vec{v}_{\omega_\beta}) = \frac{\vec{v}_{\omega_\alpha} \cdot \vec{v}_{\omega_\beta}}{|\vec{v}_{\omega_\alpha}| \left|\vec{v}_{\omega_\beta}\right|}$$

By normalizing the comment term vector to unit length in the L2-norm, the similarity of two clusters $cos(\vec{v}_{\omega_\alpha}, \vec{v}_{\omega_\beta})$ is simply the dot product $\vec{v}_{\omega_\alpha} \cdot \vec{v}_{\omega_\beta}$. This approach has the advantage of more efficiently computing cluster similarity by reusing similarity measures at lower levels of the cluster merge hierarchy.

For implementation, we maintain with each cluster ω_h a heap of clusters ordered by largest similarity $cos(\omega_h, \omega_g)$ where $\omega_h, \omega_g \in \Omega$. Thus, for each cluster, we are able to access the most similar cluster in constant time. The time complexity of the whole procedure is $O(n^2 log(n))$.

2.3 Evaluation Metrics and Dataset

To evaluate the quality of the clusters produced by a hierarchical clustering algorithm, we need a test dataset and an evaluation metric.

YouTube Dataset. Since there are no standard comment-oriented datasets, we collected a sample dataset from YouTube using the open source Tubekit toolkit [20]. By querying YouTube either with a randomly selected word from an English dictionary or with a popular query based on Google Trends (totaling 1,493 queries over September to November 2009), we sampled 5,676 unique videos. Along with each video, we extracted the video's submitter and the information provided by the submitter – including the title of the video, a short text description, keywords, and a self-assigned YouTube category (e.g., News, Politics) – as well as the comments associated with each video as contributed by the YouTube community. The sample dataset consists of 5,676 videos and 11,341,849 total comments.

Measuring Cluster Quality. For evaluation, we can rely on either *external* or *internal* measures of goodness. External measures rely on a known "gold standard" clustering and compare the results of the clustering to this known gold standard. Unfortunately, there is no standard organization for YouTube videos (aside from the very poor quality YouTube categories, which we discussed shortly). As a result, we focus on measuring cluster quality through an internal measurement. Internal measures typically rely on identifying tight clusters (with high intracluster similarity among resources) that are far apart (the clusters have high intercluster dissimilarity) and do not require knowledge of a "correct" clustering. We evaluate the quality of the clustering using the Silhouette Coefficient which combines both of these cohesion and separation properties into a single metric:

$$Silhouette(o_i) = \frac{(\beta(o_i) - \alpha(o_i))}{\max\{\beta(o_i), \alpha(o_i)\}}$$

where α represents the cohesion of each cluster that the resource belongs to and β is a separation measurement that compares the distance to clusters that do not include the target resource:

$$\alpha(o_i, \omega_j) = \begin{cases} 0 & \text{if } o_i \notin \omega_j, \omega_j \in \Omega \\ \frac{\sum_{x \in \omega_j} distance(o_i, x)}{|C_{\omega_j}|} & \text{if } o_i \in \omega_j, \omega_j \in \Omega \end{cases}$$

$$\beta(o_i, \omega_j) = \min\{\frac{\sum_{x \in \omega_j} distance(o_i, x)}{|\omega_j|} | \forall \omega_j \in \Omega, o_i \notin \omega_j\}$$

The value of Silhouette is between -1 and 1. A positive value ($\alpha(o_i) < \beta(o_i)$) is desirable because we would like to have higher similarity for intra-cluster comparisons and lower similarity for inter-cluster comparisons. The maximum value of 1 can be achieved when we have $\alpha(o_i) \to 0$. Note that the above equation is designed for flat clustering. In our study, each resource resides in multiple nested clusters that have a hierarchical relationship with each other. To accommodate the nature of this clustering tree, we adjust the weight of each α based on the number of elements in the cluster. Intuitively, the closer the cluster is to the root, the more separate it is for intra-cluster measurement. In order to diminish the degrading effect from the core quality of the tree, we calculate the average of the weighted sum of α for each object. We set the weight as the inverse of the size of the cluster, resulting in a hierarchical Silhouette:

$$\hat{\alpha}(o_i) = \frac{\sum_{w_j}^{\omega_j \in \Omega} \frac{1}{|C_i|-1} * \alpha(o_i, \omega_j)}{|\{\omega_j | \omega_j \in \Omega, o_i \in \omega_j\}|}$$

The Silhouette score quantifies the core quality of a hierarchical tree. It shows how well the elements are positioned so as to maximize the similarity within each layer of groups as well as the distance of dissimilar groups.

2.4 Initial Results

Given the baseline comments-based representation, the hierarchical clustering framework, and an appropriate evaluation testbed, we first consider the results of applying this setup to the set of YouTube videos. As a point of comparison, we compare the results of the comments-driven hierarchical clustering versus two alternative approaches:

- **By YouTube category:** In the first alternative, we group videos according to their official YouTube category. YouTube provides the uploader of each video with a pre-set list of 15 categories to choose from including Education, Entertainment, Gaming, News and Politics, etc. Such an approach is limited, though, since the number of predefined categories in YouTube is limited, resulting in coarse-level categories that provide little discriminative power among related videos. There is also the drawback of closely-related videos appearing in different categories since the category decision is made solely by the video's uploader.
- **By YouTube keywords:** In the second alternative, we group videos according to the keywords assigned to each video by the video's uploader. Each video is assigned to tag groups if the video contains the tag. Thus, each video can be assigned to more than one group for our evaluation process. Intuitively, tags do a better job than the YouTube category since there is a higher degree of freedom for video representation. However, some videos may suffer from having too few tags or from ill-specified tags (again, because the tag decision is made by the video uploader only).

In all three cases – by YouTube category, by YouTube keywords, and by comments – we used the comments-based term vector of each video for the Silhouette calculation.

Table 1: Comparing Clustering Quality

	Silhouette	Improvement
By Category	0.139	
By Tag	0.186	
By Comment	0.556	300% / 198%

As shown in Table 1, we find that the clusters generated based on categories and on tags have low cohesion among elements (resulting in a low Silhouette score) as compared to the quality of clusters generated by the comments-based approach. Additionally, we performed a t-test to compare the comments-based approach versus grouping videos by YouTube category ($t = 217.7406$, $p < 0.0001$) and by YouTube tag ($t = 198.4679$, $p < 0.0001$). In both cases, these t-tests show that it is very unlikely to encounter this significant difference of cluster quality by chance. Together, these initial results indicate that there is a fundamental mismatch between what users are contributing (via comments) and how the resources are currently organized according to YouTube categories and YouTube tags. This mismatch indicates an opportunity for clustering over community-driven commenting to serve as an alternative enhanced approach for resource organization.

2.5 Challenges

While these initial results are encouraging for indicating that comments may provide a valuable baseline for grouping related information resources, we next turn our attention to two key challenges for the comments-based hierarchical clustering framework:

- First, user-contributed comments range widely in style, quality, and informativeness. By naively relying on all comments, the inclusion of all comment terms in the resource representation can result both in poor quality clustering (by including noisy and unimportant terms) and in less efficient clustering due to the large size of the resource vectors. Can we improve on the baseline performance to deal with these challenges?
- Second, user-contributed comments are constantly being added to Web resources, meaning that a perfectly satisfactory clustering may become stale as new resources are commented on and as the comment focus of existing resources changes. How can new resources be efficiently inserted into the existing resource organization framework? Since it can be expensive to re-cluster all resources as new comments and new resources are added, we must be careful to balance overall cluster quality with support for fast updates.

In the following two sections, we address these two challenges in turn, toward the goal of an adaptable and efficient comments-driven hierarchical resource organization.

3. COMMENT DISTILLATION

In this section, we study two approaches for distilling comments to their core "essence" through comments-oriented dimensionality reduction. Concretely, the first component is *comment term normalization* for compacting variants of terms into their user-intended root form. The second component is *key term extraction* for identifying and extracting highly informative terms from the comments for succinctly summarizing a resource.

3.1 Comment Term Normalization

Our first observation is that user-contributed comments typically contain large numbers of misspellings (often intentionally) and slang. Misspellings can be divided into unintentional typos (e.g., "wacth, watc, watcing" $\rightarrow$ "watch") and intentionally misspelled words. From our YouTube dataset, some examples of this type are:

- reallyr, reallyreally reeeeeeaaaaaaaallllllllllllllllllllllllllllllllli, reeeeeeaaaaaaaallllllllllllllllllllllllllllllllly $\rightarrow$ really
- cuuutte, cuuuttee, cuuuuutest, cutee, cuteeee, cuteeeee, cuteness $\rightarrow$ cute

This type of irregular spelling is very common in user commenting systems. Our goal is to map these user-contributed terms to their common root term so that (i) we can consider multiple forms of a term as similar to semantically merge them; and (ii) so that we can compress the size of the resource vectors $\vec{v}$ for more efficient resource clustering.

We could rely on traditional stemming approaches popularized in the information retrieval community [17]. These stemmers are typically designed to collapse variants of correctly spelled terms to a common root (e.g., "buying", "buys" $\rightarrow$ "buy") and so are not appropriate for this task since many semantically related terms – "cutee" and "cuuutte" – do not share the same suffix. Alternatively, traditional spell checking approaches typically compare encountered words against a known list of correctly spelled words (i.e., the dictionary). Most domains such as search queries, email, and document composition are straightforward in terms of the "correct spelling" term corpus. These approaches – including using probability theory[1], incorporating query log [3], and machine learning approach [7] – are typically not designed to tolerate the type of unseen user-contributed words and intentional misspellings in user commenting systems.

[1] `http://norvig.com/spell-correct.html`

Toward the goal of normalizing comments terms, we propose a comment-tailored spell corrector. Since users may use standard words, newly invented slang, and intentionally misspelled words, we need to develop a spell correcting engine that can map misspelled words to a base term. Additionally, such a spell corrector should be able to identify slang terms and record them as new words in the dictionary rather than considering them as misspelled variants of existing words. In our Comment Term Normalization, we say that a standard English word is a term that can be found in the standard English dictionary; A non-standard English term is slang that people use as special Internet language; A variant term is either an intentional or unintentionally misspelling.

The comments-based term normalization algorithm is shown in Algorithm 2. For preprocessing, we remove stop words and apply Porter's Stemmer (step 3 & 5). We need a customized Comment Term Dictionary (CTD) originally set to empty. During the processing of comments, terms may push in or pop out from the CTD. Entering the main procedure, we examine each remaining term sequentially. Firstly, a standard English dictionary is used to identify whether it is a standard English term or not. If positive, we add this term to our CTD and identify its variants from the existing CTD terms which were marked as "non-standard" using the Jaccard Coefficient (JC) (step 10 & 11). Otherwise, we apply the Jaccard Coefficient to check whether this term is a variant of any of the existing CTD standard terms (step 15). If so, the root form of this term is found. Otherwise, we insert this term into CTD and mark it as a non-standard term (step 20). Before each non-standard term is clarified as a variant of a newly discovered standard term, it is treated as a slang term and placed in the CTD like all other standard English words.

Algorithm 2 Comment Term Normalization

```
1: for v = 1 to all resources do
2:   for c = 1 to all comments ∈ v do
3:     remove stop words
4:     Encounter a term t parsed from a comment
5:     apply Porter stemming on t
6:     if t ∈ CTD then
7:       return t
8:     else
9:       if t ∈ standardEnglishword then
10:        Insert t into CTD
11:        check JC of all non-standard term in CTD to find a
           list of variants of t ⟹ γ
12:        if γ¬∅ then
13:          attach γ to the variants list of t
14:      else
15:        check JC of all standard term in CTD to find the term
           that t is the variants of ⟹ χ
16:        if χ found then
17:          attach t to the variants list of χ
18:          return χ
19:        else
20:          Insert t into CTD and mark as non-standard term
21:          return t
```

The Jaccard Similarity Coefficient is used in our algorithm to determine whether a term is a variant of another recorded term in CTD. The implementation operates at a token level, comparing two terms by first tokenizing them and then dividing the number of tokens shared by the terms by the total number of tokens. Concretely, each term t is tokenized into n characters $\Phi_t = \{\alpha_1, \alpha_2, ..., \alpha_n\}$.

$$J(\Phi_{t1}|\Phi_{t2}) = \frac{|\Phi_{t1} \cap \Phi_{t2}|}{|\Phi_{t1} \cup \Phi_{t2}|}$$

We calculate the Jaccard Similarity Coefficient between the encountered term t and each term recorded in the CTD dictionary. Whenever this comparison is needed, we return the term with the highest Jaccard value which must also exceed a threshold of 0.8.

3.2 Key Term Extraction from Comments

In this section, we want to show that user generated comments are valuable in terms of identifying hidden information that may not be contained in the information resource itself (e.g., in the text of a news article) or in the metadata associated with the resource (e.g., the title, description, category, and keywords for YouTube videos that are provided by the video's uploader). In contrast to a single perspective provided by the author of a resource, mass contributed user comments may provide additional perspectives on the concepts associated with a resource and by mining these comments, we may find a concise summary for representing each resource for the purposes of hierarchical clustering.

However, as what has been observed previously [10, 13], comments are typically messy and unstructured. Among all the terms used in comments only some may be truly informative or representative of the resource itself. Intuitively, as a first step toward extracting the terms from each resource's set of comments that can be used to represent the resource itself, we propose to rank the terms using the standard TFIDF weighting.

Formally, we define TFIDF as follows:

- tf: The term frequency is defined as the number of times a given term i appears in a set of comments of a resource j

$$tf_{i,j} = n_{i,j}$$

- idf: The inverse document frequency is obtained by dividing the total number of resources k by the number of resources containing the term, and then taking the logarithm of that.

$$idf_i = log \frac{k}{|\{o_j : t_i \in C_j\}|}$$

However, it is not clear how many terms are necessary to sufficiently represent the underlying resource. Toward optimizing a top-k number of terms to choose, as ranked by TFIDF, we observe that choosing too many terms (a large k) may introduce too much noise and lead to poor clustering, whereas choosing too few terms (a small k) may be insufficient to capture the essence of the comments. On the other hand, defining a strict threshold TFIDF value for keyword consideration (e.g., consider all terms with a TFIDF above some threshold) may result in a skewed selection of terms across resources, since the term TFIDF distribution may vary greatly. Therefore, we propose a dynamic selection criterion to identify an appropriate number of representative terms for each resource.

3.2.1 Using KL-Divergence To Identify Key Terms

As discussed earlier, it is not wise to apply the same rule for term selection to each resource. To select the right amount of terms, or more precisely, best set of words from usually thousands of unique comment terms, we want to develop a new approach that is as dynamic in nature as the comments of the online resources, in our case, the YouTube videos. A logical starting point is to rank each of our comment terms of a video by their TFIDF value, which represents the importance of words for each resource. Once we obtain this rank list, we want to find the cutting point tailored for each resource accordingly. Here, we apply the Kullback-Leibler divergence technique to measure the discriminating power of the keywords parsed from the user generated comments. For each resource, we start with an empty set of keywords. Iteratively, we add the most important terms from the rank list to our keyword set and calculate the KL-Divergence of the current representation over the average term representation in our corpus. The idea of finding the cutting point θ is that, once the distinguishing power does not increase anymore, this subset of keywords can best describe the video and distinguish itself from other resources. Any further inclusion of terms with lower TFIDF may diminish the quality of the resource representation. Concretely, the KL-Divergence calculates the information gain of using a language model θ_1 over the base language model θ_2.

$$KL(\theta_1|\theta_2) = \sum_w p(w|\theta_1) log(p(w|\theta_1)/p(w|\theta_2)) \quad (1)$$

where in our case, θ_1 is the comments-based language model of a resource when only considering the top-k terms and θ_2 is the overall comments-based language model using the average TFIDF of each term in the corpus.

After we preprocess the dataset and calculate Equation 1 for every term in each resource, we identify the appropriate number of discriminating comment terms for each resource by Equation 2.

$$\delta_{v_i} = argmax(k, KL(\theta_1^k|\theta_2)) \quad (2)$$

At the end, we extract the terms ranked from 1 to δ in the term TFIDF ranked list to represent the community view of hidden concepts of each resource.

3.3 Evaluating Comment Distillation

We next evaluate the impact of comment term distillation both in terms of the sizes of comment vectors and in terms of the impact on cluster quality.

Comment Vector Compression. Recall that the first goal of comment distillation is to reduce the potentially large and noisy comment term vectors to an informative core, resulting in more robust and efficient clustering. Starting with the 11,341,849 total comments in the YouTube dataset, we first identify 274,055 unique terms. Applying the Comment Term Normalization algorithm of Section 3.1 results in a reduced set of 120,558 terms, 56% savings (see Table 2).

Table 2: Compression: Comment Term Normalization

	Number of terms
Without comment normalization	274,055
With comment normalization	120,558
Percent change	56%

We next apply the dynamic Key Term Extraction approach described in Section 3.2, which achieves another 67% savings in terms of the total terms extracted from user comments (see Table 3).

To further explore the dynamic Key Term Extraction, we show in Figure 1 the distribution of compression rates after applying the KL-Divergence approach for selecting discriminating keywords. About 800 of the 5,676 unique videos achieve more than a 95% compression rate. Even in the worst case, the KL-Divergence approach results in a 15% compression rate. We see that most videos

Table 3: Compression: Key Term Extraction

	Number of terms
With comment normalization	120,558
Using key term extraction	39,229
Percent change	67%

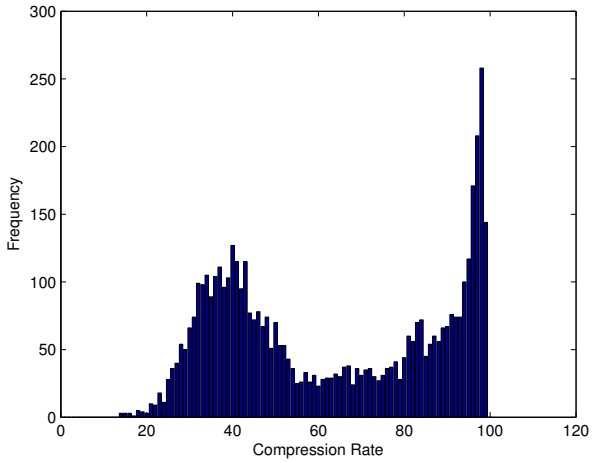


Figure 1: Compression rate of keywords considering using KL Divergence approach over all available comment terms of each video

are within the compression rate of 30% to 50%, meaning that about 1/3 to 1/2 of all comment terms can be disregarded by the clustering algorithm. We find that the compression rate has a positive correlation (0.75) with the number of unique terms appearing in the comments of a video.

Impact on Clustering. We have seen how the two comment term distillation techniques can reduce the size of the comment vectors. We next explore the impact of these techniques on the quality of clustering.

We first consider the impact of comment normalization in Table 4. The first row shows the quality of the clustering structure using all raw terms from the user comments to construct the vector representation for each resource. The second row of the table shows the case where we applied comment normalization, resulting in a 20% increase in the quality of clustering. We additionally applied a t-test to compare the two cases, with a significant difference (t=28.6145, $p < 0.0001$). These results show that comment normalization can improve the quality of clustering by reducing the noise in comments.

We next consider the impact of key term extraction on clustering quality. Following by the term normalization is the experimental performance of various ways on further feature selection for video representation. We compare a baseline with no key term extraction against three alternatives – the first is a threshold-based approach,

Table 4: Clustering Quality: Impact of Comment Term Normalization

	Silhouette	Improvement
Without normalization	0.556	
With normalization	0.670	20%

Table 5: Clustering Quality: Key Term Extraction

	Silhouette	Improvement
Baseline	0.670	
Threshold=0.5	0.689	3%
Threshold=1	0.700	4%
Threshold=5	0.694	4%
Top-50	0.747	11%
Top-100	0.691	3%
Top-500	0.701	5%
Dynamic (KL)	**0.764**	**14%**

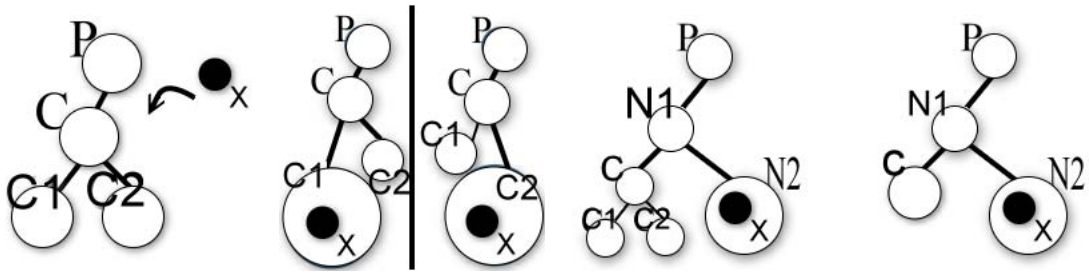
where all terms with a TFIDF score higher than a particular threshold are extracted; the second is a top-k approach, where all terms ranked in the top-k according to TFIDF are extracted; the final approach is the KL-Divergence based approach for dynamically selecting key terms based on their informativeness. Table 5 shows that KL-Divergence approach results in the best overall clustering performance, indicating that Key Term Extraction is important for improving the quality of resource representation. We additionally tested the statistical significance of the KL-Divergence approach (t = 6.7542, $p < 0.0001$) and find that it is, indeed, a significant improvement.

4. INCREMENTAL COMMENTS-DRIVEN RESOURCE INSERTION

Algorithm 3 Incremental Resource Insertion

```
1: Build an initial Hierarchical Agglomerative Clustering tree T
   (dendrogram) over a baseline set of information resources
2: for all resources to be inserted do
3:   for 10 random walkers do
4:     place the incoming new resource x at the root of T
5:     while current node is not the leaf do
6:       compute ŝ1 , ŝ2 , ŝ3
7:       if ŝ1 or ŝ2 is the largest then
8:         (Figure 2(b) explains this case)
9:         insert the new resource o into cluster C1/C2
10:        set C1/C2 as current node
11:      else
12:        (Figure 2(c) explains this case)
13:        create a new cluster N2 to include video o
14:        create a new cluster N1 to include all resources in
           cluster C and N2
15:        set C and N2 as children of N1
16:        connect N1 to the parent P
17:        stop traversing
18:    x arrives at a leaf node.
19:    Figure 2(d) explains this case
20:    create a new node N2 to hold x
21:    create a new node N1 to hold N2 and C
22:  choose the route from the best random walker
```

Using Algorithm 1, we can develop a hierarchical structure for a collection of resources by applying the comments-based resource representation and the comment distillation techniques introduced in the previous section. However, we find it is not practical to run the clustering algorithm every time a new resource is added to the system or when new comments are contributed. As a step toward overcoming this challenge, we propose to incrementally add new resources to an existing hierarchy, so that expensive re-clusterings



(a) Adding x to Tree. (b) max = s(x,c1/c2) (c) max = s(c1,c2) (d) leaf case

Figure 2: Resource Insertion Traversal Rule

are reduced.

Suppose we have initially built a dendrogram (the rooted hierarchical tree that is the output of the hierarchical clustering algorithm) with a base set of information resources using Algorithm 1. Now we want to insert a new resource x into the dendrogram without having to re-cluster the entire collection of resources. Our intuition is that a new resource alone will not require a full re-clustering and will most likely fit into one of the existing nodes of the dendrogram or require only a small change to the dendrogram. Over time, the accumulated errors of introducing many new resources may result in a "concept drift" of the overall hierarchy, necessitating the need for a full (or at least partial) re-clustering. Our goal is to reduce these errors by making good insertion decisions, so that the need to re-cluster is reduced.

We propose to incrementally insert a new resource into the tree by starting at the root of the tree and then descending the tree according to a greedy selection mechanism to select the best path. After descending the tree, the resource will be incorporated into the tree, leading to a new cluster tree until the next resource is inserted. Algorithm 3 shows the rules for incoming resource node traversal and the related data structure updating. At each step, x will make a binary selection to find the best path as it descends the tree. The time complexity of this approach is O(k*N) if we have k new resources and N resources in the original dendrogram T.

We now begin with a deterministic approach which compares the comments-based resource representation of x against the current cluster model at that point in the tree, selecting the route that has the highest similarity value:

1.
   - $\hat{s1} = similarity(x, C1)$
   - $\hat{s2} = similarity(x, C2)$
   - $\hat{s3} = similarity(C1, C2)$

Such a route traversal can result in a poor quality placement of the new resource if the greedy route results in a local maximum (that is, there may be a better placement in the tree through a route that initially looks less promising to the greedy selection mechanism). To overcome this problem, we consider a set of probabilistic random walkers that descend the tree in an effort to increase the coverage of possible placement locations for the new resource. Concretely, there are three extra steps in place of the deterministic model comparison.

1.
   - $s1 = similarity(x, C1)$
   - $s2 = similarity(x, C2)$
   - $s3 = similarity(C1, C2)$
2. assign 3 to the $max(s1, s2, s3)$, 2 to the $median(s1, s2, s3)$, 1 to the $min(s1, s2, s3)$ and get $\bar{s1}$,$\bar{s2}$,$\bar{s3}$
3. produce three random numbers, $p1$, p, $p3$ between 0 and 1
4. $\hat{s1} = p1 * \bar{s1}$, $\hat{s2} = p2 * \bar{s2}$, $\hat{s3} = p3 * \bar{s3}$

The time complexity for this binary comparison approach O(k * 10* log N) if we set up 10 random walkers for selecting the best route. Of course this is an approximate algorithm for finding the best position in the tree for the incoming resource, but without the cost of an exhaustive search of the tree.

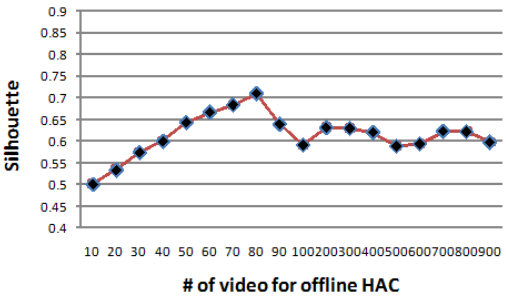


Figure 3: Clustering Performance for Incremental Resource Insertion

4.1 Evaluating Incremental Insertion

Our goal is to understand how many resources are necessary to build the initial tree so that the incremental resource insertion still results in high-quality clustering. In the best case, we would need only an empty initial tree; in the worst case, nearly all resources would be necessary to build the initial tree, meaning that the incremental insertion could only rarely be performed.

To experimentally evaluate incremental resource insertion, we first built an initial dendrogram using a varying k number of resources (YouTube videos) and then inserted the remaining $N-k$ resources using the approach described in the first part of this section. We evaluated the quality of the clustering using the Silhouette measure against a gold standard in the case when all N resources are used to build the dendrogram. We repeated this experiment 10 times and report the average results in Figure 3. After an initial improvement as the number of resources in the initial tree increases, we see that the quality of clustering stabilizes at around 300 resources, indicating that using just a fraction of resources (300 out of more than 5,000) to build the tree can result in good quality clustering.

5. RELATED WORK

The impact of user-contributed metadata such as tags, ratings and comments has been explored in several related works. For example, researchers have incorporated information extracted from comments to increase search accuracy, e.g., [23]. They discovered that comments can further distinguish relevant videos from each other especially on popular videos where the comment set is large. Others have investigated approaches for summarizing user-contributed content for better visualization and navigation, e.g., in cloud-style summaries [9]. Along this line, [14] incorporates user contributed comments in the data cloud generation process. Some other researchers have explored using user-generated tags and comments for flat clustering, e.g., [18] and [16].

An incremental hierarchical clustering algorithm on text documents as discussed in [19]. This algorithm inherited from a non-text incremental hierarchical clustering algorithm called COBWEB by Fisher [5]. In order to adapt the algorithm for text domain, they assume the word occurrences follow the Katz's distribution. Their approach is different from ours. The difference is that they have an incremental approach start with an empty tree. As documents arrive, the tree is expanded leaves are added, merged, split, etc. That is, starting with an empty tree, N documents arrive; you end up with a final tree where all N documents live in each leaf in the tree. On the other hand, our approach is to take N documents, do an initial hierarchical clustering to derive an unbalanced binary tree. As new documents arrive we insert them to the tree incrementally.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have explored the use of user-contributed comments as a basis for organizing information resources in a hierarchical fashion. We have proposed using comment term normalization and key term extraction via KL-Divergence for distilling noisy comments for effective clustering. We also developed an incremental insertion component for updating the comments-based hierarchy so that resources can be efficiently placed in the hierarchy as comments arise and without the need to re-generate the (potentially) expensive hierarchy. Our experimental study over YouTube provides evidence that the proposed approach can lead to effective and efficient comments-based video organizing even in a YouTube-like environment. As part of our future work, we are interested in to integrate these results as part of our broader research effort to build enhanced Social Web information management applications that leverage this social collective intelligence.

7. ACKNOWLEDGMENTS

We thank Jia-Hao Fan for his discussions on the Incremental Comments-Driven Resource Insertion algorithm. We also thank the three anonymous reviewers for their valuable feedback.

8. REFERENCES

[1] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *the 8th international conference on Knowledge discovery and data mining*, pages 436–442, 2002.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. pages 993 – 1022, 2003.

[3] S. Cucerzan and E. Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *the Conference on Empirical Methods in Natural Language Processing*, 2004.

[4] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *the 14th international conference on World Wide Web*, pages 801–810, 2005.

[5] D. Fisher. Knowledge acquisition via incremental conceptual clustering. pages 139–172, 1987.

[6] B. C. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *the international conference on data mining*, 2003.

[7] A. R. Golding and D. Roth. A winnow-based approach to context-sensitive spelling correction. pages 107–130, 1999.

[8] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, pages 100–108, 1979.

[9] Y. Hassan-Montero and V. Herrero-Solana. Improving tag-clouds as visual information retrieval interfaces. In *International Conference on Multidisciplinary Information Sciences and Technologies*, 2006.

[10] C.-F. Hsu, E. Khabiri, and J. Caverlee. Ranking comments on the social web. In *International Conference on Social Computing*, pages 90–97, 2009.

[11] M. Iwayama and T. Tokunaga. Hierarchical bayesian clustering for automatic text classification. In *the 14th international joint conference on Artificial intelligence*, pages 1322–1327, 1995.

[12] I. Jonyer, D. J. Cook, and L. B. Holder. Graph-based hierarchical conceptual clustering. pages 19–43, 2002.

[13] E. Khabiri, C.-F. Hsu, and J. Caverlee. Analyzing and predicting community preference of socially generated metadata: A case study on comments in the digg community. In *the 3rd International Conference on Weblogs and Social Media*, 2009.

[14] G. Koutrika, Z. M. Zadeh, and H. Garcia-Molina. Course cloud: Summarizing and refining keyword searches over structured data. In *Stanford Infolab, Technical Report*, 2009.

[15] K. Kummamuru and R. Lotlikar. A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In *the 13th international conference on World Wide Web*, 2004.

[16] B. Li, S. Xu, and J. Zhang. Enhancing clustering blog documents by utilizing author/reader comments. In *the 45th annual southeast regional conference*, pages 94–99, 2007.

[17] M. F. Porter. An algorithm for suffix stripping. *Readings in information retrieval*, pages 313–316, 1997.

[18] D. Ramage and P. Heymann. Clustering the tagged web. In *2nd International Conference on Web Search and Data Mining*, pages 54–63, 1991.

[19] N. Sahoo and J. Callan. Incremental hierarchical clustering of text documents. In *the 15th international conference on Information and knowledge management*, pages 357–366, 2006.

[20] C. Shah. Tubekit: a query-based youtube crawling toolkit. In *the 8th conference on Digital libraries*, pages 433–433, 2008.

[21] V. Torra, S. Miyamoto, and S. Lanau. Exploration of textual document archives using a fuzzy hierarchical clustering algorithm in the gambal system. *Inf. Process. Manage.*, 2005.

[22] R. Weiss, B. Vélez, and M. A. Sheldon. Hypursuit: a hierarchical network search engine that exploits content-link hypertext clustering. In *the the 7th ACM conference on Hypertext*, pages 180–193, 1996.

[23] W. G. Yee, A. Yates, S. Liu, and O. Frieder. Are web user comments useful for search? In *7th Workshop on Large-Scale Distributed Systems for Information Retrieval*, 2009.