Building Keyword-Indexed Virtual Libraries in a Logic Programming Environment

Edirlei Soares de Lima, Simone Diniz Junqueira Barbosa, Bruno Feijó, Antonio Luz Furtado
Department of Informatics – Pontifical Catholic University of Rio de Janeiro (PUC-RIO)
Rua Marquês de São Vicente, 225 – Rio de Janeiro – Brazil
{elima, simone, bfeijo, furtado}@inf.puc-rio.br

Submited to Webmedia 2014 Conference

ABSTRACT
KW-GPS is a system to assist users intent on enjoying Web resources related to a domain-restricted collection of stories. In this system, each story is referenced in a virtual library in terms of the following data: (1) the URLs of resources associated with the story, which include but are not limited to plot-summaries, narrative texts, and videos; and (2) keywords of different classes, which serve as a multi-aspect index mechanism. Library items also include story templates, representing narrative motifs. Furthermore, a reduced version of the tool runs the basic rank-and-show process on mobile devices.

Categories and Subject Descriptors
H.3.5 **[Information Systems]**: Online Information Services – *Web-based services.*

General Terms
Documentation, Languages, Design, Experimentation.

Keywords
Virtual Libraries, Web Resources, Story Templates, Digital Entertainment, Detective Stories, Logic Programming.

1. INTRODUCTION
Most recommender systems are based on collaborative filtering algorithms and user models that predict the preferences of consumers from a large database of previous interactions with the system. For a user who is searching for specific aspects of a particular series of entertainment products, however, content-based recommender systems may prove to be more adequate in providing such assistance. Nevertheless, it is quite difficult to find Web or mobile services that follow a knowledge-based strategy of recommendation, which is essential to the digital entertainment industry. On the one hand, consumers may need guidance related to the content's structure of their favorite series. For instance, how to find a novel or a game in which a woman is the head of Scotland Yard? On the other hand, authors of digital entertainment may want to go beyond finding inspiration and start creating new stories from interesting combinations of existing ones that fit certain characteristics.

In this paper we propose a simple but effective system, called **KW-GPS**, to assist users intent on enjoying Web resources related to a previously located domain-restricted collection of stories. Each story is referenced in a virtual library in terms of the following data: (1) the URLs of Web-residing resources associated with the story, which include, but are not limited to, plot-summaries, narrative texts, and videos (mostly trailers, due to copyright restrictions); and (2) keywords of different classes, which serve as a multi-aspect index mechanism. The system was initially applied to Agatha Christie's Poirot detective-stories [4] – hence its acronym, which stands for "**K**ey**W**ord-based **G**uide to **P**oirot **S**tories", also reflecting the authors' effort towards a positioning system (i.e. a **GPS** device) to help users to navigate the Web.

The keywords are optionally structured as logical terms, admitting variables as parameters. Moreover this keyword-based structure allows performing other more hardworking processes, involving complex selections and the use of templates to explore similarity.

A logic-programming tool was developed to implement the system. The modular structure of the tool caters for the different roles of prospective users. Experts on logic programming may want to revise some of our design decisions and modify parts of the main program, thus playing, like we originally did, the role of **designers**. Experts on the domain on hand, not expected to be skilled programmers (though, even in their case, some knowledge of logic programming is desirable, to be able to formulate logic expressions), would act as **providers**, searching through the Web for stories and associated resources and choosing appropriate indexing keywords; theirs is the task of producing the domain-specification program, optionally with the help of the authoring module and of a separate tool. Finally a simplified rank-and-show facility, that hides the logic formalisms, and an interface for mobile devices were designed for those whose sole interest is to watch the stories, i.e. the end users (henceforward simply **users**).

The rest of the paper is organized as follows. Section 2 describes the **KW-GPS** system, through examples taken from the domain of Poirot stories (cf. the technical report[1] for details). Section 3 elaborates on criteria to formulate keyword repertoires. The mobile device interface is covered in section 4. Section 5 surveys related works. Concluding remarks are presented in section 6.

[1] ftp://ftp.inf.puc-rio.br/pub/docs/techreports/13_10_lima.pdf

2. THE KW-GPS TOOL

2.1 The rank-and-show facility

The system operates on virtual library data previously located on the Web, pertaining to a given domain such as a set of Poirot stories. Two main sets of clauses represent, respectively, (1) the numbered *library entries*, giving the title of each story and the URLs of the associated resources; and (2) the *index entries*, consisting of keyword lists of different classes for each story (numbered as in the library entries). Three clauses precede them, the first to name the keyword classes, and the others to act as *conditioners* to the ranking process, as will be explained later.

Our choice of keyword classes was influenced by a seminal study by Todorov [31], wherein the highly-reputed literary theorist remarked that in a detective-story there actually coexist two narratives: that of the crime, and that of the investigation. One more class was added, due to Poirot's observation (in *Evil Under the Sun*) that: "Murder springs, nine times out of ten, out of the character and circumstances of the murdered person. Because the victim was the kind of person he or she was, therefore was he or she murdered!". We did *not* include a class about the criminal, because we felt we should leave out whatever might function as a "spoiler", ruining the author's effort to keep the suspense until the end. A small example follows (with the URLs in hyperlink format, for the reader's convenience):

```
/* domain My Poirot 1 */

kw_classes([victim, crime, investigation]).
thresholds(St, [V, C, I], T).
keyword_lists(St, [V, C, I]).

% LIBRARY

lib(1, 'Evil Under the Sun',
       [plot_summary: 'http://goo.gl/CMQtK',
        wiki: 'http://goo.gl/3AQvk',
        video: 'http://goo.gl/7RKxV1']).

lib(2, 'Cards on the Table',
   [plot_summary: 'http://goo.gl/mVmUO',
    wiki: 'http://goo.gl/JXQGy',
    video: 'http://goo.gl/QmykOh']).

lib(3, 'The Mysterious Affair at Styles',
   [plot_summary: 'http://goo.gl/40MRg',
    wiki: 'http://goo.gl/OEWHL',
    full_text: 'http://goo.gl/3qp1R ']).

lib(4, 'The Chocolate Box',
   [plot_summary: 'http://goo.gl/kDnVF',
    wiki: 'http://goo.gl/piaspM',
    video: 'http://goo.gl/BxRzPv']).

% KEYWORDS

kws(1, victim, [gender: female, marital_status: married,
                occupation: actress,
                character: credulous,
                swindled_by: 'younger man',
                age_bracket: 30, economic_status: rich]).
kws(1, crime, [action: murder, means: strangulation,
               place: beach, motive: 'financial gain',
               circumstance: 'holiday season',
               companion: (lover,'younger man')]).

kws(1, investigation, [clue: 'character of the victim',
                       snag: 'time of death',
                       tactic: 'break self-control']).

kws(2, victim, [gender: male, marital_status: single,
                occupation: 'art collector',
                character: bizarre, age_bracket: 40,
                economic_status: rich]).
kws(2, crime,  [action: murder, means: stabbing,
                place: 'drawing room',
                motive: 'avoid accusation',
                circumstance: 'dinner party']).
kws(2, investigation, [clue: 'bridge scores',
                       tactic: 'deceiving trick']).

kws(3, victim, [gender: female, marital_status: married,
                age_bracket: 70, character: credulous,
                economic_status: 'large fortune',
                attitude: autocratic,
                swindled_by: 'younger man',
                occupation: 'social work']).
kws(3, crime, [action: murder, means: poisoning,
               place: bedroom, motive: 'financial gain',
               circumstance: 'medical treatment',
               companion: (someone,'younger man')]).
kws(3, investigation, [clue: 'incriminating letter',
                       tactic: 'expose evidence',
                       snag: 'time of death']).

kws(4, victim, [gender: male, age_bracket: 30,
                character: evil,
                occupation: politician,
                economic_status: middle]).
kws(4, crime, [action: execution, means: poisoning,
               motive: 'moral reasons', place: study,
               circumstance: conversation]).
kws(4, investigation, [clue: chocolates,
                       snag: 'mistaken suspect',
                       tactic: confession]).
```

The ranking process is started by entering the `rank(S)` command, whose single parameter must be a variable to be instantiated at the end with a list of story numbers in decreasing order of total number of hits. The user is asked, for each keyword class, to choose from the respective set of keywords, which are taken from all the stories in the library and displayed on the screen. For the class `victim`, for example, these are:

```
1:age_bracket:30
2:age_bracket:40
3:age_bracket:70
4:character:autocratic
5:character:bizarre
6:character:credulous
7:character:evil
8:economic_status:large fortune
9:economic_status:middle
10:economic_status:rich
11:gender:female
12:gender:male
13:marital_status:married
14:marital_status:single
15:occupation:actress
16:occupation:art collector
17:occupation:politician
18:occupation:social work
19:swindled_by:younger man
choose for victim:
```

To choose, the user types the corresponding numbers, e.g. `7, 12` (for `character:evil` and `gender:male`), and presses the enter key. If the class is not at that moment an aspect of interest, the user simply presses the key without supplying any numbers. Hits are added-up for stories that possess the chosen keywords, but stories with no hits are not excluded. However the user has the option to prefix a number with either a '+' or a '-' sign, to indicate, respectively, that only stories *with* that keyword or only stories *without* it are acceptable. At the end, the output parameter variable is instantiated as mentioned before and, in addition, the result of the ranking process is shown in sentential form. With the choices `7, 12` for `victim`, `4, -11, 14` for `crime`, and `1, 3, 8` for `investigation`, one would have:

```
The Chocolate Box with 5 hits
Cards on the Table with 3 hits
The Mysterious Affair at Styles with 0 hits

S = [4, 2, 3]
```

Notice that `-11`, referring to `means:strangulation`, caused the exclusion of *Evil Under the Sun*. Also notice that a story with 0 hits was kept, which is, in general, useless. To remedy this inconvenience, the `thresholds` clause, shown before in a, so to speak, "neutral" format, can be rewritten so as to impose conditions, both on the total of hits and on the number of hits per class. The purpose of the other conditioning clause, `keyword_lists`, is to act as a *filter*, typically considering the permissible user's choices in a general context. For example, it can specify that if the user explicitly rejects stories with `means:stabbing`, then the even more gruesome stories with `means:strangulation` will also be excluded.

To activate the resources provided for their best-ranked stories, users have a `show` command, whose parameters designate the resource and the story-number. The line below will explore whatever is available for *The Chocolate Box*, a story in which Poirot, in his own opinion, acted with less than his usual genius:

```
:- has_resources(St, 'The Chocolate Box', R),
   forall(member(Ri, R), show(Ri, St)).
```

2.2 Other basic facilities

The `similar` command uses the keywords of a story to rank the others, consequently giving a measure of how close they are to it. Applying this command to *Evil Under the Sun* we learn that it has something in common with all the other stories, most notably with *The Mysterious Affair at Styles*:

```
?- similar(1,S).

The Mysterious Affair at Styles with 7 hits
Cards on the Table with 2 hits
The Chocolate Box with 1 hits

S = [3, 2, 4].
```

Users with a knowledge of the domain's specification have available a flexible `select` command to rank the stories on the basis of explicitly indicated keywords, covering one or more classes. For instance, the same result of the example of the preceding section would be obtained by entering the line:

```
?- select([[character:evil, gender:male],
         [circumstance:'dinner party',
          -means:strangulation,
          motive:'moral reasons'],
         [clue:'bridge scores',
          clue:chocolates,tactic:confession]], S).
```

Of course typing errors should be expected, which led us to introduce a checking device that performs a preliminary comparison of the indicated keywords against those figuring in the `kws` clauses. Automatic substitution will occur if one is found within a *Levenshtein distance* [25] less or equal to 2 from the misspelled keyword. In all such cases a message is displayed, such as:

```
cicunstance:dinner party not found for class {crime} - similar: circumstance:dinner party
```

Automatic substitution will also happen in order to accommodate a number of related terms not included in the `kws` clauses, which, nevertheless, would spontaneously occur to persons familiar with the domain. Indeed we realized that cluttering the `kws` clauses with hypernyms, hyponyms, and other *related terms* would affect the intended user-friendliness of the `rank` command in a negative way, since most people would not like to choose from excessively long lists. A compromise solution was adopted, consisting of the addition to the domain specification of specific (as well as somewhat more general) `related_kw` clauses such as:

```
rel_kw(means:poisoning, means:arsenic).
rel_kw(K, K_rel) :- is_a(K_rel, K).
```

assuming that, to enable the second clause, appropriate `is_a` clauses are also provided. If the system can neither handle the user's indicated term as a misspelled or as a related reference to a registered keyword, the intractable term is not used in the selection and a warning message is issued.

Yet the most significant feature of the select command is its ability to deal with variables, optionally referred to in logical expressions following the '/' separator. The example below searches for stories with victims of both genders younger than 50, noting that whenever variables are involved, the selection list is displayed to reveal how they were instantiated upon the execution of the command – thereby performing a complementary query-answering task:

```
?- select([[age_bracket:A, gender:G], [], []]/(A < 50), S).

1 - [[age_bracket:30, gender:female], [], []]
2 - [[age_bracket:40, gender:male], [], []]
4 - [[age_bracket:30, gender:male], [], []]

The Chocolate Box with 2 hits
Cards on the Table with 2 hits
Evil Under the Sun with 2 hits

S = [4, 2, 1].
```

In view of a thesis cogently exposed in [18] about categorization, we may consider that the two commands in this section, `select` and `similar`, complement each other in a nice way, given that human beings tend to classify things by just wondering to what other (prototypical) thing they resemble, rarely trying to verify systematically whether they have the properties postulated in scholarly taxonomies. Thus people would promptly categorize an animal as a bird if it looks like a robin, a bird *par excellence* after that author. Applying the notion to stories, instead of asking: "give me a story with such and such characteristics" (or "such keywords"), they would say: "give me a story like that one".

2.3 Templates and narrative motifs

The indicated similarity between *Evil Under the Sun* and *The Mysterious Affair at Styles* can be attributed to the presence of a common *narrative motif*, which we call the `Swindler motif`. The motif occurs in other Poirot stories, e.g. *Death in the Nile*.

Keyword classes composed of lists of property:value pairs, such as we have been using in the examples, can be treated as *frames*, a data structure fully compatible with the **Entity-Relationship** model and with **RDF** formalisms [8], which lends itself well to a method for handling narrative motifs. The method utilizes two operations on frames: *unification* and its dual, *most specific generalization* (**msg** for short) [17]. With **msg**, it is possible to combine two or more stories so as to create story *templates* to represent a common motif. Then, by unifying a template with frames with property:value pairs both common and not common with those in the template, one can instantiate and at the same time extend the narrative expressed by the motif. It is a known fact that evoking one or more motifs helps to compose new stories.

Besides the `Swindler motif`, a second motif, taking us to the far-removed domain of Elizabethan drama, will be included here, because Poirot himself explicitly brought it in to find the culprit in his last case (reported in the *Curtain* story). Several apparently unrelated crimes had been committed by different individuals, but all cases had one thing in common: the presence of a person, whom Poirot simply named "X", who had contact with each of the accused. The little Belgian solved the mystery and identified "X" through an analogy with Shakespeare's *Othello*. Accordingly, we expanded our library with *story templates* for both motifs:

```
lib_t(1,'Swindler motif',[]).

kws_t(1,victim, [gender: female, character: credulous,
                 economic_status: rich, swindled_by: X]).
kws_t(1,crime, [action: murder, motive: 'financial gain',
                companion: (Y,X)]).
kws_t(1,investigation, [snag: 'time of death',
                        swindled_by: X,companion: (Y,X),
                        culprit: X, accomplice: Y]).

lib_t(2,'Inducer motif',[]).

kws_t(2,victim, [kills: (B, A), victim_name:A]).
kws_t(2,crime, [kills: (B, A), loves: (B, A),
                tells: (C, B, infidel(A))]).
kws_t(2,investigation, [tells: (C, B, infidel(A)),
                        culprit: C]).
```

To be able to confirm that the templates match the two stories, we first rewrite the `kws` clauses of *Evil Under the Sun* and of *The Mysterious Affair at Styles*, giving the names of the participating characters, and adding `culprit` and `accomplice` terms with variable parameters to the investigation `kws` clauses:

```
kws(1, victim, [gender: female, marital_status: married,
                occupation: actress,
                character:credulous,
                swindled_by: 'Redfern', age_bracket: 30,
                economic_status: rich]).
kws(1, crime, [action: murder, means: strangulation,
               place: beach, motive: 'financial gain',
               circumstance: 'holiday season',
               companion: ('Christine', 'Redfern')]).
kws(1, investigation, [clue: 'character of the victim',
                       snag: 'time of death',
                       tactic: 'break self-control',
                       culprit: X, accomplice: Y]).

kws(3, victim, [gender: female, marital_status: married,
                age_bracket: 70, character: credulous,
                economic_status: 'large fortune',
                attitude: autocratic,
                swindled_by: 'Inglethorp',
                occupation: 'social work']).
kws(3, crime, [action: murder, means: poisoning,
               place: bedroom, motive: 'financial gain',
               circumstance: 'medical treatment',
               companion: (someone,'Inglethorp')]).

kws(3, investigation, [clue: 'chemical property',
                       tactic: 'expose evidence',
                       snag: 'time of death',
                       culprit: X, accomplice: Y]).
```

We are now in a position to introduce a command that uses frame-unification to find which stories incorporate a given motif. Its input parameter is the story template representing the motif, and the output parameter is the list of stories detected. Informally, what the command does is the inverse of the template-generating process: instead of generalizing and introducing variables when needed, it *specializes* by instantiating the variables with constants according to a pattern-matching discipline. So the execution of

```
:- similar_t(1,S).
```

denounces the criminals of the two stories (except that Poirot does not immediately disclose who is the "someone" – a certain Miss Howard – in the second story responsible for the `snag` involving the time of death, which served as an alibi to Mr. Inglethorp):

```
Evil Under the Sun - [[gender:female,
                marital_status:married,
                occupation:actress,
                character:credulous,
                swindled_by:Redfern,
                age_bracket:30, economic_status:rich],
                [action:murder, means:strangulation,
                 place:beach, motive:financial gain,
                 circumstance:holiday season,
                 companion: (Christine, Redfern)],
                [clue:character of the victim,
                 snag:time of death,
                 tactic:break self-control,
                 culprit:Redfern, accomplice:Christine]]

The Mysterious Affair at Styles - [[gender:female,
                 marital_status:married, age_bracket:70,
                 character:credulous,
                 economic_status:large fortune,
                 attitude:autocratic,
                 swindled_by:Inglethorp,                                occupation:social work],
                [action:murder, means:poisoning,
                 place:bedroom, motive:financial gain,
                 circumstance:medical treatment,
                 companion: (someone, Inglethorp)],
                [clue:chemical property,
                 tactic:expose evidence,
                 snag:time of death,
                 culprit:Inglethorp,
                 accomplice:someone]]

S = [1, 3].
```

Proceeding along the same line with the stories incorporating the `Inducer motif`, one obtains:

```
?- similar_t(2,S).

Curtain - [[victim_name:wife], [tells: (Norton, Riggs,
           infidel(wife)), loves: (Riggs, wife),
           kills: (Riggs, wife)],
          [culprit:Norton, executes: (Poirot, Norton)]]

Othello - [[victim_name:Desdemona], [loves: (Othello,
            Desdemona), tells: (Iago, Othello,
            infidel(Desdemona)), kills: (Othello,
            Desdemona),suicides:Othello],
           [culprit:Iago]]

S = [5, 6].
```

As anticipated, templates, as representation of literary motifs, also serve a more ambitious goal: helping to compose new stories. For this purpose, the `select_t` command is used, having as input parameter, like the `select` command of the previous section, explicit keyword lists covering all classes (though for some of them empty lists can be supplied). The command uses these terms to instantiate the `kws_t` clauses of the

templates, its output parameter indicating which templates were successfully matched. Since the command employs frame-unification to achieve instantiation, properties not shared by the two operands, i.e. that do not figure either in the input or in the template, are kept. Thus the story that begins to emerge prolongs, so to speak, the narrative even beyond the motif expressed by the template.

The example introduces two characters unknown in the Poirot world, a man called Archie and his companion Miss Neele; it also adds the novel circumstance that the victim of the supposedly criminal action has disappeared (in itself a motif – see e.g. *The Disappearance of Mr. Davenheim*). The man's female companion performs another evil act: she accuses his wife, called Teresa, of infidelity, thereby reinforcing his murderous impulse. As a result, both templates are separately instantiated, and by virtue of the first template, a culprit and an accomplice are revealed:

```
?- select_t([[swindled_by:'Archie'],
            [companion:('Miss Neele','Archie'),
             tells:('Miss Neele','Archie',
             infidel('Teresa')),
             circumstance:'victim disappears'],[]],S).

Swindler motif - [[swindled_by:Archie, gender:female,
             character:credulous, economic_status:rich],
            [companion: (Miss Neele, Archie),
             tells: (Miss Neele, Archie,infidel(Teresa)),
             circumstance:victim disappears,                          action:murder,motive:financial gain],
            [snag:time of death,
             swindled_by:Archie, companion: (Miss Neele,
             Archie), culprit:Archie,
             accomplice:Miss Neele]]

Inducer motif - [[swindled_by:Archie, victim_name:Teresa,
             loves: (Archie, Teresa)],
            [companion: (Miss Neele, Archie),
             tells: (Miss Neele, Archie,infidel(Teresa)),
             circumstance:victim disappears,
             loves: (Archie,Teresa),
             kills: (Archie, Teresa)],
            [tells: (Miss Neele, Archie,infidel(Teresa)),
             culprit:Miss Neele]]

S = [1, 2].
```

Could a story combining the two motifs be composed? The trouble is that putting the two lines together would be rejected by the frame-unification discipline: a conflict arises regarding the identity of the main culprit. Solving conflicts, in order to be able to combine narrative lines, is a task that often requires much creativity; under the name of blending, it is extensively discussed in [6]. The predicate below (for details, cf. technical report cited before) illustrates one way to face the problem: to simply ask the user to make a choice wherever a property:value conflict arises:

```
?- select_blend([[swindled_by:'Archie'],
     [companion:('Miss Neele','Archie'),
      tells:('Miss Neele','Archie',infidel('Teresa')),
      circumstance:'victim disappears'],[]],S).

conflict with culprit:
1. Archie
2. Miss Neele
Your choice: 2

victim:
[swindled_by:Archie, gender:female, character:credulous,
 economic_status:rich, victim_name:Teresa,
 loves: (Archie, Teresa)]

crime:
[companion: (Miss Neele, Archie), tells: (Miss Neele,
 Archie, infidel(Teresa)),
 circumstance:victim disappears,
 action:murder, motive:financial gain,
 loves: (Archie, Teresa), kills: (Archie, Teresa)]

investigation:
[snag:time of death, swindled_by:Archie,
 companion: (Miss Neele, Archie), accomplice:Miss Neele,
 tells: (Miss Neele, Archie, infidel(Teresa)),
 culprit:Miss Neele]
```

Archie murders his wife for monetary gain, while still in love and feeling jealous for her. Mixed feelings are not uncommon, unwarranted as they may seem, but if Miss Neele is deemed the main culprit, would it make sense to also call her an accomplice? Nonetheless the example illustrates the notion that new stories can arise by combining two or more motifs, and extending the combination with further events. And, although this little story was surely never composed by Agatha Christie, we suspect that some such plot might well have crossed her imagination. (Why? – we leave that to the reader, as a Web-searching exercise).

2.4 A simple rank-and-show user interface

A reduced version of the **KW-GPS** tool, containing only the `rank` and the `show` facilities, is activated by double-clicking an icon, which may conveniently be placed on the monitor's screen. The user does not have to enter any command line, every move being menu-directed. The keyword lists are successively presented for each class, and the user chooses them as in section 2.1.

After the stories are ranked in decreasing order of total hits, the user is invited to indicate what resource should be activated. Suppose the *Cards on the Table* is chosen, and the user asks for its plot-summary. While the plot-summary remains open, other resources can be requested and can be visualized side-by-side if non-overlapping windows are adequately disposed (Figure 1).

When `end` is chosen to the activation of resources, the list of stories is displayed again. Suppose *The Chocolate Box* is then chosen and, from its resources, the Wikipedia entry is called for. Next, if the user twice replies `end` (to the choice of resources and to the choice of stories), these two

recursive loops terminate, and the system backtracks to the outermost loop, asking whether the user wants to perform another selection, thereby starting again the whole process. If the answer is negative, a `halt` command is executed and the Prolog window vanishes from the screen. Thanks to this strictly menu-driven usage mode, the programming language formalisms stay hidden, so that the Prolog machinery becomes practically transparent to users.

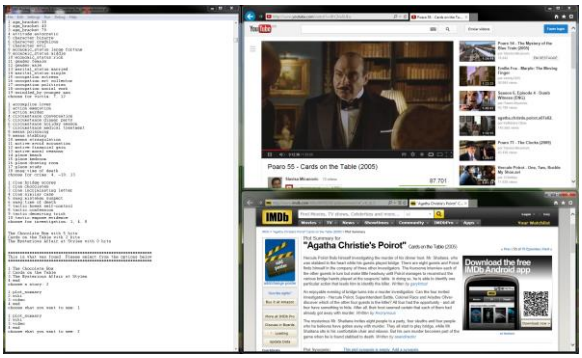

Figure 1. The simple rank-and-show user interface.

2.5 Installing a new domain

To create or redesign a domain, providers have an authoring module that requires only a minimum of familiarity with the notational details of Prolog. The first step is to write a text file specifying how the domain will be called, the names of the classes of keywords, and information about the stories that will constitute the library. The information entries for each story, numbered consecutively, indicate the title of the story and of the URLs of the included resources. The `kws` clauses are inserted by the module, which extracts them from Wikipedia or IMDB plot summaries, asking the user to indicate whether each keyword should be retained or rejected, and in the positive case in which class it should be placed. The user can test the adequacy of a keyword by asking for its **tf-idf** (term-frequency–inverse document-frequency) evaluation [32]. For the meaning of unknown terms, WordNet or DBpedia pages can be opened on demand.

In its present implementation, the module does not provide a general mechanism to locate the resources and retrieve their URLs. However we have separately developed another system, called **LOG-SNIP (**for details, cf. the technical report[2]), which captures the snippets of the resources found in the course of a Google search. To guide the search, a list containing keywords and directives of various kinds is specified, so as to define the domain of current interest. The captured snippets are kept in a Prolog file as frame-structured clauses, decomposed into four fields: `name`, `date`, `url`, `info`. A fifth `kws` field is added by extracting resource-specific keywords from the `name` and `info` fields. A facility is provided to transform the snippets clauses into the clausal notation required by **KW-GPS**: the `lib` clauses are created from the first four fields and the `kws` clauses from the fifth field (which keeps the extracted keywords). As keyword extractor our implementation now uses the **AlchemyAPI** service.[3]

The choice of truly representative keywords is critical. Although plot summaries should in principle be richer sources than snippets, they are contributed by different people, with unequal competence, who may often be misled by personal idiosyncrasies.

3. CONSIDERATIONS ON KEYWORD CHOICE

Keyword repertoires, either featuring ordinary words and phrases or more complex structures such as **property:value** pairs, can either be formed from what is found in the resources themselves (taken from snippets or from plot summaries, as indicated in the previous section) or be borrowed from an external source, such as:

a) terminology of the genre

b) traits of the audience

Option (a) is particularly attractive, since with a limited number of terms the stories can be meaningfully characterized, and neatly compared with each other. For folktales, one may take the 31 functions described in [27] or (some subset of) the many types and motifs of the index compilation in [1]; for drama in general, 36 situations have been identified in [26]. Story segments (scenes), named after a dramatic situation and with keyword classes indicating preconditions and postconditions, can be chained together to form branching plot sequences, furnishing to authors a suitable storyboard scheme. Even to characterize the "story" of sportive games there exist official lists of events, sometimes called *scouts*. The **Fédération Internationale de Volleyball** (**FIVB**)[4] shows in its site the statistics of each game, which can easily be represented in property:value format, where the properties are scouts including `attack`, `block`, `serve`, and the values (for each player and the totals for the two contending teams) are the number of points gained through each of these so-called skills.

For our Poirot examples we utilized, as seen, a number of properties, such as `motive`, `clue`, etc., commonly associated with the genre of detective stories. Surely several other properties in the same line could be added, taken both from studies on fictional works (e.g. the various phases of the crime and the investigation narratives, following Todorov's scheme [2]) and on criminal law (e.g. `mitigating` and `aggravating circumstances` [12]). When adding legal terms, however, one may find advisable to keep compatibility with the author's language, which is not always rigorously correct in this regard – for instance, in *Taken at the Flood*, Agatha Christie allows Poirot to discount as a mere accident what would still draw a verdict of involuntary manslaughter.

---

[2] ftp://ftp.inf.puc-rio.br/pub/docs/techreports/14_01_lima.pdf

[3] http://www.alchemyapi.com/api/keyword-extraction/

[4] http://www.fivb.org/

In line with option (b), the stereotype-based recommendation strategy reported in [28] offers a fascinating possibility, trying to guess, from the personalities of the users, which stories each user would be expected to like. A promising way to implement this notion is to provide for each story property:value keywords evaluating the story according to the traits of the **Big Five** [9] proposal, possibly with percentile intervals as values. Each prospective user might then go through one of the short tests available in the Web,[5] thus obtaining grades to be matched against the intervals estimated (roughly, to begin with, and later refined through usage [28]) for the stories in the virtual library. Another intriguing possibility is to revert the direction: consultants with a psychology background may, by inspecting a usage log, be able to evaluate each user's **Big Five** percentiles from the stories the user has accessed in an extended period of time.

4. THE RANK-AND- SHOW FACILITY IN MOBILE DEVICES

In order to run the rank-and-show process in mobile devices (tablets and cell-phones) we developed an **Android** application that provides a graphical user interface to the **KW-GPS** system (Figure 2). The mobile application is based on a client-server architecture, where the server hosts the **KW-GPS** system and provides access to its functions, and the client contains the mobile user interface that allows users to search and enjoy web resources provided by the system (as in the photo of Figure 3).

Figure 2. Graphical user interface of the mobile application.

In the mobile interface, keywords are organized and displayed in classes (`victim`, `crime` and `investigation`) (Figure 2a). Each class contains three types of keywords: optional (green button), required (blue button), and excluded (red button). When a keyword class and type are designated by the user, the respective list of keywords is displayed for selection (Figure 2b). After the intended keywords are selected, the server performs the rank-and-show process, and the results are shown in the mobile interface (Figure 2c). Looking at the resulting sequence, the user is then free to indicate one of the stories, not necessarily the best ranked, causing the corresponding web resources to be displayed (Figure 2d). The interface is automatically adjusted to the domain specified in the **KW-GPS** system. All the required information about keywords is retrieved from the server that hosts the **KW-GPS** system. In this way, keyword classes and lists are automatically created and labelled according to the specifics of the current domain. The communication between the mobile application and the **KW-GPS** server is performed through a TCP/IP connection.

In order to assess the mobile interface, we have conducted a user evaluation with 9 participants, 8 male and 1 female, aged 16 to 17. Seven of them had some knowledge of detective stories, and three knew about the detective stories of Poirot. All of the participants were frequent users of **Google Web Search**.

We asked the participants to utilize both our mobile application (S) and **Google Web Search** through the default **Android** web browser (G) to find a Poirot detective story to their taste. Our aim was therefore to compare our proposal with the most commonly used method of web search in mobile devices. In order to reduce learning effects, half of the participants used S first, and the other half used G first. On average, each session of S lasted 4.05 minutes (σ=0.86), and each session of G lasted 9.22 minutes (σ=1.48).

[5] http://www.ocf.berkeley.edu/~johnlab/bfi.php

Figure 3. Using the mobile interface.

After using each version, the participants filled in a questionnaire with 26 questions derived from the **USE** Questionnaire [23], concerning the system usefulness, user satisfaction and how easy was the use of the system. Each reply was graded within a 7-point **Likert scale** ranging from "strongly disagree" (-3) through "neutral" (0) to "strongly agree" (+3). After interacting with both systems, the participants were interviewed about their experience.

Figure 4 summarizes the results. Both **Google Web Search** and our mobile **KW-GPS** system obtained similar grades for system usefulness. On the other hand, our system clearly improved user satisfaction and ease of use; in the course of the interviews, the participants declared that it was easy to use, gave more accurate results, and allowed to find interesting stories without risking to enter into unknown web pages. In contrast, some participants said that **Google Web Search** gave them more freedom.

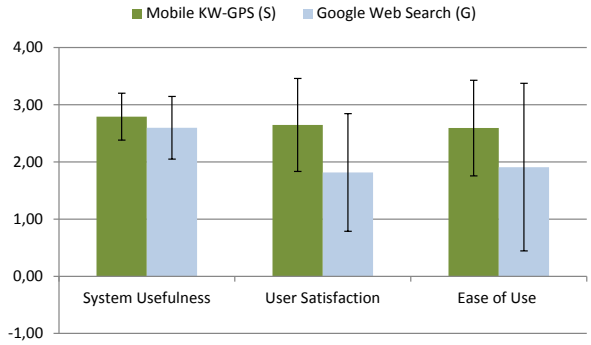

Figure 4. Average number of points (within a 7-point Likert scale) of the system usefulness, user satisfaction, and ease of use, with error bars indicating standard deviation around the mean.

During the user experiment, we also collected some statistical data about the time users spent to complete the task, number of searches, and number of clicks on wrong results. As clearly shown in Figure 5, the mobile **KW-GPS** system cut in more than a half the time needed to complete the task, and reduced substantially the number of searches and clicks on wrong results.

Although the user study reported here cannot be judged entirely conclusive, due to the small number of participants, the positive user feedback is a welcome stimulus for the continuation of our development efforts.

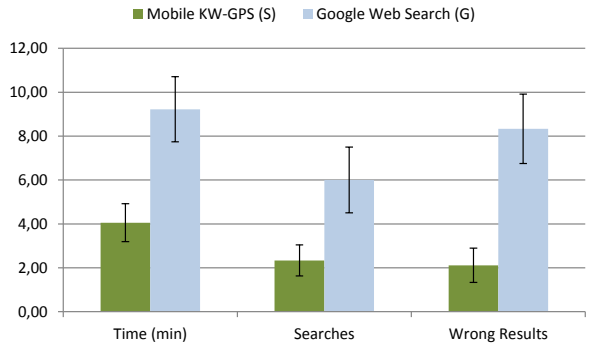

Figure 5. Statistical data collected during the user evaluation about the time users spent to complete the task, number of searches, and number of wrong results.

5. RELATED WORK

Keyword-based search is a well-studied problem in the area of information retrieval. Certain approaches [10][19] explore the use of keyword-based search for XML data; formulating queries with keywords, they retrieve document fragments and use a ranking mechanism to increase

search result quality. Some works [5][3] investigate keyword-based search for the Semantic Web and RDF data, in order to provide ranked retrieval using content-based relevance estimation. Others propose to extend keyword-based search with structured query capabilities [16][24] and logic applied in the context of ambient media [21][22].

In the field of entertainment computing, a sports video search and retrieval system, called DAVVI [29][14], offers the capability of delivering sports video content for mobile and desktop devices. The system is based on automatic summarization and recommendation techniques, wherein sports videos are semi-automatically annotated with metadata extracted from live text commentary web pages. Users can search and query for game events using keywords and phrases found in the live text commentaries. A similar system is described in [11].

There are several mobile applications that provide access to movie databases, such as the "*IMDb Movies & TV*"[6] [13], which is a mobile application developed for **Android**, iOS, and Windows Phone that provides direct access to the **IMDB** movie information database, allowing users to search and navigate through a huge collection of movies and TV series. Another application is "*Movies by Flixster*" [7] for **Android**, **iOS** and **Windows Phone**, which permits to browse and search for movies, read reviews and watch trailers in mobile devices. Several studies suggest the importance of domain-specific search applications for mobiles devices. Both [15] and [30] emphasize that task-specific search applications are better to design in ways that more adequately serve the users' needs. In [20] it is argued that, with such applications, users are able to retrieve documents with fewer interactions and less data traffic.

6. CONCLUDING REMARKS

Originally employed to organize private virtual libraries of Poirot detective-stories, providing a multiple aspect keyword-based index mechanism, the development of the **KW-GPS** system led us to a closer study of keywords as story descriptors. With promising results, we experimented with the property:value frame format, the inclusion of variable parameters to establish links between keywords, and the use of templates to represent story motifs

Regarding the modular architecture of the implementation, it must be stressed that it serves two different purposes. Firstly, the simple `rank-and-show` process, running both on fixed and mobile devices, was designed having in mind what we thought **end users** might find easy to handle – and eventually would like to share with other people. Secondly, the other basic commands, `similar` and `select`, as well as their extensions running on templates, seemed adequate to be incorporated in larger applications, to be built by **designers** with programming expertise, with the help of **providers** with sound domain knowledge.

The choice of logic programming for the implementation proved to be a major asset. Among its unique features are the rule-driven paradigm and the outstanding pattern-matching capability built into the interpreter. Future research will explore other domains, related or not with storytelling, and will submit the **KW-GPS** system to more extensive user-evaluation experiments.

7. REFERENCES

[1] Aarne, A., and Thompson, S. 1987. *The Types of the Folktale*. Suomalainen Tiedeakatemia.

[2] Bordwell, D. 1985. *Narration in the Fiction Film*. University of Wisconsin Press, Madison, WI.

[3] Cheng, G., Ge, W., and Qu, T. 2008. Falcons: searching and browsing entities on the semantic web. *Proc. of 17th International Conference on World Wide Web*, ACM Press, 1101-1102.

[4] Christie, A. 2008. *Hercule Poirot - the Complete Short Stories*. Harper, London.

[5] Ding, L., Pan, R., Finin, T., Joshi, A., Peng, Y., and Kolari, P. 2005. Finding and ranking knowledge on the semantic web. *Proc. of 4th International Semantic Web Conference*, 156-170.

[6] Fauconnier G., and Turner, M. 2002. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books, NY.

[7] Flixster Mobile, 2014. Available at: http://community.flixster.com/m obile/apps [Accessed: Jun. 2014].

[8] Furtado, A.L., Casanova, M.A., Breitman, K.K., and Barbosa, S.D.J. 2009. A Frame Manipulation Algebra for ER Logical Stage Modeling. Proc. 28th International Conference on Conceptual Modeling, Springer, 9-24.

[9] Gosling, S.D., Rentfrow, P.J., and Swann, W.B. 2003. A very brief measure of the Big-Five personality domain. *Journal of Research in Personality*, vol. 37, 505-528.

[10] Guo, L., Shao F., Botev, C., and Shanmugasundaram, J. 2003. XRANK: ranked keyword search over XML documents. *Proc. of ACM SIGMOD International Conference on Management of Data*, ACM Press, 16-27.

[11] Halvorsen, P., Johansen, D., Olstad, B., Kupka, T., Tennøe, S. 2010. vESP: A Video-Enabled Enterprise Search Platform. *Proc. of 4th International Conference on Network and System Security*, Melbourne, 534-541.

[12] Hessick, C.B. 2006. Motive Role in Criminal Punishment. *Southern California Law Review*, vol. 80, 2006.

[13] IMDB, 2014. Movie Apps for iPhone, Android, iPad, WP7 & iPod. Available at: http://www.imdb.com/apps/ [Accessed: Jun. 2014].

[14] Johansen, D., Johansen, H., Aarflot, T., Hurley, J. Kvalnes, A. Gurrin, C., Sav, S., Olstad, B., Aaberg, E., Endestad, T., Riiser, H., Griwodz, C., and Halvorsen, P. 2009. DAVVI: A Prototype for the Next Generation Multimedia Entertainment Platform. *Proc. of 17th ACM international conference on Multimedia*, ACM Press, 989-990.

[15] Kamvar, M., Kellar, M., Patel, R., and Xu, Y. 2009. Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices. *Proc. of 18th international conference on World Wide Web*, ACM Press, 801-810.

[16] Kasneci, G., Suchanek, F.M., Ifrim, G. Elbassuoni, S., Ramanath, M., and Weikum, G., 2008. NAGA: harvesting, searching and ranking knowledge. *Proc. of ACM SIGMOD International Conference on Management of Data*, Vancouver, 1285-1288.

[17] Knight, K. 1989. Unification: A Multidisciplinary Survey. *ACM Computing Surveys*, vol. 21 (1), 93-124.

[6] http://www.youtube.com/watch?v=IVMylQEJUGs

[18] Lakoff, G. and Johnson, M. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago.

[19] Liu , Z., Walker, J., and Chen, Y. 2007. XSeek: a semantic XML search engine using keywords. *Proc. of the 33rd International Conference on Very Large Data Bases*, 1330-1333.

[20] Luca, E.W.D., and Nürnberger, A. 2005. Supporting information retrieval on mobile devices. *Proc. of 7th international conference on Human computer interaction with mobile devices & services*, ACM Press, 347-348.

[21] Lugmayr, A., Zou, Y., Stockleben, B. Lindfors, K. and Melakoski, C., 2013. Categorization of ambient media projects on their business models, innovativeness, and characteristics - evaluation of Nokia Ubimedia MindTrek Award Projects of 2010. Multimedia Tools and Applications, vol. 66 (1), 33-57.

[22] Lugmayr. A., 2012. Connecting the real world with the digital overlay with smart ambient media - applying Peirce's categories in the context of ambient media. Multimedia Tools and Applications, vol. 58 (2), 385-398.

[23] Lund, A. 2001. Measuring Usability with the USE Questionnaire. *STC Usability SIG Newsletter*. Available at: http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html [Accessed: Jun. 2014].

[24] Mandreoli, F., Martoglia, R., Villani, G. and Penzo, W. 2009. Flexible query answering on graph-modeled data. *Proc. 12th International Conference on Extending Database Technology: Advances in Database Technology*, ACM Press, 216-227.

[25] Navarro, G. 2001. A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, vol. 33 (1), 31-88.

[26] Polti, G. 1924. *The Thirty-Six Dramatic Situations*. L. Ray (trans.). James Knapp Reeve.

[27] Propp, V. 1968. *Morphology of the Folktale*. S. Laurence (trans.). University of Texas Press, TX.

[28] Rich, E. 1979. User modeling via stereotypes. *Cognitive Science*, vol. 3 (4), 329-354.

[29] Scott , D., Gurrin, C., Johansen, D., and Johansen, G. 2010. Searching and Recommending Sports Content on Mobile Devices. *Proc. of 16th International Multimedia Modeling Conference*, Chongqing, 779-781.

[30] Sohn, T., Li, K.A., Griswold, W.G., and Hollan, J.D. 2008. A diary study of mobile information needs. *Proc. of 26th annual SIGCHI conference on Human factors in computing systems*, ACM Press, 433-442.

[31] Todorov, T. 1977. *The Poetics of Prose*. Cornell University Press, Cornell, NY.

[32] Wu, H.C., Luk, R.W.P., Wong, K.F., and Kwok, K.L. 2008. Interpreting TF-IDF Term Weights as Making Relevance Decisions. *ACM Transactions on Information Systems*, vol. 26 (3), No 3.