

Optimizing Visual Search with Implicit User Feedback in Interactive Video Retrieval

Stefanos Vrochidis
Queen Mary University of London
Mile End Road, London, UK /
Informatics and Telematics Institute
Thermi, Thessaloniki, Greece
+302311257754
stefanos@iti.gr

Ioannis Kompatsiaris
Informatics and Telematics Institute
Thermi, Thessaloniki, Greece
+302311257774
ikom@iti.gr

Ioannis Patras
Queen Mary University of London
Mile End Road, London, UK
+442078827523
I.Patras@elec.qmul.ac.uk

ABSTRACT

This paper describes an approach to optimize query by visual example results, by combining visual features and implicit user feedback in interactive video retrieval. To this end, we propose a framework, in which video processing is performed by employing well established techniques, while implicit user feedback analysis is realized with a graph based approach that processes the user actions and navigation patterns during a search session, in order to initiate semantic relations between the video segments. To combine the visual and implicit feedback information, we train a support vector machine classifier with positive and negative examples generated from the graph structured past user interaction data. Then, the classifier reranks the results of visual search that were initially based on visual features. This framework is embedded in an interactive video search engine and evaluated by conducting a user experiment in two phases: first, we record the user actions during typical retrieval sessions and then, we evaluate the reranking of the results of visual query by example. The evaluation and the results demonstrate that the proposed approach provides an improved ranking in most of the evaluated queries.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *relevance feedback, retrieval models, search process.*

General Terms

Algorithms, Performance, Experimentation.

Keywords

Implicit feedback, machine learning, search engine, interactive, video retrieval.

1. INTRODUCTION

Nowadays, due to the rapid progress in network technologies and the enormous storage capabilities of computers, very large amounts of audiovisual content have become available worldwide.

The availability of such content places the demand for advanced multimedia search engines; therefore video retrieval remains one of the most challenging tasks of research. Despite the significant advances in this area recently, further advancements in multiple fields of video retrieval are required to improve the performance of video search engines. One of the recent techniques applied, in order to improve the results of search engines, attempts to exploit the implicit feedback provided by the users [1] of a video search engine. We consider as implicit user feedback any action or navigation behavior of the user during the interactive query session, including mouse movements and clicks, as well as keyboard inputs and keystrokes. The main advantage of using implicit techniques is that they do not require any explicit feedback from the user side. Although implicit information is in general thought to be less accurate than explicit [2], large quantities of implicit data (e.g. log files in web search engines) can be gathered at no extra effort to the user. Recent approaches in interactive video retrieval analyze and process past user interaction data either for performing retrieval [1], or for complementing existing content based search modalities [3]. In this context, this paper proposes a novel approach for combining implicit user feedback with visual features, in order to improve the results of query by visual example. Motivated by the fact that content-based search, which relies only on visual features, could yield results that resemble the query image only in terms of color or texture but without having any semantic resemblance with it, this paper argues that taking into account the implicit user feedback to rerank these results, would improve the performance of the system and add a semantic flavor in the visual search.

To this end, we design a video indexing and retrieval framework, in which, this heterogeneously extracted information (i.e. implicit feedback and visual features) is combined by employing machine learning methods. Video processing is performed with well established techniques techniques, including video to shot segmentation, keyframe extraction and generation of visual features. Then, we exploit the implicit user feedback, in order to initiate semantic relations between the video segments and define positive and negative examples for a given visual query. This is realized with the following methodology. First, the user actions such as mouse clicks and keyboard inputs are recorded and an action graph that describes the navigation of the user during the search process is constructed, considering as nodes the queries and the segmented video shots and as links the user actions. Subsequently, this graph is converted to a weighted graph by translating the actions into weights. During a query by visual example, this graph is utilized in order to define positive and

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CIVR'10, July 5-7, Xi'an, China

Copyright © 2010 ACM 978-1-4503-0117-6/10/07... \$10.00

negative samples by employing a distance-based algorithm. The latter are merged with a set of visually similar and dissimilar examples based on visual features, in order to construct a final training set, which is used to train a Support Vector Machine (SVM) classifier that reranks the results of the visual search. This framework is embedded in an interactive video search engine, which supports basic retrieval functionalities including text, visual and temporal search. The evaluation of the system is performed by conducting a user experiment in two phases: in the first part, the users are searching for 6 different topics and their actions are being recorded, while in the second phase, different users are recruited to evaluate the reranking of the proposed approach. The main contribution of this work is the methodology for combining visual features with implicit user feedback, which adds a semantic flavor to visual search. To the best of the authors' knowledge this is the first attempt of improving visual search in video retrieval by combining visual features with graph structured implicit user feedback expressed as clickthrough data.

This paper is structured as follows: section 2 presents the related work, while in section 3 we introduce the SVMs employed in this approach. The video indexing and retrieval framework is presented in section 4, while section 5 deals with the video content analysis. Section 6 describes the processing of user implicit actions based on a graph approach and section 7 presents the methodology for combining visual features with graph structured implicit user feedback. Section 8 describes the experimental results and the evaluation and finally, section 9 concludes the paper.

2. RELATED WORK

Implicit feedback approaches have been effective in textual retrieval, where they were mostly employed for query expansion, in order to retrieve, filter and recommend items of interest [4]. The "Implicit Interest Indicators" were introduced in [5] as a definition of specific user actions that can be considered as meaningful implicit feedback. In [6], the authors performed a comparison between an explicit and an implicit feedback system concluding that there were not significant differences between these two systems and that substituting the former with the latter could be feasible. In another interesting work, in which text information was combined with user implicit feedback, the query-logs of a search engine were utilized to learn retrieval functions with the aid of machine learning [7]. More specifically the clickthrough data are translated into ranking user preferences and then they are used to train a retrieval function with a SVM approach. In [8] the authors propose to detect query chains (i.e. a sequence of queries) and then learn a retrieval function with SVM.

Implicit feedback techniques have not been fully explored in the multimedia domain [9]. In text retrieval, the usual implicit information that can be taken into account is the user selection (i.e. the user clicks on an interesting link or textual description to view the complete document), while in video retrieval we have multiple interactions between the user and the system, which could be utilized to provide meaningful feedback. The main idea to exploit the user feedback during video retrieval interactive sessions, is to extend the idea of "query chains" [8] and construct a graph that describes a series of user actions. Such a graph is transformed to a weighted graph by aggregating the links between the same nodes and weights are introduced based on the different actions taken into account. Some recent works [1], [10] employ

the aforementioned technique to deal with user clicks. More specifically, in [1] a video retrieval system, which is based on Okapi BM25 retrieval model supporting text queries and enhanced by a recommendation generator utilizing the weighted graph structure, is evaluated. In [10] the authors evaluate 4 different algorithms that can be applied on such weighted graphs to provide recommendations. However, these works consider only textual queries, while basic video retrieval options as visual and temporal based search are ignored. In addition, fusion or combination of implicit feedback data with the content-based approaches is not attempted.

In another work [3], a video retrieval system is presented, which employs relevance feedback and multimodal fusion of different sources (textual, visual and click-through data), in order to generate recommendations for the user. In this approach the textual, visual and aural data of the video shots are processed separately and compared with the video document clicked. Then, these results are fused. A further adjustment of the fusion weights is performed with the aid of the click through data, which denote the interest of the user to a specific document based on the time she watched the video shot. The approach of fusing content analysis information with the implicit feedback seems to be very interesting and promising, however in this specific work the implicit information is not very deeply exploited as no sequence of query actions are taken into account, failing in that way to semantically connect subsequent queries and shots. In comparison to the proposed approach, this work employs implicit user feedback for adjusting weights, in order to fuse efficiently the results from different search modalities (e.g. text, visual) and not for improving the results of a specific retrieval module (i.e. in this case visual search).

3. SUPPORT VECTOR MACHINES

Support vector machines constitute a set of supervised learning methods used for classification and regression. When a set of training positive and negative examples is available, a SVM training algorithm builds a model that predicts in which category a new example falls into. To achieve that, a support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite dimensional space. In general, it is assumed that the best separation is achieved by the hyperplane that has the largest distance from the nearest training datapoints of any class.

In this approach, we employ a SVM implementation that realizes the alternative structural formulation of the SVM optimization problem for conventional binary classification with error rate described in [11]. For a given training set $(x_1, y_1), \dots, (x_n, y_n)$ with $x_i \in R^N$ and $y_i \in \{-1, +1\}$, training this binary classification SVM solves the following optimization problem, which was proposed for predicting structured outputs and optimizing multivariate performance measures like F_1 -Score or the Precision/Recall Break-Event Point [12].

$$\text{mean}_{w, \xi \geq 0} \frac{1}{2} w^T w + C \xi \quad (1)$$

$$\text{subject to: } \forall c \in \{0, 1\}^n: \frac{1}{n} w^T \sum_{i=1}^n c_i y_i x_i \geq \frac{1}{n} \sum_{i=1}^n c_i - \xi \quad (2)$$

where C is the capacity constant and w a parameter vector. This approach has 2^n constraints, one for each possible vector $c = (c_1, \dots, c_n) \in \{0, 1\}^n$ and it has only one slack variable ξ that is shared across all constraints. The algorithm that is employed to solve the aforementioned classification SVM optimization problem is an adaptation of the Cutting-Plane Algorithm. This

algorithm iteratively constructs a sufficient subset W of the set of constraints. Starting with an empty set of constraints W , in each iteration, it first computes the optimum over the current working set W (i.e. $w = 0$ and $\xi = 0$ in the first iteration) and then it finds the most violated constraint in the optimization problem and adds it to the working set W [11].

The reason for selecting this specific SVM in this approach, was the very fast performance it demonstrates, which was important for performing the ranking optimization at real time during the query process, with an adequate training set.

4. VIDEO RETRIEVAL FRAMEWORK

The overall video indexing and retrieval framework employed in this work, is illustrated in Figure 1. At the bottom part, standard video analysis is performed, including video segmentation to shots, textual analysis of recorded audio and visual processing of extracted keyframes. On the top part of the framework, implicit feedback is captured in log files and analyzed. The outputs of these two parts are combined in a separate component, which employs training of a SVM classifier, in order to rerank the results of visual search. Processing of implicit information and video content based analysis take place off line, while the combination of visual and implicit information is realized on the fly (i.e. in real time when a new query is submitted). In the next sections these parts will be described in detail.

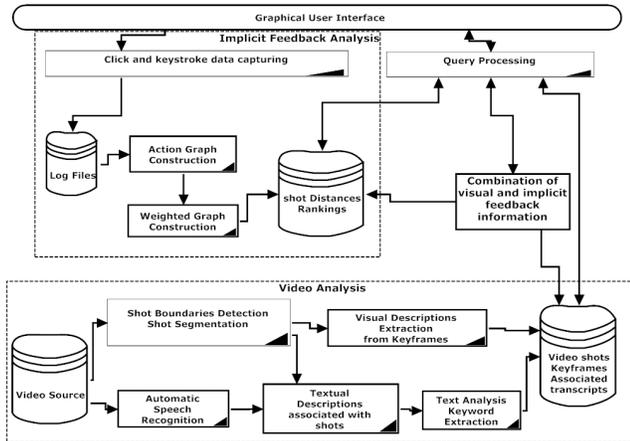


Figure 1. Video Indexing and Retrieval Framework

5. VIDEO ANALYSIS

The role of video analysis is to process the initial video source and create multiple indexing structures that can be utilized for retrieval. Indexing is performed according to the following heterogeneous information:

- Temporal information
- Textual description
- Visual characteristics

5.1 Temporal Indexing

In order to index the initial video source according to temporal information, the video is split into shots by performing shot boundaries detection and shot segmentation. Shot detection is achieved by thresholding the distance between color histograms corresponding to two consecutive frames in a video [13]. Then each video is segmented into smaller shots the middle keyframe

for each shot, which is considered as the representative one, is extracted. In that way a temporal indexing is constructed, so each video can be represented by a sequence of images (i.e. one image per video shot).

5.2 Keyword Indexing

Indexing of video shots according to the associated textual information is realized following the approach of [14]. The audio information is processed off-line with the application of Automatic Speech Recognition (ASR) to the initial video, so that specific sets of keywords can be assigned to each shot. Indexing and query functions are implemented utilizing the KinoSearch full-text search engine library [15]. First, stopwords are removed from keyword files followed by the application of the Porter stemming algorithm [16]. Then, term weights for each keyword are computed utilizing the BM25 text algorithm [17].

5.3 Visual Similarity Indexing

Indexing of shots according to visual similarity is performed by extracting low level descriptors following the approach of [17]. In this case five MPEG-7 descriptors namely Color Layout, Color Structure, Scalable Color, Edge Histogram, Homogeneous Texture are generated from the representative keyframe of each video shot [18]. By concatenating these descriptors, a feature vector is formulated to compactly represent each image in the multidimensional space. An empirical evaluation of the system performance using different combinations of the aforementioned descriptors, advocated the choice of one MPEG-7 based scheme, which relies on color and texture (i.e., ColorLayout and EdgeHistogram are concatenated) [14]. Distance calculation between the descriptors of two shots is performed by employing the functions proposed in the MPEG eXperimentation Model [19]. In order to improve the performance of the system in terms of time response, we employ a multi-dimensional r-tree indexing structure, which is constructed off line using the feature vectors of all images. R-tree(s) [20] are structures suitable for indexing multidimensional objects and known to facilitate fast and efficient retrieval on large scale. In the query phase, the feature vector of the query shot (i.e. from the representative keyframe of it) Q is extracted and it is submitted to the r-tree indexing structure. The latter returns a set of not ranked K results R_V , which contains the shots s_i , ($0 \leq i \leq K$) that are found to resemble the query one. The final visual ranking is performed by calculating the visual distances d_V between Q and all shots in R_V , so we have $d_V(Q, i) = f_V(Q, s_i)$, where f_V is the visual distance computing function and $s_i \in R_V$. The ranking for a query Q can be described as a new ordered set $RK_V = (s_a, s_b, s_c, \dots)$, where $d_V(Q, a) \leq d_V(Q, b) \leq d_V(Q, c), \dots$ and the cardinality of RK_V elements is K .

6. IMPLICIT FEEDBACK ANALYSIS

This section presents the analysis of implicit user feedback expressed in mouse clicks and keyboard inputs. First, we define the implicit interest indicators that we will consider for video search, then we construct action graphs based on the user navigation patterns and finally we convert them to a weighted graph, which can be utilized to generate distances between the video shots.

6.1 Implicit Interest Indicators

To define implicit interest indicators [5] for interactive video retrieval, we need to identify the actions of a user that could

declare interest and could convey meaningful information about his preferences. Based on the search functionalities offered by the video analysis and indexing modules, we introduce the following minimum set of user actions that can be considered as the main implicit interest indicators for video retrieval:

1. Text-based query (TQ): the user inserts a keyword and submits the query. In this case, we assume that this keyword satisfies the query of the user with a very high probability.
2. Visual query (VQ): the user submits a visual query by example. We assume that, when a user selects a keyframe and searches for visually similar images, then there is also interest in the example that used.
3. Side-shot query (SQ): the user selects a shot in order to view the temporally adjacent shots. In that case, the user is very likely to be interested in the shot he selected.
4. Video-shot query (VSQ): the user selects a shot and retrieves all the shots of the same video. In such a scenario, we consider that he is interested in the initial shot to a certain extend.
5. Submit a shot (SS): the user marks a shot as relevant. In this case, we assume that the user is very interested in this shot. In a web search system this action could be equivalent with watching the video shot.

6.2 Action Graph

We exploit implicit feedback information by employing an extended variation of the graph construction methodology proposed in [1]. While [1] considers only text-based queries, we deal with a more complex situation as visual-based and temporal-based queries are also included. We define as “search session” the time period that a certain user spends on searching. To construct such a graph, we exploit the properties of the involved user actions. First, we introduce the property of “transitivity”, which characterizes an action according to its output. More specifically, we consider an action as “transitive”, when it generates an output (i.e. $input \rightarrow action \rightarrow output$). On the other hand, when an action does not provide any output, (i.e. $input \rightarrow action$) it is characterized as “intransitive”. During a search session it is possible to have a series of transitive actions, where part of the output of one action is the input for another (e.g. a result from a text search is the input for a visual search). Consequently, to create a link between two nodes of an action graph, we need to have a sequence of two actions, where at least the first one has to be transitive.

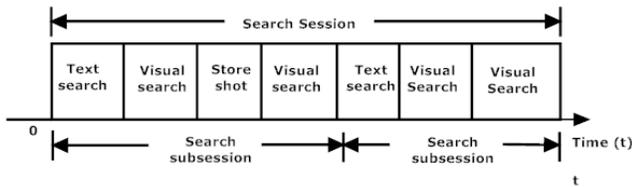


Figure 2. Search session and subsessions.

In addition, we classify the query actions into two main categories based on their dependency on previous actions: a) the autonomous queries, which do not depend on previous results and b) the dependent queries, which take as input results from previous search actions. During a search session, the user may search for a specific topic, however it is possible to perform search having a

very broad or complex topic in mind or decide to change the search topic during the session. Assuming that every autonomous query could initiate a different topic search, we divide the search sessions into “search subsessions” using as break points the autonomous queries. We consider the “search subsession” as the time periods, in which a certain user spends searching for a specific topic. Taking into account the corresponding functionalities of the introduced implicit interest indicators, only the text-based search can be denoted as autonomous query, while the other queries are considered as dependent.

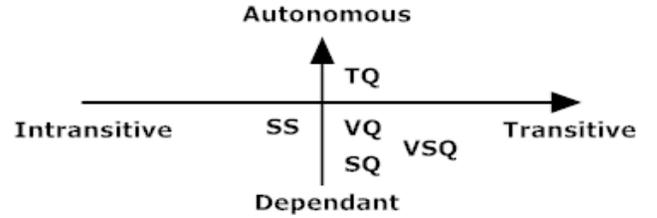


Figure 3. Actions classification

In such a case the text-based query is utilized as a break point between the subsessions as illustrated in the example of Figure 2. The overall classification of these functionalities can be visualized in the two different axes of transitivity and dependency as illustrated in Figure 3.

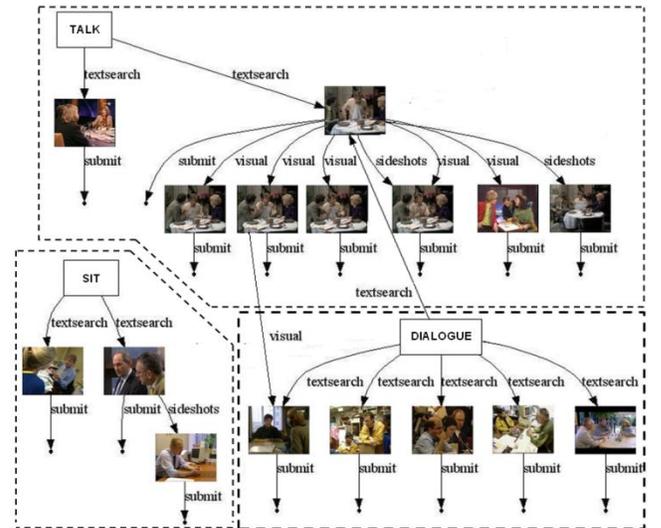


Figure 4. Action graph after user interaction.

In the general case, a search subsession S consists of a set of actions A_S that includes one autonomous and a number of dependent query actions. The proposed subgraph G_S is comprised by a set of nodes (i.e. shots and keywords that represent inputs and outputs of a set of actions A_S) and links that represent the corresponding actions $a_i \in A_S$, where $i \in \{1, \dots, N_S\}$ and N_S is the cardinality of the elements of A_S . These are illustrated in the example of Figure 4, where an action graph for a search session is presented. Here, the user is searching for shots, in which people are sitting in a table talking are depicted. We observe that the three keywords that were used to start the search (i.e. talk, sit and dialogue) were considered as the parents for new subgraphs. Then we construct a single action graph aggregating the action graphs

from the different user sessions. More specifically, all the nodes from the individual action graphs are mapped to single action graph, and then all the action edges are mapped onto the same graph, generating in that way multiple links between the nodes. Although the implicit user feedback is considered noisy (i.e. it can contain irrelevant information), we assume that by aggregating inputs from different users, we are able to minimize the introduced error.

6.3 Weighted Graph

After the construction of a single action graph, we generate the weighted graph by a) linking the relevant results to the parent query, transforming in that way the intransitive actions into transitive, b) collapsing the multiple links between the same nodes into one and c) translating actions into weights.

The final normalized weight w (with a range of values between 0 and 1) for a link n between two nodes k and m is given by the formula:

$$w(n) = 1 - \frac{1}{x(n)} \quad (3)$$

where $x(n)$ is the sum of weights of each action that interconnects nodes k and m . This sum is expressed as:

$$x(n) = \sum_{a \in U_s} g(a) \quad (4)$$

where g is a function that maps each action to an implicit weight and a is an action that belongs to the set of actions $U_s \subseteq A_s$ that comprise the different links between the nodes k and m [1]. Following the analysis in section 6.1, we assign indicative values (between 0 and 10) that quantify the level of interest associated to the introduced implicit interest indicators (Table 1). In that way for instance, we incorporate to the graph the assumption that the shot selected for a visual query by example is more of interest to the user compared to a shot selected for viewing its temporally adjacent shots. Figure 5 illustrates the weighted graph that is produced after processing the action graph of Figure 4.

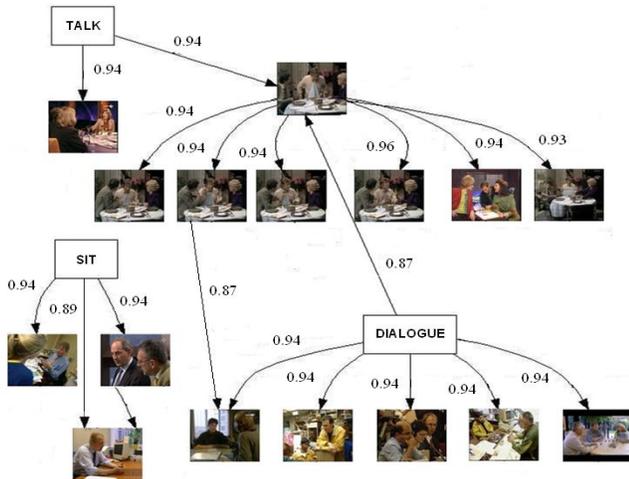


Figure 5. Weighted graph after processing the action graph of Figure 4.

In order to calculate distances between two different nodes in this graph, we apply the Dijkstra algorithm [21], which computes the shorter path between two nodes. As this distance is based on implicit information but it reveals semantic relations, we name it

“implicit semantic distance”. Hence, based on the shorter path approach, we can calculate the implicit semantic distance of each shot Q that is represented as a node in the graph, with the rest of the shots included in the graph. Formally, we compute $d_l(Q, i) = f_l(Q, s_i)$, where f_l is the implicit distance computing function, $s_i \in R_l$, R_l is the set of M shots that are interconnected through links with the query shot Q and $1 \leq i \leq M$. The ranking for query Q can be described as a new ordered set $RK_l = (s_{\hat{a}}, s_{\hat{b}}, s_{\hat{c}}, \dots)$, where $d_l(Q, \hat{a}) \leq d_l(Q, \hat{b}) \leq d_l(Q, \hat{c}), \dots$ with a cardinality of M elements.

Table 1. Assign weights for each action

| Actions(a_i) | $g(a_i)$ | Actions(a_i) | $g(a_i)$ |
|------------------------|----------|--------------------|----------|
| Text-based query (TQ) | 8 | Visual query (VQ) | 8 |
| Side-shot query (SQ) | 7 | Submit a shot (SS) | 9 |
| Video-shot query (VSQ) | 6 | | |

7. COMBINING VISUAL AND IMPLICIT FEEDBACK INFORMATION

The objective of this part, is to rerank the initial results of visual search by exploiting the weighted graph. Although the results obtained based on visual descriptors are usually quite satisfactory, there are cases, in which visual search fails to fetch results of the same semantic meaning confused by similar colors or textures of irrelevant depictions. As discussed in section 5.3, visual search is performed in two steps: i) by submitting the query descriptors to the R-tree structure and ii) by ranking the results returned utilizing the distances between visual descriptors. The idea is to replace the ranking function of the second step with a classifier, which is trained with semantically positive and negative shots, in order to emphasize more on the specific visual features that can be of importance for each query. More specifically, we train a classifier for each visual example query by utilizing as training set a combination of visually similar and dissimilar examples, as well as positive and negative samples generated by implicit feedback information, in order to rerank the initial visual results. In Figure 6 the overall algorithm of the proposed approach is presented.

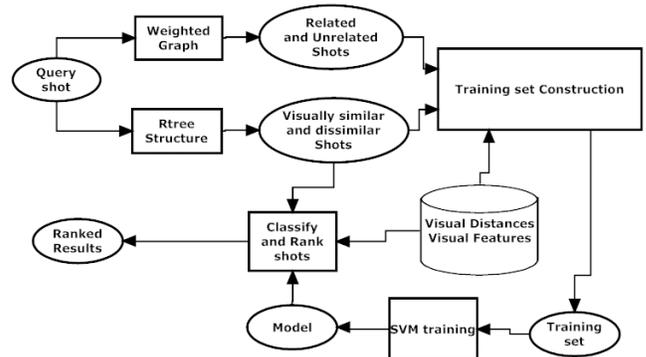


Figure 6. Algorithm to combine visual features and implicit user feedback

As shown in the diagram, when a query by shot example Q is submitted, we produce the following ranked datasets of results: two sets R_l and U_l using the weighted graph that is obtained utilizing the user implicit feedback and one set R_V using the r-tree structure that includes visually related shots according to the

extracted visual features. Subsequently these sets are merged, in order to construct a training set T and then train a support vector machine classifier utilizing as features the visual descriptors. Finally, we employ the R_V (i.e. the set of results from visual search) as the test set, which is ranked according to the degrees of coefficients that are outputted by the classifier and represent a similarity metric between each segment of the test set and the query shot. In the next sections we will provide the details about the training set construction and the SVM training.

7.1 Training set construction

In order to train a binary support vector machine classifier, we need to identify a training set $T = T_P \cup T_N$, where T_P is the set of the positive and T_N the set of the negative samples. Utilizing the weighted graph, we can extract a number of positive samples that are closer to the query shot and a number of negative samples that are placed as further as possible in this graph. Hence we create the set of positive samples $T_{I,P} = R_{I,P}$, where $\forall s_i \in R_{I,P}, d_I(Q, i) < d_{I,thresh}$ and $d_{I,thresh}$ is an experimentally set threshold. In the same way we define a set of negative examples by employing another distance threshold $d_{I,hthresh}$ and we consider the $T_{I,N} = U_I$, where $\forall s_i \in U_I, d_I(Q, i) > d_{I,hthresh}$. In the best case, these shots should not be interconnected with the query shot in the graph (i.e. $d_I(Q, i) = \infty$). Alternatively, we can select a predefined number of negative and positive samples from the graph and apply the distance thresholds only if required.

The obvious approach could be to simply train the classifier with $T_{I,P}$ and $T_{I,N}$. However, due to the fact that implicit feedback is not always precise, such an approach could rerank the visual results in non efficient way. On the other hand, visual search is usually capable of retrieving very similar keyframes, which demonstrate an almost zero visual distance. So, in order to minimize such effects and in addition to exploit the results from visual ranking that are of good quality, we include in the positive samples set the shots that are visually closer to the query example by employing an experimentally set threshold. This threshold is actually tuned in such a way in order to maintain a good precision (i.e. by including in the training set the almost identical images to the query example that are retrieved by content based search). Furthermore, we include in the negative samples some of the visual results that were very far from the query image taking into account again a distance threshold. Formally, we construct a new set of positive samples $T_{V,P} = R_{V,P} \subseteq R_V$, where R_V is the set of results of visual search where, $\forall s_i \in R_{V,P}, d(Q, i)_V < d_{V,thresh}$ and $d_{V,thresh}$ is an experimentally set threshold. Subsequently we define a set of negative samples $T_{V,N} = R_{V,N} \subseteq R_V$, where $\forall s_i \in R_{V,N}, d_V(Q, i) > d_{V,hthresh}$. These distance thresholds could either experimentally set to specific values, where always $d_{V,hthresh} > d_{V,thresh}$ or they could be manually adjusted by the user in a manual assisted combination of implicit information and visual data according to the user needs. The final training set would be:

$$T = T_P \cup T_N = (T_{I,P} \cup T_{V,P}) \cup (T_{I,N} \cup T_{V,N}) \quad (5)$$

7.2 Support Vector Machine Classifier

As the training and the reranking of the results is performed in real time during the query phase, we need to select a fast SVM implementation, which can provide quick results. Of course apart from the implementation, the size of the training dataset is an important factor that could keep the speed low. As our purpose is

to rerank results, the initial idea was to employ the ranking SVM of [7], which could be trained by user preferences between two different shots. However, based on how the weighted graph is constructed and especially the quantitative assignment of the actions' weights, it is clear that the shots that are closer to the example shots are better to be considered as positive samples instead of declaration of relative preference.

Hence, we employ the fast performing SVM classifier implementation described in section 3 utilizing as features of the positive and negative samples the concatenation of the EdgeHistogram and ColorLayout descriptor. Assuming that the concatenated visual descriptor is $V = \{v_0, v_1, \dots, v_p\}$, then (2) is transformed into:

$$\forall c \in \{0,1\}^n: \frac{1}{n} w^T \sum_{i=1}^n c_i y_i V_i \geq \frac{1}{n} \sum_{i=1}^n c_i - \xi \quad (6)$$

After having trained the classifier with T , we provide as test set the initial results based on visual descriptors R_V , which is finally ranked according to the degrees of confidence predicted by the model.

8. EXPERIMENTS AND EVALUATION

In order to test and evaluate the proposed approach, we have implemented a video search engine and we performed two different experiments. A detailed description of the interface of the implemented search engine that realizes the proposed framework is illustrated in Figure 7. All the highlighted functionalities are recorded during the user interaction.



Figure 7. Search Engine Interface

Then, an experiment was conducted, which took place in two phases. In this experiment, we made use of the test video set of TRECVID 2008, which includes about 100 hours of Dutch video (news magazine, news, documentaries, etc.) segmented into about 30.000 shots. The audiovisual information was indexed according to the methodology described in section 5.

In the first phase, we employed 24 users, who used the system and searched for 6 different query topics (4 users for each topic). Examples of topics used, were: "Find shots of people sitting at a table" and "Find shots of people where a body of water can be seen". During this part, the users were free to use all the text and content-based retrieval functions of the system and their actions were recorded. Each search session for one topic lasted 15 minutes. By processing the interaction data of this experimental

phase, we have constructed the action and the weighted graph as explained in section 6. In Table 2 we can see the statistics of the constructed weighted graph in terms of nodes and links.

Table 2. Statistics for the weighted graph

| | Nodes | Shots | Keywords | Links |
|----------------|-------|-------|----------|-------|
| Weighted Graph | 1298 | 1229 | 69 | 2659 |



Figure 8. Visual ranking of results for the query shot on the top left corner

Then, in order to measure and evaluate the performance of the suggested algorithm, we have to compare the two different rankings: a) the visual ranking, which is provided by the distances of the visual descriptors and b) the hybrid ranking, which is generated after the application of the proposed algorithm. To compare the aforementioned retrieval functions, we utilize the method suggested in [7]. Considering that RK_H is the hybrid ranking and RK_V the visual, we construct a combined ranking $RK_{H,V}$ that includes the top links of both rankings. More specifically, $RK_{H,V}$ consists of l results, which are actually the k_a top results of RK_H and the k_b of RK_V , where $|k_a - k_b| \leq 1$. This form of presentation leads to a blind statistical test so that the user selections, expressed as clicks on the ranked results, demonstrate the unbiased user preferences. Such method can be considered even more appropriate when applied in query by visual example instead of text web search, as in this case, the results are not so subjective to what the user has in mind. Figures 8 and 9 illustrate a visual and a hybrid ranking respectively, while in Figure 10, the combined ranking is depicted.

In the second experimental phase, 8 users were employed to identify visually similar results for 100 queries by visual example, that were included in the weighted graph. To construct efficiently the training set, the thresholds have been experimentally tuned, while 30 positive and 30 negative examples were extracted by the weighted graph for each query. In tables 3 and 4 we observe the statistic results for all the users. It can be seen that for 68% of the queries the hybrid ranking is preferred by the user, for 6% of queries the two functions seem to perform equally, while for the rest 26% of queries the visual ranking outperforms the hybrid. In Figure 11, we present the corresponding histogram, which shows the frequency of user preference (i.e. selection clicks on hybrid ranking minus clicks on visual ranking for a query) for the involved queries. We constructed the histogram by considering absolute values of the clicks and not normalized (i.e. divided by the total clicks in a query), as the actual number of clicks seems to be of more importance. For instance, in a case that a user clicks and selects only one item more from the one of the rankings, the conclusion should be that this ranking is with higher probability

slightly better than the other. This can be reflected when considering the absolute difference of the clicks (e.g. $8-7=1$ and $1-0=1$), where the value 1 describes the user preference. However if we normalize the metrics according to the total number of selections in a query, we could get misleading results (e.g. 12,5% and 100% respectively for the previous example).



Figure 9. Hybrid ranking of results for the query shot on the top left corner



Figure 10. Combined Ranking of results for the query shot on the top left corner

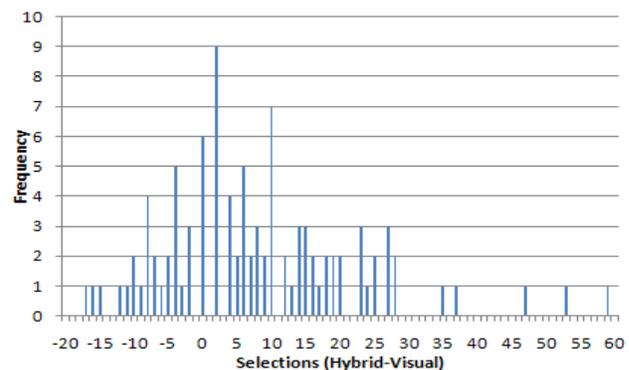


Figure 11. Histogram of the clicks. The horizontal axis stands for the number of clicks (positive for hybrid ranking and negative for visual), while the vertical for the frequency.

In this specific case it seems that when retrieval is performed only by considering the visual descriptors, low distances are estimated also for shots that have similar colors but no semantic resemblance with the query. On the other hand, when the hybrid

ranking is applied, it seems that the implicit user feedback have given a semantic flavor to the ranking as the shots that shared only common color characteristics were ranked lower. Finally, as can be seen from the example in Figures 9 and 10. It is clear that the top 10 results of the hybrid approach are much better.

Table 3. Pairwise comparison of the hybrid retrieval function with the visual one.

| Total Queries | More selections on Hybrid | More selections on Visual | Equal selections |
|---------------|---------------------------|---------------------------|------------------|
| 100 | 68 | 26 | 6 |

Table 4. Clicks on Hybrid and Visual ranking

| Total Clicks | Selections on Hybrid ranking | Selections on Visual ranking |
|--------------|------------------------------|------------------------------|
| 1198 | 1008 | 190 |

9. CONCLUSIONS

In this paper we described an approach of combining visual features with implicit user feedback by employing a SVM classifier. From the experimental results it seems that the proposed method is capable of improving the visual search results in most of the cases. We could conclude that utilizing implicit user feedback to optimize content based retrieval seems to insert a semantic flavor to the query by visual example results, by ranking higher shots that are semantically similar, despite the initial lower visual resemblance. Future work could include more extended evaluation of the method in order to better observe the behavior of this hybrid approach especially in cases that the implicit feedback is of lower quality.

10. ACKNOWLEDGMENTS

This work was supported by the projects PESCaDO (FP7-248594) and PetaMedia (FP7-216444).

11. REFERENCES

- [1] Hopfgartner, F., Vallet, D., Halvey, M, Jose, J. M. 2008. Search trails using user feedback to improve video search. In Proceedings of the ACM Multimedia (Vancouver, Canada, 2008), 339-348.
- [2] Nichols, D. M., 1997. Implicit ratings and filtering. In Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering (Hungary, 1997), 31-36.
- [3] Yang, B., Mei, T., Hua, X-S., Yang, L., Yang, S-Q., Li, M. 2007. Online video recommendation based on multimodal fusion and relevance feedback. In Proceedings of the 6th ACM international conference on Image and video retrieval, (Amsterdam, The Netherlands, 2007), 73 – 80.
- [4] Kelly, D., and Teevan, J. 2003. Implicit Feedback for Inferring User Preference: A Bibliography". SIGIR Forum, 32(2).
- [5] Claypool, M., Le, P., Waseda M. and Brown, D. 2001. Implicit Interest Indicators. In Proceedings of the ACM Intelligent User Interfaces Conference (Santa Fe, New Mexico, USA, 2001). IUI'01, 14-17.
- [6] White, R., Ruthven, I., Jose, J. M., 2002. The Use of Implicit Evidence for Relevance Feedback in Web Retrieval. In Proceedings of the 24th BCS-IRSG European Colloquium on IR Research: Advances in Information Retrieval (Glasgow, UK, 2002), 93-109.
- [7] Joachims, T. 2002. Optimizing Search Engines Using Clickthrough Data. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (Edmonton, Alberta, Canada, 2002). KDD'02.
- [8] Radlinski, F. Joachims, T. 2005. Query chains: learning to rank from implicit feedback. In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (Chicago, Illinois, USA, 2005), 239 – 248.
- [9] Hopfgartner, F. Jose, J. 2007. Evaluating the implicit feedback models for adaptive video retrieval. In Proceedings of the International on Multimedia Information Retrieval (Augsburg, Bavaria, Germany, 2007).
- [10] Vallet, D., Hopfgartner, F., Jose, J. M. 2008. Use of Implicit Graph for Recommending Relevant Videos: A Simulated Evaluation. In Proceedings of the he annual European Conference on Information Retrieval (Glasgow, Scotland, 2008). ECIR'08, 199-210.
- [11] Joachims, T. 2006. Training Linear SVMs in Linear Time. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (Philadelphia, USA, 2006). KDD'06.
- [12] Joachims, T. 2005. A Support Vector Method for Multivariate Performance Measures. In Proceedings of the International Conference on Machine Learning (Bonn, Germany, 2005). ICML'05.
- [13] Hanjalic, A., Lagendijk R.L. and Biemond, J. 1997. A New Method for Key Frame based Video Content Representation. In Image Databases and Multimedia Search, A. Smeulders and R. Jain, Eds., World Scientific. 97-107.
- [14] Zhang, Q., Tolia, G. et. al, 2008. COST292 experimental framework for TRECVID 2008. In Proceedings of TRECVID 2008 Workshop (Gaithersburg, MD, USA, 2008).
- [15] Kinosearch search engine library. <http://www.rectangular.com/kinosearch/>.
- [16] Porter, M.F. 1980. An algorithm for suffix stripping. Program, 14(3), 130-137.
- [17] Robertson S.E. and Jones K.S., 1994. Simple, proven approaches to text retrieval. Technical Report UCAM-CL-TR-356, University of Cambridge, Computer Laboratory.
- [18] Kittler, J., Hatef, M., Duin, R. P. W. and Matas, J. 1998. On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3), 226-239.
- [19] MPEG-7 XM software, http://www.lis.ei.tum.de/research/bv/topics/mmdb/e_mpeg7.html.
- [20] Gutmann, A. 1984. R-trees: a dynamic index structure for spatial searching, In Proceedings of the ACM International Conference on Management and Data (New York, USA, 1984), SIGMOD'84.
- [21] Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. In Numerische Mathematik, 1, 269–271.