Tolkien: An Event Based Storytelling System

Arjun Satish
University of California, Irvine
Irvine, CA
arjun@uci.edu

Ramesh Jain
University of California, Irvine
Irvine, CA
jain@ics.uci.edu

Amarnath Gupta
University of California, San Diego
San Diego, CA
gupta@sdsc.edu

ABSTRACT

Since the dawn of human civilization, stories have been a popular medium of communication, both synchronously and asynchronously. Technically, a story is a time-ordered coherent sequence of events. In many applications, heterogeneous data is collected and organized so appropriate stories could be told. In this paper, we present a system that helps in generation of stories using a large database of events with associated multimodal data, called eventbase. We define storytelling as a two step process in which a storyteller can retrieve appropriate events and associated data, and then those are further filtered using preferences of the viewer. We develop this model using a measure of interestingness based on attributes of selected events and the preferences. Using an event system developed in our laboratory, we demonstrate the story telling process in Tolkien as one that generates multiple queries to select coherent interesting events to form a story.

1. INTRODUCTION

At the time of writing this paper, Wikipedia had the following to say about storytelling "Storytelling has existed as long as humanity has had language. Every culture has its stories and legends, just as every culture has its storytellers and often revered figures with the magic of the tale in their voices and minds." Storytelling has found its use in all aspects of life, be it in organizations [3] or educational institutions like Center of Digital Storytelling or at a personal level. A story is an organization of events which provides a real life experience to a viewer.

A storyteller has lots of event related data in his collection, but must select only a few that are most relevant considering his audience. As suggested in [8], all stories, including movies, are based on a database of events and related data and the story teller must filter this data considering his target audience. Essentially, story telling is a script of query processing operations, whose results are filtered using interests of the audience. Considering the fact that it has become so easy to generate data in different formats, it is increasingly important to use this data to tell effective stories.

Here is a use-case of our system. Consider a personal collection of events. Such a repository would effectively contain a log of events a person has been part of [10]. This could mean records of trips, conferences, professional meetings, private parties etc. Many people especially in the United States, at the end of each calendar year, build a tiny presentation of what happened that year and send it to the their family members, friends and colleagues. Each of this presentation is custom tailored to the recepient. Let us prepare a presentation for a colleague, a family member and a close friend. First the author selects all the events he desires to show (these would be important events of that year). When the family member sees it, more importance (and hence, more details are provided) for events like festival celebration, birth of a child in the family, birthdays, whereas important professional activities are just summarized. For a friend, those events are given more importance which have people whom this friend is familiar with, or contain events which interest him. For instance, if this friend is a music lover, we would show concert related activites more than say, sports related events. For the colleague, the list of professional events are provided in more detail and no personal events are shown. Tolkien requires the author to only specify the intial set of events and the list of people to whom the story shall be sent.

The eventbase used in this paper uses a large event collection. Each event is represented using general event information, experiential data (photos, videos, audio, and text) related to the event, time and location of the event, links to its sub-events, causal links to other events using the model presented by [12] and also discussed in the context of the EventWeb in [6]. It must be mentioned here that efforts related to [4, 5] have presented systems mostly related to organization of such data and simple queries on that data. Our work extends that direction by defining storytelling as a process which subsumes queries as a step towards building a coherent interesting stories that uses multiple queries in a particular sequence and presents data in an interesting way. In this paper, we will first give a formal description of what events and stories and how one can be obtained from the other. Next, we describe the system and provide details on its implementation. Finally, we describe the demonstration.

1.1 Related Work

There have been a lot of approaches to system assisted storytelling. Most of the approaches are interactive storytelling approaches, [7, 4]. These turned out to be largely

VLDB '09, August 24-28, 2009, Lyon, France

inefficient while managing large numbers of events. It was humanly impossible to filter out 1000 events out of 10,000 using a limited UI. [2] depends largely on video content to tell a story. In most of these systems, the control of information organization is mainly in the hands of the system, which makes it hard to tweak the final output. Previous script based approaches put complete control in the hands of the author and it is usually cumbersome to change these scripts [9, 1, 11]. Also, most systems assume some static audience viewers, whose preferences are largely ignored.

2. STORIES AND EVENTS

In this section, we will mathematically formalize the problem and describe our storytelling model.

2.1 The Event Data Model

The world around us can be best described using events occurring at various places at different times. The various objects, their behavior, interactions and the digital data created in these interactions constitue an event. An event e consists of six facets [6] . These are its structural, spatio-temporal, causal, experiential and informational aspects. Events occur within a time interval and at a location. Hence, they must have a spatio-temporal description. Also, events are connected to each other by causal references. The inference of a causal reference between two events is beyond the scope of this project, but we assume that such references exist in the system. The informational aspect specifies other elements of an event like participants, their roles etc, and their interactions. The experiential aspect constitutes references to media generated during the course of the event. These could be photos taken during a party event, logs generated during a client-server transaction etc.

With these properties, we can model a set of events using a directed acyclic graph $G_e(V_e, E_e, L_e)$, where each node constitues an event and edge indicates the relation between them. The set of events is indicated by V_e and the set of edges is denoted by E_e. Since the event relations can have varying semantics, we use a vocabulary L_e to label these edges. Each node can have a set of attributes which provide contextual information. This includes a spatio-temporal description [T, L], where T could be a time interval or a timestamp (indicating a point event). L specifies any arbitrarily shaped polygonal region, in which case it is a series of (Latitude,Longitude) pairs. It could also be a label like "My Home" in which case we translate it to a series of Latitude-Longitude pairs. Each event also has a type I_{type} associated with it. This provides the semantics of the event. This can be a string like "trip", "wedding" or "meeting" which gives the user of the system understanding of the event.

An important observation about events is to be made here. An event can have subevent heirarchy. In other words, an event can be divided into as many levels as possible depending on the requirement of the application on the basis of time, space and type (I_{type}) attributes.

2.2 Problem Definition

A story has the following properties which must be kept in mind while organizing events into stories.

1. Time ordering is an important parameter of the story. As mentioned before, events have directional links between them. The story must preserve this structure. For instance, if event e_1 led to e_2. Then, the story must contain both these events with the edge.

2. A story has multiple episodes. Each episode describes an event of a particular type. An author would want to tell different episodes of his story in different ways. For instance, the description of a wedding event is very different from that of a sports events which is very differnt from that of a natural disaster event.

3. We separate the storytelling process into two steps. The first one requires the author to select a set of events. The second step is to project this story customized to the viewer.

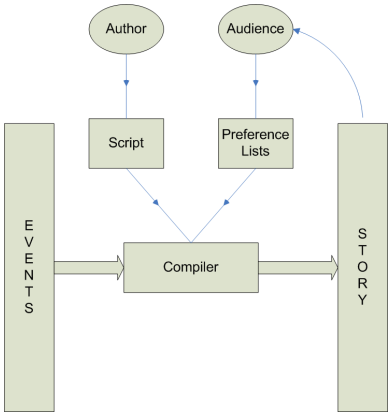

Figure 1: Creating a story from a set of events.

In order to support the aforementioned features of a story in a convenient way, we introduce our specialized script language which allows the user to specify the predicates used to select the event structures. Tolkien provides ways to input details of the viewer, which can result in different projections of the story.

2.3 Stories

As mentioned before stories are a sequence of time ordered coherent events. Figure 1 provides a better understanding of how a story is created from a set of events. Consider an event repository E which has N events. $E = \{e_1, e_2, e_3, ... e_N\}$. These events are nodes (V_e) in the graph previously mentioned. We assume a set view for clarity purposes. An author A writes a story script. We represent the audience with another set $A_U = \{A_1, A_2, A_3 ... A_{|U|})\}$. Each of this audience member has a preference list. The semantics of entries this list may vary from entry to entry. Effectively, each preference is a triple where the subject of the predicate indicates the person, the predicate indicates the semantics of the preference towards the object. We convert these triples to a score list. In this list, we have the object reference which is a string S and a related score w, $w \epsilon [-1, 1]$. Hence, the preference list P is a set of object references and their scores, $P = \{ \{S_1, w_1\}, \{S_2, w_2\}, \{S_3, w_3\} ... \{S_{|p|}, w_{|p|}\} \}$. Each preference list is associated with an audience member. For example, a dislike of a particular item would result in a score of -1. Given a script written by the author A, the compiler selects a set of events and arranges them into a story S_T for a viewer from the set A_U. Any story has to follow the following two properties:

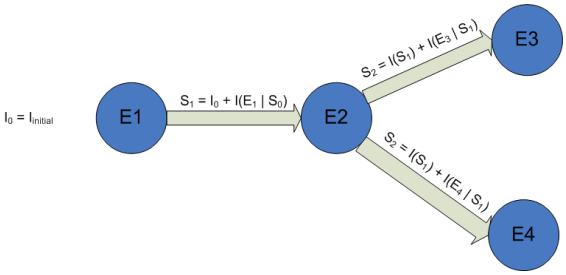

Figure 2: Flow of interestingness in the story.

1. Each episode is time ordered. That is, adjacent events are related by time. This property of the story is called diachronicity.

2. A preceding event sets the context for the next event. Thus, events in an episode are linked together to form a chain. These links can also have labels to indicate relationships like diachronicity or causality.

We segregate the storytelling process into two steps. The first in which, an author *selects* a set of events which he wants to be part of his story. The second is to *project* them in a manner which appropriate for the viewer. Note that these select and project operators are significantly different from those in relational algebra. We define the selection operator as follows:

$$S_c = \sigma_E(G_e, \Re_E) \tag{1}$$

Here, S_c is the list of events which the author wants to communicate to the result. σ_E is the selection operator. $\Re_E$ is the set of predicates based on which the selection takes place.

To quantify a good projection of the story, we introduce the parameter, event interestingness, which specifies the degree of similarity between an event and preferences of a person. Event Interestingness is a viewer dependent parameter which is computed from his scores in the preference list. We can formalize this parameter more by paraphrasing the above definition as follows: Given a particular person and his preference, Event Interestingness is the weighted sum of scores of those preferences which are also the attributes of an event.

$$I_e(e_i) = \sum_{S_i \epsilon e_i} (w_i) \tag{2}$$

$I_e(e_i)$ is normalized by the number of attributes to keep its value within bounds. Story projection is a context dependent process. By this we mean, the decision to project an event or not depends not only on the event itself, but also on what will be projected later, and what has already been projected. Once a selection is made, we iterate through the events and project only those events, which have higher interestingness, maintaining the structure of the entire story as specified by the script. If S_i is the story containing the first i events. S_T is the complete story with T events. We introduce Story Interestingness, which is the story counterpart of event interestingness. $I_e(e_i \mid S_{i-1})$ is the contribution to the story interestingness by the event e_i given that S_{i-1} has been projected to the user. Mathematically, we write:

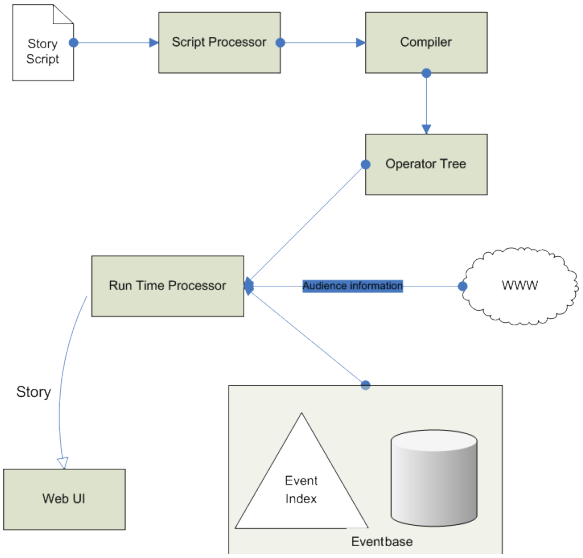

Figure 3: Tolkien's System Architecture.

$$I(S_i) = I(S_{i-1}) + I_e(e_i|S_{i-1}) \tag{3}$$

where,

$$I(S_0) = I_{initial} \tag{4}$$

and

$I_e(e_i \mid S_{i-1}) = 0$, when $e_i \epsilon S_{i-1}$, and $I_e(e_i \mid S_{i-1}) = I_e(e_i)$ otherwise.

The goal of a good storytelling system is to choose the events in such a way that it keeps the story interestingness as high as possible. This is not always possible as the event interestingness might be a negative value.

3. SYSTEM ARCHITECTURE

The system architecture is shown in figure 3. The input is fed in terms of a script written by the author. We will explain the semantics of such a script shortly. The script contains a specification of the story as well as instructions on how to modify it depending of the audience.

The script is first analyzed by a *Script Processor* to check for lexical errors. This provides the *Compiler* an error-free script with which it creates an operator tree. This operator tree would be stored in a cache. Once the preference list of the audience is known to the *Run Time Processor*, this tree is converted to a series of *Index* lookups and queries to the *Eventbase*. This database contains detailed event descriptions with the relevant media items. The results of these queries are then collected and sent to the Web-UI.

3.1 Story Script Semantics

In this section, we describe another important aspect of our system, which is the story script. The figure 4 shows an example story script. The intention to be able to describe a football event. The first ***find*** statement describes the source of our events, which is in a XML document *eRoot.xml*. We also select from this document, only those events which are within a given time interval. Other event predicates can also mentioned here. The variable *$events* is used to reference

```
FIND $events FROM /home/arjun/test/eRoot.xml WHERE datetime >= '1 July 2006'
                                              AND datetime <= '20 July 2006';
FOR ($e IN $events) {
   IF ($e.instanceOf == 'FootballMatch') {
      FIND $badFouls WHERE type = 'Injury'
                           FOLLOWED BY type = 'PlayerDisqualification';
      FIND $stealGoals WHERE (type = 'Steal' AND Object = 'Ball')
                              FOLLOWED BY role = 'Passes'
                              FOLLOWED BY type = 'Goal';
      FIND $intermission WHERE type = 'HalfTime';
      FIND $argument WHERE activity = 'Argument'

      TELL $badFouls;
      TELL $stealGoals;
      TELL $intermission;
      TELL $argument;
   }
}
```

Figure 4: A story script describing a football game. The preferences for this game can be as simple as the team the viewer is supporting.

these events later in the script. The next ***for*** loop iterates through the root events which were found in $events. The variable *$e* is used as a cursor to point to the current event in this iterator. The next ***if*** statement checks if the type of this event is an FootballMatch. The ***find*** and ***followed by*** clause permits the user to search for event structures rather than individual events. Finally, once we have the required events, ***tell*** statements publish the required data in time order. Authors can also format the data by using HTML (not shown in the example). Currently, we rely on translating these statements to XQuery/XPath statements to retrieve the required data and less on the index lookups. In the future work of this project, we aim to construct stories more efficiently using more sophisticated algorithms.

4. DEMONSTRATION

We will demonstrate Tolkien by showing stories created from a personal collection of events. These events were created by looking at various personal media, including Digital Photographs, Email conversations, Calendar entries, IM conversations and events from social networking websites like Facebook, Twitter. Tolkien is a standalone application which will allow a user to create a story script and visualize the selection events as well as the projection events. Tolkien also offers a variety of templates where the user can start editing from (instead of starting from scratch always). Later, he can enter the list of viewers who would want to see this story, with their preferences. The preferences can be entered in the form of Facebook accounts (which tell us the list of people whom the viewer would be familiar with), Amazon wishlist (which tells us what kind of objects he is interested in) and Google calendar (to tell us what kind of events he might be interested in). In figure 5 the script demonstrates the description of a personal event like sight seeing in a trip. We also present the model, architecture and algorithms involved in the Tolkien system.

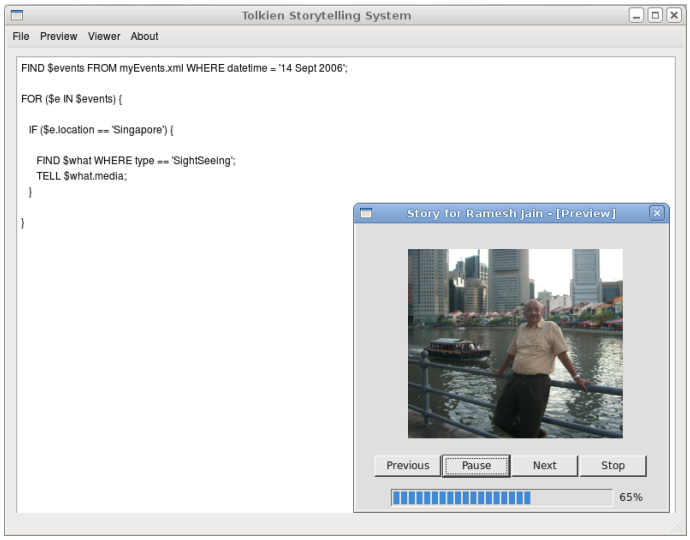

Figure 5: Screenshot of the script editing environment for a simple personal event. The media contained in the event is displayed as a presentation. Note the picture shows Ramesh himself when the story is being prepared for him.

5. REFERENCES

[1] E. André, K. Concepcion, I. Mani, and L. Van Guilder. Autobriefer: A System for Authoring Narrated Briefings. *Multimodal Intelligent Information Presentation*, page 143, 2005.

[2] S. Bocconi. Vox Populi: generating video documentaries from semantically annotated media repositories. *Eindhoven: Technische Universiteit Eindhoven*, 2006.

[3] J. Brown. *Storytelling in organizations: Why storytelling is transforming 21st century organizations and management*. Butterworth-Heinemann, 2004.

[4] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. MyLifeBits: fulfilling the Memex vision. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 235–238. ACM New York, NY, USA, 2002.

[5] R. Jain. Media Vision: eChronicles. *IEEE Multimedia*, 2003.

[6] R. Jain. Eventweb: Developing a human-centered computing system. *IEEE Computer Society*, 41(2):42, 2008.

[7] B. Landry and M. Guzdial. iTell: supporting retrospective storytelling with digital photos. In *Proceedings of the 6th conference on Designing Interactive systems*, pages 160–168. ACM New York, NY, USA, 2006.

[8] L. Manovich. *The language of new media*. The MIT press, 2001.

[9] C. Rocchi and M. Zancanaro. Rhetorical patterns for adaptive video documentaries. *Lecture notes in computer science*, pages 324–327, 2004.

[10] A. Scherp, S. Agaram, and R. Jain. Event-centric media management. In *Proceedings of SPIE*, volume 6820, page 68200C, 2008.

[11] K. Sumi and K. Tanaka. Automatic Conversion from E-content into Virtual Storytelling. *ICVS-VirtStory*, pages 260–269, 2005.

[12] U. Westermann and R. Jain. Events in multimedia electronic chronicles (e-chronicles). *International Journal on Semantic Web & Information Systems*, 2(2):1–23, 2006.