Exploiting Semantics for Personalized Story Creation

Mark D. Wood
Eastman Kodak Company
mark.d.wood@kodak.com

Abstract

The task of creating albums or multimedia output from consumer content is becoming increasingly difficult as the amount of content grows. This work presents a system for using semantic information to automate the process of selecting and combining digital assets into summary presentations or storylines, as well as determining triggers for when to generate such content. The system obtains semantic information from a variety of sources, including the capture metadata, image and video understanding algorithms, user profiles and third party ontologies; all such semantic information is stored in a triple store. Prolog-based rules leverage the triple store to provide a knowledgebase for determining when to create particular types of output and how to select assets for such output. This knowledgebase greatly simplifies the task of creating consumer-grade multimedia content.

1. Introduction

The widespread use of digital capture devices, including digital still cameras, video cameras, and camera cell phones, has resulted in a burgeoning growth in personal digital content. People capture this content because they wish to preserve and relive or share some moment or event, but the huge volume of digital assets has resulted in many users feeling overwhelmed and their captured content lies underutilized, perhaps buried in some folder but seldom, if ever, seen. While the pictures and video have meaning for the person who captured them, that meaning is seldom, if ever, understood by the digital systems people use to store and access their content.

The Semantic System Demonstration Framework, or SSDF, leverages the semantics associated with digital assets to help consumers organize and enjoy their digital assets. The key principle is that the system attempts to understand the meaning or significance of each digital asset, and to leverage that understanding to automate organization and retrieval. The emphasis in this work is leveraging semantics to automate the process of constructing albums or multimedia presentations, but the same technology can also enable semantically rich interfaces for searching and browsing through a personal or shared multimedia collection.

The SSDF harnesses semantic information associated with the digital asset, information about the user, and third-party information to automate the process of creating albums or multimedia presentations. The system first determines an appropriate story/event theme and product representation, and then selects appropriate assets to create an actual album or multimedia presentation. While the created output may not be exactly what the user would have created, the system has gotten the user past the initial hurdle. The user may decide the 80% solution is good enough, or the user may choose to spend a little time tweaking the output to further reflect the user's preferences.

Others in related work have attempted to automate some aspects of content management by using semantic information. Creating albums requires first selecting the appropriate assets, and then determining how to arrange them spatially and temporally. In [1], the authors present a system that uses semantic information to inform the process of laying out content, by grouping assets based upon metadata associated with the asset such as capture time or orientation of the main subject. The Story Picturing Engine [2] uses semantic information to help illustrate text-based stories. Hyperdoc [3] uses templates to drive the selection of assets to tell a particular story. The work described here is differentiated in that it uses semantic information to determine both when a particular type of story should be created, as well as what assets should be selected to go into that story. Furthermore, this system is designed to work with thousands of individual assets, selecting only those appropriate for a particular story, using rule-based logic.

978-0-7695-3279-0/08 $25.00 © 2008 IEEE
DOI 10.1109/ICSC.2008.10

Section 2 briefly outlines the structure of the SSDF and describes the types of semantic information used by the system. Section 3 describes the rule-based inferencing used in the SSDF; Sections 4 and 5 describe how the rule system is used to select stories and assets to populate those stories. Section 6 discusses the lessons learned from building this system and Section 7 concludes the paper.

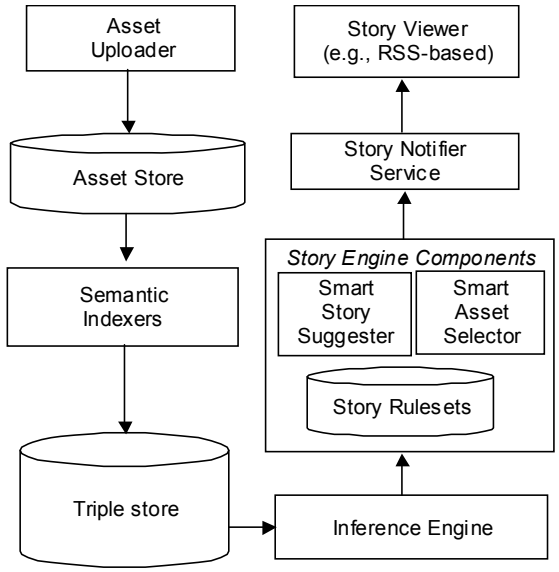

Figure 1 System Architecture

2. System Architecture

Figure 1 illustrates the high-level system components comprising the SSDF. The SSDF is architected as a client-server system. The overall framework contains other system components not relevant to this discussion; such components are not shown here.

Picture and video assets are uploaded into the system into the asset store. The system runs semantic indexers over the assets as they are added, with the extracted and derived metadata stored into the triple store. The SSDF uses the AllegroGraph triple store [4], a commercial product sold by Franz Inc. The Smart Story Initiator evaluates rules, using the inferencing engine, to determine if one or more products should be created. If so, the Smart Story Initiator calls the Smart Asset Selector to obtain the appropriate set of assets, and then creates the XML-based product representation. The current system renders the XML-based description to SVG or HTML. Finally, the Story Notifier component informs the user when new products are available for viewing. The current system uses an RSS-based notification mechanism, with the created story viewed using a web browser. The arrows in Figure 1 illustrate the general flow of data, but note that the Story Suggester is run as an independent agent and may suggest stories both based upon recent uploads and other system events.

2.1. User and Asset Metadata

Users may upload both still and video digital asset types. The server stores such assets in a file system-based asset store, and then invokes a variety of semantic indexers that run against the freshly introduced assets. These semantic indexers include algorithms for face detection and people recognition [5], scene classification [6], image appeal or value [7], event clustering and classification [8], and video analysis. In addition, the indexers extract metadata normally recorded as part of the digital asset at the time of capture. For example, images stored using the Exif file format typically include the capture date and time and various camera settings. Captured metadata may also include GPS information or even an indication of the photographer's mood [9].

An important part of the semantic indexing process is the grouping and classification of assets into events. The temporal event clustering algorithm groups a series of assets into super events, events, and subevents, based upon an analysis of the capture timestamps and image similarity. A set of pictures taken over several days during a vacation might be grouped together into a super event, with the different picture-taking events occurring during the vacation, such as visiting a historical site, forming an event. An event might be further broken down into subevents if the scene content changed significantly from one picture to another, as would occur, for example, when taking pictures both inside and outside a monument. For each super event, event, and subevent detected by the temporal event recognizer, the system creates a separate node representing the event object, with the node pointing to the appropriate constituent elements. For example, the node for the super event would point to the nodes for the events making up the super event, with those nodes in turn pointing to the appropriate subevents, which finally point to the various assets. Each of the various event nodes in turn might have other metadata associated with it. For example, the system uses probabilistic techniques to compute an event type classification for super events, using categories such as "outdoor sports" or "family moment."

Many of the asset selection rules depend upon knowledge of who is portrayed in a given asset. This

information may be obtained automatically, using people recognition algorithms, or the user may explicitly label pictures as containing specific individuals. Windows Vista provides an efficient mechanism whereby the user can rapidly label large sets of pictures; the system understands the metadata elements introduced by Vista and translates them into the appropriate set of triple store statements. For each metadata attribute associated with a particular asset, whether the metadata was recorded by the capture device, derived algorithmically, or manually specified by the user, a corresponding statement is generated and added to the triple store. Figure 2 illustrates some metadata that might be associated with a particular image asset, 986_102_990.jpg, using the N-Triples representation [10]. The first line states that the asset portrays a person identified by the specified URI; the next line states that the asset is of type JPEG; the following line states that one face was detected in the asset, and so on.

In addition to containing metadata generated by the various semantic indexers, the triple store also contains user profile information and product information. For example, for a given system user, the system could record information about a user's family as well as a user's interests and hobbies. The user may enter additional metadata for each person known to the system. By entering parent/child and spousal relationships, the system will be able to infer additional familial relationships such as aunt or grandparent. Other information such as birthdays, anniversaries, addresses, and personal interests or hobbies may also be entered into the system.

Additional types of metadata not supported by the user interface may be directly added to the system by encoding the information in an RDF-based file and loading the file into the triple store. The RDF data model allows a potentially arbitrary set of facts to be associated with a given person or object.

2.2. Other Metadata

A significant part of the reasoning carried out by the system pertains to output product type selection. The information the system needs to reason about product types comes from the system output product catalog. Each output type in the system is identified by a guid, which is trivially made into a URI. At a minimum, the triple store must contain the list of available output types. Additional metadata may be associated with a given output type. For example, a certain output type may be associated with a particular holiday or hobby. Other metadata might include the output modality or stylistic theme.

Additional third-party sources of information such as the U.S. Government's Geographic Name Information System (GNIS) data may also be added to the triple store. The GNIS database provides information mapping commonly used placenames to latitude and longitude coordinates as well as feature classes (e.g., park, school, church, etc.). Such information may be readily converted into an RDF format and then loaded into the triple store. Other sources of knowledge include calendars of commonly observed holidays, third-party ontologies, and other types of common sense knowledge.

<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/SceneContent/Contains/Person> <urn:83EB2D0B-26B4-8CB0-3DB0-F1A0AB3B55B7> .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/SceneContent/MIMEType> "image/jpeg" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/SceneContent/Face/Faces> "1" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/CaptureConditions/ShutterSpeed> "6" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/SceneContent/ImageCaptureDateTime/ImageCaptureDateTime> "2005-10-02 10:59:49" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/ImageContainer/Orientation> "1" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/SceneContent/ImageValueIndex/Technical> "4" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/ImageContainer/Height> "1728" .
<http://rocrl998zb3t:8080/SSDFAssetServer/2/20070228/2/986_102_9990.JPG>
<http://pstc.rl.kodak.com/properties/ImageContainer/Width> "2304" .

Figure 2 Sample Metadata

2.3. Rule-Based System Components

The primary system components exploiting the semantic understanding infrastructure are the Story Suggester and the Smart Asset Selector. The Story Suggester invokes Prolog rules to determine what story to produce, if any. The Smart Asset Selector references an XML file containing a set of separate rulesets, where each ruleset is a set of rules defined typically in Prolog, although SPARQL-based rules are also supported. The Smart Asset Selector, in response to a query to produce the required assets for a specified story, evaluates the associated rules and then collates the final results as needed. Together these components provide the basis for the story generation functionality.

3. Rule-based Inferencing

The story generation component of the SSDF functions as a knowledgebase system, with its behavior driven by a set of extensible rules specifying both the circumstances under which stories should be suggested and how the assets should be selected to create those stories. As previously noted, the core of the rule-based reasoning is written in Prolog. Our version of Prolog, Allegro Prolog, uses S-expression syntax as opposed to the usual Clocksin & Mellish syntax. This version of Prolog was chosen because it is tightly integrated into the AllegroGraph triple store. The rules ultimately query the triple store to access the various types of system metadata.

AllegroGraph 2.5 provides two basic functors for querying the triple store from Prolog. The q functor takes three arguments, each of which may be bound or unbound, with the arguments corresponding to the subject, predicate, and object of a statement. The q functor returns the set of variable bindings for the unbound variables such that the corresponding subject, predicate, and object corresponds to a statement in the triple store. A variation of the q functor, the qs functor, returns the set of variable bindings for the unbound variables such that the corresponding subject, predicate, and object correspond to a statement in the triple store, or where such a statement could be inferred using certain RDFS or OWL-based inferencing rules.

The Prolog rulebase was divided into a set of different Prolog definitions. For notational simplicity and to abstract metadata implementation details, most of the commonly accessed pieces of metadata were defined as simple Prolog rules where the body of the rule is a reference to the appropriate q operator. For example, the rule videoHasLength is defined by the following clause:

```
(<-- (videoHasLength ?video ?len)
(q ?video !ek:ImageContainer/Movie/Duration
    ?len))
```

In this S-expression syntax, the first parenthesized subexpression denotes the head of the Prolog rule; the subsequent parenthesized subexpressions denote the goals making up the body of the rule. This clause would be written in traditional Prolog syntax as the following:

```
videoHasLength(?video ?len) :-
q(?video !ek:ImageContainer/Movie/Duration ?len)
```

The term !ek:ImageContainer/Movie/Duration is an abbreviation supported by AllegroGraph; it is expanded to the appropriate full URI, substituting for the prefix ek: the appropriate namespace substitution.

In addition to simple rules serving as metadata accessors, the Prolog rule base contains an extensive set of rules describing personal relationships. These rules reference the metadata associated with people, including gender and the three types of interpersonal relationships currently supported by the system—parent, spouse, and friend—to define concepts such as mother, grandmother, aunt, etc. The system represents the marriage of two individuals by creating a new node to represent the union; metadata associated with the marriage such as the anniversary and possible divorce date are associated with that marriage node. The rule

```
(<-- (spouse ?x ?y)
(q ?x !ek:Person/HasMarriage ?m)
(q ?y !ek:Person/HasMarriage ?m)
(not (personEquals ?x ?y))
(not (q ?m !ek:Person/Marriage/DivorcedOn ?d)))
```

defines two people as being currently married if they both link to the same marriage node and that marriage node does not contain a divorce attribute.

Sections 2.1 and 2.2 described the types of metadata used by the system. Some of this metadata is based upon probabilistic algorithms. Additional rules may be specified referencing such metadata to mitigate the uncertainty. For example, the event classifier described in Section 2.1 may produce the classification "family moment" using Bayesian techniques based on various picture attributes. The validity of this classification may be further tested by a rule that explicitly tests to see if any of the people portrayed in the picture are in fact close relatives of the user.

4. The Story Suggester

The knowledgebase for selecting when to create stories uses two types of rules: one type based upon the date and another based upon event nodes. The story triggers need to determine two dimensions when

deciding that a particular story should be created: they need to pick the appropriate product from the catalog and the appropriate ruleset for selecting the assets to go into that product. An arbitrary number of rules can be added to the system to cover the various likely (and not so likely) story generation opportunities.

4.1. Calendar-Driven Stories

To illustrate calendar-driven stories, suppose user Alex is married to Ann and they have young children. Mother's Day is coming up; the system anticipates this and automatically creates a Mother's Day Surprise Story one week in advance for Alex to preview. The story is a multimedia creation, which includes pictures of Ann and her family over the past few years.

The date-driven rules for triggering story generation consider what stories might be appropriate given a particular date. Typical dates of interest include birthdays, anniversaries, and holidays.

In the current implementation of calendar-driven stories, the trigger rules determine several parameters, including the user for whom the story should be produced, a potential recipient of the product, the output type, and the ruleset to use in picking assets.

Rule R-1 illustrates one possible rule to determine whether a Mother's Day album should be created for a user.

R-1. Given target date *Date*, suggest to user *User* story type "Mother's Day Album" and product "Mother's Day Multimedia Album" intended for recipient *Recipient* if:
R-1.1. Target date *Date* is a known recurring holiday *Holiday*
R-1.2. *Holiday* is Mother's Day
R-1.3. The system user *User* is the spouse of the recipient *Recipient*
R-1.4. The recipient *Recipient* is a mother

In the Prolog equivalent for this rule, the user, recipient, and product are unbound variables; the Prolog inferencing engine seeks to satisfy this goal with a particular user, recipient, and product variable binding. In the example above, the user would be Alex with the recipient being his wife and the product being the Mother's Day multimedia album.

4.2. Event-Driven Stories

In addition to choosing stories based upon an upcoming date, the system may suggest a story based upon a recently occurring event. For example, suppose a user has returned from vacation and has just uploaded their pictures into the system; the system might automatically put those pictures and memories into an album or multimedia presentation for the user to enjoy and share with others.

In general, the event-driven story triggers consider a particular event and determine whether or not an appropriate story could be generated from that event. This trigger uses ontological reasoning to determine if the event classification for the event corresponds to a user's hobby or interest for which the system offers an appropriately themed product. To execute this rule, the triple store must contain:

- An interest/activity ontology
- A product catalog ontology, with the ability to specify that specific products go with specific interest/activities
- Statements associating people with interests from the interest/activity ontology

The interest/activity ontology defines an extensible list of possible activities, interests, and hobbies. For example, a (small) subset of the ontology might look like:

(1) Musical Activities
  1.a) Singing
  1.b) Playing a musical instrument
(2) Sporting Activities
  2.a) Outdoor sports
    2.a.1) Baseball
    2.a.2) Soccer
    2.a.3) Football
(3) Social Gatherings
  3.a) Parties
    3.a.1) Birthday parties
  3.b) Solemn Occasions

A full ontology would obviously contain far more information. While the system currently uses a custom-designed ontology, one or more appropriate third-party ontologies may also be used.

As previously noted, the system uses probabilistic techniques to categorize super events. The triple store maps the general event category "outdoor sports" to the appropriate outdoor sports class in the ontology. As technologies for event classification advance, the classifiers will be able to more narrowly map events to classes within the interests and hobbies ontology.

The product catalog likewise contains a set of possible product types along with the activities or interests with which the product might be associated.

Using this data, the system includes the following story generation trigger:

R-2. For a set of assets comprising a given event *Event*, suggest product *Product* for user *User* if:
R-2.1. *User* owns event *Event*
R-2.2. *Event* has classification *EventType*
R-2.3. *Event* contains picture(s) featuring *Person*
R-2.4. *User* is a parent of *Person*

R-2.5. *Person* likes activity *ActivityType*
R-2.6. *Product* goes with activity *ActivityType*
R-2.7. *Activity* is a subclass of *EventType*

Given the above, the system can suggest a themed story based upon an upload of a set of digital media assets. For example, suppose a father Alex uploads a set of pictures from his daughter Jane's recent Little League game, and the system knows the following information:

- The event is classified as an outdoor sport event.
- Baseball is a type of outdoor sport.
- The event contains Jane, a child of the user.
- Jane likes baseball.
- The baseball album product is associated with the activity baseball.

Based on this information, the rule R-2 would be satisfied, causing the system to create a baseball-themed album for Alex featuring Jane's recent sporting event. Clearly this rule is overly simplistic; a production-quality rule might consider whether there might be other sports Jane is interested in, and whether the time of the event was a time of the year when baseball was commonly played, etc.

5. Smart Asset Selectors

Once the system has identified a candidate story to create for the user, the system must then select the appropriate set of assets to populate that story. The SSDF provides a mechanism to define an arbitrary number of rulesets, where each ruleset consists of a set of rules resulting in potentially the selection of one or more assets. Rulesets are named and stored in an XML-based file containing the Prolog or SPARQL definition as well as certain other modifiers controlling how the rule results are interpreted; a Java-based mechanism is used to control the execution of the appropriate ruleset.

Referring back to the Mother's Day album of Section 4.1, the following rules might be used to actually select the appropriate assets to create the story:

- Select the two best pictures of the mother alone from any year.
- Select the three best pictures of the mother with all children from the past year.
- Select the single best picture of the mother with each child individually from any year.
- Select the two best pictures of the mother with her mother from any year.
- Select the three best pictures of the mother with family (children and spouse) from past year, from distinct subevents.
- Select the two best short video clips, less than 60 seconds, where the video clip is from a set of pictures classified as a "Family Moment."

Figure 3 illustrates the Mother's Day asset selection ruleset, showing specifically the Prolog rules corresponding to the first and last rules identified above. Before evaluating rules, the system substitutes the appropriate values for #OWNER, #PERSON, and #EVENT. Where applicable, the string #OWNER is substituted with the user's URI; #PERSON is substituted with the URI of the person suggested as the subject by the story trigger, and #EVENT is substituted with the URI denoting the event identified by the story trigger. In the Mother's Day example, the #OWNER would be replaced with the URI for Alex, and #PERSON would be replaced with the URI for Ann.

Each Prolog rule may be satisfied by one or more variable bindings. The first parenthesized expression in the definition portion of the rule declares the unbound variables. The asset selection system expects each rule to list three unbound variables, resulting in each rule returning a set of triples. The use of triples here is unrelated to the concept of subject, predicate, and object within the triple store; the system happens to use three values in collating the results of the various rules making up a rule set. The first variable is always bound to the asset. The second variable, given as ?date in both rules, provides a value against which to sort, in the case where the assets returned by the different rules are to be returned in a sorted order. In the example ruleset, the attribute sortResults is false, so this sorting step is not carried out. The third variable, ?ivi in the first rule, is used for the purposes of sorting the values returned by a single rule. For example, the first rule matches all of the pictures containing only the mother. As this could potentially be a large number of pictures, the max attribute on the rule element limits the number to no more than two assets. To pick which of the assets should be returned from all of the assets that satisfy the rule, the assets are sorted by the third variable, ?ivi, with the top two assets actually returned. The last two clauses cause ?ivi and ?date to be bound to the image value index and capture date for the asset, enabling the sorting to be carried out.

In addition to the sorting and selection done after the rules are evaluated, the Prolog rules themselves may also incorporate the logic necessary to select the appropriate number of assets. In some cases the selection criteria are better expressed in a more procedural language. The system includes a variety of predefined relations to pick the best from among a set of assets according to some criteria, where the relation

is implemented using Prolog and/or Lisp as appropriate.

6. Discussion

The process of selecting and populating multimedia story products lends itself to being expressed as a knowledgebase. Simple rules can be readily defined for when to create stories and how to choose assets to go into those stories. These definitions can be readily modified and extended, even while the system is running.

A number of issues surface in constructing a rule-based system for story generation. The parameters fed into the reasoner often were computed using probabilistic means and have an associated degree of uncertainty. This uncertainty is handled by applying thresholds to make binary decisions; additional constraints or checks could be expressed as rules. Future research may wish to consider whether the use of other techniques for handling uncertainty would be better applied.

In a similar manner, the current implementation requires the use of rigid ontologies in describing hobbies and interests. Few such ontologies exist, and again, the use of a rigid ontology does not always correspond to real world situations.

Understanding who and what an image portrays is critical to story generation, yet the technologies for people and object recognition are still in their infancy.

```
<ruleset name="MothersDayAlbum"
  friendlyName="Mothers' Day Album" sortResults="false">
      <rule max="2" type="prolog">
            <desc>Pictures only containing #PERSON</desc>
            <definition>
              (?pic ?date ?ivi)
              (containsOnlyIdentifiedPerson ?pic #PERSON)
              (belongsTo ?pic #OWNER)
              (hasIVI ?pic ?ivi)
              (capturedOn ?pic ?date)
            </definition>
      </rule>
      ...
      <rule max="2" type="prolog">
            <desc>video snippet</desc>
            <definition>
              (?movie ?date ?date)
              (eventIsOfEventType ?e !!"FamilyMoment")
              (eventContains ?e ?movie)
              (assetIsVideo ?movie)
              (videoHasLength ?movie ?len)
              (literalStringIsLT ?len "00:00:60" )
              (capturedOn ?movie ?date)
            </definition>
      </rule>
</ruleset>
```

Figure 3 Sample Ruleset

This problem can be alleviated by allowing manual tagging, but such tagging may be tedious for the consumer and runs counter to the goal of automatic story creation. As these technologies continue to mature, this will be less of an issue.

The current implementation separates the process of selecting a story and choosing the assets for a particular story. However, the availability of assets may influence the choice of story. In the future, the system should support rules for story triggering that consider whether sufficient assets are available to adequately populate the story.

This knowledge system could have been based on a variety of different database types. Besides a triple store, the other obvious choice would have been a relational database. While relational databases have rightfully earned their place as a powerful tool for storing structured data, they are less suited for representing semantic networks [11]. With the relational data model, introducing new types of data typically requires changing the database schema: either new tables need to be added, or the type of existing tables must be changed. In the RDF data model, new data is simply added to the system. However, this flexibility comes at a cost: a triple store requires an explicit triple for each piece of metadata, resulting in potentially redundant storage of the subject (the asset id) and predicate (the metadata type).

Although the system uses the RDF data model for reasoning and can make use of RDFS and OWL-based reasoners, the bulk of the query logic is expressed in Prolog. Story triggers are exclusively written in Prolog. Rules used by the Smart Asset Selector may be written in either Prolog or SPARQL. The SPARQL query language for RDF provides a straightforward means for querying a triple store for resources that satisfy a particular pattern. However, SPARQL does not support inferencing, and so a simple query requesting pictures of a person and the person's mother could not be directly expressed using SPARQL; the system would have had to infer the URI for the mother before making the query. Likewise, RDFS and OWL-based reasoning is of limited utility in the current system. OWL-based reasoning lets one infer the type of an object, but it does not let one infer relationships between objects. For example, an OWL-based reasoner might be able to infer that Frank is an uncle, but it would not be able to infer that Frank is Ann's uncle. Consequently, most rules used by the Smart Asset Selector are written in Prolog, which gives complete flexibility and expressiveness. Although Prolog is very expressive, writing efficient Prolog code can be challenging at times. The close tie-in between Prolog and Lisp in the AllegroGraph product facilitated writing efficient rules, as the more

computationally expensive components can be easily expressed in Lisp, if necessary.

A detailed performance analysis has not yet been carried out. However, on a 32-bit dual processor 2.4 GHz machine with approximately 8000 digital assets from which to select, executing individual rules takes between approximately 35 ms and approximately 400 ms, with most rules in our test set executing in under 100 ms. The rule taking 400 ms was particularly complex, searching for the best pictures containing various combinations of people over a reasonably large people set.

A variety of multimedia product types have been implemented and tested on previously collected consumer imagery, including the Mother's Day and special event stories described in this paper, a tribute to a person story, a yearly calendar, and a personalized group event summary. Further work is needed to verify that the chosen results are indeed acceptable to consumers. Note that the consumer's satisfaction with the final product is dependent upon both which assets are selected and how they are presented in the output product. To give a preliminary assessment of the system's utility, it was used to produce output for two externally recruited consumers, who were then given the opportunity to comment on the resulting products. The main comment noted on the choice of assets was that the system did not include as many assets as the consumer would have preferred, but that could be easily accommodated by a simple change to the rules. Additional research would be needed to determine what elements consumers typically expect to go into different types of products, and to what extent the rules need to be personalized for individual preferences. In many cases, an automatically created story may function as a trigger to get the consumer to do more with their pictures; if the story requires only minimal editing by the consumer, then it will be a success.

7. Summary

This work demonstrates the feasibility of using a knowledgebase to create consumer multimedia stories from a user's collection of pictures and videos. This framework quickly produced compelling stories by selecting a small number of assets out of thousands of assets within a collection. The RDF data model combined with a logic programming language enables easy construction and extension of rules driving system behavior. The SSDF implements these concepts to produce selective consumer-grade albums and multimedia presentations from thousands of consumer digital assets.

8. Acknowledgments

Catherine Newell contributed the high-level description of the stories used in this work. Bryan Kraus, Kevin Delong, Kathleen Costello and numerous other people participated in the development and testing of the broader SSDF system. Alexander Loui, Marcello Balduccini, Dhiraj Joshi, and the anonymous reviewers provided feedback on earlier versions of this paper.

9. References

[1] N. Diakopoulos and I. Essa, "Mediating Photo Collage Authoring," in *ACM Symposium on User Interface Software and Technology (UIST)*, Seattle, WA 2005.

[2] D. Joshi, J. Z. Wang, and J. Li, "The Story Picturing Engine-A System for Automatic Text Illustration," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 2, pp. 1-22, 2006.

[3] D. E. Millard, C. Bailey, T. Brody, D. Dupplaw, W. Hall, S. Harris, K. R. Page, G. Power, and M. J. Weal, "Hyperdoc: An Adaptive Narrative System for Dynamic Multimedia Presentations," University of Southampton ECSTR-IAM02-006 2003.

[4] Franz Inc., "AllegroGraph Product Description," 2008. http://agraph.franz.com/allegrograph/.

[5] A. Gallagher, M. Das, and A. Loui, "User-Assisted People Search in Consumer Image Collections," in *IEEE Intern. Conf. on Multimedia and Expo (ICME)*, Beijing, China: IEEE, July 2-5 2007.

[6] J. Luo and M. Boutell, "Natural Scene Classification Using Overcomplete ICA," *Pattern Recognition*, vol. 38, pp. 1507-1519, 2005.

[7] A. Savakis, S. Etz, and A. Loui, "Evaluation of Image Appeal in Consumer Photography," in *SPIE Human Vision and Electronic Imaging*, January 2000.

[8] A. Loui and A. Savakis, "Automated Event Clustering and Quality Screening of Consumer Pictures for Digital Albuming," *IEEE Transactions on Multimedia*, vol. 5, pp. 390-402, 2003.

[9] E. A. Fedorovskaya, S. Endrikhovski, T. A. Matraszek, K. A. Parulski, C. A. Zacks, K. M. Taxier, M. J. Telek, F. Marino, and D. Harel, "Imaging Method and System Using Affective Information," United States Patent 7,233,684, 2007.

[10] J. Grant and D. Beckett, Eds., "RDF Test Cases," World Wide Web Consortium, February, 2004. http://www.w3.org/TR/rdf-testcases/.

[11] J. Aasman, "Using Social Network Analysis, Geotemporal Reasoning, and RDFS++ Reasoning for Business Intelligence," tutorial at The 6th International Semantic Web Conference, November, 2007.