Contextual Trace-Based Video Recommendations

Raafat Zarka
Université de Lyon, CNRS
INSA-Lyon, LIRIS,
UMR5205, F-69621,
France
raafat.zarka
@liris.cnrs.fr

Amélie Cordier
Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205, F-69622,
France
amelie.cordier
@liris.cnrs.fr

Elöd Egyed-Zsigmond
Université de Lyon, CNRS
INSA-Lyon, LIRIS,
UMR5205, F-69621,
France
elod.egyed-zsigmond
@liris.cnrs.fr

Alain Mille
Université de Lyon, CNRS
Université Lyon 1, LIRIS,
UMR5205, F-69622,
France
alain.mille@liris.cnrs.fr

ABSTRACT
People like creating their own videos by mixing various contents. Many applications allow us to generate video clips by merging different media like videos clips, photos, text and sounds. Some of these applications enable us to combine online content with our own resources. Given the large amount of content available, the problem is to quickly find content that truly meet our needs. This is when recommender systems come in. In this paper, we propose an approach for contextual video recommendations based on a Trace-Based Reasoning approach.

Categories and Subject Descriptors
H.5.3 [**Information Interfaces and Representation**]: Group and Organization Interfaces – *web-based interaction.*

General Terms
Algorithms, Human Factors.

Keywords
Recommendations, trace-based reasoning, user assistance, temporal pattern mining, storytelling, modeled trace.

1. INTRODUCTION
The Web constitutes a worldwide interactive information system providing a potentially infinite number of resources for a constantly growing number of human activities. Choosing the appropriate information to perform a given task is a complex process. Recommendation systems can save us time and efforts in this context. When performing an activity, the context is of major importance. However, most recommendation systems deal with applications having only two types of entities, users and items and do not put them into a context when providing recommendations. For example, using the temporal context, a travel recommender system should provide a vacation recommendation in the winter that can be very different from the one in the summer. YouTube video recommendations are mainly based on the videos that users have viewed but not on the order in which the user has actually watched these videos. Contextual recommendations are particularly important for domains like: e-commerce, e-learning, e-health, media applications. In this paper, we focus on video recommendations in a semi automatic video clip systems.

WWW 2012 Companion, April 16-20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

We are contributing to a web application called Wanaclip (http://www.wanaclip.eu). The aim of this application is to generate a video clip by selecting and merging different media: photos, videos, music and sounds. Wanaclip allows users to import and annotate their own media in the media base. User content is added to the public content of the application. Users can organize media, adjust the duration, and customize content by adding comments, choose their fonts and colors and adjust the display time. In Wanaclip, users enter keywords, the system searches video sequences (rushes) annotated with these keywords and lets the users drag them into a timeline in order to compose a video clip. Several search cycles can refer to the same result clip. Wanaclip offers publishing and sharing functionalities. It is plugged with social networks. In order to collect interaction traces, we have instrumented Wanaclip application. In Figure 1, the users' trace is displayed on the top of the screen.

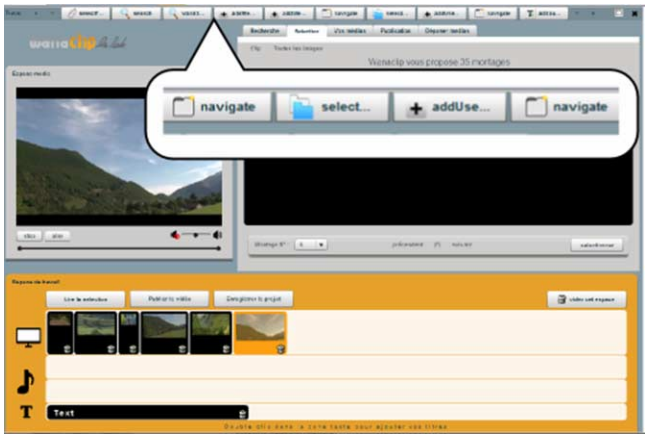

Figure 1. Wanaclip Interface with collected traces

It is sometimes difficult to find the media in the huge database. We show how context-based recommendations can provide efficient help to users. For this, we rely on recorded interaction traces. We claim that interaction traces can be used to record user experiences over the web. Traces can then be reused by a reasoning process for different purposes (replay, mining, etc.). Here, we propose two complementary reasoning mechanisms exploiting traces in order to provide recommendations, namely: Temporal Pattern Mining and Trace-Based Reasoning (TBR) [3]. In order to provide relevant recommendations, we base ourselves on the notion of *episode*. An episode is a temporal pattern composed of an ordered set of events corresponding to a specific task. Given this definition, the goal is then to define similarity measures allowing us to find and compare episodes.

The remaining of this paper is organized as follows. Section 2 reviews some background knowledge, especially about traces. In section 3, we present our contextual recommendation mechanisms. Implementation progress is discussed in Section 4. In section 5 we conclude by comparing our work with other related projects.

2. BACKGROUND

We consider that a trace is a digital representation of a sequence of events occurring during an activity. These events are represented by *obsels* (observed element). Each obsel has, at least, a type and two timestamps (begin and end). Obsels also can have an arbitrary number of attributes and relations with other obsels. Traces are different from log files in the sense that they come with a model. Each trace must have a *trace model* which describes the obsel types that the trace may contain, their attributes, and their relations. A *trace model* can be the same for several traces. A trace associated with its model is called a ***M-Trace.*** Traces are stored and managed in a TBMS (Trace Base Management System) that handles storage, manipulation and exploitation of traces [12]. Transformations are specific operations available on traces. They apply on *M-Traces* and produce *transformed M-Traces*. Transformations can be used for several purposes (filtering, abstraction, reformulation, etc.). Predefined transformation functions are provided by the TBMS.

The collection process consists in observing an application and storing obsels (according to the observations) in the *M-Traces* managed by the TBMS. A transformation process consists in transforming a *M-Trace* into another *M-Trace* within the TBMS. The output of the trace collection process is called a *primary M-trace*, whereas the traces produced by a transformation are called *transformed M-traces*.

These concepts enable us to implement Trace-Based Reasoning (TBR). TBR draws its inspiration from Case-Based Reasoning and reuses past experiences to solve new problems. The main difference is that, in TBR, the main knowledge container is a set of traces [3]. Several researchers have studied the suitability of CBR to cope with temporal domain. In [5], they developed a method for representing temporal CBR cases, but in TBR traces replace cases. TBR is particularly suitable for dynamic Web applications. *M-Traces* can be reused in different ways: task automation, replay, exploration, modification, user assistance, recommendation, stream mining, and visualization. In previous work, we defined an approach to support replay of user's traces to return to a particular state of an application by exploiting traces [14]. Here, we focus on recommendation mechanisms.

3. CONTEXTUAL RECOMMENDATIONS

In this section, we describe our contextual recommendation mechanism. We have implemented a collect process in Wanaclip application. In this process, trace handler captures events, models them as obsels and stores them. The description of the collect process is out of the scope of this paper, hence it is not described. For the recommendation phase, we define two different mechanisms. The first mechanism is Trace-Based Reasoning that applies a retrieval algorithm to get similar episodes and that reuses these episodes to provide recommendations. The second mechanism is temporal pattern mining that generates temporal association rules which are then used to support recommendations. TBR can combine several contextual measures like time, location, humor, etc and it can work for a small amount of traces, whereas temporal association rules are generated offline but need huge amounts of data to work efficiently (see CONCLUSION section for a comparison).

3.1 General recommendation algorithm

We define a general algorithm for recommendation. Depending on the context, the appropriate recommendation mechanism is triggered. The general algorithm is defined as follows:

For each new event leading to a change in a video selection, the algorithm starts by applying a sequence of three different transformations to the current *M-Trace* (filtering, segmentation, switching). These transformations produce a *transformed M-trace* that represents temporal video patterns, ordered sequences of selected videos. Then, the retrieval process can be a trace-based reasoning process or a temporal pattern mining process depending on the contextual measures and the number of *M-Traces*. In both methods, the algorithm ends by visualizing the *N* recommended videos to the user to help him create the clip.

3.2 Trace Transformation

The primary trace issued from the collect process contains all the collected obsels (from all types). Depending on the task at hand, only a subset of these obsels is relevant. In addition, some obsels have to be combined during a transformation process to produce a *transformed M-Trace* ready for its exploitation. Our idea is to use three different transformations to produce temporal video patterns from *M-traces*. As shown in Figure 2, we apply 3 transformations: Filtering, Segmentation and Switching.

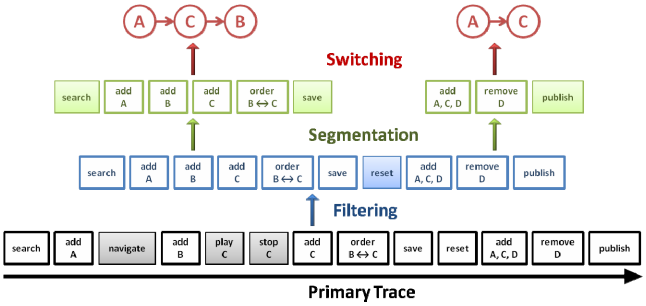

Figure 2. Trace transformation example

First, we apply filtering transformation on the *primary M-trace* to get only the obsels that are related to videos and led to a change in the video selection. This filter is applied on obselTypes to ignore types like: *navigate*, *playVideo*, *changeColor*, etc. It keeps types like: *addVideo*, *changeOrder*, *removeVideo*, etc. After that, a segmentation transformation is applied to the *transformed M-trace*. The segmentation splits the *M-trace* into different segments where each segment corresponds to a specific task. Finally, we apply a switching transformation to each segmented *M-trace*. Switching produces temporal video patterns representing the videos that the user has selected in chronological order. Obsels representing videos are dynamically built from existing obsels through a transformation process. For example, we can build an obsel of type "A" by transforming all the obsels related to the manipulation of the video "A".

3.3 Trace-Based Reasoning

TBR uses traces as a knowledge source. Reusing traces allows solving new problems by finding similar situations in *M-traces* and adapting them to the current situation. In our case, the TBR reuses of past *M-traces* in order to find episodes having similarity score bigger that a predefined similarity threshold. The TBR process comprises three phases:

3.3.1 Elaboration phase:

The goal of the elaboration step is to identify an *episode signature*. In our case, episode signatures represent video sequences that have been extracted using trace transformation.

3.3.2 Search phase:

The search phase consists in finding a set of episodes that match the episode signature. Episodes matching the signature can be used to get the recommended videos. In this step, we define episode similarity measures based on temporal distance and order of videos. To simplify the problem, we ignore video duration. We focus on the chronological order of videos. Therefore, our similarity measure is similar to sequence alignment. Sequence alignment is widely used in different domains like natural language processing, bioinformatics for DNA matching and business analysing purchase series over time [11].

Pair wise sequence alignment methods are used to find the best-matching piecewise (local) or global alignments of two query sequences. A variety of computational algorithms have been applied to the sequence alignment problem using different methods like dynamic programming, efficient, heuristic algorithms or probabilistic methods. The three primary methods of producing pair wise alignments are dot-matrix, dynamic programming, and word methods. Word methods are heuristic and best known for their implementation in the database search tools FASTA and the BLAST family [4]. We will not get into the details of the algorithms, but we will discuss the similarity score.

The search algorithm is based on the similarity measure. For each episode it calculates the similarity measure between the current episode and the stored one. The algorithm returns all the episodes for which similarity scores are bigger than an *episode similarity threshold.* To improve the performance we eliminate some episodes using techniques like self-organizing map [6]. This issue will be treated later in a detailed work. Table 1 shows an example list of episodes and their similarity scores to the current episode (estimated). The current episode is composed of an ordered sequence of 3 videos "ACB". If we considered that episode similarity threshold is 0.5 then the search step returns 5 similar episodes from the episodes in this table.

3.3.3 Utilization phase:

The utilization step reuses the retrieved episodes of the search step to get the top N video recommendations. During this step, we define *query operations* on the retrieved episodes; these queries are actually *weight functions* for ranking videos. A weight function is composed of different measures applied to the videos in the retrieved episodes like: *support*, *distance* and the pre-calculated *episode similarity*. It is possible to include some kind of similarity between video sequences to calculate *distance* like: *edit distance* and *video annotations*. For example: Video *A* can be similar to video *B* if they had similar annotations. Therefore, the distance between *ABC* and *XYZ* is not necessarily 0. Finally, the algorithm visualizes the top *N* recommended video.

Definition 1. Let *NE* be the number of retrieved episodes and *NV* the number of distinct videos in the retrieved episodes. Let □*(P)* be the episode similarity score for an episode *P*, G is the maximum gap or window size. Then, the support and closeness of a video *V* are defined as:

Support (V) = Number of occurrences of *V* / *NE*

Closeness (P, V) = □*(P)* * (1- (distance) / G)

Weight (V) is a combination of different measures to rank the videos. The user can specify the measures and he can change the importance of each measure.

For example, in Table 1: NE = 5, NV = 7, support (F) = 0.6, and closeness (DACBEF, F) = 2*0.5 = 0.375 for G = 4. If we considered Weight(X) = support(X) * $sum_{1 \to NE}$ (closeness (P_i, X):

Weight (F) = 0.6 * (0.6 + 0.8+ 0.6) = 1.2
Weight (G) = 0.4 * (0.75 + 0.7) = 0.58

The previous example was designed using several assumptions, like closeness and weight which will be enhanced later in a complete work. We repeat the calculations for all the retrieved videos and finally we find that F has the highest weight so it has to be recommended first.

Table 1. An Example of TBR Approach (similarity to "ACB")

Episode	Similarity score □
ACDBF	0.8
~~BDBFDA~~	~~0.2~~
ACBFG	1
CABDAE	0.7
DACBEF	0.5
ABCGA	0.7
~~ADGBE~~	~~0.3~~

3.4 Temporal Pattern Mining

In this section, we describe another recommendation method using data mining techniques. We can convert the problem into a temporal pattern mining issue because transformation outputs are sequences. A sequence is an ordered list of items. We consider sequences as time point patterns (duration is not important). Different time point patterns express concepts of order. Substring patterns don't allow gaps whereas regular expression patterns allow gaps and repetitions. In regular expressions, at each time point there is only one item. Episode patterns express concurrency with constraint partial order of items. Partial orders are the most flexible representation but also more complex to mine [9].

We show how we can apply a data mining approach to the *transformed M-traces*. In our case, gaps and ordering are required and we need a simple and fast approach, so sequential pattern mining suits the problem. The next issue is to find all the sequences that have a user-specified minimum support.

Definition 2. Let *N* be the number of video sequences in the *transformed M-traces*, user-specified *window-size*, *min-gap* and *max-gap* are time constraints. The problem of mining interesting sequential patterns is to find all sequences whose *support* is greater than the *minimum support* or its *confidence* is greater than the *minimum confidence*. A *temporal association rule* is defined as a frequent pattern, if it satisfies user defined support and confidence with temporal constraints.

There are different algorithms for sequential pattern mining such as GSP (Apriori-based), PSP (prefix tree), and SPAM [8]. The choice of the algorithm is out of the scope of this paper. We make the assumption that we have selected an algorithm that takes the video sequences generated from *M-trace* transformation, and it generates *temporal association rules*. These rules are generated and stored offline to be used later for the recommendations during the selection process of videos. Each time a user changes the selection of videos, we apply *temporal association rules* to the current video selection in order to get the rules that respect it. Then we retrieve the top *N* videos from these rules and we recommend them to the user.

4. IMPLEMENTATION

We have instrumented Wanaclip application to collect users' traces and displaying them in the top of the screen (See Figure 1). Each time the user performs a new action, a trace handler generates a new obsel and stores it in the TBMS. This obsel will be displayed with other obsels on top of the application. The user can click on any obsel of the trace to see additional information about it (for example, when clicking on "*addVideo*", the user sees information about the added video, such as name, duration, keywords, etc. in addition to obsel id, obsel type, and its time-stamps).

We have developed TStore, a Trace Base Management System that handles the storage, transformation and exploitation of *M-Traces*. TStore is a PHP web service tool that allows anyone to store traces from various applications. TStore allows different programs to concurrently access the same trace and trace model. It provides facilities for web service access, for creating models, for storing obsels, for importing and exporting traces, and for querying. Of course, it supports transformations. We are now working on the implementation part of the recommendations.

5. RELATED WORK

Recommender systems have different approaches like collaborative filtering and content-based filtering. Collaborative filtering builds models according to a user's past behavior as well as similar decisions made by other users [13]. Content-based filtering utilizes item characteristics and a profile of the users' interests [10]. These approaches are often combined. Collaborative filtering requires a large amount of information on a user in order to make accurate recommendations. Content-based filtering needs little information but it is limited to the original item and doesn't take in account user's behavior.

Facebook, MySpace, LinkedIn, and other social networks use collaborative filtering to recommend new friends, groups, and other social connections. When viewing a product on Amazon.com, the store will recommend additional items based on a matrix of what other shoppers bought along with the currently-selected item [7]. Video systems such as YouTube recommend videos that a user might like to watch based on the user's previous ratings and watching habits. It is based on the behavior of other users and the characteristics of the video. Case-Based Reasoning has been used to generate a sequence of songs customized for a community. It reuses the preferences of the audience to customize the selection for the group of listeners [2].

Different recommendation systems do not take the context in consideration. Mostly, they are based on item sequences and users' behavior but not the chronological order of the items. In context-aware recommendation systems [1], they show the importance of contextual information. They introduce three different algorithmic paradigms contextual pre- filtering, post-filtering, and modeling for incorporating contextual information into the recommendation process. Our approach reuses traces that represent user's behavior to provide contextual recommendation.

6. CONCLUSION

In this paper we have described an approach for contextual video recommendation. We have discussed two complementary approaches for recommendation. Each approach has specific properties and achieves good results in a particular context. Table 2 compares the two approaches.

Table 2. Trace-Based Reasoning VS Temporal Pattern Mining

	Temporal Pattern Mining	Trace-Based Reasoning
Mode	Offline	Online
Match	Tolerant matching	Strict matching
Time	Fast recommendation	Depends on traces size
Measures	Support and Confidence	Customized measures
Data size	Needs huge data	Can works for sample data

We are applying our approach to Wanaclip; it allows users of all skills to create their own video clips from different media (rushes). This work is still in progress and we have implemented a collect process in order to gather interaction traces, thus recording experiences of users in a reusable format. We have also implemented the TStore web service tool that controls the storage, transformation and exploiting of traces. Currently, we are working on the implementation of the recommendation. Our approach is based on *M-Traces* to reuse users' experiences on the web to provide contextual video recommendation.

7. REFERENCES

[1] Adomavicius, G. and Tuzhilin, A. 2008. Context-aware recommender systems. *Proceedings of the 2008 ACM conference on Recommender systems RecSys 08*. 16, RecSys (2008), 335.

[2] Baccigalupo, C. and Plaza, E. 2007. A case-based song scheduler for group customised radio. *Lecture Notes in Computer Science*. 4626, (2007), 433-448.

[3] Cordier, A. et al. 2009. Extending Case-Based Reasoning with Traces. *Grand Challenges for reasoning from experiences, Workshop at IJCAI'09* (Jul. 2009).

[4] Goeta, B. 2002. Bioinformatics-Sequence and Genome Analysis. *Briefings in Bioinformatics*.

[5] Jære, M.D. et al. 2002. Representing Temporal Knowledge for Case-Based Prediction. *Advances in CaseBased Reasoning 6th European Conference ECCBR 2002 Aberdeen Scotland UK September 47 2002*. (2002), 174-188.

[6] Kohonen, T. 1990. The self-organizing map. *Proceedings of the IEEE*. 78, 9 (1990), 1464-1480.

[7] Linden, G.D. et al. 2001. collaborative recommendations using item-to-item similarity mappings. *US Patent 6266649*. Google Patents.

[8] Mabroukeh, N.R. and Ezeife, C.I. 2010. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*. 43, 1 (2010), 1-41.

[9] Moerchen, F. 2010. Temporal pattern mining in symbolic time point and time interval data. *Knowledge Creation Diffusion Utilization*. (2010).

[10] Pazzani, M.J. and Billsus, D. 2007. Content-Based Recommendation Systems. *The adaptive web*. 4321, 2 (2007), 325 - 341.

[11] Prinzie, A. and Vandenpoel, D. 2006. Incorporating sequential information into traditional classification models by using an element/position-sensitive SAM. *Decision Support Systems*. 42, 2 (2006), 508-526.

[12] Settouti, L.S. et al. 2009. A Trace-Based Systems Framework□: Models, Languages and Semantics.

[13] Su, X. and Khoshgoftaar, T.M. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*. 2009, Section 3 (2009), 1-20.

[14] Zarka, R. et al. 2011. Rule-based impact propagation for trace replay. *19th International Conference on Case Based Reasoning (ICCBR-2011)* (2011).