World Wide Web manuscript No.
(will be inserted by the editor)

Clustering Web Video Search Results based on Integration of Multiple Features

Alex Hindle · Jie Shao · Dan Lin · Jiaheng Lu · Rui Zhang

Received: date / Accepted: date

Abstract The usage of Web video search engines has been growing at an explosive rate. Due to the ambiguity of query terms and duplicate results, a good clustering of video search results is essential to enhance user experience as well as improve retrieval performance. Existing systems that cluster videos only consider the video content itself. This paper presents the first system that clusters Web video search results by fusing the evidences from a variety of information sources besides the video content such as title, tags and description. We propose a novel framework that can integrate multiple features and enable us to adopt existing clustering algorithms. We discuss our careful design of different components of the system and a number of implementation decisions to achieve high effectiveness and efficiency. A thorough user study shows that with an innovative interface showing the clustering output, our system delivers a much better presentation of search results and hence increases the usability of video search engines significantly.

Keywords Web video · YouTube · search results clustering · user interface

1 Introduction

The exponential growth of the number of multimedia documents distributed on the Internet, in personal collections and organizational depositories have brought extensive attention to multimedia search and data management. Among the different multimedia types, video

A. Hindle · J. Shao · R. Zhang
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
E-mail: {ahindle, jsh, rui}@csse.unimelb.edu.au

D. Lin
Department of Computer Science
Missouri University of Science and Technology, USA
E-mail: lindan@mst.edu

J. Lu
School of Information and DEKE, MOE
Renmin University of China, China
E-mail: jiahenglu@ruc.edu.cn

carries the richest content and people are using it to communicate frequently. With the massive influx of video clips on the Web, video search has become an increasingly compelling information service that provides users with videos relevant to their queries [12].

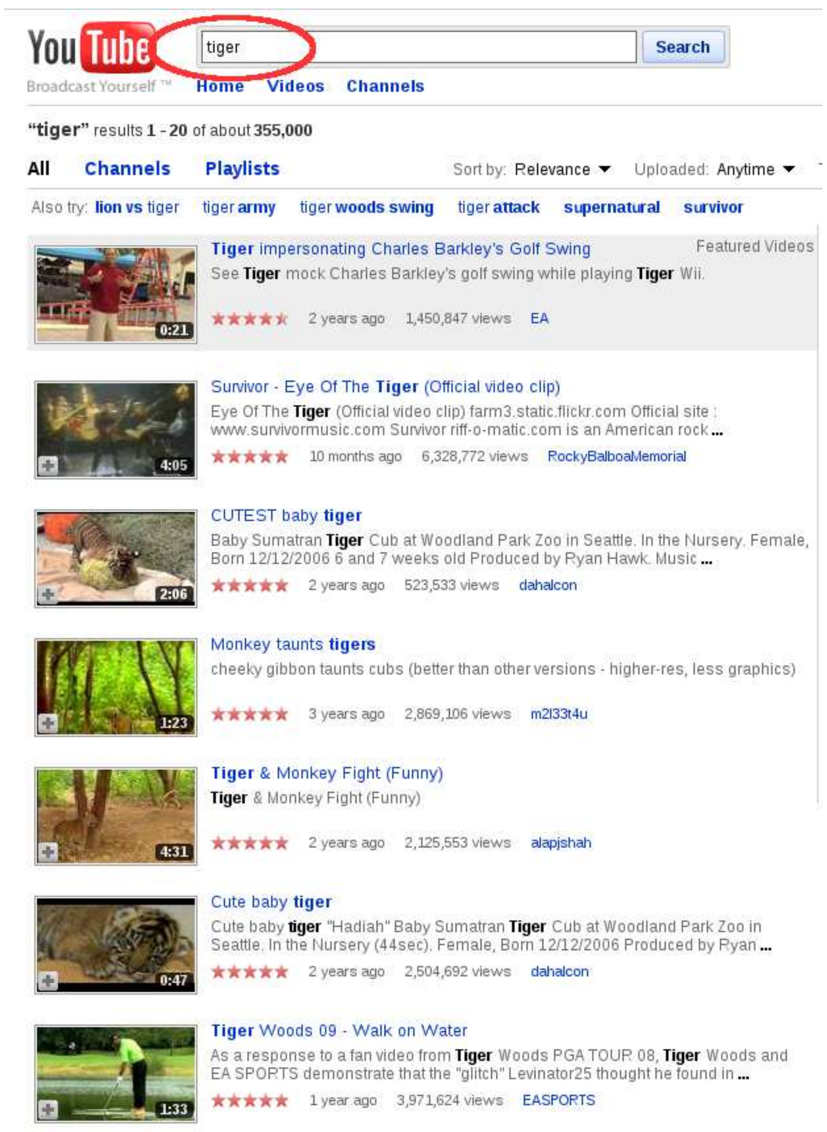
Since numerous videos are indexed, and digital videos are easy to reformat, modify and republish, a Web video search engine may return a large number of results for any given query. Moreover, considering that queries tend to be short [17,16] (especially those submitted by less skilled users) and sometimes ambiguous (due to polysemy of query terms), the returned videos usually contain multiple topics at semantic level (see an example in Fig. 1). Even semantically consistent videos have diverse appearances at visual level, and they are often intermixed in a flat-ranked list and spread over many results pages.

In terms of relevance and quality, videos returned in the first page are not necessarily better than those in the following pages. As a result, users often have to sift through a long undifferentiated list to locate videos of interest. This becomes even worse if one topic's results are overwhelming but that topic is not what the user actually desires, or the dominant results ranked at the top are different versions of duplicate or near-duplicate videos (i.e., lack of diversity). In such a scenario, *clustering search results* is essential to make the search results easy to browse, and improve the overall search effectiveness.

Clustering the raw result set into different semantic categories has been investigated in text retrieval (e.g., [38,24,39]) and image retrieval (e.g., [3,18]), as a means of improving retrieval performance for search engines. To the best of our knowledge, this work is the first attempt to address a more challenging problem of *video search results clustering*. Web video search results clustering is clearly related to the general-purpose clustering but it has some specific requirements concerning both the effectiveness and the efficiency of the underlying algorithms that are addressed by conventional techniques.

Currently available commercial video search engines generally provide searches only based on keywords but do not exploit the context information in a natural and intuitive way. This paper presents the first system that clusters Web video search results by fusing the evidences from a variety of information sources besides the video content such as title, tags and description. We propose a novel framework that can effectively integrate multiple features and enable us to adopt existing clustering algorithms. In addition, unlike only optimizing clustering structure as in the traditional clustering algorithms, we emphasize the role played by other expressive messages such as representative thumbnails and appropriate labels of generated clusters. Besides clustering organization, these messages are very helpful to facilitate users for fast browsing. A thorough user study carried out by human assessors confirms that our proposed strategy can deliver a much more meaningful presentation of search results, which complements the output of current video search engines and leads to a considerately enhanced search experience. The proposed framework for information integration enables us to exploit state-of-the-art clustering algorithms to organize returned videos into semantically and visually coherent groups - their efficiency ensures almost no delays caused by the post-processing procedure. In summary, our system returns much higher quality results yet with similar response time to other systems.

The rest of the paper is organized as follows. We discuss related work in Section 2 and provide some preliminaries on two general clustering algorithms in Section 3. Section 4 presents our system by describing the information integration framework, the purpose of each module, and the key comparison and clustering algorithms. Section 5 reports the results of our experimental study. Finally, Section 6 gives the concluding remarks and future work directions.



"tiger" results **1 - 20** of about **355,000**

All · Channels · Playlists · Sort by: Relevance · Uploaded: Anytime

Also try: **lion vs** tiger · tiger **army** · tiger **woods swing** · tiger **attack** · **supernatural** · **survivor**

Tiger impersonating Charles Barkley's Golf Swing — Featured Videos
See **Tiger** mock Charles Barkley's golf swing while playing **Tiger** Wii.
0:21 · 2 years ago · 1,450,847 views · EA

Survivor - Eye Of The Tiger (Official video clip)
Eye Of The **Tiger** (Official video clip) farm3.static.flickr.com Official site : www.survivormusic.com Survivor riff-o-matic.com is an American rock **...**
4:05 · 10 months ago · 6,328,772 views · RockyBalboaMemorial

CUTEST baby tiger
Baby Sumatran **Tiger** Cub at Woodland Park Zoo in Seattle. In the Nursery. Female, Born 12/12/2006 6 and 7 weeks old Produced by Ryan Hawk. Music **...**
2:06 · 2 years ago · 523,533 views · dahalcon

Monkey taunts tigers
cheeky gibbon taunts cubs (better than other versions - higher-res, less graphics)
1:23 · 3 years ago · 2,869,106 views · m2l33t4u

Tiger & Monkey Fight (Funny)
Tiger & Monkey Fight (Funny)
4:31 · 2 years ago · 2,125,553 views · alapjshah

Cute baby tiger
Cute baby **tiger** "Hadiah" Baby Sumatran **Tiger** Cub at Woodland Park Zoo in Seattle. In the Nursery (44sec). Female, Born 12/12/2006 Produced by Ryan **...**
0:47 · 2 years ago · 2,504,692 views · dahalcon

Tiger Woods 09 - Walk on Water
As a response to a fan video from **Tiger** Woods PGA TOUR 08, **Tiger** Woods and EA SPORTS demonstrate that the "glitch" Levinator25 thought he found in **...**
1:33 · 1 year ago · 3,971,624 views · EASPORTS

Fig. 1 A search results page showing the flat-ranked list for a "tiger" query, with multiple semantic categories mixed together (as of October 2009).

2 Related work

In this section, we review some previous research efforts on search results clustering and video clip comparison, respectively.

2.1 Search results clustering

Currently, there are several commercial Web page search engines that incorporate some form of result clustering[1]. The seminal research work in information retrieval uses *scatter/gather* as a tool for browsing large to very large document collections [7,6]. This system divides a document corpus into groups and allows users to iteratively examine the resultant document groups or sub-groups for content navigation. Specifically, *scatter/gather* provides a simple graphical user interface. After the user has posed a query, s/he can decide to "scatter" the results into a fixed number of clusters; then, s/he can "gather" the most promising clusters, possibly to scatter them again in order to further refine the search. Many other works on text (in particular, Web page) search results clustering are along this line, such as [38,24,19, 39,22,36]. For more details, there is an excellent survey [4] regarding this topic published recently.

There are also some works on general image clustering [10,25,34] and particularly, a few Web image search results clustering algorithms [3,18] have been proposed to cluster the top returned images using visual and/or textual features. Nevertheless, different from an image, normally the content of a video can be hardly taken in at a glance or be captured in a single vector. This brings more challenges. Compared with the previous design [18] solely based on textual analysis for clustering, our system of video search results clustering can yield a certain degree of coherence on visual appearance of each cluster. While [3] takes a two-level approach that first clusters the image search results into different semantic categories and then further groups images in each category with visual features for a better visual perception, we propose to integrate textual and visual features *simultaneously* rather than *successively*, to avoid propagating the potential errors from the first clustering level to the next level. Although there are some previous studies on image [28] and video [37] retrieval on the Web utilizing the integration of multiple features, fusion of the heterogeneous information from various sources for clustering Web video search results in a single cross-modality framework has not been addressed before.

Existing systems of general video clustering only consider the content information but not the context information. For example, in [21] video clips are clustered according to video signature (ViSig) [5] similarity, which is discussed in the next section.

2.2 Video clip comparison

Video clips are short videos in digital format predominantly found on the Web and express a single moment of significance. The term "video clip" is loosely used to mean any short video typically less than 15 minutes. It is reported in the official YouTube blog that, over 99% of videos uploaded are less than 10 minutes. Traditional videos such as full movies and TV programs with longer durations can be segmented into short clips, each of which may represent a scene or story.

Generally, a video can be viewed as being multi-modal by having visual, audio, textual and motion features [31]. In this work we exploit the inherent visual information by representing a video clip as a sequence of frames, each of which is represented by some low-level feature [33] which is referred to as *video content*, such as color distribution, texture pattern or shape structure.

[1] An example is clusty.com.

One of the most popular methods to compare video clips is to estimate the percentage of visually similar frames. Along this line, [5] proposed a randomized algorithm to summarize each video with a small set of sampled frames named video signature (ViSig). However, depending on the relative positions of the seed frames to generate ViSigs, this randomized algorithm may sample non-similar frames from two almost-identical videos. [27] proposed to summarize each video with a set of frame clusters, each of which is modelled as a hyper-sphere named video triplet (ViTri) described by its position, radius, and density. Each video is then represented by a much smaller number of hyper-spheres. Video similarity is then approximated by the total volume of intersections between two hyper-spheres multiplying the smaller density of clusters. In our system, we partially employ a more advanced method called bounded coordinate system (BCS) [14]. Given two video clips, their video similarity can be approximated by comparing their BCSs. This is described in Section 4.2.

2.3 Semantic Web and information fusion

Considering semantics in Web applications have been studied extensively. Recently, García et. al. [11] explored using the object-action interaction paradigm to improve the usability and accessibility of Web applications. Fusion of information from various sources has also been studied in different application scenarios such as [28,37,32]. However, none of them have addressed the problem of video clustering.

3 Preliminaries

This section briefly reviews two general-purpose clustering algorithms *normalized cuts* and *affinity propagation*, which will be used in our system for comparing their outputs.

3.1 Normalized cuts (NC)

The first clustering algorithm represents a similarity matrix M as a weighted graph, in which the nodes correspond to videos and the edges correspond to the similarities between two videos. The algorithm recursively finds partitions (A,B) of the nodes V subject to the constraints that $A \cap B = \varnothing$ and $A \cup B = V$, to minimize the following objective function [29]:

$$N_{cut}(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{1}$$

where $assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$ is the total connection from nodes in A to all nodes in V and $assoc(B,V)$ is defined similarly. $Cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$ is the connection from nodes in A to those in B. It can be seen that the clustering objective is equivalent to minimizing the cut $N_{cut}(A,B)$, which can be solved as a generalized eigenvalue problem [13]. That is, the eigenvector corresponding to the second smallest eigenvalue (which is called *Fiedler vector*) can be used to bipartition the graph. The components of this vector are thresholded to define the class memberships of the nodes. This bipartition process is performed recursively until the desired number of clusters is reached.

A limitation of normalized cuts is the fact that the user must specify in advance the number of generated clusters. This often has an adverse effect on the quality of the clustering. We usually prefer automatically determining the number of clusters. The clustering algorithm described next can determine an adaptive number of clusters automatically.

3.2 Affinity propagation (AP)

The second clustering algorithm is the affinity propagation method proposed in [9]. Like normalized cuts, this algorithm accepts a similarity matrix and assigns data points to clusters. In this algorithm each data point to be clustered is viewed as a node in a network which passes messages to other nodes in order to determine which nodes should be exemplars and which nodes should be associated with those exemplars. An exemplar is the point which best represents other points in its cluster. The algorithm runs to maximize the overall similarity of all data points to their exemplars. The solution is approximated following the ideas of belief-propagation. There are two types of messages sent between data point i and candidate exemplar k: *responsibility* $r(i,k)$ and *availability* $a(i,k)$. Responsibility messages are sent from i to k and reflect how strongly data point i favors k over other candidate exemplars. Availability messages are sent from k to i and reflect how available i is to be assigned to k currently.

$$r(i,k) \leftarrow s(i,k) - \max_{k' | k' \neq k} \{a(i,k') + s(i,k')\} \tag{2}$$

$$a(i,k) \leftarrow \min \left\{ 0, r(k,k) + \sum_{i' | i' \notin \{i,k\}} \max\{0, r(i',k)\} \right\} \tag{3}$$

The messages are passed during a variable number of iterations. In each iteration the evidence accumulates that some points are better exemplars. It can be seen in (2) and (3) that there is a circular dependency between responsibility and availability. This is handled by initializing $a(i,k)$ to a zero value so that $r(i,k)$ can be calculated in the first iteration. After this the availabilities are calculated and stored to be ready for the next iteration.

Responsibility (2) can be thought of as a competitive update where the similarity measure between i and k, $s(i,k)$, is subtracted from by $k^{'}$. That is the greatest similarity value between i and every other potential exemplar, $s(i,k')$, plus the corresponding availability value, $a(i,k')$. It can be seen in (3) that an availability value will not be greater than zero so this factor will either have no effect on responsibility or it will increase it. Intuitively, if some other candidate exemplar for point i is less available, the current candidate exemplar being analyzed then becomes more responsible and a better fit.

Availability (3) is either zero or less depending on the self responsibility, $r(k,k)$, of the candidate exemplar and the sum of the positive responsibilities that the candidate exemplar receives from other points. Self responsibility is a measure of how much evidence there is for the point itself to be an exemplar. This measure can be adjusted by a "preference" measure given by the user. By default, all data points are equally suitable as exemplars, so the preferences are set to a common value (e.g., the median of the input similarities in the similarity matrix). It can be seen in (3) that if $r(k,k)$ is negative then this will negatively affect the availability of this point, which means this point is less available to be assigned to another cluster.

Affinity propagation is also an iterative algorithm. The two messages are passed from one node to every other, once per iteration. The algorithm reaches convergence when enough evidence has been formed about exemplars and assignments to exemplars. That is when the messages being passed have little or no effect on the current responsibility and availability measures any more. At this stage node i is assigned to whichever candidate exemplar k maximizes the value of $a(i,k) + r(i,k)$. If this value is maximized where $i = k$ then i itself is an exemplar.

4 Our system

In this section, we describe the different components of our video search results clustering system including acquisition and pre-processing of returned videos, pre-processing of context information with a focus on texts, our video clustering method and result visualization. Our system is comprised of a database, processing codes of various algorithms implemented in different languages. The key algorithms in our system are the ones used for compactly representing and comparing video clips, processing texts, and underlying clustering algorithms.

4.1 Collection of information from various sources

Our system mimics the storage and search components of a contemporary Web video search engine but has additional post-processing functionality of clustering returned results. In response to a query request, first we gather top results via a third-party Web video search engine. YouTube is an ideal third-party Web video search engine to be used in our system, since it provides an API to its system which enables developers to write content-accessing programs more easily. TubeKit[2] is an open source YouTube crawler which targets this API [26]. In our system TubeKit is used for sending text queries to YouTube and downloading returned videos and their associated metadata. It is run from a local computer and is essentially a client interface to the YouTube API. When supplied with a query, TubeKit will send it to YouTube and will in turn receive a list of videos and metadata similarly to the user actually accessing YouTube via a Web browser and entering the same query. Specifically, available metadata supplied in YouTube include video *title*, *tags*, *description*, *number of viewers*, *viewer comment counts*, *average ratings*, among others (see Fig. 2). This information is by default gathered and stored in a local database and indexed by a video ID.

Among the different metadata around a video, the *title*, *tags*, and *description* are more likely to be informative in revealing its semantic meaning. For example a video's title might be "pluto" while its tags might be "pluto", "disney", "mickey" which give a good indication that the video would belong to a Disney Pluto's cluster rather than the former planet's cluster. In our system, we only utilize the metadata which are posted by the video publisher and remain constant. We have observed that other metadata such as viewer comments can be quite noisy and less relevant. There may be a case for mining this kind of data but we leave that for future research. The embedded videos clips can be downloaded using youtube-dl[3], a Web video downloader written in Python that can be integrated with TubeKit. It uses the URL of the YouTube page to extract the video and save it to the local database. From there we can analyze its visual content as well as context information.

4.2 Video processing

4.2.1 Computing similarity based on video content analysis

A video is really a sequence of image frames so the problem of representing a video numerically can be decomposed into representing multiple, sequential images. Each video can be

[2] www.tubekit.org
[3] bitbucket.org/rg3/youtube-dl

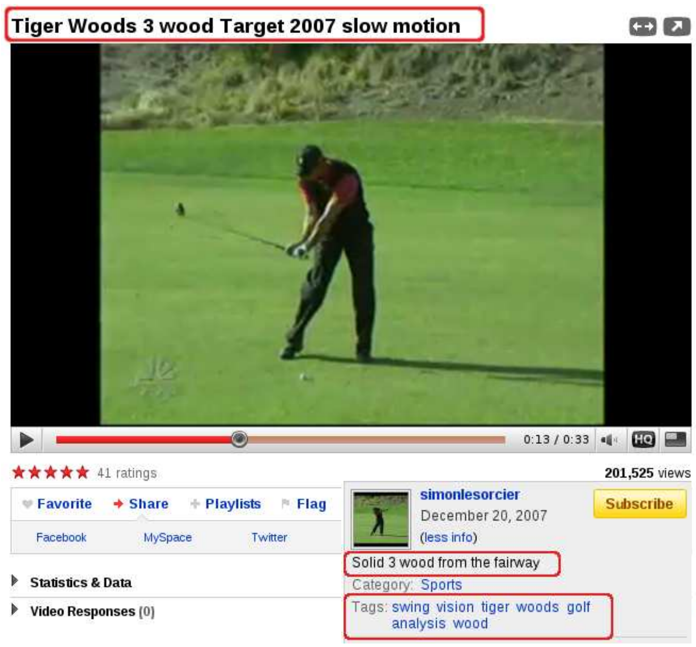


Fig. 2 A YouTube page with highlighted title, tags and description.

represented by a sequence of d-dimensional frame features obtained from image histograms, in order of appearance. An image histogram is constructed by counting how many pixels in an image fall into certain parts of the color spectrum. It is represented by a single vector where the dimensionality d relates to the number of parts the spectrum is divided into. This low-level visual feature is far less complicated to analyze when compared with higher level features such as local points of interest in an image.

In order to compare two video clips, we may compare their histograms, using a frame-by-frame comparison approach (or possibly based on key-frame or sub-sampling representations). Unfortunately this approach has a quadratic time complexity because each frame must be compared with every other frame. This is undesirable because there may be at least hundreds, maybe tens of thousands of videos which may need to be compared with each other and leads to unacceptable response time. In our previous research [14], the bounded coordinate system (BCS), a statistical summarization model of content features is introduced. It can capture dominating content and content changing trends in a video clip by exploring the tendencies of low-level visual feature distribution. BCS can represent a video as a compact signature, which is suitable for efficient comparison.

To transform a sequence of frame features into a BCS model, principal component analysis (PCA) is applied to project each histogram vector (or frame) to a new coordinate system. The basic idea of BCS is shown by an example in Fig. 3, where the black dots illustrate the frame feature vector distribution of a sample video clip, where each frame can be thought of as a data point in this system. The data is projected so that the greatest variance in it lies on the first principal component, the second greatest variance on the second and so on. The number of principal components is equal to the dimensionality of the histogram vector. Web videos can be noisy, for example there might be anomalies in the recording process which could significantly alter a frame's histogram and disrupt the continuity of the sequence. The

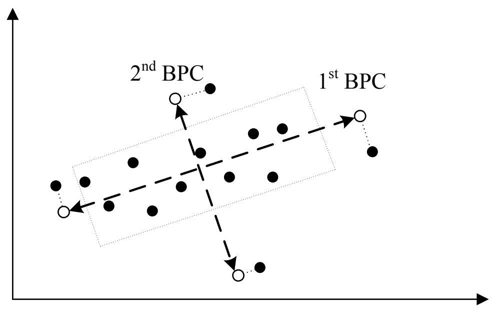


Fig. 3 Bounded coordinate system.

noise can be eliminated by using *bounded principal components* (BPC) where bounds are placed on the furthest data points in the directions of the principal components (see Fig. 3). Using standard deviations as thresholds, these outlying data points can be systematically dropped from the coordinate system. Another component in the BCS model is the *mean* of the data points. It is the origin of the system and is used in the visual similarity computation together with the BPC values.

A video X in BCS form has the notation $BCS(X) = (O^X, \ddot{\Phi}_1^X, \ldots, \ddot{\Phi}_d^X)$ where O is the mean and $\ddot{\Phi}$ is a BPC. d represents the dimensionality of the feature space, which in our case is the dimensionality of the image histograms. We compare two videos X and Y by using (4) which produces a dissimilarity measure D.

$$D(BCS(X), BCS(Y)) = \underbrace{\|O^X - O^Y\|}_{\text{by translation}} + \underbrace{(\sum_{i=1}^{d^Y} \|\ddot{\Phi}_i^X - \ddot{\Phi}_i^Y\| + \sum_{i=d^Y+1}^{d^X} \|\ddot{\Phi}_i^X\|)/2}_{\text{by rotation and scaling}} \quad (4)$$

This dissimilarity measure accounts for three operations which are useful in comparing the BCSs of two videos. The difference in mean values represents a *translation* distance between the origins of the two videos. Another operation is *rotation*: how far must one axis (BPC) be rotated to be the same as its counterpart? The other is *scaling*: how much must one axis shrink or stretch to be the same length as its counterpart? Intuitively, if we applied (4) to the BCSs of two videos for which the only difference was that one video was color shifted (i.e., every pixel had a constant intensity difference in relation to its counterpart), then the shift would be accounted for by a small operation and their dissimilarity would be minimal. In general, as a compact signature for the original video BCS is robust to various video transformations ranging from simple formatting to complex mixture of different editing effects [14]. Different from the traditional frame-based comparisons that involve quadratic computational complexity, visual similarity computation with BCS is only linear in the dimensionality of feature space and independent of video length. These properties make BCS an ideal visual representation model in our system.

4.2.2 Implementation issues

We assume that visual feature extraction and representation are pre-processed offline (by the *server side* of video search engine at the same time when videos are indexed). The process done online in real-time is the actual clustering of videos. This is appropriate for a real-world system because users have the expectation that after they enter a query, the relevant results should be returned quickly. The clustering may operate between query entry and results return time. Before it can operate, the videos must have undergone certain offline pre-processing.

We extended the TubeKit database schema so that the progress of each video through various processing stages can be monitored. This is useful because the system has a number of scripts which operate on bulk datasets. We want them to operate only on data which have not been processed. For a video, the stages which are monitored are whether the video has been *downloaded*, *converted from Flash to MPEG format*, *histogram-analyzed*, *BCS-analyzed*, and *processed for similarities with other videos*. A logical data flow is shown in Fig. 4.

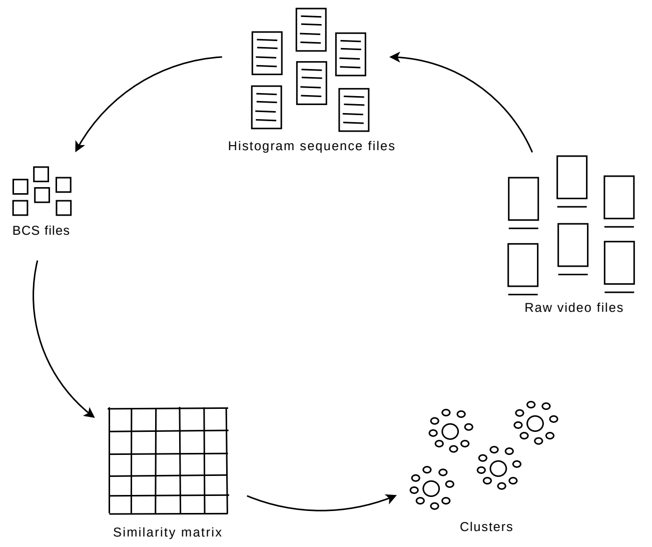


Fig. 4 System data flow.

After a video has been downloaded it is converted to MPEG format because we found that our feature extraction algorithm module (or perhaps the software library it uses) was more reliable taking MPEG as input rather than Flash video. The extraction produces a file with a histogram vector on each line, one line for a video frame. The histogram file of a video is used as input to the BCS algorithm. The BCS files are much smaller than histogram files as they only contain the BPC, mean and standard deviation vectors.

The BCS comparison algorithm accepts two BCS files from different videos. It produces a dissimilarity measure which is written to a video dissimilarity matrix stored in the database. This is done for every video pair which has already been BCS-analyzed - an example of the use of the tracking system.

4.3 Text processing

4.3.1 Computing similarity based on text analysis

With the maturity of Web 2.0, the context of a video is often very closer to its semantic meaning [30] and thus comparing textual features is beneficial. As we do for the videos, we also make comparisons for the associated metadata. The classical information retrieval method of computing text similarity is to use a set of pre-determined index terms to represent a *document* in the form of document-term vector. Vector similarity is then used to identify documents that are most related to a query. This is however inappropriate in our problem for comparing the similarity of *short sentences*. A sentence represented using a large number of pre-determined terms will result in a very sparse vector (i.e., the vector has a very small number of non-zero elements). The textual similarity computation method we adopt here is straightforward: two sentences or groups of words are searched for common words. For every common word they share, their similarity measure is increased by one. This measure is initialized to zero. In the example below, the similarity measure is 4 because there are four common words between the short sentences s_1 and s_2.

$$s_1 = \text{“Tiger Woods Greatest Golf Shot Ever”}$$
$$s_2 = \text{“Tiger Woods Amazing Golf Shot”}$$
$$Sim(s_1, s_2) = 4$$

Although we acknowledge the fact that a comprehensive scoring function of text semantic similarity should produce a more accurate similarity measure, we take a first rough cut at this problem and attempt to model the semantic similarity of texts as a function of the component words. This lexical matching method should be quite effective on the context information for which there is no concept of full sentence semantic meaning and which is more like a “bag of words”. For context information which are actual sentences, our method based on word co-occurrence might give reasonable results to some extend but it would ignore full sentence meaning.

4.3.2 Implementation issues

The context information that accompanies a video also has to be processed before any clustering of that video occurs. As with video processing, the stages that each video’s metadata has passed are monitored. The stages include *normalization* and *comparison*. The normalization process involves *stemming* and *removing stop words*. Stemming is achieved by a WordNet stemmer and can be summarized by an example: changing the word “realization” to the stem “realiz-”. For comparison purposes, this is useful because the words “realize” and “realization” will now be recognized by our text comparison algorithm as a similar word through their common stem. The second part of the normalization involves eliminating stop words such as “I”, “do” or “the” because they will inflate the score of the comparison algorithm and affect clustering quality. For example, two sentences being compared may have many stop words in common but this does not necessarily imply that the sentences are indeed relevant. All punctuation symbols are removed and capitalized letters substituted for similar reasons. We run a Perl script which normalizes given texts and then writes them into the main database. After normalization is complete every text is compared using the text comparison algorithm and given a similarity score which is written into another similarity matrix in the database. Tags are only compared with tags, title with title, and description with description. Therefore, for context information we store three similarity matrices, one for each source of metadata.

4.4 The clustering process

In this section, we present our framework for integrating information from various sources and how to exploit existing clustering algorithms to cluster videos in our framework. We also present an innovative interface for grouping and visualizing the search results.

4.4.1 Framework for information integration

Almost all video sharing websites have valuable social annotations [2] in the form of structured surrounding texts. We view a video as a multimedia object, which consists of not only the video content itself, but also lots of other types of context information (title, tags, description, etc.). Clustering videos based on just one of the information sources does not harness all the available information and may not yield satisfactory results. For example, if we cluster videos based on visual similarity alone, we cannot always get satisfactory outcomes because of the problem of semantic gap and excessively large number of clusters generated (how to effectively cluster videos based on the visual features is still an open problem and even a video and its edited fraction may be not grouped correctly). On the other hand, if we cluster videos based on some other type of information alone, e.g., textual similarity, we may be able to group videos by semantic topic, but their visual appearances are often quite diverse, especially for large clusters.

To address the above problem, we propose a framework for clustering videos, which simultaneously considers information from various sources (video content, title, tags, description, etc.). Formally, we refer to a video with all its information from various sources as a *video object* and the information from each individual source a *feature* of the video object. Our framework for information integration has three steps:

1. First, for each feature (video content, title, tags, or description), we compute the similarity between any two objects and obtain a similarity matrix using the methods described in Sections 4.2 and 4.3 (tags and description are also texts, and how to compute their similarities are also discussed in Section 4.3).
2. Second, for any two video objects X and Y, we obtain an *integrated similarity* by combining similarities of different features into one using the following formula:

$$Sim(X,Y) = \sum_{\text{for every feature } i} w_i \cdot Sim_i(X,Y) \tag{5}$$

   where $Sim(X,Y)$ is the integrated similarity, $Sim_i(X,Y)$ is the similarity of X and Y for feature i, and w_i is the weight of feature i. In our system, the current set of features is $\{visual, title, tags, description\}$. The weights are customizable to reflect the emphasis on certain features. For example, if *tags* is the main feature we would like to cluster the video objects on, then *tags* will be given a high weight. After computing the integrated similarity of every pair of objects, we obtain a square matrix for the integrated similarity with every video object corresponding to a row as well as a column. We call this matrix the *integrated similarity matrix*.
3. Third, a general-purpose clustering algorithm is used to cluster video objects based on the integrated similarity matrix.

The framework can incorporate any number of features. For example, almost all Web videos have an audio track which is also regarded a content feature. Thus, audio features such as Mel-frequency cesptral coefficients (MFCCs) might be further exploited to highlight similarities of videos.

Via our framework, many general-purpose clustering algorithm can be adopted to cluster the video objects. In our system, we implemented two state-of-the-art clustering algorithms, normalized cuts (NC) [29] and affinity propagation (AP) [9] as described in Section 3. The reason for choosing these two algorithms is that, NC is the most representative spectral clustering algorithm, while AP is one of few clustering algorithms that do not require users to specify the number of generated clusters. They both accept as input a similarity matrix which is indexed by video ID and is populated with pairwise similarities. The efficiency of the clustering process largely depends on the computation cost of generating the similarity matrix. Computing the integrated similarities through the elements from different similarity matrices is sufficiently fast, so that our proposed strategy is suitable to be practically deployed as a post-processing procedure for Web video search engines, where timely response is critical. We will compare the quality of the clustering results of these two algorithms in the experimental study.

4.4.2 Video clustering and result visualization

The integrated similarity matrix is the common input for the both clustering algorithms. It is derived from four similarity matrices which are computed from the different features we have analyzed (visual, title, tags and description) after the videos and associated metadata are pre-processed as described earlier.

We modified the NC and AP algorithms implemented by their respective authors so that the system now could retrieve subsets of the matrices from the database. This is needed by selecting the similarity matrices for video results for one query, say “tiger” or query number 2. After the clustering algorithm has been executed, a report is produced which provides information on how many clusters were generated for the given dataset, which videos were assigned to which clusters, and which videos are exemplars (for AP clustering only). We use such a report to evaluate the quality of clustering.

To make the best use of the clustering organization, we design an innovative interface for visualization of the clustering output, as shown in Fig. 5. We divide the available space of user interface into multiple displaying regions, one for each cluster. For each generated cluster, we summarize the information contained by the representative thumbnails and appropriate labels, in the hope that they can give users an immediate understanding of the cluster. Users can choose to navigate to the clusters of interest where further details can be obtained. Besides clustering organization, we believe their search experience also depends on these expressive messages. Particularly, for AP clustering some exemplar videos emerge after the algorithm finishing. Naturally their thumbnail images can be chosen as the representative thumbnails of the corresponding clusters. On the other hand, each cluster label should concisely indicate the contents of the cluster, while being consistent with the meanings of the labels of more general and more specific clusters. In our system, some query-related key phrases are extracted by exploring Web page search result clustering method [39], to enumerate the specific semantic of each cluster. Our design of selecting both titles and images as highlights of a search result document to be presented is supported by prior study [35], which shows that using mixed text summaries and thumbnails achieves a better performance than using either text summaries or thumbnail images in informing users of search results. With this interface, users can identify their desired results with much less effort and time compared with conventional browsing of a flat-ranked list. They can quickly grasp the different meanings of the query terms, and select a subset of relevant clusters easily.

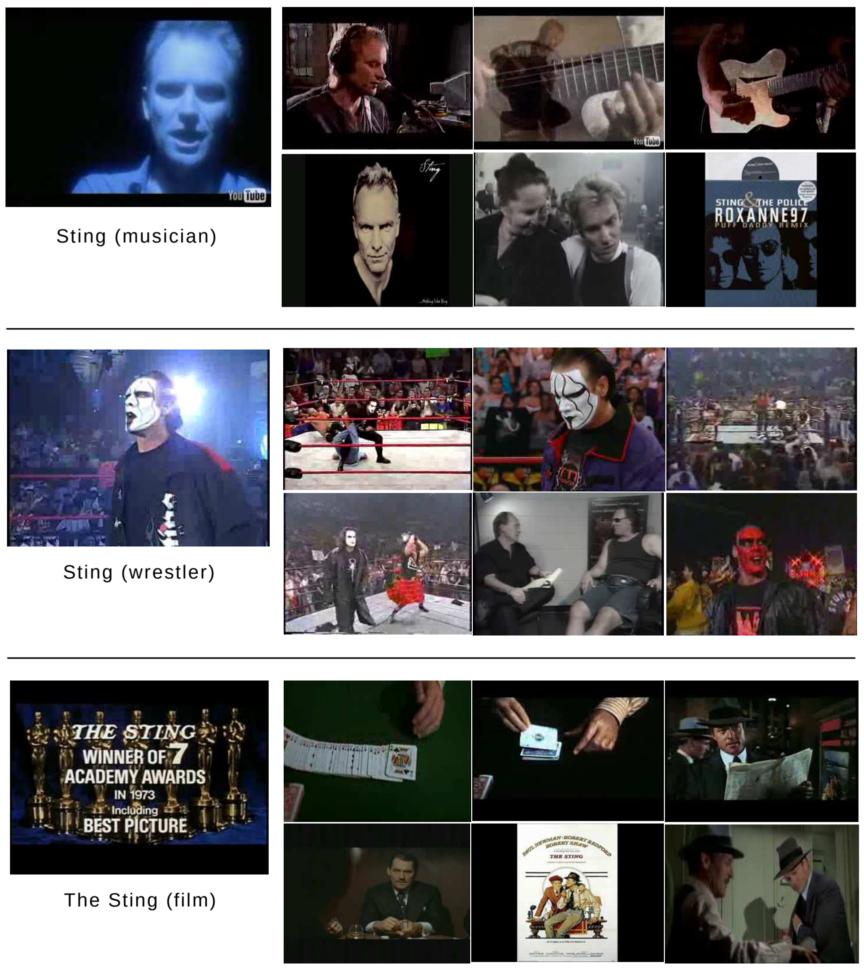


Fig. 5 Visualization of cluster results for the “sting” query.

5 Experiments

To evaluate the quality of the system’s clustering output in terms of its ability to correctly group video clips, we compared the computer-produced clusters to human-produced categories for a same query result set. First we discuss how this was done and what parameters were used to generate the results.

5.1 Methods

To obtain a result set of videos we supply queries to the Web video crawler component of the system. We assume that the users are interested in a particular aspect of the query, but do not know how to express this explicitly and submit a broader, more general query. It could be that the search results contain multiple topics, and there is no way to exactly figure out what is relevant to the user given that the queries are short and their interpretation is inherently vague in the absence of a context. For example, some polysemous terms such as "tiger", "pluto" and "sting" can be submitted. For "tiger" we may get results including videos relating to real feline, Tiger Woods the golfer, the Chinese horoscope and others. This is appropriate for our evaluation purpose because our system is expected to dynamically build a bunch of clusters that can reflect the semantically different categories in the raw set returned by the video search engine.

The result sets were filtered to contain video clips no longer than four minutes. Most video clips on YouTube fall beneath this threshold [21] and we found that the visual similarity computation algorithm does not perform well when videos have very long durations (BCS is designed for short video clips). We downloaded about one hundred video clips per query so that there would be an adequate number of videos per cluster. Very occasionally some videos in the result set were removed because they either had little or no relevance to search query or were isolated in their semantic meaning and thus would be alone in their category. This elimination is reasonable because we are testing a post-processing clustering functionality but not the search accuracy of a Web video search engine's retrieval component.

The affinity propagation (AP) clustering algorithm can automatically determine the number of clusters it generates, given any input as discussed above. The normalized cuts (NC) algorithm however cannot achieve this and must be given an expected number of clusters for the supplied integrated similarity matrix. In order to make a fair comparison between these two clustering algorithms, we run the AP algorithm on a result set, get the number of clusters it generates, and then supply the NC algorithm with the same number for the same result set.

We chose three weighting combinations which we thought would give a range of different results, as shown in Table 1. This table shows keys for different weights which correspond with results in the next section. For example, weight key A puts relatively more emphasis on visual similarities and less on textual similarities.

Weight Key	Visual	Tags	Title	Description	Combined
A	0.70	0.21	0.03	0.06	1.00
B	0.50	0.35	0.05	0.10	1.00
C	0.30	0.49	0.07	0.14	1.00

Table 1 Keys for matrix weights.

For evaluation, each video clip is manually labelled as belonging to a certain category by a human assessor. Cues used to determine memberships are the visual content and context information (tags, titles and descriptions) of videos. After a result set is divided into multiple categories by the human assessor, we can measure the performance of a clustering algorithm by comparing the categories and the generated clusters for the same set. The performance measure we use is *average precision*. Assume we are given a set of *n* returned videos belonging to c distinctive categories manually labelled, which are denoted by $1,\ldots,c$, and the clustering algorithm clusters the n videos into m groups C_j, $j = 1,\ldots,m$ indexed by j. The precision of a category is defined as

$$P(k) = \frac{\sum_{j=1}^{m_k} |C_{j,k}|}{\sum_{j=1}^{m_k} |C_j|} \tag{6}$$

where k identifies a category ($k = 1,\ldots,c$). $|C_j|$ is the size of cluster C_j (how many videos are in that cluster). $|C_{j,k}|$ is the number of videos in cluster C_j that belong to category k, according to manual labelling. m_k denotes the number of clusters containing video clips from category k. As an example, the clustering precision of one category for a "pluto" query could be $P(\text{former planet}) = 0.85$. This indicates that for this query, a high number of the former planet videos were indeed grouped correctly in the right cluster(s), but not all. For the same query term, we also calculate $P(\text{cartoon dog})$ and precisions for other "pluto" categories. Finally we take the average precision as an overall measure of the algorithm's performance for this query, which is in this case $P(\text{pluto})$. We show a range of average precision measures next.

5.2 Results

Table 2 shows some test query terms used to retrieve the video clip result sets, the number of video clips used in the clustering process, and the number of clusters generated by the AP algorithm for each feature weighting combination. Note that the algorithm often generated a different number of clusters for each weighting scheme. Table 3 shows the average precisions of the two clustering methods (AP and NC, respectively) across different queries and feature weighting combinations. It seems that in general, a higher textual feature weighting results in higher average precision for both affinity propagation and normalized cut clustering methods. There is a noticeable exception to this trend in the case of the "sting" query in AP - we believe that this was due to one category having significantly fewer labelled videos than the other categories. In cases where textual features are relatively lowly weighted or evenly weighted with visual features, the two clustering algorithms perform fairly evenly. When textual features have greater weighting, NC clustering generally outperforms AP clustering in average precision. This result is tempered by the fact that NC has to utilize AP's number of clusters as input. Despite this, in a real Web video search engine, the number of categories for certain queries may be fixed. In this case, the expected number of clusters could be stored as a heuristic in order to use the better performing algorithm.

Query	#Clips	#Clusters		
		A	B	C
Panda	90	15	13	13
Tiger	80	10	10	10
Sting	86	14	15	8
Python	100	12	12	7
Pluto	84	8	10	11

Table 2 Query list we tested in the system.

Query	Weight	AP	NC
Panda	A	0.44	0.45
	B	0.54	0.55
	C	0.58	0.78
Tiger	A	0.69	0.57
	B	0.74	0.63
	C	0.74	0.78
Sting	A	0.81	0.61
	B	0.61	0.74
	C	0.44	0.77
Python	A	0.79	0.56
	B	0.89	0.85
	C	0.53	0.70
Pluto	A	0.37	0.35
	B	0.31	0.61
	C	0.32	0.79

Table 3 Average precision by query.

The trend in the results indicates that the visual feature component could be less effective in producing good clusters when highly weighted relative to textual features. This is expected as there are quite a number of video clips in the returned result set which actually have the same semantic meaning but different visual appearances. We also observed that videos with similar visual content were almost always grouped together, even when the visual feature weighting was lower. In other words, the majority of duplicate or near-duplicate videos can be correctly clustered and no longer be repeatedly represented as in the output of current video search engines.

We noted that the computer-produced clusters were often smaller than the corresponding human-produced categories. That is, often the system tends to generate more than one clusters which belonged to the same category judged by the assessor. This is probably an indication that the categories determined by the assessor are subjective and that other assessors might separate the categories further.

The results indicate that there is no unanimously optimum feature weighting combination which can be applied across all query result sets, although generally a higher textual feature weighting increases average precision. A solution might be to use empirical evidence to determine the best weighting schemes for certain queries and use heuristics to apply them at run-time. Our best result was achieved with weight key B for the "python" query where there were not many labelled categories (the snake, Monty Python, etc.). Both clustering algorithms performed well here with AP outperforming NC. Unfortunately there was no perfect result yet (i.e., $P(\text{query}) = 1$). Future work is outlined in the next section which would help in approaching this score.

In summary, our system can deliver a much better presentation of search results, yet with similar response time to other video search engines. Overall speaking, AP is recommended to be employed as the underlying clustering algorithm, due to three reasons. First, it can automatically determine an adaptive number of clusters. Second, the exemplar of each generated cluster can be naturally used as the most representative video in our user interface for result visualization. Third, its clustering performance is slightly better than NC in our larger experiments with more queries.

6 Conclusion and future work

We have developed a Web video search system which has additional post-processing functionality of clustering returned results. This enables users to identify their desired videos more conveniently. Our proposed information integration framework is the first attempt to investigate the fusion of the heterogeneous information from various sources for clustering. The main infrastructure of the system is complete and if we wish it is readily extendible to integrate and test other video clip and text comparison algorithms, as well as clustering algorithms, which may further improve the quality of clustering.

Currently, comparing visual features is based on global color histogram distribution of video frames. The effects of other features on clustering quality could be investigated. Analyzing high-level visual features is computationally costly but if the processing is done offline then the reward may be worthwhile. The textual similarity measure we used looks simple compared with more advanced techniques for inferring short text semantic similarity [23, 20, 15], which probably could be used on the sentence-oriented title and description metadata to good effect. Probabilistic latent semantic indexing can be used to discriminate subjective tags and build a hierarchy with objective tags [8]. In addition, by translating and comparing context information in different languages, it would be possible to source and cluster desired video clips even though they might have context information in a language different to that of the query.

Moreover, currently we assume that every video ought to be assigned to at least one cluster. However, due to the limitations of search engine technology, only partial returned videos are indeed relevant to query and some irrelevant videos are mixed (i.e., some noisy results are likely to be returned). To tackle this problem, another research direction might be in investigating a new scheme of clustering, Bregman bubble clustering [1], to cluster returned videos. The algorithm can obtain the dominant clusters by partially grouping videos in the whole set while discarding some scattered noisy ones.

References

1. Banerjee, A., Merugu, S., Dhillon, I.S., Ghosh, J.: Clustering with bregman divergences. Journal of Machine Learning Research **6**, 1705–1749 (2005)
2. Bao, S., Yang, B., Fei, B., Xu, S., Su, Z., Yu, Y.: Social propagation: Boosting social annotations for web mining. World Wide Web **12**(4), 399–420 (2009)
3. Cai, D., He, X., Li, Z., Ma, W.Y., Wen, J.R.: Hierarchical clustering of www image search results using visual, textual and link information. In: ACM Multimedia, pp. 952–959 (2004)
4. Carpineto, C., Osinski, S., Romano, G., Weiss, D.: A survey of web clustering engines. ACM Comput. Surv. **41**(3) (2009)
5. Cheung, S.C.S., Zakhor, A.: Efficient video similarity measurement with video signature. IEEE Trans. Circuits Syst. Video Techn. **13**(1), 59–74 (2003)
6. Cutting, D.R., Karger, D.R., Pedersen, J.O.: Constant interaction-time scatter/gather browsing of very large document collections. In: SIGIR, pp. 126–134 (1993)
7. Cutting, D.R., Pedersen, J.O., Karger, D.R., Tukey, J.W.: Scatter/gather: A cluster-based approach to browsing large document collections. In: SIGIR, pp. 318–329 (1992)
8. Eda, T., Yoshikawa, M., Uchiyama, T., Uchiyama, T.: The effectiveness of latent semantic analysis for building up a bottom-up taxonomy from folksonomy tags. World Wide Web **12**(4), 421–440 (2009)
9. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**(5814), 972–976 (2007)
10. Gao, B., Liu, T.Y., Qin, T., Zheng, X., Cheng, Q., Ma, W.Y.: Web image clustering by consistent utilization of visual features and surrounding texts. In: ACM Multimedia, pp. 112–121 (2005)
11. García, R., Gimeno, J.M., Perdrix, F., Gil, R., Oliva, M., López, J.M., Pascual, A., Sendín, M.: Building a usable and accessible semantic web interaction platform. World Wide Web **13**(1-2), 143–167 (2010)
12. Gibbon, D.C., Liu, Z.: Introduction to Video Search Engines. Springer (2008)

13. Golub, G.H., Loan, C.F.V.: Matrix Computations., third edn. The Johns Hopkins University Press (1996)
14. Huang, Z., Shen, H.T., Shao, J., Zhou, X., Cui, B.: Bounded coordinate system indexing for real-time video clip search. ACM Trans. Inf. Syst. **27**(3) (2009)
15. Islam, A., Inkpen, D.Z.: Semantic text similarity using corpus-based word similarity and string similarity. TKDD **2**(2) (2008)
16. Jansen, B.J., Campbell, G., Gregg, M.: Real time search user behavior. In: CHI Extended Abstracts, pp. 3961–3966 (2010)
17. Jansen, B.J., Spink, A., Saracevic, T.: Real life, real users, and real needs: a study and analysis of user queries on the web. Inf. Process. Manage. **36**(2), 207–227 (2000)
18. Jing, F., Wang, C., Yao, Y., Deng, K., Zhang, L., Ma, W.Y.: Igroup: web image search results clustering. In: ACM Multimedia, pp. 377–384 (2006)
19. Kummamuru, K., Lotlikar, R., Roy, S., Singal, K., Krishnapuram, R.: A hierarchical monothetic document clustering algorithm for summarization and browsing search results. In: WWW, pp. 658–665 (2004)
20. Li, Y., McLean, D., Bandar, Z., O'Shea, J., Crockett, K.A.: Sentence similarity based on semantic nets and corpus statistics. IEEE Trans. Knowl. Data Eng. **18**(8), 1138–1150 (2006)
21. Liu, S., Zhu, M., Zheng, Q.: Mining similarities for clustering web video clips. In: CSSE (4), pp. 759–762 (2008)
22. Mecca, G., Raunich, S., Pappalardo, A.: A new algorithm for clustering search results. Data Knowl. Eng. **62**(3), 504–522 (2007)
23. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI (2006)
24. Osinski, S., Weiss, D.: A concept-driven algorithm for clustering search results. IEEE Intelligent Systems **20**(3), 48–54 (2005)
25. Rege, M., Dong, M., Hua, J.: Graph theoretical framework for simultaneously integrating visual and textual features for efficient web image clustering. In: WWW, pp. 317–326 (2008)
26. Shah, C.: Tubekit: a query-based youtube crawling toolkit. In: JCDL, p. 433 (2008)
27. Shen, H.T., Ooi, B.C., Zhou, X., Huang, Z.: Towards effective indexing for very large video sequence database. In: SIGMOD Conference, pp. 730–741 (2005)
28. Shen, H.T., Zhou, X., Cui, B.: Indexing and integrating multiple features for www images. World Wide Web **9**(3), 343–364 (2006)
29. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
30. Siorpaes, K., Simperl, E.P.B.: Human intelligence in the process of semantic content creation. World Wide Web **13**(1-2), 33–59 (2010)
31. Snoek, C., Worring, M.: Multimodal video indexing: A review of the state-of-the-art. Multimedia Tools Appl. **25**(1), 5–35 (2005)
32. Taddesse, F.G., Tekli, J., Chbeir, R., Viviani, M., Yétongnon, K.: Semantic-based merging of rss items. World Wide Web **13**(1-2), 169–207 (2010)
33. Wang, H., Divakaran, A., Vetro, A., Chang, S.F., Sun, H.: Survey of compressed-domain features used in audio-visual indexing and analysis. J. Visual Communication and Image Representation **14**(2), 150–183 (2003)
34. Wang, X.J., Ma, W.Y., Zhang, L., Li, X.: Iteratively clustering web images based on link and attribute reinforcements. In: ACM Multimedia, pp. 122–131 (2005)
35. Woodruff, A., Rosenholtz, R., Morrison, J.B., Faulring, A., Pirolli, P.: A comparison of the use of text summaries, plain thumbnails, and enhanced thumbnails for web search tasks. JASIST **53**(2), 172–185 (2002)
36. Xu, S., Jin, T., Lau, F.C.M.: A new visual search interface for web browsing. In: WSDM, pp. 152–161 (2009)
37. Yang, J., Li, Q., Wenyin, L., Zhuang, Y.: Searching for flash movies on the web: A content and context based framework. World Wide Web **8**(4), 495–517 (2005)
38. Zamir, O., Etzioni, O.: Grouper: A dynamic clustering interface to web search results. Computer Networks **31**(11-16), 1361–1374 (1999)
39. Zeng, H.J., He, Q.C., Chen, Z., Ma, W.Y., Ma, J.: Learning to cluster web search results. In: SIGIR, pp. 210–217 (2004)