

Semantics-based Framework for Personalized Access to TV Content: Personalized TV Guide Use Case

Pieter Bellekens¹⁴, Kees van der Sluijs¹, Lora Aroyo¹², Geert-Jan Houben¹³,
and Annelies Kaptein⁴

¹ Eindhoven University of Technology (TU/e), Computer Science
{p.a.e.bellekens, k.a.m.sluijs}@tue.nl

² Vrije Universiteit Amsterdam (VU), Computer Science l.m.aroyo@cs.vu.nl

³ Vrije Universiteit Brussels, Computer Science Geert-Jan.Houben@vub.ac.be

⁴ Stoneroos Digital Television, Hilversum <http://www.stoneroos.nl>
annelies.kaptein@stoneroos.nl

1 Introduction

The ICT landscape is developing into a highly-interactive distributed environment in which people interact with multiple devices (e.g. portable devices such as mobile phones and home equipment such as TV's) and multiple applications (e.g. computer programs such as Web browsers and dedicated Web services)[1]. Globally the industry is being driven by the shift away from old models - from physical to digital. New methods emerge for getting content such as TV programs via the Web. Almost half of the people want to watch TV content on their PC's. They want to make a bridge between a TV and a PC, perhaps even sitting in a home office[2]. The information overload is enormous and the *content presented is hardly adapted* to the prior knowledge, to the preferences and to the current situation of the user.

Personalization in information retrieval and information presentation has therefore become a key issue and a key ingredient of the so-called "Web 2.0" applications. However, such personalization is still local (e.g. personalized information is only valid for www.tvgids.nl; it cannot be used for other information services, nor can TVGids cater for different "modes" of a user, e.g. when looking for something personal, or when watching together with friends[3]. Thus, main aspects we cover in this work are **Data integration** of distributed collections; **Context modeling** framework for temporal and spatial-specific viewpoints; **User modeling** in a contextualized form; and **Personalized presentation** of the combined information about data, context and user.

In this paper we present SenSee, a semantics-based framework for providing personalized access to TV content in a cross-media environment. It allows for an integrated view on data harvested from heterogeneous and distributed Web sources. The ultimate goal is to support individual and group TV viewers (operating multiple devices, e.g. PC, set-top box and a mobile) in finding programs that suit best their interests. Personalization here has to consider both

data-integration issues (how is information from different applications and devices related) as well as context-modeling issues (in which space/time/mode are statements about a user valid). iFanzzy is a personalized TV guide application using SenSee framework. iFanzzy consists of Web-based front-end serving as a Web-based remote control point to the set-top front-end. Future extension is considered on a mobile platform.

SenSee integrates multiple data sources, such as BBC data from BBC backstage⁵, XMLTV⁶ and IMDB⁷. Moreover, these sources are interconnected and mapped to external vocabularies, like OWL Time[4], Geo Ontology, TV Anytime genre classification⁸ and WordNet^{9,10}. The resulting abundance of data is controlled in the user interface by offering the user a faceted browsing view on the data, i.e. they can search and browse the data based on facets for time, location and genre. We daily retrieve the metadata from the different multimedia sources on the Web. How we integrated this data is described in section3.

Large part of this framework, primarily supporting the set-top box front-end, has been developed within the context of the Passepartout project [1] in collaboration with Philips Applied Technologies and Stoneroos Interactive Television¹¹. Currently Stoneroos and Technical University of Eindhoven are working towards the commercial deployment of the framework in the form of the personalized electronic program guide iFanzzy¹². Here we present the iFanzzy Web-based front-end using SenSee framework.

This is work in progress so expect to find changes over time, both on the platform side as on the interface side. However, the basic functionality is ready and available at <http://wwwis.win.tue.nl:8888/SenSee>. Also take a look at the tutorial we included with the demo to give a feeThe SenSee Webclient is built on top of GWT¹³, which allows it to run on most modern JavaScript enabled Web-browsers. However, so far we have only tested it in Firefox 2 and Internet Explorer 7. Note that between 2:00h and 4:00h (GMT +1:00) the servers retrieve, parse and transform broadcast information, so expect the application to be slower in that period of time.

2 Architecture

Figure1 gives an overview of the architecture of the SenSee framework (i.e. excluding the front-end interfaces) complying with the main requirements to be scalable, extensible and flexible. For more details on the architecture see[3].

⁵ <http://backstage.bbc.co.uk/>

⁶ <http://xmltv.org/wiki/>

⁷ <http://imdb.com>

⁸ <http://www.tv-anytime.org/>

⁹ <http://wordnet.princeton.edu/>

¹⁰ <http://www.w3.org/2001/sw/BestPractices/WNET/wn-conversion>

¹¹ <http://www.stoneroos.nl/>

¹² <http://www.stoneroos.nl/portfolio/case-study/Passepartout-personalised-EPG>

¹³ <http://code.google.com/webtoolkit/>

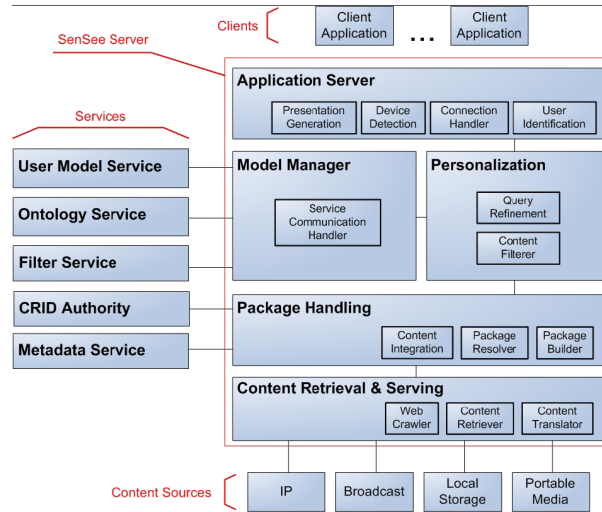


Fig. 1. SenSee Architecture

A key element in this framework is the use of Semantic Web languages RDF(S)/OWL to model all input data and its relationships, so that we can reason and navigate over it in an uniform connected way. In this case it means navigating via shared facets like time, location and genre.

For the scalability we rely on the RDF storage, inferencing and querying framework Sesame. All data handled within the SenSee platform is stored and retrieved by Sesame, including all metadata sources as well as the ontological sources, the user model and context models.

Various clients applications can use the SenSee platform. These are not only front-end interfaces (e.g. iFancy settop box and web-based applications)¹⁴, but could also be sensed-devices like the sensor-enhanced pillow[5] we have also developed in collaboration with V2 and CWI.

3 Data Integration

Program information from BBC broadcasts are retrieved directly from BBC Backstage in TV-Anytime XML format. We have transformed this into OWL/RDF and used SKOS to describe relations between concepts in the loosely structured vocabulary (i.e. by using the relations skos:broader, skos:narrower and skos:related). Furthermore we mapped the TV-Anytime time definition to the OWL-TIME notion of time. So, for every crawl of the data source we transform the input XML data to RDF-instances of our schema and we transform the data by using XPath. Another broadcast source is XMLTV program information grabbed from several program information websites. XMLTV covers various

¹⁴ <http://en.wikipedia.org/wiki/iFancy>

countries worldwide¹⁵. For demonstration purposes we have chosen to focus on the broadcast information from UK and the Netherlands. This choice does not imply any restrictions in the generality of the framework. Both data sources are crawled on a daily basis.

Movie-related information we gather from the IMDB dataset (text dump of the IMDB database and with data model transformed by us in RDF(S)). This text dump however does not contain photos and movie trailers URLs. Thus, we made scripts that on access of movie details retrieve such information live from the Web with screen scrapers and then cache that data in the database. The IMDB information is used for detailed descriptions of movies in the regular broadcasted TV programming as well as a in the "movie on demand" option, e.g. on a pay-per-view basis. As there is currently no commercial pay-per-view party involved in the project we show-case only that it is possible to integrate such a movie source with no time-dimension, e.g. broadcast time. For the pay-per-view option we use the 10.000 most popular movies in the IMDB dataset (determined by the number of IMDB votes).

An overview of all the datasets we use within the demo:

Data source	#triples
User Model (schema)	319
IMDB schema	408
TV Anytime Genre Classification	3.971
Geo Ontology	59.835
Time Ontology	1.547
Country Codes	2.023
WordNet	1.942.887
BBC dataset (random pick)	91.447
XMLTV dataset (random pick)	1.278.718
IMDB dataset	7.969.199

Table 1. Size of data sources

4 Faceted Browsing

Bringing together TV content, available from high number of broadcasting channels and a number of shared vocabularies for its semantics-rich representation, requires a special attention of the search and browse interface presented to the user. Thus, we provide two ways to access and navigate TV content, i.e. by search and by facet. Both ways of navigation can also be combined, but both are not compulsorily.

¹⁵ see <http://xmltv.org/wiki/xmltvworlddomination.html> for an overview



Fig. 2. SenSee Screenshot

Figure 2 shows a screenshot of the Web interface search panel. On the left one can browse selected *facets*, e.g. time, genre and location. *Time* is visualized as a calendar showing the current day, and the following six days. Users can select a day and further specify the time period. Only broadcast information is time-bound as opposed to on-demand movies, so a time-selection will only filter broadcast results. *Genre* is visualized as a TV-Anytime classification tree. Users can select one or more genres for the program their searching for. *Location* information is visualized as a IMDB-location tree. In the IMDB dataset every movie has filming locations. The form of such location is built in a hierarchical string. For instance, for the movie *The Godfather* one of the filming locations is *Bronx, New York City, New York, USA*. We parsed these strings building a hierarchy, and created a structured modeled similar to the RFDS subclass structure, i.e. *Bronx -subComponentOf-> New York City -subComponentOf-> USA*. When we encountered equally named location names with different parents, we created different unique nodes for them. For instance there is a *Athens* in both the *USA* and in *Greece*, but the locations are still different. It appeared that locations were uniquely named in the database (i.e. there were no derivations like *USA* in one place and *United States of America* in another). Users can select one or more locations in the tree, similar as we do for genres, which limits movies that were shot in the selected locations. Locations are in a separate store with inferencing, so we compute closure. If a user selects a location that has sub-components, those will also be returned as a result.

The *basic search* field accepts a Google-like string input. By default the input strings are matched to title, keywords and actor names in all sources available. The *advanced search* triggers the following actions before the query execution:

- **Keywords broadening:** configured to query for synonyms in WordNet, i.e. terms appearing in the same set as the input term. We currently limited the keyword expansion to 5 new terms per keyword. The output is a set of keywords.
- **Keyword conceptualization:** All facet-sources (e.g. time, genre and location) are parsed for match for every keyword in the broader set. This matching has three modes, i.e. strict, loose and free. In the *strict match* the keyword should be syntactically equal to the concept (ignoring case), e.g. the keyword **News** and the genre concept **News** match. *Loose match* looks for sub-strings, e.g. the keyword **News** and the genre concept **Daily News** match, where it would not match in strict mode. *Free match* does pattern matching and would find the term **day** as a match with **Friday**, where it would not match in loose match. The output is a set of concepts.
- **Concepts broadening:** for a given object property we traverse the corresponding graphs and find related concepts (including via inverse relationships). For example, the current TIME ontology configuration is to consider inverse **time:after** relationship, i.e. we find all concepts that are connected to a given time concept via the **time:after** relationship. Given, for instance, the **time:Noon** concept, we will also find the **time:afterNoon** concept. For the TV-Anytime genre classification we consider the **skos:broader** relationship, resulting in all concepts that are *narrower* than the input concept. For example, for the **tva:3.2 (Sports)** concept we will also find the **tva:3.1.1.9 (Sports News)**, as all programs of the genre **Sports News** are also about **Sports**.

These actions lead to execution of a number of queries that takes time in an order that is linear to the *number of keywords*, the *number of ontologies*, the *size of ontologies* and the *configured number of extension steps*. The more generic a search keyword is, the more extensive the set of additional terms and concepts is. In overall, these broadening and conceptualization actions take on average about 4 or 5 seconds to execute on a reasonable modern single server machine. That is without querying the data sources yet with more complex query than average and without any special optimizations, e.g. keyword indexes, which could considerably speed-up the process. Currently, we undergo various optimizations of this part of the demonstrator.

Results can be grouped in several facets. Which facets and in which order the results will be grouped can be controlled at the bottom of the screen by drag and drop the facet names in order. Based on the selection the grouping will be arranged in a tree. The top priority facet will make out the first level of the tree. If **Genre** is of highest importance to the user, the first level tree will contain the genres of the first N results in alphabetic order. Grouping occurs in a similar fashion for the extra facets. Note that programs can have several genres, and that results will therefore occur more than once in the tree.

5 Personalization and Adaptation

In our previous work with a sensor-enhanced pillow[5] in combination with the set-top box front-end, we have collected bio-sensor data in order to calibrate the user profile. In the current web application use case we gather user data by monitoring the user behavior, or as we have shown in previous research by re-using data from other application[6]. The current implementation does not include a user interface to create user profiles yet. For the purposes of this demo we created three different user and context profiles. The current time is determined dynamically, however other context information, such as current location, is for demo purposes pre-set in the context user profiles. One user preference is language. Currently we support English, Dutch and Swedish. If several user log at the same time there profiles are combined and the common language is chosen. The default language is English, however if all logged users prefer another language that language will be chosen (i.e. in general we compute an intersection of the profiles).

The key personalization feature is adapting the search results with respect to the user profile, i.e. content with user preferred terms and genres are ranked higher. Also explicit non-preferent terms and genres are taken into account. Context information is applied in a similar fashion. It can be configured in the concrete application and adapted to the context needed there. In the current demo we assume that people would prefer to look at certain type of TV programs that relate to the location they are in. For example, traffic information, news and police reports for a location in London if the user is situated there.

6 Lessons Learned and Future Work

Several problems and issues have been tackled during the work on this project. In this section we present a brief reflection on them and draw lessons learned.

Sesame performance and scalability: In close collaboration with Aduna¹⁶ we used Sesame as the backbone RDF storage and querying. The new alpha (and later beta) allowed us to use the newest Sesame features related to context modeling. However, this came in a package with a *poor query optimization* (for now), especially critical when using large datasets like IMDB. The overall result is a slower and less scalable demonstrator. By cross testing queries in Sesame 1 and 2, and regarding that the query and data model should lead to better optimization in the second version lead us to believe that some queries will improve in evaluation time up to a factor 10. Similarly, by using Googles GWT toolkit for programming the front-end we depend on its portability and efficiency (which is for example currently rather slow and inflexible with tree-rendering).

Code optimization: Currently we undergo various optimizations of the system, e.g. free text indexing for matching keywords. The application now runs on a single (single-core) server because of resource limitations, while the application lends itself quite well for parallelization, especially if we move specific

¹⁶ <http://www.aduna-software.com>

functionality to different machines. So if the application feels a bit sluggish now (especially with multiple users), expect it to greatly speed up in the following months.

Use of live data: Initially we aimed at working with all live data. For instance, we did not want to store the IMDB-text dump (as it is not refreshed that often), but query it live all of the time and scrape its HTML-pages. However, after experiencing many layout changes, and thus rewrites of the screen scrapers, we had to give that up because of the amount of work it gave us. The problems with the BBC Backstage data was even more rigorous. Apparently, after a test phase of exactly two years, BBC decided to stop publishing this daily data at all. As this happened rather recently our current solution is a bit of an improvised one, we weekly change dates of the last published data so that it matches the coming week. Even though the program data is not really relevant anymore, we considered that is was the concept that counted.

User interface: The current faceted-based presentation of the search result is a good demonstration of combining multiple perspectives in one view. However, we are exploring options to use timeline¹⁷ for the temporal aspect and a map¹⁸ for the location facet. Furthermore, we aim at realizing combinations of different facets, e.g. conventional paper tv-guides typically combine the channel facet with the time facet.

Recommendations: Recommendation functionality (currently implemented for an empty query) needs features currently not available in our backbone database's query engine, such as sorting, aggregation, query nesting and update queries. Further developments in the context of recommendations will involve social aspects and focus more on group recommendations, content sharing, etc.

References

1. Bjorkman, M., Aroyo, L., Bellekens, P., Dekker, T., Loef, E., Pulles, R.: Personalised home media centre using semantically enriched tv-anytime content. In: EuroITV 2006 Conference. (2006) 156–165
2. Aroyo, L., Bellekens, P., Bjorkman, M., Houben, G.J.: Semantic-based framework for personalised ambient media. *Multimedia Tools and Applications in print* (2007)
3. Aroyo, L., Bellekens, P., Bjorkman, M., Houben, G.J., Akkermans, P., Kaptein, A.: Sensee framework for personalized access to tv content. In: *Interactive TV: a Shared Experience*, Amsterdam, the Netherlands, Springer (2007) 156–165
4. Hobbs, J.R., Pan, F.: An ontology of time for the semantic web. *ACM Transactions on Asian Language Information Processing (TALIP)* **3**(1) (2004) 66–85
5. Aroyo, L., Nack, F., Schiphorst, T., Schut, H., KauwATjoe, M.: Personalized ambient media experience: move.me case study. In: *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, New York, NY, USA, ACM Press (2007) 298–301
6. van der Sluijs, K., Houben, G.J.: A generic component for exchanging user models between web-based systems. *IJCELL Journal* **16**(1/2) (2006) 64–76

¹⁷ <http://simile.mit.edu/timeline/>

¹⁸ like Google maps: <http://www.google.com/apis/maps/>