



LINKEDTV



Deliverable 4.6 Contextualisation solution and implementation

Matei Mancas, Fabien Grisard, François Rocca (UMONS)

Dorothea Tsatsou, Georgios Lazaridis, Pantelis Ieronimakis, Vasileios Mezaris (CERTH)

Tomáš Kliegr, Jaroslav Kuchař, Milan Šimůnek, Stanislav Vojříř (UEP)

Werner Halft, Aya Kamel, Daniel Stein, Jingquan Xie (FRAUNHOFER)

Lotte Belice Baltussen (SOUND AND VISION)

Nico Patz (RBB)

31.10.2014

Work Package 4: Contextualisation and Personalization

LinkedTV

Television Linked To The Web

Integrated Project (IP)

FP7-ICT-2011-7. Information and Communication Technologies

Grant Agreement Number 287911

| | |
|------------------------------------|---|
| Dissemination level ¹ | <i>PU</i> |
| Contractual date of delivery | <i>30th September 2014</i> |
| Actual date of delivery | <i>31st October 2014</i> |
| Deliverable number | <i>D4.6</i> |
| Deliverable name | <i>Contextualisation solution and implementation</i> |
| File | <i>LinkedTV_D4.6.docx</i> |
| Nature | <i>Report</i> |
| Status & version | <i>v1.0</i> |
| Number of pages | <i>96</i> |
| WP contributing to the deliverable | <i>WP 4</i> |
| Task responsible | <i>UMONS</i> |
| Other contributors | <i>UEP CERTH FRAUNHOFER IAIS SOUND AND VISION RBB</i> |
| Author(s) | <i>Matei Mancas, Fabien Grisard, François Rocca (UMONS) Dorothea Tsatsou, Georgios Lazaridis, Pantelis Ieronimakis, Vasileios Mezaris (CERTH) Tomáš Kliegr, Jaroslav Kuchař, Milan Šimůnek, Stanislav Vojtíš (UEP) Werner Halft, Aya Kamel, Daniel Stein, Jingquan Xie (FRAUNHOFER) Lotte Belice Baltussen (SOUND AND VISION)</i> |

¹ • PU = Public

- PP = Restricted to other programme participants (including the Commission Services)
- RE = Restricted to a group specified by the consortium (including the Commission Services)
- CO = Confidential, only for members of the consortium (including the Commission Services)

| | |
|------------------------------|---|
| | <i>Nico Patz (RBB)</i> |
| Reviewer | <i>Jan Thomsen, CONDAT</i> |
| EC Project Officer | <i>Thomas Küpper</i> |
| Keywords | <i>Contextualization, Implementation, Ontology, semantic user model, interest tracking, context tracking, preference learning, association rules, user modelling, context</i> |
| Abstract (for dissemination) | <i>This deliverable presents the WP4 contextualisation final implementation. As contextualization has a high impact on all the other modules of WP4 (especially personalization and recommendation), the deliverable intends to provide a picture of the final WP4 workflow implementation.</i> |

Table of contents

| | | |
|----------|---|-----------|
| 1 | Contextualisation overview | 10 |
| 1.1 | History of the document | 12 |
| 1.2 | List of related deliverables..... | 12 |
| 2 | Contextualisation and LinkedTV Scenarios | 13 |
| 2.1 | Personalization-aware scenarios..... | 13 |
| 2.1.1 | TKK Scenarios | 13 |
| 2.1.2 | RBB Scenarios | 17 |
| 2.1.2.1 | Nina, 33, urban mom | 17 |
| 2.1.2.2 | Peter, 65, retired..... | 20 |
| 2.2 | Context-aware scenarios..... | 22 |
| 3 | The Core Technology..... | 24 |
| 4 | Core Reference Knowledge | 25 |
| 4.1 | LUMO v2..... | 25 |
| 4.1.1 | LUMO-arts..... | 27 |
| 5 | Implicit user interactions | 29 |
| 5.1 | Behavioural features extraction | 29 |
| 5.1.1 | Head direction validation: the setup..... | 34 |
| 5.1.2 | Head direction validation: some results | 35 |
| 5.2 | Communication of behavioural features with the LinkedTV player..... | 39 |
| 5.3 | Communication of LinkedTV player with GAIN/InBeat module | 41 |
| 5.3.1 | API description | 41 |
| 5.3.1.1 | Player Actions | 42 |
| 5.3.1.2 | User Actions..... | 43 |
| 5.3.1.3 | Application specific actions..... | 43 |
| 5.3.1.4 | Contextual Features | 43 |
| 5.4 | InBeat | 44 |
| 5.4.1 | Import of annotations for media content..... | 45 |
| 5.4.2 | Support of contextualization | 46 |
| 5.4.3 | InBeat Recommender System | 47 |

| | | |
|-----------|--|-----------|
| 5.4.3.1 | Components | 47 |
| 5.4.3.2 | Recommender Algorithms | 48 |
| 5.4.3.3 | In-Beat: Matching Preference Rules with Content..... | 48 |
| 5.4.3.4 | InBeat: Ensemble as combination of multiple recommenders | 48 |
| 6 | User model..... | 50 |
| 6.1 | Linked Profiler contextual adaptation..... | 50 |
| 6.2 | Scenario-based user models..... | 52 |
| 6.2.1 | TKK scenario user profiles..... | 53 |
| 6.2.2 | RBB scenario | 54 |
| 7 | Core Recommendation | 58 |
| 7.1 | LiFR reasoner-based recommendation and evaluation | 58 |
| 7.1.1 | LiFR performance evaluation..... | 62 |
| 7.1.2 | Bringing recommendations to the general workflow | 64 |
| 8 | The Experimental Technology..... | 65 |
| 9 | Experimental Reference Knowledge..... | 66 |
| 9.1 | LUMOPedia | 66 |
| 9.1.1 | Design considerations | 66 |
| 9.1.1.1 | Dedicated Temporal-aware Relational Schema..... | 66 |
| 9.1.1.2 | Reasoning with Open World and Closed World Assumptions..... | 67 |
| 9.1.1.3 | Unified Ontology for User and Content Modelling | 69 |
| 9.1.2 | Statistics of the LUMOPedia knowledge base | 69 |
| 9.1.3 | LUMOPedia Browser as the frontend | 70 |
| 9.1.3.1 | The class taxonomy..... | 71 |
| 9.1.3.2 | The schema and property definitions..... | 71 |
| 9.1.3.3 | The instances with temporal constraints | 72 |
| 9.1.4 | Backend with JavaEE and PostgreSQL..... | 72 |
| 9.1.5 | Summary..... | 74 |
| 10 | Experimental Explicit User Interaction | 75 |
| 10.1 | LUME: the user profile editor..... | 75 |
| 10.1.1 | System requirements..... | 75 |
| 10.1.2 | HTML5-based frontend design | 76 |
| 10.1.2.1 | Manage user models | 77 |

| | | |
|-----------|--|-----------|
| 10.1.3 | NodeJS powered service layer | 81 |
| 10.1.4 | Data Management with PostgreSQL..... | 82 |
| 10.1.5 | Summary..... | 83 |
| 11 | Experimental Recommendation | 84 |
| 11.1 | Personal Recommender..... | 84 |
| 11.1.1 | System design..... | 84 |
| 11.1.1.1 | Incrementally Building the Knowledge Base | 84 |
| 11.1.1.2 | Materialise the Semantics via Enrichment | 85 |
| 11.1.1.3 | Semantic Recommendation Generation | 86 |
| 11.1.1.4 | The sample video base..... | 87 |
| 11.1.1.5 | Related web contents | 87 |
| 11.1.2 | Personal Recommender – the prototype frontend..... | 88 |
| 11.1.3 | The RESTful web services | 90 |
| 11.1.4 | Summary..... | 93 |
| 12 | Conclusions & Future Work | 94 |
| 13 | Bibliography | 95 |

List of Figures

| | |
|--|----|
| Figure 1: The WP4 core implicit personalization and contextualization workflow which is under implementation and will be a part of the final demonstrator..... | 11 |
| Figure 2: The WP4 extended workflow containing both the core and experimental (LUMOPedia, LUME, LSF Recommender) modules. | 11 |
| Figure 3: Literary work products (article, novel, etc) were moved under the Intangible > Work category in v2, as opposed to under the Topic > Literature category in v1. They were related to Literature via the hasSubtopic property in a corresponding axiom | 26 |
| Figure 4: Object properties in LUMO v2 and the semantics, domain and range of property "authorOf" | 27 |
| Figure 5: Extract of the LUMO-arts expansion | 28 |
| Figure 6: User racking for default mode (left) and seated mode (right). | 30 |
| Figure 7: Seated tracking with face tracking. | 31 |
| Figure 8: The different action units given by the Microsoft SDK and their position on the face [WYR]..... | 32 |
| Figure 9: Action units on the left and expression discrimination on the right | 32 |
| Figure 10: Three different degrees of freedom: pitch, roll and yaw [FAC]. | 33 |
| Figure 11: User face windows with head pose estimation and age estimation..... | 33 |
| Figure 12: The user is placed in front of the TV, and covers his head with a hat with infrared reflectors for the Qualisys system. | 34 |
| Figure 13: Setup for facial tracking recording. The Kinect for the head tracking algorithm is marked in green. We can also see the infrared reflectors for the Qualisys on the TV corners. | 35 |
| Figure 14: Mean correlation with the reference for the pitch depending on the distance from TV..... | 36 |
| Figure 15: Mean correlation with the reference for the yaw depending on the distance from TV..... | 36 |
| Figure 16: Mean correlation with the reference for the roll depending on the distance from TV | 37 |
| Figure 17: Mean RMSE (in degrees) for the pitch depending on the distance to TV. | 37 |
| Figure 18: Mean RMSE (in degrees) for the yaw depending on the distance to TV. | 38 |
| Figure 19: Mean RMSE (in degrees) for the roll depending on the distance. | 38 |
| Figure 20: pause action performed by Rita at 32s of video | 42 |
| Figure 21: bookmark of specific chapter performed by Rita | 42 |
| Figure 22: view action of presented enrichment performed by Rita | 42 |
| Figure 23: User Action Example: user Rita logged in..... | 43 |
| Figure 24: Application Specific Actions Example: user Rita opens a new screen (TV, second screen ...) | 43 |
| Figure 25: Rita started looking at second screen device at 15th second of video | 44 |
| Figure 26: Example of annotation send from player along with event | 46 |
| Figure 27: Example of "keepalive" event for propagation of context | 47 |

| | |
|---|----|
| Figure 28: LiFR's time performance for topic detection. Points denote each content item's processing time. The line shows the polynomial trendline (order of 6) of the data points. | 63 |
| Figure 29: Relational database schema hosting the LUMOPedia knowledge base | 68 |
| Figure 30: Histogram of the curated instance relations | 70 |
| Figure 31: LUMOPedia Browser - the web-based frontend of the LUMOPedia knowledge base | 71 |
| Figure 32: The defined and inherited properties for the class "movie" | 72 |
| Figure 33: Architecture of the LUMOPedia Browser application | 73 |
| Figure 34: The revised architecture of LUME | 76 |
| Figure 35: A screenshot of the LUME user profile editor | 78 |
| Figure 36: Add an instance as a UME in LUME | 78 |
| Figure 37: Add a class with constraints as a UME in LUME | 79 |
| Figure 38: Natural Language interface in LUME | 79 |
| Figure 39: Eliminate the semantic misunderstanding by selecting the structured information | 80 |
| Figure 40: The confirmation popup dialog for the deletion of a UME | 80 |
| Figure 41: Fast filtering and ranking the UME list | 81 |
| Figure 42: The database schema for storing the user models | 83 |
| Figure 43: UML deployment diagram of the Personal Recommender | 85 |
| Figure 44: The illustration of the enrichment process for an instance | 86 |
| Figure 45: Top 30 LUMOPedia instances used in the video annotations | 87 |
| Figure 46: One screenshot of the video base displayed in the Personal Recommender frontend | 88 |
| Figure 47: One screen shot of the Personal Recommender | 89 |
| Figure 48: The enrichments of the entity "Berlin" | 90 |

List of Tables

| | |
|---|----|
| Table 1: History of the document | 12 |
| Table 2: OSC messages from Interest Tracker to the player and the produced action | 23 |
| Table 3: List of behavioural and contextual features available from Interest Tracker. The features can be sent through HTTP or websocket protocols. | 39 |
| Table 4: Description of REST service used for tracking of interactions | 41 |
| Table 5: GAIN output example. prefix d_r_ indicates that the feature corresponds to a DBpedia resource from the English DBpedia, the prefix d_o_ to a concept from the DBpedia Ontology. For DBpedia in other languages, the prefix is d_r_lang_. | 44 |
| Table 6: GAIN interaction for Nina: <i>bookmarking</i> a media item while in the company of her kids | 50 |
| Table 7: Nina's interaction serialized in her (previously empty) user profile | 51 |
| Table 8: An example of a user profile of the first use case | 59 |
| Table 9: Use case 1: Precision, recall, f-measure for the recommendation of the 73 manually annotated content items, over the 7 manual user profiles | 60 |

| | |
|--|----|
| Table 10: Use case 2: Precision, recall, f-measure for the recommendation of the 50 automatically annotated RBB content items, over the 5 manual user profiles | 60 |
| Table 11: Average precision, recall, f-measure of the automatic annotation of 50 RBB videos in comparison with the ground truth annotations..... | 60 |
| Table 12: Use case 3/RBB scenario: Precision, recall, f-measure for the recommendation of the 4 automatically annotated RBB chapters, over the <i>Nina</i> and <i>Peter</i> manual user profiles..... | 61 |
| Table 13: Use case 3/TKK scenario: Precision, recall, f-measure for the recommendation of the 9 automatically annotated RBB chapters, over the <i>Anne</i> , <i>Bert</i> , <i>Michael</i> and <i>Rita</i> manual user profiles | 62 |
| Table 14: Time performance and memory consumption of LiFR, FiRE and FuzzyDL on global GLB calculation | 63 |
| Table 15: Statistics of the LUMOPedia knowledge base..... | 70 |
| Table 16: The list of all RESTful services implemented in LUME service layer..... | 82 |

1 Contextualisation overview

This deliverable deals with contextualizing content information, a process which is used for more efficient user profile personalization. As contextualization impacts the entire workflow of WP4, in this deliverable, advances in the final personalization and contextualization workflow implementation is detailed in two steps.

The first step is the core workflow which is already implemented or in the process of implementation (Figure 1) within the LinkedTV workflow. The core workflow comprises of implicit personalization and contextualization, and subsequent concept and content recommendation, and will be demonstrated by LinkedTV partners.

The second step is an extended experimental branch, consisting of an optional explicit personalization and contextualization approach which is optimized and that can be used for testing with available REST services (Figure 2).

This deliverable is structured around those two steps to describe the different blocks visible in Figure 2 and Figure 1.

Chapter 2 illustrates personalization and contextualization within the 3 LinkedTV scenarios. To this end, the 3 scenarios are summarized and their link with contextualisation and personalization (for the two first) and contextualization (only for the third one) is shown.

Chapter 3 introduces the core personalization and contextualization workflow and details the chapters that deal with this workflow (chapters 4, 5, 6, 7).

Chapter 4 presents updates on the core background knowledge and to this end it describes the LUMO v2 ontology and its arts and artefacts oriented expansion, namely LUMO-arts.

Chapter 5 focuses on the implicit contextualized user tracking and preference extraction, which comprises of the attention/context tracker and Inbeat mainly through its GAIN and PL module.

Chapter 6 describes the process of setting up of a contextualized user model using information from the core LUMO ontology (Chapter 4) and the implicit contextualization (Chapter 5). It also presents the final user profiles of the personas presented in Chapter 2.

Chapter 7 deals with providing and evaluating content recommendations based on the user models presented in Chapter 6. This process is conducted via the core recommender which is based on the LiFR reasoner. In addition, it presents evaluations on the reasoner's algorithmic efficiency.

Chapter 8 introduces the optional experimental branch and details the chapters that deal with this workflow (chapters 9, 10, 11).

Chapter 9 deals with the optional knowledge base called LUMOPedia.

Chapter 10 talks about the optional explicit preference induction of the LUME module.

Chapter 11 details the optional Personal Recommender module (LSF).

While the present deliverable describes how the WP4 workflow is implemented, tests on real data going through the entire pipeline will be detailed in the next deliverable (D4.7 about Validation).

Compared to previous deliverables exposing the contextualization and personalization ideas at a conceptual level, in the present deliverable the final pipeline is set up with all the necessary technical details needed for implementation (Figure 1).

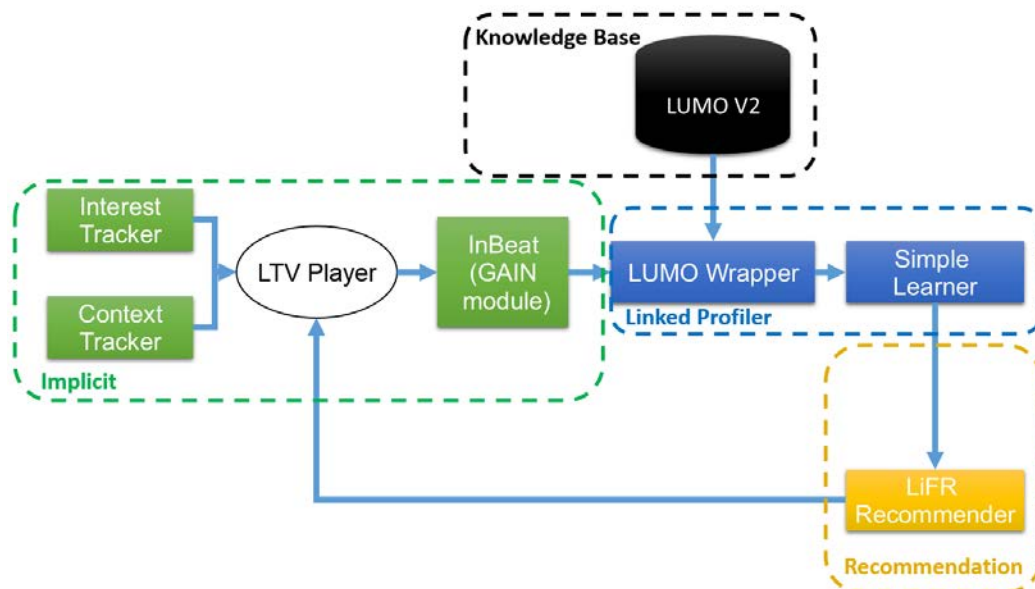


Figure 1: The WP4 core implicit personalization and contextualization workflow which is under implementation and will be a part of the final demonstrator.

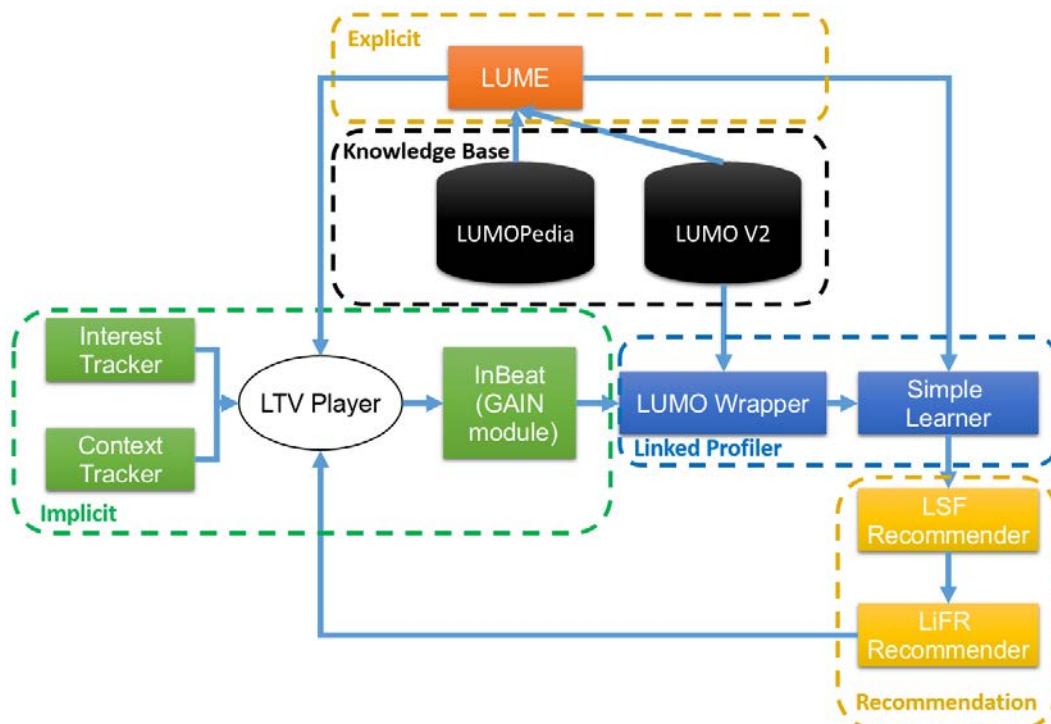


Figure 2: The WP4 extended workflow containing both the core and experimental (LUMOPedia, LUME, LSF Recommender) modules.

1.1 History of the document

Table 1: History of the document

| Date | Version | Name | Comment |
|------------|---------|------------------|---|
| 2014/05/21 | V0.1 | Matei Mancas | Empty document with initial ToC to be discussed |
| 2014/08/8 | V0.2 | Tomas Kliegr | UEP sections 5.3, 5.4 |
| 2014/08/14 | V0.3 | Daniel Stein | FhG sections |
| 2014/08/15 | V0.4 | Lotte Baltussen | Added section on contextualisation and scenarios from Sound and Vision use case perspective |
| 2014/08/27 | V0.4.1 | Daniel Stein | Minor update FhG sections |
| 2014/09/12 | V0.5 | Matei Mancas | Adding attention tracker validation |
| 2014/09/22 | V0.6 | Nico Patz | Adding RBB scenario |
| 2014/10/15 | V0.7 | Dorothea Tsatsou | Adding CERTH chapters |
| 2014/10/22 | V0.8 | Tomas Kliegr | 1 st QA addressed for UEP sections |
| 2014/10/24 | V0.9 | Dorothea Tsatsou | 1 st QA addressed for CERTH sections |
| 2014/10/27 | V1.0 | Matei Mancas | Fusion/Formatting/Ready for final QA |
| 2014/10/29 | V1.0.1 | Dorothea Tsatsou | Final QA addressed for CERTH sections, formatting and spell check. |
| 2014/10/30 | V1.0.3 | Jaroslav Ku-car | Final QA addressed for UEP sections |
| 2014/10/30 | V1.0.4 | Matei Mancas | Final QA addressed |

1.2 List of related deliverables

This deliverable is related to the previous ones which focus on contextualization and personalization (D4.2, D4.4 and D4.5). There are also links with the proposed scenarios in terms of personalization and contextualization information with deliverable D6.4. Also, there is a connection with D2.6 regarding the topic detection module which utilizes components inherent of WP4 tools. Finally, the communication between the final results of the WP4 workflow, namely the recommendations, and the platform are described in D5.6.

2 Contextualisation and LinkedTV Scenarios

LinkedTV proposes 3 different scenarios which are detailed in the deliverable D6.4. Two of them use the entire WP4 pipeline and they are referred in the following sections as “personalization-aware” scenarios. The third one does not use the personalization but it uses the Interest and Context trackers which are the first brick of the WP4 pipeline (Figure 1). This scenario which is not personalization-aware but only context-aware is referred as “context-aware” scenario.

In the following sections, the scenarios are summarized and their interaction with WP4 in terms of context and/or personalization are shown.

2.1 Personalization-aware scenarios

2.1.1 TKK Scenarios

The Sound and Vision scenarios are based on the programme Tussen Kunst & Kitsch (henceforth: TKK) by Dutch public broadcaster AVRO. In the show, people can bring in art objects, which are then appraised by experts, who give information about e.g. the object’s creator, creation period, art style and value. The general aim of the scenarios is to describe how the information need of the Antiques Roadshow viewers can be satisfied from both their couch and on-the-go, supporting both passive and more active needs. Linking to external information and content, such as Europeana [EUR], museum collections but also auction information has been incorporated. These scenarios (three in total) can be found in full in D6.4 Scenario demonstrator v2. The personas and scenario summaries are provided below, after which the specific personalization issues and an example of a personalised profile will be provided.

Rita: Tussen Kunst & Kitsch lover (young, medium media literacy)



- Name and occupation: Rita, administrative assistant at Art History department of the University of Amsterdam
- Age: 34
- Nationality / place of residence: Dutch / Amsterdam
- Search behaviour: Explorative
- Digital literacy: Medium

1. Rita logs in to the LinkedTV application, so she can bookmark chapters that interest her.
2. Rita is interested to find out more about the host Nelleke van der Krogt.
3. Rita wants more information on the location of the programme, the Museum Martena and the concept of period rooms.
4. Rita wants more information on an object, the Frisian silver tea jar, and Frisian silver in particular.
5. Rita wants to bookmark this information to look at more in-depth later.
6. Rita wants to learn more about painter Jan Sluijters and the art styles he and his contemporaries represent.
7. Rita wants to plan a visit to the Museum Martena.
8. Rita invites her sister to join her when she visits the Museum Martena.
9. Rita checks the resources she's added to her favourites.
10. Rita sends a link to all chapters with expert Emiel Aardewerk to her sister.
11. Rita switches off.

Bert and Anne: Antiques Dealer, volunteer (older, high + low media literacy)



- Name and occupation: Bert, antiques dealer. Anne, volunteer at retirement home.
 - Age: Bert - 61, Anne - 59
 - Nationality / place of residence: Dutch / Leiden
 - Search behaviour: Bert - Focused. Anne - Explorative
 - Digital literacy: Bert - High. Anne – Low
1. Bert sees a chapter about a statuette from the late 17th century, which is worth 12,5K, which is similar to a statuette he recently bought.
 2. Bert bookmarks this chapter, so he can view it and the information sources related to it later on.
 3. Bert immediately gets the chance to do so, because a chapter about a watch is next, something he doesn't really care for.
 4. Anne is however very interested in the watch chapter: it depicts gods from Greek mythology and she wants to brush up on her knowledge. She asks Bert to bookmark the information to the Greek gods to read later.
 5. Anne would like to know more on why tea was so valuable (more than its silver container!) in the 18th century. Bert bookmarks the silver tea jar chapter for her.
 6. Bert and Anne read and watch the additional information related to the wooden statue chapter and the Greek mythology after the show. Bert has sent the latter to Anne through email, so she can read it on her own device.

Michael: library manager (middle-aged, high media literacy)

- Name and occupation: Michael, library manager at a public library.
 - Age: 55
 - Nationality / place of residence: Dutch / Laren
 - Search behaviour: Explorative and focussed
 - Digital literacy: High
1. Michael comes home late and has missed the latest Tussen Kunst & Kitsch episode. He logs in to the LinkedTV application and starts watching it from there.
 2. He skips the first chapter that doesn't interest him, and then starts watching one about a Delftware plate.
 3. He likes Delftware, and sends the chapter to the main screen to explore more information about the plate on his tablet. It turns out doesn't like this specific plate much.
 4. He selected a related chapter filmed at De Porceleynse Fles, a renowned Delftware factory in Delft. This is about a plate he does like.
 5. He adds relevant Delftware chapters to his "Delftware" playlist.
 6. After this, there's a chapter on a silver box, which reminds him of silver box he inherited from his grandparents.
 7. Michael sees a link to similar content related to the chapter and finds another box similar to the one he owns. He bookmarks the chapter and shares the link via Twitter.

Personalization in the S&V scenarios

In order to make clear how personalization appears in these user scenarios, the Michael scenario summary is expanded below to indicate 1) which concepts the persona finds interesting (or not) and 2) to make clear how the persona acts in various situations.

CONCEPTS:

- Michael is interested in [boxes] [made out of silver] and that were made in [Europe].
 - Short term preference: 100%
 - Middle term preference: 90%
 - Long term preference: 85%
- Michael would like to learn more about the [Jewish] [Sukkot] Festival, in which the spice [etrog] place an important role.
 - Short term preference: 60%
 - Middle term preference: 20%
 - Long term preference: 5%
- When Michael really likes a type of art object, like [Delftware plates] made at [De Porcelyne Fles], he wants to see [all related chapters] from TKK.
 - Short term preference: 100%
 - Middle term preference: 80%

- Long term preference: 60%
- Michael is not interested in [Delftware plates] with [oriental depictions].
 - Short term preference: -100%
 - Middle term preference: -50%
 - Long term preference: -50%
- Michael wants to learn more about the [designer] [Jan Eisenlöffel], who made objects in the [art nouveau]-related style [New Art], since Michael really loves art nouveau.
 - Short term preference: 90%
 - Middle term preference: 80%
 - Long term preference: 70%
- Michael bookmarks the top three recommended TKK chapters related to [art nouveau] to watch later.
 - Short term preference: 90%
 - Middle term preference: 60%
 - Long term preference: 40%
- Michael is not interested in [African] [masks].
 - Short term preference: -85%
 - Middle term preference: -85%
 - Long term preference: -85%
- Michael is interested in [paintings] with value [over 10,000 euros]
 - Short term preference: 90%
 - Middle term preference: 90%
 - Long term preference: 90%

SITUATIONAL CONTEXT

Rita

When Rita is not very interested in a chapter, she likes to use that time to [pick up her weights] and does [some weight-lifting] until the chapter is over.

Bert and Anne

- When Anne likes a chapter, but Bert doesn't, he will [look away from the main screen and browse the web on this tablet], whereas Anne will keep watching the main screen.
- When Anne doesn't like a chapter, but Bert does, she will [get up and make a coffee], whereas Michael will keep watching the main screen.

Michael

- When Michael [views TKK with his wife] they specifically like to plan [visits to the museum] in which the episode is recorded.
- When Michael has [missed an episode], and a chapter come up that he doesn't find interesting (e.g. one on an [African mask], he will [skip to the next chapter].
- However, when he watches an episode [together with his wife], he will [not skip the chapter], because she like to see the whole show, so he will then not [watch the television screen] but [use his tablet to surf or check his mail].
- Sometimes Michael uses the TKK Linked Culture app to [browse through the show's archive] based on his interests (e.g. 'art nouveau'), [bookmark chapters related to his interest] and then [watches the bookmarked chapters one after the other].
- When Michael is watching a chapter while [browsing the TKK archive], and he sees related information he likes on the second screen, he will [click the related information], [pause the episode] and [resume it when he's checked out the information].

2.1.2 RBB Scenarios

The RBB scenarios are based on the daily news show RBB AKTUELL which is being enriched with the help of the LinkedTV process. A combination of devices (HbbTV set or set-top box plus optional tablet) allows different depth of (extra) information to cater for varying information needs. Levels of information can be chosen step by step:

1. Watching the “pure” news show, users will see notifications in the top right corner whenever extra information is available
2. If a user is interested in this extra information, s/he can get an introductory text about this person, location or topic easily on the TV screen with just a push of a button
3. Whenever a user feels that this introduction was not enough, s/he will pick up the tablet and find more information and links to further resources in the second screen app

The following describes how different users apply these steps differently according to different interests and information needs as well as different device preferences. [Entities] will be followed by an interest value in (XY %).

2.1.2.1 Nina, 33, urban mom

Nina, now 33, is a young, urban mom. Her baby is growing and getting more active so Nina has to be even more flexible, also with respect to where she is watching the news and when she is consuming additional information; e.g. the baby is sleeping or playing in her room, but Nina has to keep an eye on her and be able to pause the interactive LinkedNews at any time.

The tablet is her main screen, not only because she is young and innovative and keeps playing around with the tablet any free minute to escape from her daily responsibilities, but also because it makes her more mobile.

Nina's show always has that chapter first on the list which was detected as the most relevant according to her profile settings - but switching back to default view is very simple. According to her preferences Nina will receive the topics in the order described in the following (#1, #2, #4, #5, #8, #9, #7, #3, #10, #6, #11).

How Nina is watching the show of 02 June 2014

Nina is generally not interested in the [host] (0%), and this guy, [Arndt Breitfeld] (0%) doesn't change her mind.

1. Chapter #1: New Reproach against BER Management

Nina is generally interested in Berlin politics ([Berlin] AND [politics]: 90%) and has been watching the developments around [BER airport] (80%) closely. She found it especially interesting that [Wowereit] (90%) and the BER holding [Flughafengesellschaft Berlin-Brandenburg BER] (60%) invited [Hartmut Mehdorn] to become BER top manager, although under his lead [Deutsche Bahn] (30%) had almost gone bankrupt.

She is very interested in [Federal politics] (90%), but only to a limited extent when it comes to the [Ministry of Transport] and its Minister [Alexander Dobrindt] (40%).

She doesn't like the Pirates party [Die Piraten] (-40%), incl. [Martin Delius] (30%), but they started playing an interesting role in German politics, so she couldn't afford to miss what they

are saying - in effect that would mean that she would not be interested in reading background information, but she would not want to miss news items where they give their comments!

2. Chapter #2: Danger of a Blackout?

Would police, fire and other rescue service still work, if all electricity went off? [Emergency Management] (70%) This is definitely a matter of importance for an urban mother!_She consumes all the information about [Energy] (80%) / [Energy AND Security] (80%), the Berlin [Fire Dept.] (50%), Berlin [police] (50%), Berlin's public transport service providers [Berliner Verkehrsbetriebe BVG] (75%) and [S-Bahn Berlin GmbH] (85%).

Nina really can't stand Berlin's Senator of the Interior, [Frank Henkel] (-70%), but as she is very interested in such security issues she fights the wish to skip and listens to what he has to say.

Listening to [Christopher Lauer] (-40%), another member of the Pirates party [Die Piraten] (-40%), is equally hard for her to bear, so, as she thinks that this news item seems to be done anyway, she eventually skips to the next chapter.

Then there is this expert interview: As an ecology-minded person, Nina is interested in hearing about how the much discussed [Energy Transition] (90%) can even foster her need for energy security. She picks up her tablet again to check what it might hold for her and takes the time to check the enrichments on the German Institute for Economic Research [Deutsches Institut für Wirtschaftsforschung DIW] (50%) which is here represented by the expert interviewee, [Claudia Kemfert] (0%). When the discussion turns to [Renewable Energies] (80%) and the expert defends these with strong and convincing arguments, Nina's interest unexpectedly rises and she picks up the app's enrichments on [Claudia Kemfert] (50%).

3. Chapter 4: Brown Coal in Brandenburg

Nina is wondering a little why a news item on [Brandenburg] (20%) should be ranked so high on her preference list, but soon she realises that this is about [Renewable Energy] (80%), [Greenpeace] (90%) and [people's rights] (60%), so she listens intensely. Seeing that politicians of [SPD] (60%) and [Die Linke] (65%) act against their own promises really makes her angry, but the fact that people from the area will be relocated [Umsiedlung] (50%) from their homes to other places is even more annoying. Nina is interested in the mentioned plans both on the pullout from fossil energies and the relocation of whole villages, so she checks the enrichments to learn more.

The [Renewable Energy] (80%) expert again. Nina liked her arguments in the other interview so she stays interested.

4. Chapter 5: Refugee Camps in Berlin

Nina has followed the story of the refugees [Flüchtlinge] (70%) on Oranienplatz and the development of the discussions closely. [Human Rights] (70%) and the stories of refugees and how they are treated is always interesting for her. Usually she likes to look at the situation in other countries, now she is very interested in seeing what is going on in Berlin and Germany.

5. Chapter 8: New RBB Smart Apps

Here is a new app!? Of course, Nina is interested in [Smartphone]s (65%), [Tablet]s (70%) and other [New Media] apps and devices, so she listens carefully how the new apps intend to enable user participation.

6. Chapter 9: Arthaus Festival celebrates 20th anniversary of Arthaus Films

[Die Blechtrommel] (70%) always used to be one of her favourite movies and Nina loves going to the [Cinema] (80%).

[Günter Grass] (65%) has been discussed a lot in past years for his diverse history: he seems to have been in the [SS] (50%) and thus a servant to [Nationalsozialismus] (National Socialism) (65%), but in the 1970s and 1980s he used to be famous for his left-wing activities.

7. Chapter 7: Short News Block 2

[Charity] (70%) is always a nice topic, so Nina keeps her attention high while watching this short news item on a campaign where people leave their change behind at the supermarket cash desk, so it can be transferred to kid's foster homes.

And here is another heart-melting activity: someone supported the building of a hospice for end-of-life care for [children] (95%) and [youths] (75%). How could Nina not support this!?

[Science] (50%) is generally a topic which needs to be handled carefully, but Nina is definitely not interested in huge telescopes in Arizona's deadlands.

8. Chapter 3: Short News Block 1

Nina is shocked that Berlin's [police] (50%) apparently keeps records of mental illnesses and even transferable diseases like HIV. What about the [German Constitution]'s (70%) [first article] (90%) ("Human dignity shall be inviolable. To respect and protect it shall be the duty of all state authority.")??? This is an outrageous provocation of this basic law!

Another car accident; bitter but nothing to look behind the scenes. Before she could even consider skipping, the spot was over.

Oh, but this accident between a bicycle and a car happened just around the corner, in her neighborhood in [Prenzlauer Berg] (90%)! Maybe she even knew this guy? She quickly thinks who she might have to call, but of course, the news doesn't mention names in such events.

9. Chapter 10: Medien Morgen

[Glienicke Brücke] (70%) is a beautiful spot between [Potsdam] (20%) and [Berlin] (70%), but Nina neither likes [Steven Spielberg] (-50%) nor [Tom Hanks] (-30%) too much.

Hearing about the [Geisterbahnhof] (unused station) underneath [Kreuzberg]'s (60%) Dresdner Straße really makes Nina curious and she is very interested in checking available enrichments

10. Chapter 6: Public Viewing on the Sofa

Nina loves [Brazil] (75%), but is not interested in [Soccer] (-40%), but she absolutely detests [FIFA] (-95%) and what they did to take as much as possible out of the [FIFA World Cup] (-100%). Therefore, Nina ignores this news chapter which was automatically sorted at the end of her list.

2.1.2.2 Peter, 65, retired

Since Peter retired he is mainly interested in culture and sports and everything that happens around him, in Potsdam and the region of western Brandenburg. Peter knows what is going on, but he is always interested in taking a closer look.

When it comes to personalization, Peter is rather conservative. He trusts the editors that when they deem something very important, it will be very important. To him an editor is almost like a director: he (or she) has a tension curve in mind and Peter wouldn't want to destroy this, so his settings read: "Editor's order".

How Peter is watching the show of 02 June 2014

Peter generally likes looking at the Information cards for speakers/anchormen. This young man [Arndt Breitfeld] (50%) seems to be new in the anchorman position, but Peter thinks that he may have seen him before - so he checks the tablet for background information on the young man.

1. Chapter 1: New Reproach against BER Management

Peter is generally interested in regional topics [Brandenburg] (90%) and especially in [BER] (75%), as it is the nearest airport. Furthermore, this is about corruption, that is to say that these people like [Jochen Großmann] (0%) waste our public money and that is absolutely unacceptable! Who is this guy, anyway? Peter had never heard of him before, so it is time to check this new rbb Tablet Service!

Even Federal Minister [Alexander Dobrindt] (-20%) [Federal politics] (40%) now joins the discussion and announces action in this tragedy.

Peter is not particularly fond of the representatives of the Pirates party [Die Piraten] (-70%), but this guy [Martin Delius] (30%) surprisingly speaks out Peter's thoughts!

2. Chapter 2: Danger of a Blackout?

Would police, fire and other rescue service still work, if all electricity went off? [Emergency Management] (90%) is definitely an issue for everyone! Peter listens closely and meanwhile bookmarks the additional information on his favourite topics: [Fire Rescue] (90%), [Police] (80%) and [Technology] (95%).

[Berlin] (0%)! It's always Berlin! Does anyone care about the weak infrastructure in [Brandenburg] (95%)? They always talk about [Berliner Verkehrsbetriebe BVG] (-80%) and [S-Bahn Berlin GmbH] (30%), which at least operates trains to Potsdam, too. The ways are much longer between the small townships in the country and the network of busses and trains is by far weaker.

Peter switches to the next spot to see, if it brings anything about [Potsdam] (90%)

3. Chapter 3: Short News Block 1

[Berlin] (0%) again! But the Short News are usually too short to skip, so Peter stays with it. Hm, so Berlin's [Police] (80%) keep records of people with mental illnesses and Transferable Diseases? Yea, so what!? That is absolutely logical and fair, because they have to know about these special dangers, or not?

Oh, a car accident on the Autobahn A10 near [Ludwigsfelde] (80%)! That is actually quite nearby! [Brandenburg] (95%).

Oh, a biker got killed in an accident!? No one knows why this man, 42, unexpectedly changed from bike track to the road, but these bikers are crazy, anyway!

4. Chapter 4: Brown Coal in Brandenburg

This next chapter is about the [Social Democrats] (60%) and the [Socialists] (85%) who rule in [Brandenburg] (95%) and how they lied to get voted! Peter is truly disappointed that even his preferred party cannot be trusted!

While Peter is still checking the tablet for information about what the people of the region think, a new spot about refugees in [Berlin] (0%), [Kreuzberg] (-70%), starts.

5. Chapter 5: Refugee Camps in Berlin

As Peter is not at all interested in what the Hippies do in Berlin's streets, he quickly pushed the Arrow Up and skips to the next spot by pushing the Arrow Left.

6. Chapter 6: Public Viewing on the Sofa

Peter is not so much into [sports] (40%), let alone into [soccer] (20%), but with the [FIFA World Cup] (55%) coming, it may be worth listening and indeed...

This looks like a lot of fun: People can bring their sofas into the football stadium and meet there for public viewings! Sitting on the sofa and not being alone – how could he not love the idea!? But, unfortunately, the Stadium at [Alte Försterei] (0%) in [Berlin] (40%) [Köpenick] (20%) is much too far away and he has no idea how to get his sofa on the green! But he likes the idea.

7. Chapter 7: Short News Block 2

Peter had seen this [charity] (65%) campaign at the supermarket and he likes this grey-haired guy, but somehow he still didn't get how he could do any good, i.e. how he could help in this campaign. The tablet certainly has links to further information, so Peter quickly grabs it and pushes the „Charity“ box with the image of this famous guy to the bookmarks section at the top to check it later.

[Science] AND [Technology] (80%) has always been a favourite topic for Peter, so he is especially proud that scientists from [Potsdam] (90%) now send a huge telescope or something to [America] (-40%). This may help these Ami guys see that Potsdam is much bigger than they thought!

8. Chapter 8: New rbb Smart Apps

There is the nice young man again, announcing that rbb's news shows, both the one for [Berlin] (0%) and the one for [Brandenburg] [85%], now launched apps for Tablets [Technology] (80%). Peter listens closely, trying to understand what makes these better than the one he is using just now – probably it's the option to send comments and even Photos or videos if you happen to witness any accident or so. Now that sounds nice, so Peter quickly bookmarks this spot for download information, so he may try them later. ...and it is also nice to see the speakers and moderators of RBB like [Tatjana Jury] (80%), [Dirk Platt] (60%), [Cathrin Böhme] (70%) and [Sascha Hingst] (90%) and even some people from behind the scenes, like [Christoph Singelstein] (0%), the main editor in chief.

9. Chapter 9: Arthaus Festival celebrates 20th anniversary of Arthaus Films

“[Die Blechtrommel]”? (30%) by [Gunter Grass] (-65%). Yes, Peter had heard this book title numerous times, but he doesn't know much about it as he preferred reading East-German books at the time. SO, he calls up the Information Cards on the TV screen again to get a first notion and see, if he should explore further. After the first bits of information he decides, he has seen enough. Eventually, Peter closes the service and the TV in general to go and check the bookmarks he had made during the show.

2.2 Context-aware scenarios

The proposed artistic scenarios managed by the UMONS partner explore various opportunities arising from the merge of LinkedTV technologies and media arts. They aim to present demonstrations of current achievement and trigger conceptual ideas from several media artists. Artists who participated to the call for projects were assisted by UMONS to define the outline, and then refine their projects to use in the most relevant way the technologies developed for LinkedTV. From the three retained scenarios, one had a specific interest in identifying context and behaviours. This scenario does not make use of personalization techniques, but it uses contextual features (number of people, looking at the main/second screen, joint attention, viewing time) on viewer's reactions provided by the interest and context tracker developed in WP4.

This scenario is called *Social Documentary* and is detailed in the deliverable D6.4, section 4.2. Shortly, it consists in an interactive and evolving artistic installation created to navigate through a collection of multimedia content. The project has emerged after the social events that happened in the Gezi Park in Istanbul (Turkey, June 2013) and during which a lot of content has been produced by both TV channels and protestors themselves. The artists, four former students from Istanbul Technical University, wanted to re-use some of this content and present it “as-it” in their installation. They also wanted to use the visitors' behaviour as an input of their system, to compare it to the behaviour of people on the images and violence of the videos. The attention of the visitors is used as a Facebook “like”, each time a visitor watches the video for at least 5.5 seconds, the rate of this video is increased. The rate is directly linked to the probability to display this video later to other visitors. In case of joint attention - two visitors looking at the same screen - the player adds a Tweet from our selection (tagged with #GeziPark, #occupyGezi, etc.) on the main screen.

The installation software is divided in three parts communicating through OSC² protocol. This protocol which is UDP based is easily accessible using a lot of softwares used by media artists. The player part, programmed in Processing (www.processing.org) a simplified Java platform extensively used by media artists, receives messages from a Reactable software and the Interest Tracker developed in WP4 and manages the video feedback (through video projection and effects). We use a MS Kinect sensor and a modified version of the Interest Tracker to send specific messages using the OSC protocol. These messages (see Table 2) inform the player about the number of visitors in the room, if they look to the main screen (projection wall) or the second screen (projected on a table in front of them), and the time they have spent looking at it, related to the attention level [HAW05]. Only the two visitors closest to the Kinect sensor are taken into account.

Table 2: OSC messages from Interest Tracker to the player and the produced action

| OSC message | Values | Description | Player effect |
|----------------------------------|--|--|------------------------------------|
| /context/nbusers | [0..6] | Number of users tracked in the room | Stop the video if 0 |
| /context/facetrackedusers | userID [0..5], (userID [0..5]) | IDs of the users for whom we have a face tracking | Updates internal state |
| /context/jointattention | 0, 1 | 1 if the face tracked users are watching the same screen, 0 else | Add tweet on the main screen |
| /context/user/attention | userID [0..5], screenID [0, 1], attention [0..3] | Attention level of a user when it changes | Update currently played video rate |
| /context/user/coordinates | userID [0..5], screenID [0, 1], x[-1..1], y[-1..1] | Approximate gaze coordinates on the screen for a single user, not used by the player | none |

² <http://opensoundcontrol.org/>

3 The Core Technology

The implicit personalization and contextualization workflow is displayed in Figure 1, which is also available below. This consists of the core WP4 technology which will be fully implemented and tested throughout with LinkedTV data in the next deliverable (D4.7).

Concerning the modules communication implementation (Figure 1), the Kinect-based behavioural Interest/Context Tracker sends events (through a HTTP protocol) to the player. The player enriches the events with the video ID and time when the events occurred and passes them to the GAIN module using the GAIN API (which is also HTTP-based), to enable retrieval of the specific media fragment for which an Interest/Context event was manifested. In addition to the behavioural Interest/Context events, the player also sends player interaction events (like pause, play, bookmark etc...) using the same channel to GAIN.

The GAIN module fuses this data and provides a singular measure of user interest for all the entities describing a given media fragment and in a given context (alone, with other people, etc...). In addition, the PL module detects associations between entities for a given user, which it formulates in association rules. The communication between these modules is detailed in Chapter 5.

This information is sent using a RESTful service to the model building step, namely the Linked Profiler. This step comprises conveying entities into the LUMO “world” via the LUMO Wrapper utility and using this data to progressively learn user preferences based on the Simple Learner component. Communication of the implicit tracking and the Linked Profiler module is detailed in Chapter 6.

Finally, the user models created by the Linked Profiler are passed onto the LiFR-based recommender, which matches user preferences to candidate concepts and content and as a result provides recommendations over this data. Recommendation results are lastly provided to the LinkedTV platform as described in Chapter 7.

The core pipeline provides all the functionalities needed by the three scenarios of LinkedTV described in Chapter 2. Additional experimental modules can be used for explicit model management for example and they will be detailed in Chapters 8 to 11.

4 Core Reference Knowledge

In year 3 of LinkedTV, the ontologies engineered to consist of the background knowledge for the personalization and contextualization services were revised and updated. To this end, a new version of LUMO [TSA14a] (v2)³ was released, along with an arts and artefacts expansion, namely LUMO-arts.

4.1 LUMO v2

As described in deliverable D4.4, ch. 2, LUMO serves as a uniform, lightweight schema with well-formed semantics and advanced concept interrelations which models the networked media superdomain from an end-user's perspective. It balances between being not too abstract or too specific, in order to scale well and maintain the decidability of formal reasoning algorithms. These traits might make LUMO useful to a plurality of semantic services designed to manage/deliver content to an end user, even passed its use within LinkedTV and besides or alongside personalization. Semantic search, semantic categorization, semantic profiling, semantic matchmaking and recommendation technologies that are hindered by the volume and inconsistency of other vocabularies and/or need to take advantage of advanced inferencing algorithms might benefit from reusing LUMO as their background ontology. Its reusability from semantic technologies is further strengthened by its connection to the most prominent LOD vocabularies, as described in D2.6, ch. 5, which prevail in Semantic Web applications.

In the scope of personalisation and contextualisation in LinkedTV, LUMO aims to (a) homogenize implicitly tracked user interests under a uniform user-pertinent vocabulary, (b) express and combine content information with contextual features under this common vocabulary and (c) provide hierarchical and non-taxonomical concept connections at the schema level that will enable advanced semantic matchmaking between user profiles and candidate content, both in the concept as well as at the content filtering layer.

LUMO is primarily the schema behind the implicit tracking core workflow, where it is used to formulate both user preferences as well as homogenize information about the content, but can also be used in the explicit tracking branch (ch. 8-11), to express user interests.

In comparison to v1, v2 of LUMO has been updated and extended on 4 layers:

1) New concepts

New concepts (classes) were added at the schema level, to enhance completeness of the ontology. This provides greater coverage of (a) the relevant concept space and (b) the newest versions of the vocabularies that WP2 uses to annotate content and LUMO maps to (refer to D2.6, ch. 5 for LUMO mappings to other vocabularies).

³ For LUMO engineering principles, design decisions, core ontology/v1 presentation, refer to D4.4, ch. 2.

Covering these vocabularies is important, since WP2 annotations are the information from which implicit user preferences are built and that is used to determine relevance of content to the user profiles. However, this extension and coverage was not exhaustive, to stay in line with the primary LUMO design decision: maintain the ontology lightweight yet at the same time meaningful from a user perspective.

Therefore, over 100 new classes were added, mostly under the “Agent”, “Tangible” and “Intangible” categories. These were based mostly on the updated schema of the DBPedia ontology (version 2014) [LEH14] and in part for including some concepts within YAGO2 [BIE13] relevant to the news scenario.

2) New axioms

New concepts brought along the need to model new non-taxonomical concept relations. To this end, several new universal quantification⁴ axioms were added, in order to maintain connections under the “Agent”, “Tangible” and “Intangible” categories with related “Topics”, based on the “hasTopic” and “hasSubtopic” LUMO properties.

3) Revised semantics

Semantics of existing concepts in v1 have been revised and updated to better reflect their hierarchical and non-taxonomical relations in the ontology. E.g. concepts in the “Topics” subhierarchy were deemed as belonging under the “Tangible” or “Intangible” subhierarchies, but connected to their related topics with the “hasTopic” and “hasSubtopic” relations. An example can be seen in Figure 3. In addition, some concepts of v1 were omitted as too specific in the interest of maintaining the ontology lightweight.



Figure 3: Literary work products (article, novel, etc) were moved under the Intangible > Work category in v2, as opposed to under the Topic > Literature category in v1. They were related to Literature via the hasSubtopic property in a corresponding axiom

⁴ I.e., relations of the form: $entity \sqsubseteq \forall has(Sub)Topic.topic$, where $entity$ is subsumed by the Agent, Tangible and Intangible concepts/categories of LUMO and $topic$ is subsumed by the Topics concept/category of LUMO. Cf. D4.4, ch. 2.1.2 for more details.

4) New object properties

In the interest of accommodating the needs of the explicit profiling branch (ch. 8-11) of WP4, which demands for extensive concept interconnections at the schema level, almost 30 new object properties were added in the ontology, with corresponding semantics and domain/range attributes assigned to them. Figure 4 illustrates the object properties in v2, and an example of the semantics, domain and range of property “authorOf”.



Figure 4: Object properties in LUMO v2 and the semantics, domain and range of property “authorOf”

The update in version 2 brings LUMO to 929 classes, 38 object properties and more than 130 universal quantification axioms.

4.1.1 LUMO-arts

In the interest of accommodating the LinkedTV cultural scenario (TKK scenario), an expansion of LUMO was engineered to provide more detailed coverage of the arts and artefacts domain. In order to maintain a reduced concept space, this expansion was modeled separately from the core, more generic, LUMO v2, but was built as an extension of the core hierarchy.

This expansion was heavily based on the Arts & Architecture Thesaurus (AAT)⁵. The recent release of AAT as LOD enabled to advise a well-formed hierarchy, while we adjusted the semantics to the core LUMO v2 schema.

⁵ <http://www.getty.edu/research/tools/vocabularies/aat/about.html>

The TKK scenario partners (S&V) have carefully examined the AAT and, out of the vast information in the vocabulary, defined several facets that were deemed the most relevant to describe TKK content. To this end, LUMO-arts models details on materials, clothing, furnishing, art styles etc which outline the contents of the TKK scenario. An extract of the ontology can be seen in Figure 5. The next version of LUMO-arts will delve deeper into the TKK scenario requirements and, based on AAT, will expand more on the requirements of the context-aware artistic scenario.

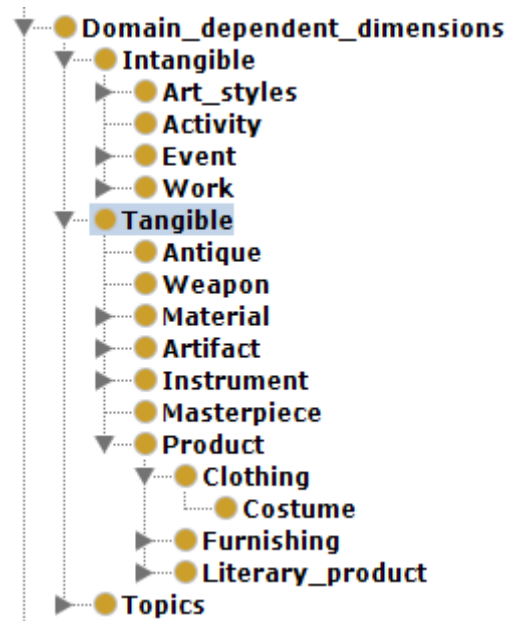


Figure 5: Extract of the LUMO-arts expansion

5 Implicit user interactions

The user interactions can be implicitly captured from the user behaviour. The features which are captured come from a sensor watching the users (like the Microsoft Kinect sensor [KIN10]) or from the player (logging the user actions). All those interactions are linked to a video shot and sent to the GAIN module which process them into a value of “interest” quantifying the way the user is interested (or not) in the current video shot.

5.1 Behavioural features extraction

As already stated in previous deliverables, the use of implicit and contextual features which can be extracted by analysing the behaviour of the people watching the TV such as the number of people watching, their engagement from the body language, their emotions or the viewing direction bring crucial additional information for content personalization and adaptation.

One issue in using cameras to watch people while they view TV is of course the degree of acceptance and the ethical issues which are brought in there by this situation. Nevertheless, the camera or the depth sensor are not recorded by any means and all the processing is made in real time. Only precise features which can be controlled are sent to the system. Moreover, the degree of acceptance of cameras watching people is higher and higher since the Xbox systems using the Kinect sensor invaded a lot of houses for game purposes. Extension from games to TV is very natural and already happened since Microsoft proposed an add-on for XBOX One to watch TV and use the Kinect gesture recognition capabilities⁶. Also Google just acquired Flutter⁷, a company which provides webcam-based gestures and this feature could be added to Chromecast. Apple also bought Primesense⁸, the company which built up the first Kinect version, and this one could be used as a new feature for Apple TV. Finally classical TV manufacturers like Samsung already propose cameras for communication or gesture control. This trend might be a first step in the use of cameras and depth sensors for TV in the future with further steps which are to go beyond gestural controls (already available on the market) where specific features will be acquired to enhance and personalize the TV experience. Viewer acceptance of cameras inside homes is also growing from simple games to TVs. In this context, the work on the Interest/Context tracker based on the first version of the Kinect sensor is very important as new contextual features will need to be used by the WP4 pipeline to enhance personalization. WP4 will test the information brought by such kind of future TV sensors and see how much it can enhance the TV experience personalization.

⁶ <http://www.polygon.com/2014/8/7/5979055/xbox-one-digital-tv-tuner-europe>

⁷ <http://www.bbc.com/news/technology-24380202>

⁸ <http://www.forbes.com/sites/anthonykosner/2013/11/26/apple-buys-primesense-for-radical-refresh-of-apple-tv-as-gaming-console/>

More details about the Kinect sensor used for the Interest/Context tracker are available in previous deliverables of WP4 (as D4.4). The Kinect sensor is a low-cost depth and RGB camera. It contains two CMOS sensors, one for the RGB image (640 x 480 pixels at 30 frames per second) and another for the infrared images from which the depth map is calculated, based on the deformation of an infrared projected pattern. The depth sensor has an optimal utilisation in a range of 1.2 meter (precision better than 10 mm) to 3.5 m (precision better than 30 mm) [GON13].

The main use of the Kinect is the user tracking. It allows tracking up to 6 users and giving a skeletal tracking up to two users to follow their actions in the Microsoft SDK 1.8 which we used. It is possible to detect and track several points of the human body and reconstruct a “skeleton” of the user (see Figure 6). Skeletal Tracking is able to recognize users standing or sitting (Figure 7), and it is optimized for users facing the Kinect; sideways poses imply higher chances to have tracking loss or errors.

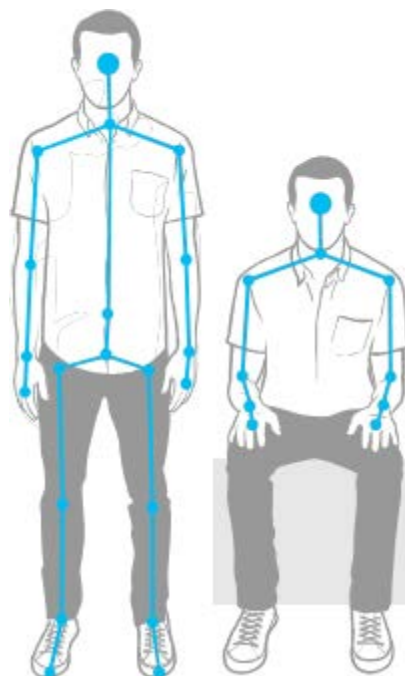


Figure 6: User tracking for default mode (left) and seated mode (right).

To be correctly tracked, the users need to be in front of the sensor, making sure the sensor can see their head and upper body. There tracking is possible when the users are seated. The seated tracking mode is designed to track people who are seated on a chair or couch, only the upper body is tracking (arm, neck and head). The default tracking mode, in contrast, is optimized to recognize and track people who are standing and fully visible to the sensor, this mode gives legs tracking. We thus used for the interest/context tracking the standing version of the skeleton as in a TV configuration, there are a lot of chances to have hidden legs.

We can estimate user's engagement by his sitting position. From the skeleton we use extract torso orientation to determine whether the user is leaning forward or backward.

Other features are compute based on the skeleton features: the child/adults discrimination and the attention and the user interest. Regarding the age, the distance between the two shoulders is used and it is compared to a statistical set of anthropometry of child body size. The development of the body, and shoulder width, specifies whether the person is closer to the body of a child or of an adult (Figure 11).

In addition to the skeleton features, Microsoft provides a Face Tracking module with the Kinect SDK since the version 1.5. These SDKs can be used together to “create applications that can track human faces in real time” To achieve face tracking, at least the upper part of the user's Kinect skeleton had to be tracked in order to identify the position of the head (Figure 8).

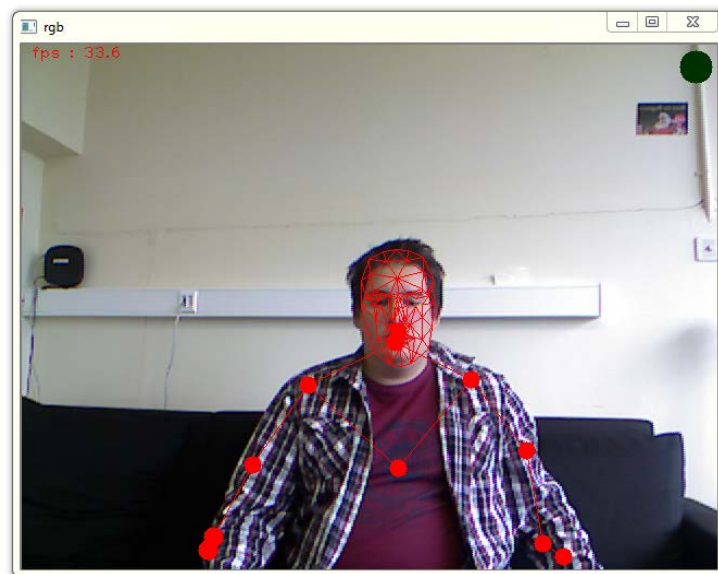


Figure 7: Seated tracking with face tracking.

Based on the face tracking, it's also possible to extract facial feature to obtain more information about the users faces. The Microsoft SDK gives 6 facial features and they are called “animation units”. The tracking quality may be affected by the image quality of the RGB input frames (that is, darker or fuzzier frames track worse than brighter or sharp frames). Also, larger or closer faces are tracked better than smaller faces. The system estimate the basic information of the user's head: the neutral position of their mouth, brows, eyes, and so on. The Action Units represents the difference between the actual face and the neutral face. Each AU is expressed as a numeric weight varying between -1 and +1.

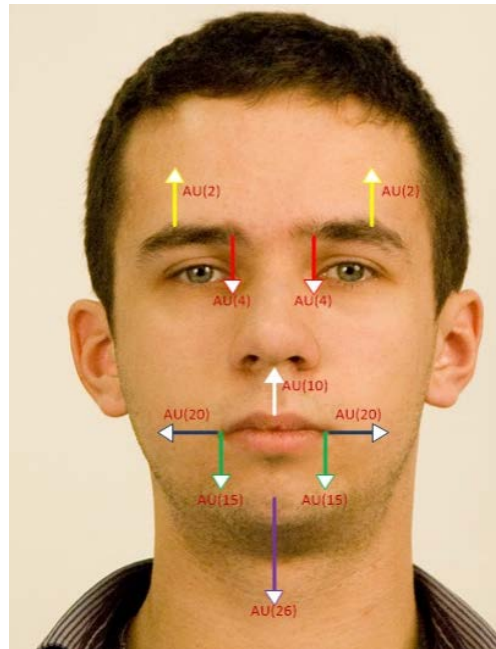


Figure 8: The different action units given by the Microsoft SDK and their position on the face [WYR]

Based on these action units, it's possible to compute and discriminate basic expressions. Nevertheless, in real-life TV conditions, the sensor is not precise enough to provide usable information about the precise emotions. We managed to provide relatively reliable information about the discrimination between the neutral pose and "non-neutral" pose (Figure 9). The system can provide events on emotion changes without being precise enough to provide the exact emotion of the viewer.



Figure 9: Action units on the left and expression discrimination on the right

Finally, the last and most important extracted feature is the head direction which is very close to the eye gaze (see previous deliverable of WP4, D4.4 for more details).

The Get3DPose() method returns two tables of three float numbers. The first one contains the Euler rotation angles in degrees for the pitch, roll and yaw as described in Figure 10, and the second contains the head position in meters. All the values are calculated relatively to the sensor which is the origin for the coordinates.

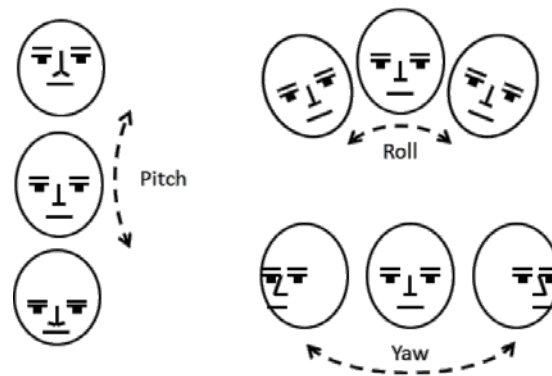


Figure 10: Three different degrees of freedom: pitch, roll and yaw [FAC].

The technique used to estimate the rotations and facial features tracking of the head is not described by Microsoft, but the method uses the RGB image and depth map. The head position is located using 3D skeleton only on the depth map. The head pose estimation itself is mainly achieved on the RGB images. Consequently, the face tracking hardly works in bad light conditions (shadow, too much contrast, etc.). This drawback will be solved in Kinect version 2, where the head tracking is made using the depth map and the infra-red image, much less sensitive to illumination changes.

Based on the head pose estimation, it is possible to know where the user is looking (main screen, second screen or elsewhere). Based on the duration of a screen watching by the user, a measure of attention is given [HAW05]:

- Gaze not taken into account if shorter than 1.5 seconds
- Orienting attention (1.5s to 5s)
- Engaged attention (5s to 15s)
- Starring (more than 15 s)

In addition to those single-user features, if two viewer look at the same screen, the attention mode becomes “joint attention” which might show mutual interest for the content display on the screen.



Figure 11: User face windows with head pose estimation and age estimation.

The features described here are summarized in the table of section 5.2. A first validation test was led on the head direction feature and it shows best robustness compared to other state-of-the-art available methods. This validation is detailed in the next section.

5.1.1 Head direction validation: the setup

For the validation of the head direction feature, the results obtained by the Kinect sensor are compared with an accurate measurement of the head movements (Figure 12). This ground truth was obtained thanks to an optical motion capture system from Qualisys [QUA]. The used setup consists of eight cameras, which emit infrared light and which track the position of reflective markers placed on the head. Qualisys Track Manager Software (QTM) provides the possibility to define a rigid body and to characterize the movement of this body with six degrees of freedom (6DOF: three Cartesian coordinates for its position and three Euler angles - roll, pitch and yaw - for its orientation).



Figure 12: The user is placed in front of the TV, and covers his head with a hat with infrared reflectors for the Qualisys system.

The Qualisys system produces marker-based accurate data in real-time for object tracking at about 150 frames per second. The infrared light and marker do not interfere with RGB image and with infrared pattern from the Kinect. The choice of Qualisys as reference has been done especially in order to compare markerless methods without interferences. This positioning is shown on Figure 13. The angles compute from the different methods are the Euler angles.



Figure 13: Setup for facial tracking recording. The Kinect for the head tracking algorithm is marked in green.
 We can also see the infrared reflectors for the Qualisys on the TV corners.

We made several recordings with 10 different candidates. Each one performs the same head movement sequence (verticals, horizontals, diagonals and rotations) at 5 different distances from the screen: 1.20m, 1.50m, 2m, 2.5m and 3m. Movements performed are conventional movements that people make when facing a TV screen (pitch, roll, and yaw; combination of these movements; slow and fast rotation). Six of the candidates had very light skin, others had darker skin color. Some of the candidates had beards and others not.

5.1.2 Head direction validation: some results

After having synchronized the results obtained by the Kinect SDK and the reference, as the sampling frequencies are different, we have interpolated reference values to obtain points at the same moments for the two systems. To make the comparison with the reference computed by the Qualisys, we use two metrics: the Root Mean Square Error (RMSE) and the correlation (CC).

The correlation is a good indicator used to establish the link between a set of given values and its reference. It is interesting to analyse the average candidates' correlation value obtained for each distance from the TV screen. If the correlation value is equal to 1, the two signals are the same. If the correlation is between 0.5 and 1, we consider a strong dependence. The 0 value shows that the two signals are independent and de -1 value correspond to the opposite of the signal. Figure 14 shows the correlation for pitch, Figure 15 for yaw and Figure 16 for roll. The curve from KinectSDK is compared with the reference obtained with the Qualisys system. The pitch, roll and yaw are described in Figure 10.

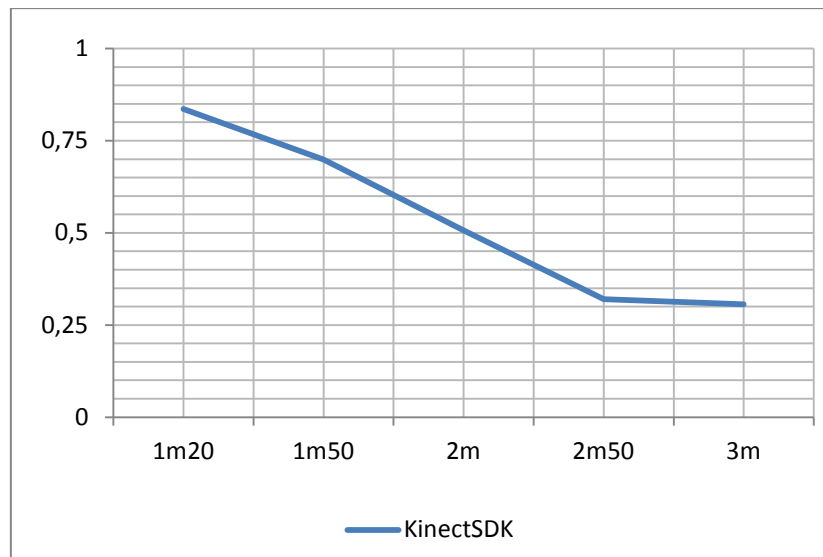


Figure 14: Mean correlation with the reference for the pitch depending on the distance from TV.

On Figure 14, we observe that the pitch (up-down movements) of the KinectSDK has a good correlation (0.84) at a distance of 1m20. And it decrease with the distance under the correlation value of 0.5 from 2 meters.

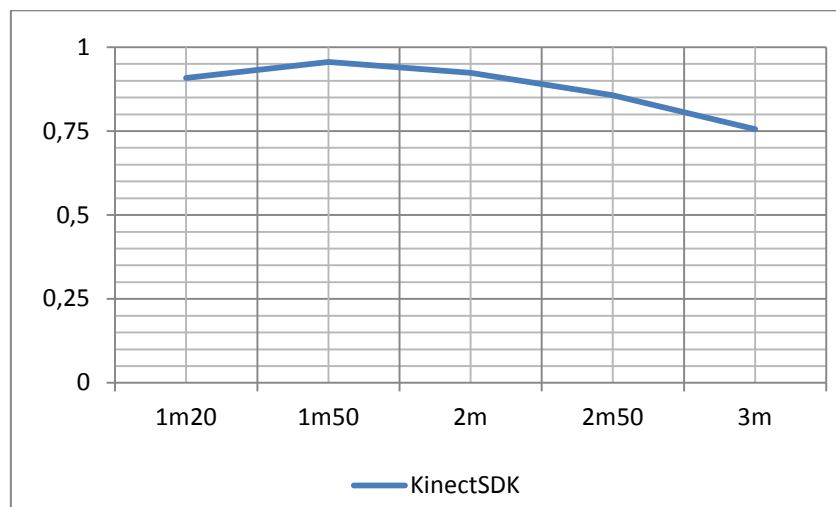


Figure 15: Mean correlation with the reference for the yaw depending on the distance from TV.

For the second angle, the yaw (right-left movement), we have on the Figure 15 good results for the KinectSDK with values upper than 0.9 for 1m20, 1m50 and 2m. Then de values decrease from 0.85 for 2m50 to 0.76 for 3m. We can consider that the values of the correlation for the KinectSDK are pretty good. Thus the yaw measure is much more reliable than the pitch measure.

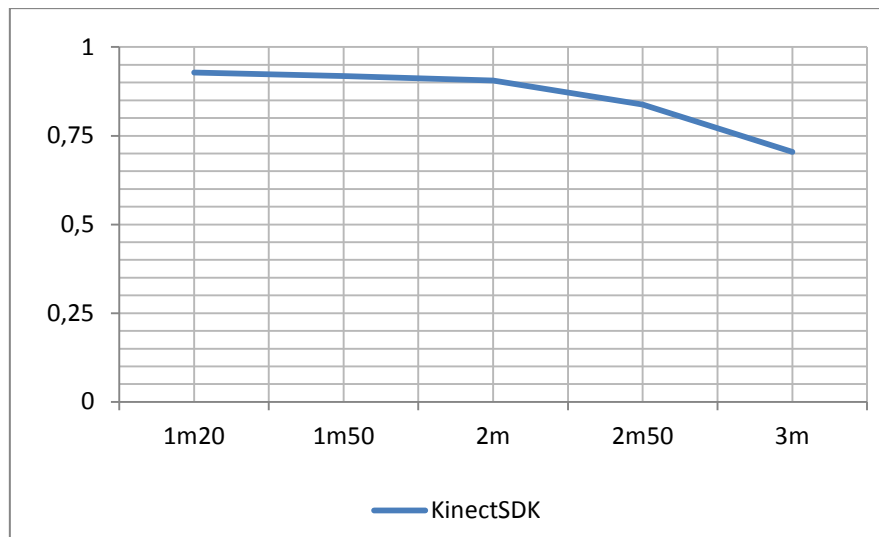


Figure 16: Mean correlation with the reference for the roll depending on the distance from TV

The KinectSDK has good correlation as the roll curve (0.93 to 0.7) on the Figure 16. After watching the correlation values, it is also interesting to look at the mean error made by each system. Indeed, a method with a big correlation and low RMSE is considered very good for head pose estimation. Figure 17 shows the RMSE for pitch, Figure 18 for yaw and Figure 19 for roll.

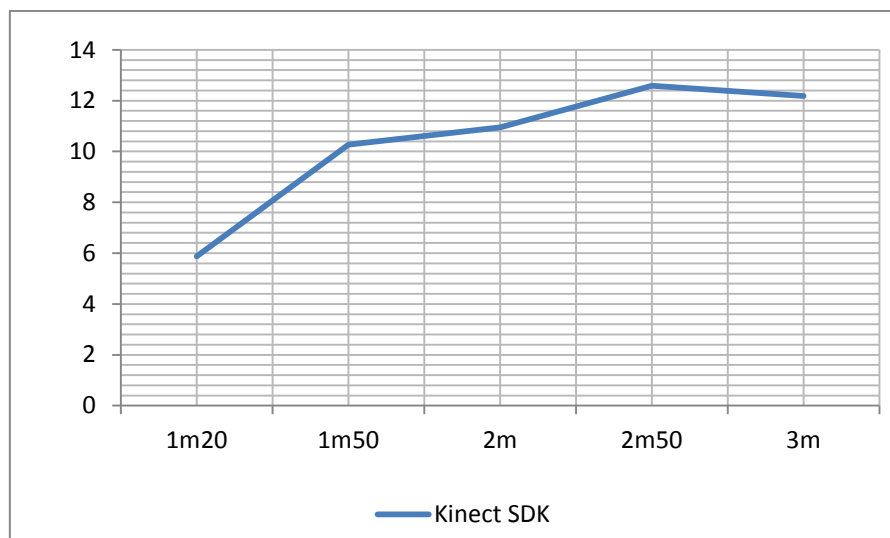


Figure 17: Mean RMSE (in degrees) for the pitch depending on the distance to TV.

We observe on Figures 17 to 19 that the RMSE obviously increases with distance to the TV (more precisely to the Kinect sensor located on the TV). For the pitch (Figure 17), the KinectSDK is good at 1m20 with a RMSE of 5.9 degrees. The error grows with the distance to 12 degrees after 2.5 m.

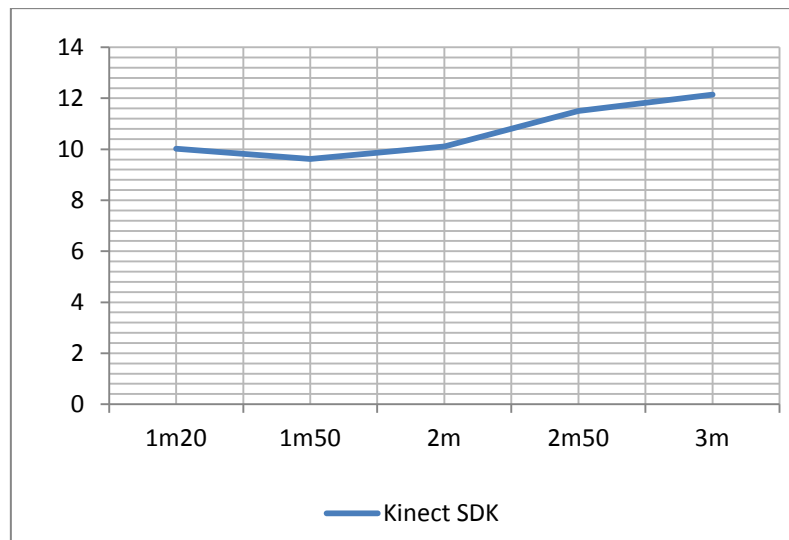


Figure 18: Mean RMSE (in degrees) for the yaw depending on the distance to TV.

On the yaw, we observe on Figure 18 higher mean error (from 10 to 12), but this error growth with the distance is very small.

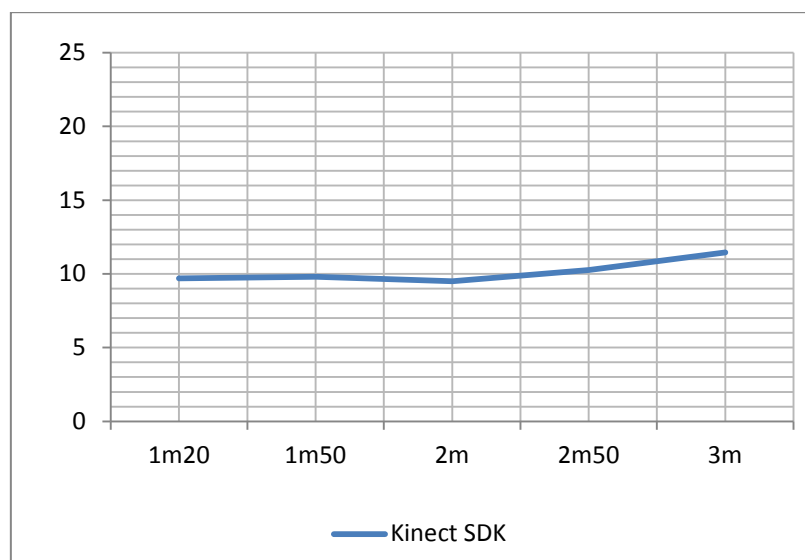


Figure 19: Mean RMSE (in degrees) for the roll depending on the distance.

In the case of roll (Figure 19), the RMSE the KinectSDK gives values around 10 degrees with a smaller error at 2m for KinectSDK.

While it is possible to extract the roll has less interest in the LinkedTV Project where mainly yaw and pitch are used. The correlation is good and place the LinkedTV interest tracker on the top of the state of the art in the field. Moreover, all those values will become even better when using the Kinect One, second version of the Kinect sensor.

5.2 Communication of behavioural features with the LinkedTV player

The Interest Tracker software offers several measures of contextual and behavioural features, such as the number of people in the room, their position or basic expression analysis. All these features which are computed in real time without any need of data recordings can be sent to the LinkedTV player. The features are computed at an average frequency of 30 per second. To avoid sending too many messages to the player, only feature variations (events) will be sent through the network.

The player gets the messages and adds the current video ID and time and forward the event to InBeat via GAIN API for user profiling and personalization. Several network protocols have been implemented in the software. We implemented HTTP (POST and GET) and websocket communication with respectively cURL⁹ and easywebsocket¹⁰ libraries. Another version of the Interest Tracker has been developed for artistic scenarios and this one uses the OSC protocol (see section 2.2 and D6.4 for more details). The list of features and their format for HTTP (GET) and websocket protocol are detailed in Table 3.

Table 3: List of behavioural and contextual features available from Interest Tracker. The features can be sent through HTTP or websocket protocols.

| Feature | Name | Value |
|--|------------------------------|--|
| Number of detected people in front of the TV | Recognized_Viewers_NB | 0, 1, 2 |
| Websocket: <code>{"interaction":{"type":"context"},"attributes":{"action":"Recognized_Viewers_NB","value":[0,1,2]}}</code> | | |
| HTTP Get: <code>http://baseUrl?Recognized_Viewers_NB=[0,1,2]</code> | | |
| The screen the user is currently watching (for each user) [HAW05] | Viewer_Looking | 0 = viewer does not look to any screen (maybe someone called him or he is simply doing something else in front of the TV) 1 = viewer looks to the main screen from more than 1.5 seconds (if less than 1.5 s nothing is sent as this corresponds with the hazard peak or the monitoring looks). From 1.5 seconds we have "orienting" looks. |

⁹ <http://curl.haxx.se/>

¹⁰ <https://github.com/dhbaird/easywsclient>

| | | |
|---|-----------------------------|---|
| | | <p>2 = main screen from more than 5 seconds. From 5 seconds we have "engaged" looks.</p> <p>3 = main screen from more than 15 seconds. From 5 seconds we have "staring" looks.</p> <p>4 = second screen from more than 1.5 seconds</p> <p>5 = second screen from more than 5 seconds</p> <p>6 = second screen from more than 15 seconds</p> |
| Websocket: <pre>{ "interaction": { "type": "context", "attributes": { "action": "Viewer_Looking", "value": [0, 1, 2, 3, 4, 5, 6], "confidence": [0..1] }, "user": { "id": [1, 2, 3, 4, 5, 6] } } }</pre> HTTP Get: <pre>http://baseUrl?UserID=[userID]&Viewer_Looking=[0,1,2,3,4,5,6]</pre> | | |
| There are two people in front of the TV and both are looking at the same screen | Viewer_Joint_Looking | 1 if true, 0 else |
| Websocket: <pre>{ "interaction": { "type": "context", "attributes": { "action": "Viewer_Joint_Looking", "value": [0, 1], "confidence": [0..1] } } }</pre> HTTP Get: <pre>http://baseUrl?Viewer_Joint_Looking=[0,1]</pre> | | |
| There are only adults in front of the TV | Viewer_adults | 1 if true, 0 else |
| Websocket: <pre>{ "interaction": { "type": "context", "attributes": { "action": "Viewer_adults", "value": [0, 1], "confidence": [0..1] } } }</pre> HTTP Get: <pre>http://baseUrl?Viewer_Adults=[0,1]</pre> | | |
| Basic emotion analysis (for each user) | Viewer_emotion | 0 if neutral, 1 else |
| Websocket: <pre>{ "interaction": { "type": "context", "attributes": { "action": "Viewer_emotion", "value": [0, 1], "confidence": [0..1] } } }</pre> HTTP Get: <pre>http://baseUrl?UserID=[userID]&Viewer_Emotion=[0,1]</pre> | | |
| User lean forward/backward | Viewer_engagement | <p>0 = lean backward</p> <p>1 = lean forward</p> <p>2 = unknown (HTTP only)</p> |

WebSocket:

```
{"interaction":{"type":"context"},"attributes":{"action":"Viewer_engagement","value":[0,1],"confidence":[0..1]}}
```

HTTP Get:

```
http://baseUrl?UserID=[userID]&Viewer_Engagement[0,1,2]
```

In the examples, the value of *baseUrl* can be any valid URL which will handle such messages in the player interface. For testing and debugging, we used <http://httpbin.org/get>, an address which returns exactly the data it receives.

5.3 Communication of LinkedTV player with GAIN/InBeat module

Communication between the LinkedTV player and the GAIN module of InBeat is performed by using REST API calls from player to the GAIN module. The API is designed to handle multiple types of interactions including standard player actions (e.g. play, pause, bookmark, view of enrichment ...), user actions (login, bookmark ...), platform specific actions (add or remove second screen ...) and contextual features (e.g. viewer looking, number of persons ...). In this section, we will describe new version of the API and the communication format for all previously described actions.

The communication was tested using a Noterik player simulator, which emulates user actions in the player, which generate respective calls to the GAIN API.

5.3.1 API description

The first version of the GAIN API was described in D4.2 – *User profile schema and profile capturing*. Since GAIN became part of InBeat [KUC13] there has been a change in the base URL of the API, and we also updated the exchange format in order to support more types of interaction.

Table 4: Description of REST service used for tracking of interactions

| Track interaction | |
|---------------------|---|
| Description | POST /gain/listener |
| HTTP Method | POST |
| Content-Type | application/json |
| URI | http://inbeat.eu/gain/listener |
| cURL | curl -v --data @data.json http://inbeat.eu/gain/listener |
| data.json | Format is described in following section. |
| Status codes | 201 – Created 400 – Bad request |

5.3.1.1 Player Actions

There is no change in the format for actions generated by user's operation of the remote control (or player control buttons or gestures in general). All such events should be assigned value `player` in the `category` attribute, and the `action` attribute is set to one value from the enumeration of possible actions. The `location` attribute specifies the time passed since the start of the video.

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "attributes": {
    "category": "player",
    "action": "pause",
    "location": "32"
  }
}
```

Figure 20: pause action performed by Rita at 32s of video

Figure 20 presents an example of action “pause” performed by Rita at 32s of video. Examples of action “bookmark” of a specific chapter and “view” of enrichment presented by player to the user are described in Figure 21 and 22 respectively.

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "attributes": {
    "category": "player",
    "action": "bookmark"
  }
}
```

Figure 21: bookmark of specific chapter performed by Rita

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/entity/8c219677-47ea-4964-815d-add91320f234",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "attributes": {
    "category": "player",
    "action": "view"
  }
}
```

Figure 22: view action of presented enrichment performed by Rita

5.3.1.2 User Actions

User actions are actions performed by a user that are not connected to any multimedia content (in contrast to player actions). For user actions, the `objectId` attribute is set to empty string value. Each type of interaction is specified by `category` and `action` attributes.

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "attributes": {
    "category": "user",
    "action": "login"
  }
}
```

Figure 23: User Action Example: user Rita logged in

5.3.1.3 Application specific actions

Application specific actions are actions invoked by the player. Figure 22 presents the typical example for an *application specific action*, which is opening of a new screen by the user. The format is similar to *User action*, the only one difference is in the `category` and the `action` attributes. These actions are also not connected to specific multimedia content.

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "attributes": {
    "category": "screen",
    "action": "new"
  }
}
```

Figure 24: Application Specific Actions Example:
user Rita opens a new screen (TV, second screen ...)

5.3.1.4 Contextual Features

Contextual features are completely a new type of interaction. This type is identified by value `context` in the `type` attribute of the communication format. Combination of `action` and `value` attributes specify the type of the contextual feature. The example depicted on Figure 23 describes user Rita, who started looking (“Viewer_looking”) at a second screen device

("value=2") at the 15th second of video. Other contextual features and their possible values are described in D4.4.

```
{
  "accountId": "LINKEDTV-TEST",
  "type": "context",
  "userId": "/domain/linkedtv/user/rita",
  "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
  "attributes": {
    "action": "Viewer_Looking",
    "value": "2",
    "location": "15",
    "confidence": "1"
  }
}
```

Figure 25: Rita started looking at second screen device at 15th second of video

5.4 InBeat

The InBeat platform is composed of three main modules: *GAIN* (General Analytics INterceptor), module for tracking and aggregating the user interactions, *Preference Learning* module for analyzing user preferences, and *Recommender System* module providing on-demand recommendation.

All components expose independent RESTful APIs, which allow to create custom workflows.

Within the scope of this deliverable we focus on the GAIN module, which processes the interactions sent by the LinkedTV player for a given user and generates aggregated output, which is consumed by further WP4 components to build user profiles. GAIN logic combines multiple interest clues it derives from the interactions into a single scalar *Interest* attribute. GAIN aggregates all types of interactions including player actions (play, bookmark, view of enrichment ...) and contextual features mainly provided by the Kinectbased Interest/Context tracker. Similarly, GAIN also performs the aggregation of the content of the shot, based on the entity annotation it receives either along with the interactions from the player or from the platform, into a feature vector usually corresponding to DBpedia or NERD concepts.

Table 5: GAIN output example. prefix *d_r_* indicates that the feature corresponds to a DBpedia resource from the English DBpedia, the prefix *d_o_* to a concept from the DBpedia Ontology. For DBpedia in other languages, the prefix is *d_r_lang_*.

| User_id | d_r_North_Korea | ... | d_o_SoccerPlayer | ... | c_userlooking | ... | interest |
|---------|-----------------|-----|------------------|-----|---------------|-----|----------|
| 1 | 0 | | 0.9 | | 1 | | 0.3 |
| 2 | 1 | | 1 | | 0 | | 1 |

GAIN module became part of the InBeat service (see next section for more details). However, GAIN has the same purpose and goals – tracking and aggregating user interactions. GAIN uses a specific JSON format (details in the previous section) as input; the main output is a tabular form of aggregated data. In this section we describe new features supported by the latest release – support for contextualization, import of annotations for media content and tabular format serialization. The developments in the GAIN module are reported in Section 5.4.1 and 5.4.2.

The InBeat Preference learner module is a standalone component which wraps EasyMiner and LISp-Miner as the underlying learning stack. We have experimented also with alternative learning backends, but the EasyMiner/LISp-Miner stack seems to provide the best experience in terms of web-based user interface and preserving compatibility with existing LinkedTV components.

The InBeat Recommender systems module (InBeat RS) has been developed simultaneously to other components of InBeat (GAIN as component for collecting and aggregating user feedback, Preference learning component that learns user preferences). This component is not part of main workflow of LinkedTV platform, but it was used as a development tool to test GAIN and PL modules, and to participate in benchmarking contests. The developments in the InBeat-RS module are reported in Section 5.4.3.

5.4.1 Import of annotations for media content

In order to reduce communication between GAIN and the LinkedTV platform and to overcome issues with updating annotations of media on the GAIN side, we designed the approach for sending annotations along with interactions. Figure 24 demonstrates the format for sending description of the object (chapter) the user interacted with. The LinkedTV player can provide annotation of the content played with entities, since the entity information is available in the player. GAIN module supports attachment of entities to interactions that are sent from the player.

This approach should solve issues with updating annotations in GAIN module. GAIN needs annotations on its input to perform the aggregations. If there will be no annotation for played media content in internal storage of GAIN, it will lead to incorrect aggregations or delays caused by on demand fetch of data from the LinkedTV platform. Each annotation needs to be sent only once during the viewer session since it is cached in the GAIN module. This approach allows to reduce the amount of data communicated between the player and GAIN. Another advantage is in temporal aspects of annotations – for next session can the platform provide updated annotations for specific media content. This approach allows active adaptation to new or updated content.

```

{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "object": {
    "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
    "entities": [
      {
        "source": "thd",
        "lod": "http://dbpedia.org/resource/Museum_Island",
        "type": "http://dbpedia.org/ontology/Place",
        "label": "Museum Island",
        "typeLabel": "Place",
        "entityType": "named entity",
        "confidence": 0.14,
        "relevance": 0.29
      }
    ]
  },
  "attributes": {
    "category": "player",
    "action": "pause",
    "location": "32"
  }
}

```

Figure 26: Example of annotation send from player along with event

5.4.2 Support of contextualization

In this section, we describe progress on the support for contextualization. Contextual features supported in GAIN were introduced in the Deliverable D4.4. However, the communication format did not account for the following situation: the viewer is watching screen without any interactions or changes in context.

In this case, the tracking module does not have any information about the content that was on the screen, since no events that would have this information attached are raised by the player, and cannot provide the correct output. For this specific situation we designed “keepalive” interaction type that will provide data and descriptions for each shot. This interaction is raised by the player even if there is no explicit user action or change in context to notify GAIN about the content being played. GAIN interprets this type of interaction as a simple “copy previous state” command. Figure 25 provides description of data format implemented in GAIN.

Example: Viewer Rita would like to watch media content with 1...N. She presses “play” button and attention tracker recognizes that she is watching the screen. Both interactions are sent to GAIN as an interaction with event “Play” and context “Viewer_looking=1”. She is watching the screen carefully without any interactions for all the remaining shots. Without support of “keepalive” interaction, GAIN could be able to derive interest clues only from the first shot and its annotations. On the other hand, when the player sends “keepalive” for each remaining shot, GAIN propagates “Viewer_looking=1” context to all these shots. Each of these “notified” shots is afterwards included to the final output and interest value can be calculated based on the propagated values.

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "object": {
    "objectId": "http://data.linkedtv.eu/chapter/b82fb032-d95e-11e2-951c-f8bdfd0abfbd",
    "entities": [
      {
        "source": "thd",
        "lod": "http://dbpedia.org/resource/Museum_Island",
        "type": "http://dbpedia.org/ontology/Place",
        "label": "Museum Island",
        "typeLabel": "Place",
        "entityType": "named entity",
        "confidence": 0.14,
        "relevance": 0.29
      }
    ]
  },
  "attributes": {
    "action": "keepalive"
  }
}
```

Figure 27: Example of "keepalive" event for propagation of context

5.4.3 InBeat Recommender System

Recommendations are one of key features of personalization provided by LinkedTV platform. In this section we will briefly introduce and describe InBeat Recommender System (InBeat RS) that is available in the platform. InBeat RS consumes inputs from both GAIN and Preference Learning modules and provides recommendation as its outputs. The InBeat Recommender Systems participated in the RecSys'13 News Recommender Challenge (2nd place) and the CLEF NewsReel Challenge'14 (3rd place).

5.4.3.1 Components

The Interest Beat (InBeat) recommender consists of several components described below.

Recommendation Interface module obtains requests for recommendation, which comprises the user identification, identification of the currently playing mediafragment (seed mediafragment), and description of user context. As a response, the module returns a ranked list of enrichment content.

Recommender algorithms module covers set of algorithms that can be used in LinkedTV platform.

BR Engine module finds the rules matching the seed content vector, aggregates their conclusions, and returns a predicted single value of interest.

BR Ranking module combines the estimated user interest in the individual enrichment content item produced by the BR Engine with the importance of the entity, for which the enrichment item was found.

5.4.3.2 Recommender Algorithms

InBeat RS contains implementations of several baseline algorithms and experimental implementations of specific algorithms that fit to the LinkedTV workflow. InBeat RS can provide recommendations based on following algorithms:

- Most recent – recommendations based on simple heuristic that selects a set of newest items from all available candidates.
- Most interacted – only top “viewed” items are selected.
- Content-based similarity – a set of most similar items to the item that user is currently viewing.
- Collaborative filtering – both user-based and item-based versions are available.
- Matching Preference Rules with Content
- Rule-based similarity of users and their contextual
- Ensemble – combining of algorithms. See next sections for more details.

The most recent and most interacted methods are described in [KUC13], the rule based similarity algorithm is described in [KUC14]. The details of the “Matching preference rules with content” algorithm are given in Subs. 5.4.3.3., and the details of the ensemble method in Subs. 5.4.3.4.

5.4.3.3 In-Beat: Matching Preference Rules with Content

Preference rules learned with EasyMiner via the In-Beat Preference Learner (see D4.4 Section 4.2.2) can be directly used to rank relevant enrichment content according to user preferences, in addition to further processing of these rules by SimpleLearner. In this section, we introduce In-Beat Recommender, an experimental recommender, which serves for direct recommendation based on preference rules learnt with EasyMiner and stored in the Rule Store.

5.4.3.4 InBeat: Ensemble as combination of multiple recommenders

Existence of different recommender algorithms that can have different quality in different situations leads to idea of combining algorithms in order to get better overall quality. One algorithm can have better quality of recommendations in the morning, since users can be interested in the newest content in the morning. The second algorithm can provide better recommendations for youth in the evening.

InBeat RS deals with combining using ensemble based on Multi-Armed Bandit algorithm [KUL00]. The core of ensemble uses probabilistic distributions to decide which algorithm is probably the best for the specific situation. At the beginning all algorithms have the same probability of selection. One of them is randomly selected and its recommendations are presented to the user. If the user chooses one of the recommended items, it is interpreted as positive case. The probability associated with the selected algorithm is increased and for all others, it is decreased. User can also provide negative feedback. The probability of the algorithm that provided the recommendation is then decreased.

The selection of the recommendation algorithm is affected by the modified probabilities. A more successful algorithm has a higher probability of selection. Since one algorithm can be successful at the beginning and its quality can rapidly get down later, ensemble also deals with the level of conservativeness. Ensemble supports different strategies in order to change speed of adaptation to new situation. It allows forgetting previous states and evolution of ensemble in time.

6 User model

This chapter will examine the advances in the final user modeling service based on WP4's Linked Profiler¹¹ service and present the user profiles created based on the LinkedTV scenarios for year 3 (Chapter 2).

6.1 Linked Profiler contextual adaptation

As described in D4.4, ch. 4, the results of the GAIN and PL modules are fed into the Linked Profiler module via InBeat's RESTful services, where the user interests (positive preferences) and disinterests (negative preferences) are mapped to LUMO via the LUMO wrapper utility. Similarly, contextual features passed onto GAIN are mapped to classes under the "Context_dimensions" subhierarchy of the LUMO ontology. These mappings were manually created to bring context feature keywords and expressions into the LUMO concept space. The correlations between features and classes can be seen in D4.4, table 5.

An example of preferences transported to the user profile of user Nina, based on a session where she is bookmarking a media item about the *Deutsche Historisches Museum* while viewing content along with her *children* can be seen in Tables 6 and 7.

Table 6: GAIN interaction for Nina: *bookmarking* a media item while in the company of her *kids*

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/media/b82fb031-d93f-14b3-768c-f3bdfdl1dabca?start=22.96&end=25.62",
  "userId": "/domain/linkedtv/user/rita",
  "type": "event",
  "attributes": {
    "category": "video",
    "action": "bookmark",
    "value": "",
    "location": "24",
    "client": {
      "type": "GAIN",
      "version": 1
    }
  }
}
```

¹¹ Cf. D4.4, ch. 4

```
{
  "accountId": "LINKEDTV-TEST",
  "objectId": "http://data.linkedtv.eu/media/b82fb031-d93f-14b3-768c-f3bdfdl1dabca?start=22.96&end=25.62",
  "userId": "/domain/linkedtv/user/rita",
  "type": "context",
  "attributes": {
    "action": "kids",
    "value": "1",
    "location": "24",
    "client": {
      "type": "GAIN",
      "version": 1
    }
  },
  "object": {
    "objectId": " http://data.linkedtv.eu/media/b82fb031-d93f-14b3-768c-f3bdfdl1dabca?start=22.96&end=25.62",
    "attributes": {
      "start": 12345
    },
    "entities": [
      {
        "source": "thd",
        "lod": "http://dbpedia.org/resource/Deutsches_Historisches_Museum",
        "type": "http://dbpedia.org/ontology/Museum",
        "label": "Deutsches Historisches Museum",
        "typeLabel": "Museum",
        "entityType": "named entity",
        "confidence": 0.72,
        "relevance": 0.83
      }
    ]
  }
}
```

Table 7: Nina's interaction serialized in her (previously empty) user profile

```
(IMPLIES (SOME hasInterest (OR profsubconcept1)) /domain/linkedtv/user/rita)
(IMPLIES (AND (SOME inRule inst_deutsches_historisches_museum) (SOME inRule with_children)) profsubconcept1)
(IMPLIES (AND (SOME inRule museum) (SOME inRule with_children)) profsubconcept2)
(IMPLIES inst_deutsches_historisches_museum museum)
(DISJOINT /domain/linkedtv/user/rita /domain/linkedtv/user/rita_dis)
(WEIGHT profsubconcept1 0.72)
(WEIGHT profsubconcept2 0.67)
(RELATED content rule hasInterest)
```

Due to ongoing technical work over the connection of the Y3 scenario content and the corresponding content annotations and enrichments produced by WP2 with the LinkedTV players at the time that this deliverable was completed, it was not yet possible to produce interactions for Y3 content. Subsequently, the rest of the implicit profiling and filtering approach will rely on manually constructed user profiles, based on the personas of chapter 2.

Immediate steps of the implicit profiling approach will be to finalize the player-content-implicit tracking connection and produce implicit profiles based on the interaction storyboards described in chapter 2 and compare them (as well as the filtering module) against the automatically produced implicit profiles.

6.2 Scenario-based user models

In this chapter, the user profiles for the personas presented in chapter 2 will be presented. The user profiles are in the KRSS¹² format, a lightweight ontological syntax used as input in the LiFR recommender¹³.

The profiling engineering process, following the serialization principles of D4.4, ch. 4.3, applies the following:

- All concepts at the schema level are classes of the LUMO v2 ontology. No IRI, URI, or prefix is employed to define LUMO concepts, to avoid memory overhead. No identification of the ontology namespace is needed anyway, since by default no other vocabulary but LUMO is used in the implicit profiling and filtering.
- All instances (named entities) in the user profiles are treated as most specific subclasses of a LUMO ontology concept. To this end, they are expressed with the prefix *inst_* (e.g. *inst_Angela_Merkel*) and are appointed as subclasses of their LUMO types with an implication axiom (e.g. (IMPLIES *inst_Angela_Merkel* Politician)). This method is devised to discern between weights in the user preferences (e.g. (WEIGHT *inst_angela_merkel* 0.82)) and confidence degrees in content annotation (affecting content profiles, cf. chapter 7.2).

It is worth noticing that the profiles created for the users in the following sections follow the assumption that the users have not previously performed any interactions within the LinkedTV platform, i.e. had an empty user profile. To this end, when starting a new user profile, all possible interests will be added/updated in their profile, from the most specific to the more general. But as the users continue their interactions, obviously their interests will grow significantly in size. In order to prevent user profiles to reach an unmanageable size, a pruning mechanism is employed (cf. D4.2, ch. 5.1 & 7.4) to delete preferences that are under a specific weighting threshold, thus discard the most obsolete or indifferent preferences and give rise to the strongest preferences for each specific user. This threshold is activated and defined based on the size of the user profile and the relative interest weights of all preferences in the profile (e.g. when the profile includes >100 preferences the pruning mechanism is activated and when the average preference weight is >0.5, then a threshold of <0.3 is defined, where preferences with a weight of <0.3 will be discarded).

¹² dl.kr.org/dl97/krss.ps

¹³ On more details on the decision to employ the KRSS format cf. D4.4. The LiFR syntax and semantics are available in <http://mklab.iti.gr/project/LiFR#semantics>

6.2.1 TKK scenario user profiles

Rita

```
(IMPLIES (SOME hasInterest (OR inst_nelleke_van_der_krogt inst_netherlands inst_museum_martena inst_period_rooms profsubconcept1 profsubconcept2 inst_jan_sluijters profsubconcept3 inst_emiel_aardewerk)) rita)
(IMPLIES inst_nelleke_van_der_krogt host)
(IMPLIES inst_netherlands country)
(IMPLIES inst_museum_martena museum)
(IMPLIES inst_period_rooms building_division)
(IMPLIES (AND (SOME inRule inst_frisia) (SOME inRule silver) (SOME inRule inst_tea_jar)) profsubconcept1)
(IMPLIES (AND (SOME inRule inst_frisia) (SOME inRule silver)) profsubconcept2)
(IMPLIES inst_frisia administrative_region)
(IMPLIES inst_tea_jar vessel)
(IMPLIES inst_jan_sluijters painter)
(IMPLIES (AND (SOME inRule inst_jan_sluijters) (SOME inRule art_styles)) profsubconcept3)
(IMPLIES inst_emiel_aardewerk expert)
(DISJOINT rita rita_dis)
(WEIGHT inst_nelleke_van_der_krogt 0.75)
(WEIGHT inst_netherlands 0.65)
(WEIGHT inst_museum_martena 0.93)
(WEIGHT inst_period_rooms 0.82)
(WEIGHT profsubconcept1 0.87)
(WEIGHT profsubconcept2 0.76)
(WEIGHT inst_jan_sluijters 0.74)
(WEIGHT profsubconcept3 0.69)
(WEIGHT inst_emiel_aardewerk 0.81)
(RELATED content rule hasInterest)
```

Michael

```
(IMPLIES (SOME hasInterest (OR profsubconcept1 profsubconcept2 profsubconcept3 inst_jan_eisenloffel art_nouveau paintings)) michael)
(IMPLIES (AND (SOME inRule box) (SOME inRule silver) (SOME inRule inst_europe)) profsubconcept1)
(IMPLIES inst_europe continent)
(IMPLIES (AND (SOME inRule inst_sukkot) (SOME inRule inst_etrog)) profsubconcept2)
(IMPLIES inst_sukkot jewish_festival_or_holiday)
(IMPLIES inst_etrog spice)
(IMPLIES (AND (SOME inRule inst_delft) (SOME inRule plate) (SOME inRule inst_de_porcelyne_fles)) profsubconcept3)
(IMPLIES inst_delft city)
(IMPLIES inst_de_porcelyne_fles factory)
(IMPLIES inst_jan_eisenloffel designer)
(IMPLIES (SOME hasInterest (OR profsubconcept4 profsubconcept5)) michael_dis)
(IMPLIES (AND (SOME inRule inst_delft) (SOME inRule plate) (SOME inRule inst_orient) (SOME inRule paintings)) profsubconcept4)
(IMPLIES inst_orient region)
(IMPLIES (AND (SOME inRule inst_africa) (SOME inRule mask)) profsubconcept5)
(IMPLIES inst_africa continent)
(DISJOINT michael michael_dis)
(WEIGHT profsubconcept1 0.9)
(WEIGHT profsubconcept2 0.2)
(WEIGHT profsubconcept3 0.8)
(WEIGHT inst_jan_eisenloffel 0.8)
(WEIGHT art_nouveau 0.6)
```

(WEIGHT paintings 0.9)
 (WEIGHT profsubconcept4 0.5)
 (RELATED content rule hasInterest)

Bert and Anne

(IMPLIES (SOME hasInterest (OR statuette)) bert)
 (IMPLIES (SOME hasInterest (OR watch)) bert_dis)
 (DISJOINT bert bert_dis)
 (WEIGHT statuette 0.62)
 (WEIGHT watch 0.92)

(IMPLIES (SOME hasInterest (OR watch profsubconcept1 profsubconcept2 inst_silver_tea_jar)) anne)
 (IMPLIES (AND (SOME inRule inst_greece) (SOME inRule mythology)) profsubconcept1)
 (IMPLIES inst_greece country)
 (IMPLIES (AND (SOME inRule inst_frisia) (SOME inRule silver) (SOME inRule inst_tea_jar)) profsubconcept2)
 (IMPLIES inst_frisia administrative_region)
 (IMPLIES inst_tea_jar vessel)
 (DISJOINT anne anne_dis)
 (WEIGHT watch 0.85)
 (WEIGHT profsubconcept1 0.80)
 (WEIGHT profsubconcept2 0.63)

(IMPLIES (AND bert anne) bert_and_anne)
 (IMPLIES (SOME hasInterest (OR profsubconcept3 profsubconcept1)) bert_and_anne)
 (IMPLIES (AND (SOME inRule wood) (SOME inRule statuette)) profsubconcept1)
 (WEIGHT profsubconcept3 0.77)

(RELATED content rule hasInterest)

6.2.2 RBB scenario

Peter

(IMPLIES (SOME hasInterest (OR Journalist soccer inst_fifa inst_Sascha_Hingst inst_Cathrin_Bohme inst_Dirk_Platt inst_Tatiana_Jury sports inst_Ludwigfelde technology inst_martin_deliuss inst_Alexander_Dobrindt inst_Ardnt_Breitfeld inst_Brandenburg profsubconcept1 profsubconcept4 profsubconcept5 politics police inst_Die_Blechtrommel inst_gunter_grass charity science sociology inst_Potsdam inst_BER public_service)) peter)
 (IMPLIES (AND (SOME inRule inst_Social_democrats) (SOME inRule inst_Socialists) (SOME inRule inst_Brandenburg)) profsubconcept1)
 (IMPLIES (AND (SOME inRule refugee) (SOME inRule inst_Kreuzberg)) profsubconcept2)
 (IMPLIES (AND (SOME inRule inst_Berlin) (SOME inRule inst_alte_forsterei)) profsubconcept3)
 (IMPLIES (AND (SOME inRule science) (SOME inRule technology)) profsubconcept4)
 (IMPLIES (AND (SOME inRule science) (SOME inRule inst_Potsdam)) profsubconcept5)
 (IMPLIES inst_Sascha_Hingst Journalist)
 (IMPLIES inst_Cathrin_Bohme Journalist)
 (IMPLIES inst_Tatiana_Jury Journalist)
 (IMPLIES inst_Dirk_Platt Journalist)
 (IMPLIES inst_alte_forsterei stadium)
 (IMPLIES inst_Social_democrats political_party)
 (IMPLIES inst_Socialists political_party)
 (IMPLIES inst_Brandenburg state)

```

(IMPLIES inst_Ludwigsfelde area_of_interest)
(IMPLIES inst_Kreuzberg area_of_interest)
(IMPLIES inst_Alexander_Dobrindt Minister)
(IMPLIES inst_Jochen_Grobmann person)
(IMPLIES inst_Brandenburg state)
(IMPLIES inst_BER airport)
(IMPLIES inst_Potsdam city)
(IMPLIES inst_gunter_grass person)
(IMPLIES inst_Die_Blechtrommel movie)
(IMPLIES inst_Berlin city)
(IMPLIES inst_Germany country)
(IMPLIES inst_martin_delius politician)
(IMPLIES inst_pirates political_party)
(IMPLIES inst_fifa sports_game)
(IMPLIES inst_Ardnt_Breitfeld Journalist)
(IMPLIES (SOME hasInterest (OR profsubconcept2 inst_Berlin inst_Jochen_Grobmann inst_pirates )) peter_dis)
(DISJOINT peter peter_dis)
(WEIGHT political_party 0.70)
(WEIGHT Journalist 0.70)
(WEIGHT sports 0.40)
(WEIGHT Technology 0.95)
(WEIGHT public_Service 0.60)
(WEIGHT soccer 0.20)
(WEIGHT science 0.50)
(WEIGHT sociology 0.60)
(WEIGHT charity 0.65)
(WEIGHT police 0.80)
(WEIGHT politics 0.40)
(WEIGHT profsubconcept1 0.70)
(WEIGHT profsubconcept2 0.70)
(WEIGHT profsubconcept3 0.20)
(WEIGHT profsubconcept4 0.80)
(WEIGHT profsubconcept5 0.90)
(WEIGHT inst_alexander_dobrindt 0.90)
(WEIGHT inst_pirates 0.70)
(WEIGHT inst_martin_delius 0.30)
(WEIGHT inst_Die_Blechtrommel 0.30)
(WEIGHT inst_gunter_grass 0.65)
(WEIGHT inst_Potsdam 0.20)
(WEIGHT inst_Berlin 0.50)
(WEIGHT inst_fifa 0.55)
(WEIGHT inst_BER 0.75)
(WEIGHT inst_Ardnt_Breitfeld 0.50)
(WEIGHT inst_Brandenburg 0.90)
(WEIGHT inst_Jochen_Grobmann 0.50)
(WEIGHT inst_Alexander_Dobrindt 0.20)
(WEIGHT inst_Ludwigsfelde 0.80)
(WEIGHT inst_Tatiana_Jury 0.80)
(WEIGHT inst_Dirk_Platt 0.60)
(WEIGHT inst_Cathrin_Bohme 0.70)
(WEIGHT inst_Sascha_Hingst 0.90)
(RELATED content rule hasInterest)

```

Nina

(IMPLIES (SOME hasInterest (OR Actor politician profsubconcept1 profsubconcept2 profsubconcept3 Refugee politics security_measures police Public_transportation Economics inst_Claudia_Kemfert Human_rights inst_Greenpeace inst_SPD inst_Die_Linke inst_Die_Blechtrummel movie_theater inst_gunter_grass charity science sociology inst_Prenzlauer_Berg inst_Glienicker_Brucke inst_Potsdam inst_Berlin inst_Brazil inst_BER public_service environment)) nina)
 (IMPLIES (AND (SOME inRule transportation) (SOME inRule alexander_dobrindt)) profsubconcept1)
 (IMPLIES (AND (SOME inRule inst_Berlin) (SOME inRule inst_Germany)) profsubconcept2)
 (IMPLIES (AND (SOME inRule inst_Berlin) (SOME inRule politics)) profsubconcept3)
 (IMPLIES inst_BER airport)
 (IMPLIES inst_Brazil Country)
 (IMPLIES inst_Potsdam city)
 (IMPLIES inst_Prenzlauer_Berg area_of_interest)
 (IMPLIES inst_Glienicker_Brucke area_of_interest)
 (IMPLIES inst_gunter_grass person)
 (IMPLIES inst_Die_Blechtrummel movie)
 (IMPLIES inst_Berlin city)
 (IMPLIES inst_Germany country)
 (IMPLIES inst_Greenpeace Non_profit_organization)
 (IMPLIES inst_Claudia_Kemfert person)
 (IMPLIES inst_martin_delius politician)
 (IMPLIES inst_alexander_dobrindt politician)
 (IMPLIES inst_pirates political_party)
 (IMPLIES inst_SPD political_party)
 (IMPLIES inst_Die_Linke political_party)
 (IMPLIES inst_Frank_Henkel politician)
 (IMPLIES inst_Christopher_Lauer politician)
 (IMPLIES inst_Steven_Spielberg Actor)
 (IMPLIES inst_Tom_Hanks Actor)
 (IMPLIES inst_fifa sports_game)
 (IMPLIES inst_Ardnt_Breitfeld Journalist)
(IMPLIES (SOME hasInterest (OR inst_martin_delius inst_pirates inst_Frank_Henkel inst_Christopher_Lauer inst_Steven_Spielberg inst_Tom_Hanks soccer inst_fifa inst_Ardnt_Breitfeld)) nina_dis)
 (DISJOINT nina nina_dis)
 (WEIGHT political_party 0.55)
 (WEIGHT Actor 0.4)
 (WEIGHT politician 0.575)
 (WEIGHT environment 0.65)
 (WEIGHT public_Service 0.60)
 (WEIGHT soccer 0.40)
 (WEIGHT science 0.50)
 (WEIGHT sociology 0.60)
 (WEIGHT charity 0.70)
 (WEIGHT movie_theater 0.80)
 (WEIGHT refugee 0.70)
 (WEIGHT Human_rights 0.60)
 (WEIGHT Economics 0.50)
 (WEIGHT Public_transportation 0.80)
 (WEIGHT police 0.5)
 (WEIGHT security_measures 0.75)
 (WEIGHT politics 0.90)
 (WEIGHT profsubconcept1 0.80)
 (WEIGHT profsubconcept2 0.70)
 (WEIGHT profsubconcept3 0.90)

(WEIGHT inst_alexander_dobrindt 0.90)
(WEIGHT inst_pirates 0.40)
(WEIGHT inst_martin_delius 0.30)
(WEIGHT inst_Frank_Henkel 0.70)
(WEIGHT inst_Claudia_Kemfert 0.50)
(WEIGHT inst_Greenpeace 0.90)
(WEIGHT inst_SPD 0.60)
(WEIGHT inst_Die_Linke 0.65)
(WEIGHT inst_Die_Blechtrommel 0.70)
(WEIGHT inst_gunter_grass 0.65)
(WEIGHT inst_Glienicker_Brucke 0.70)
(WEIGHT inst_Prenzlauer_Berg 0.90)
(WEIGHT inst_Potsdam 0.20)
(WEIGHT inst_Berlin 0.70)
(WEIGHT inst_Tom_Hanks 0.30)
(WEIGHT inst_Steven_Spielberg 0.50)
(WEIGHT inst_Brazil 0.75)
(WEIGHT inst_fifa 0.95)
(WEIGHT inst_BER 0.80)
(WEIGHT inst_Ardnt_Breitfeld 0.60)
(WEIGHT inst_Christopher_Lauer 0.40)
(RELATED content rule hasInterest)

7 Core Recommendation

Effective and fast recommendation generation plays a central role in large-scale content consumption systems like online video platform and distributed advertising systems. Utilising the semantics hidden in the content to produce effective recommendations is an attractive and challenging task for both academia and industry. In LinkedTV, a semantic recommender based on the LiFR semantic reasoner is extended and employed for this task. LiFR produces concept and content recommendations by performing *semantic matchmaking* between a given user profile and a set of candidates to be recommended. The matchmaking and subsequent recommendation process is detailed in deliverable D4.5, ch. 3.2. This section presents the communication of the LiFR recommender with the LinkedTV workflow and a set of comprehensive evaluations, both in terms of its recommendation performance as well as in terms of its algorithmic efficiency within LinkedTV.

7.1 LiFR reasoner-based recommendation and evaluation

The implicit recommendation module based on the LiFR reasoner (previously known as f-PocketKRHyper) can serve as a standalone filterer, but also as the post-filterer in the combined implicit and explicit approaches, as described in deliverable D4.5, ch. 3.2. Depending on the requested task, a collection of concepts or a collection of content items (which would include a plurality of concepts per item) is passed by the LinkedTV platform to the LiFR-based recommender as the set of candidates to filter from, via RESTful services¹⁴. The LiFR-based recommender then matches these content items/concepts to a given implicit user profile, as retrieved from the Linked Profiler. The user ID is again passed to LiFR by the platform, which in turn retrieves it from the player. As a result, the recommender ranks candidates according to the implicit user interests, while also rejecting false positives with respect to the user disinterests.

In the combined implicit and explicit approach, the LinkedTV platform will firstly receive the results of the Personal Recommender and then pass them to the LiFR-based recommender as the candidate set. LiFR will then re-rank recommendations coming from explicit preferences to also reflect implicit preferences, while again rejecting false positives.

The inference process that the LiFR reasoner employs for matchmaking is described in D4.5, ch. 3.2. It is worth noticing that in the LiFR-based inferencing service, all information about the user and the content is expressed only within the LUMO (LUMO v2 and in the case of the TKK scenario also LUMO-arts) “world”. Therefore both seed content and enrichment annotations are exposed to LUMO, as seen in deliverable D2.6, ch. 5.1.2 and then passed as input to the recommendation service.

¹⁴ The LiFR RESTful services are described in D4.5, ch. 3.2.3

In the interest of evaluating LiFR-based recommendations, three sets of evaluation experiments were conducted:

In the first use case, a controlled media content dataset was chosen, where 73 freely available media items (i.e. open-licensed videos, images, web pages) were selected online and manually annotated with LUMO concepts. In principal, the content was annotated with the kinds of concepts that can be automatically detected via LinkedTV's annotation services. Such manual annotations are considered error-free, as opposed to automatic annotations which inevitably would carry some level of error.

At the same time, seven manual user profiles were constructed (an example can be seen in Table 8), with several combinations of interests that were visible in the 73 media items, including atomic preferences (interests *and* disinterests) and associations (rules) between preferences. Each user profile was then used to filter and rank the 73 content items according to the user preferences. The evaluation results are displayed in Table 9.

Since implicit user models and in effect recommendations are inferred from relevant concepts in the content annotation, errors in the annotation, lack of semantics modelled in LUMO, or lack of mappings between the annotation and LUMO are directly reflected at the recommendation layer. Consequently, this controlled test case demonstrates (a) the level of completeness of LUMO both in terms of concept space coverage and of mappings coverage (to model user profiles and content profiles), as well as (b) the overall efficiency of the content filtering module. Results show a very good performance validating the service's efficiency.

Table 8: An example of a user profile of the first use case

```
(IMPLIES (SOME hasInterest (OR profsubconcept1 profsubconcept2 profsubconcept3 natural_disaster crime military_conflict)) profile1)
(IMPLIES (AND (SOME inRule terrorism) (SOME inRule bombing)) profsubconcept1)
(IMPLIES (AND (SOME inRule inst_angela_merkel) (SOME inRule politics)) profsubconcept2)
(IMPLIES (AND (SOME inRule inst_barrack_obama) (SOME inRule inst_syria)) profsubconcept3)
(IMPLIES inst_angela_merkel politician)
(IMPLIES inst_barrack_obama president)
(IMPLIES inst_syria country)
(IMPLIES (SOME hasInterest (OR inst_sri_lanka health animals_&_wildlife)) profile1_dis)
(IMPLIES inst_sri_lanka country)
(DISJOINT profile1 profile1_dis)
(WEIGHT profsubconcept1 0.76)
(WEIGHT profsubconcept2 0.63)
(WEIGHT profsubconcept3 0.97)
(WEIGHT natural_disaster 0.55)
(WEIGHT crime 0.64)
(WEIGHT military_conflict 0.51)
(WEIGHT inst_sri_lanka 0.81)
(WEIGHT health 0.74)
(WEIGHT animals_&_wildlife 0.94)
(RELATED content rule hasInterest)
```

Table 9: Use case 1: Precision, recall, f-measure for the recommendation of the 73 manually annotated content items, over the 7 manual user profiles

| Profile | Precision | Recall | f-Measure |
|-----------------|-------------|-------------|-------------|
| <i>profile1</i> | 1 | 0.769230769 | 0.869565217 |
| <i>profile2</i> | 0.833333333 | 0.833333333 | 0.833333333 |
| <i>profile3</i> | 0.933333333 | 0.777777778 | 0.848484848 |
| <i>profile4</i> | 1 | 0.8 | 0.888888889 |
| <i>profile5</i> | 1 | 0.714285714 | 0.833333333 |
| <i>profile6</i> | 1 | 1 | 1 |
| <i>profile7</i> | 1 | 1 | 1 |
| AVG | 0.966666667 | 0.842089656 | 0.896229374 |

The second use case consisted of evaluating recommendations over a large set of automatically annotated content. Out of a pool of 970 content items of RBB news segments, 50 items were selected randomly as the test set and annotated automatically. The automatic annotation of this particular dataset consisted only of processing the audio track of the videos through ASR analysis, therefore did not undergo the full WP2 processing pipeline (e.g. analyzing textual transcripts). In addition, a "golden standard" set of manual annotations of the same 50 items was prepared as ground truth.

Based on the content annotation, five user profiles were created manually in order to reflect the contents of the 50 videos dataset. Although not undergone the full WP2 analysis, this experiment can still demonstrate the completeness and efficiency of the content filtering module (Table 10), especially looking at the precision/recall of the automatic annotation (Table 11).

Table 10: Use case 2: Precision, recall, f-measure for the recommendation of the 50 automatically annotated RBB content items, over the 5 manual user profiles

| Profile | Precision | Recall | F-Measure |
|---------------------|-----------|-------------|-------------|
| <i>RBBProfile01</i> | 0.375 | 0.375 | 0.375 |
| <i>RBBProfile02</i> | 1 | 0.428571429 | 0.6 |
| <i>RBBProfile03</i> | 0.25 | 0.142857143 | 0.181818182 |
| <i>RBBProfile04</i> | 0.75 | 0.375 | 0.5 |
| <i>RBBProfile05</i> | 0.4 | 0.5 | 0.444444444 |
| AVG | 0.555 | 0.364285714 | 0.420252525 |

Table 11: Average precision, recall, f-measure of the automatic annotation of 50 RBB videos in comparison with the ground truth annotations.

| AVG | Precision | Recall | f-Measure |
|----------------------|-----------|-------------|-------------|
| <i>50 RBB videos</i> | 0.2931806 | 0.461643468 | 0.350653773 |

The low scores of the automatic annotations as compared to the ground truth annotations (Table 8) and their direct correspondence to the content filtering scores for this test case,

verify that the low performance of the recommendation service is a direct consequence of the quality of annotations. The fact that the recommendation scores are much higher than the annotation scores is due to the fact that the user profiles were constructed manually, and not based on the automatic annotations of this content, otherwise the misses in the annotation would also be reflected in the recommendations.

The third use case consisted of recommending chapters of the seed content of the year 3 scenarios based on the manually constructed profiles presented in chapter 6.2. The dataset consisted of a much smaller set than the previous test case: 4 chapters for the RBB scenario¹⁵ and 9 chapters for the S&V scenario. The content was annotated automatically, after having undergone the entire WP1-WP2 analysis pipeline.

Table 12: Use case 3/RBB scenario: Precision, recall, f-measure for the recommendation of the 4 automatically annotated RBB chapters, over the *Nina* and *Peter* manual user profiles

| User Profile | Precision | Recall | f-Measure |
|--------------|-----------|-------------|-----------|
| <i>Nina</i> | 1 | 0.333333333 | 0.5 |
| <i>Peter</i> | 1 | 0.666666667 | 0.8 |
| AVG | 1 | 0.5 | 0.65 |

In the RBB case (Table 12), the results verify the perfect precision of the recommendation module. Low recall was investigated and concluded to be a result of the information about the content passed by the automatic annotation tools: in some cases, content that is about something a user prefers (as seen in the curated annotation of LinkedTV's editor tool) was not annotated by WP2 with an entity that should have been present in the content; in other cases, it was due to lexical differences between the manual and automatic annotations (e.g. preference "Claudia_Kemfert" in the manual profile, vs. recognized entity "expertin_Claudia_Kemfert" which is a whole different entity and throws off matching for this case). The latter missed positive is expected to be resolved when automatic annotations formulate the user profile. However, if in different contents, an entity is recognized with different lexical definitions (e.g. if Claudia Kemfert is recognized in one content item as "Claudia_Kemfert" and in another as "expertin_Claudia_Kemfert"), this discrepancy will again reflect in both the user profiles and recommendations.

¹⁵ Out of the 9 chapters presented in this deliverable, the 4 that are part of the scenario demonstrators were used in this experiment, while evaluation based on all 9 chapters is under way.

Table 13: Use case 3/TKK scenario: Precision, recall, f-measure for the recommendation of the 9 automatically annotated RBB chapters, over the *Anne*, *Bert*, *Michael* and *Rita* manual user profiles

| User Profile | Precision | Recall | f-Measure |
|----------------|-------------|-------------|-------------|
| <i>Anne</i> | 0 | 0 | 0 |
| <i>Bert</i> | 1 | 0.666666667 | 0.8 |
| <i>Michael</i> | 0.5 | 0.333333333 | 0.4 |
| <i>Rita</i> | 1 | 0.5 | 0.666666667 |
| AVG | 0.833333333 | 0.5 | 0.622222222 |

In the TTK case (Table 13), lower scores were investigated and it was discovered that it is a reflection of the types retrieved from the content annotation. Types in the automatic content annotation of seed content are normalised under the DBpedia and NERD ontologies, which are both high-level, more generic ontologies, just like the core LUMO ontology, thus lack more specific information about the arts domain which are inherent of the TTK scenario. As a result, apart from expected misses in the content annotation, as before, it was observed that since the more specific art-related types that are important in the TTK scenario were usually not presented in the seed annotations, which was reflected in the recommendation results.

It is expected that in the case of recommending enrichments (which is prioritized in the TTK scenario) this problem will be alleviated, as enrichments are also annotated based on the YAGO ontology, which includes more specific types that are mapped to LUMO-arts. At the time that these experiments were conducted, an adequate enrichment test set was not available, but evaluation of filtering enrichments based on the manually created profiles is now under way.

7.1.1 LiFR performance evaluation

In the context of evaluating the LiFR reasoner's algorithmic performance, LiFR's memory consumption and time to perform a reasoning task was compared against two other prominent fuzzy reasoners, namely FiRE¹⁶ and fuzzyDL¹⁷. The experiment consisted of computing the global greatest lower bound (or global GLB) on several sets of randomly generated assertions (i.e. individuals/instances) based on the LUMO ontology as the background knowledge base. It is worth noticing that the global GLB calculation is the most complex reasoning task in fuzzy reasoners (of LiFR's expressivity), while a reasoner's algorithmic complexity is increased when the number of assertions in the problem at hand is increased. The results were presented in the ESWC conference [TSA14b] and can be seen in Table 14.

¹⁶ <http://www.image.ece.ntua.gr/~nsimou/FiRE/>

¹⁷ <http://nemis.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html>

Table 14: Time performance and memory consumption of LiFR, FiRE and FuzzyDL on global GLB calculation

| Number of Individuals | Time (ms) | | | Memory (MB) | | |
|-----------------------|-------------|----------------|-------------|-------------|-------------------|-------------|
| | <i>LiFR</i> | <i>FuzzyDL</i> | <i>FiRE</i> | <i>LiFR</i> | <i>FuzzyDL</i> | <i>FiRE</i> |
| 20 | 189 | 38458 | 47538 | 10 | 59.07 | 67.95 |
| 50 | 192 | 169875 | 318228 | 92.42 | 181.19 | 252.8 |
| 100 | 332 | 596292 | 665721 | 137.28 | 206.36 | 274.27 |
| 250 | 923 | 4955568 | 3137765 | 169.26 | 268.23 | 386.82 |
| 500 | 2015 | 23239036 | 6316162 | 191.64 | 294.75 | 474.02 |
| 1000 | 4208 | >12hrs | 12260563 | 239.93 | N/A ¹⁸ | 515.12 |

In addition, the time consumption of LiFR when retrieving the topics for all 970 RBB videos of the test set of the second use case, presented in the previous section, where computed (cf. D2.6, ch, 5 for more details on the topic detection experiment).

The process involved running three distinct reasoning services per content item:

- (1) *entailment*, based on the LUMO mappings ontology, in order to map DBPedia types in the content annotation to LUMO concepts (i.e. the basic LUMO wrapper functionality);
- (2) *global GLB calculation* (main task and LiFR's default reasoning service), based on the LUMO ontology as the background and the concepts retrieved in the previous step as assertions, in order to retrieve all concepts (aka the model) that describe to each content item;
- (3) iterative *subsumption* check for each of the predicates in the produced model against the LUMO *Topics* superconcept, to retrieve from the entirety of the predicates in the produced model, the ones that are subsumed by *Topics*, thus are actually topics.

The time performance distribution for this task is illustrated in Figure 28.

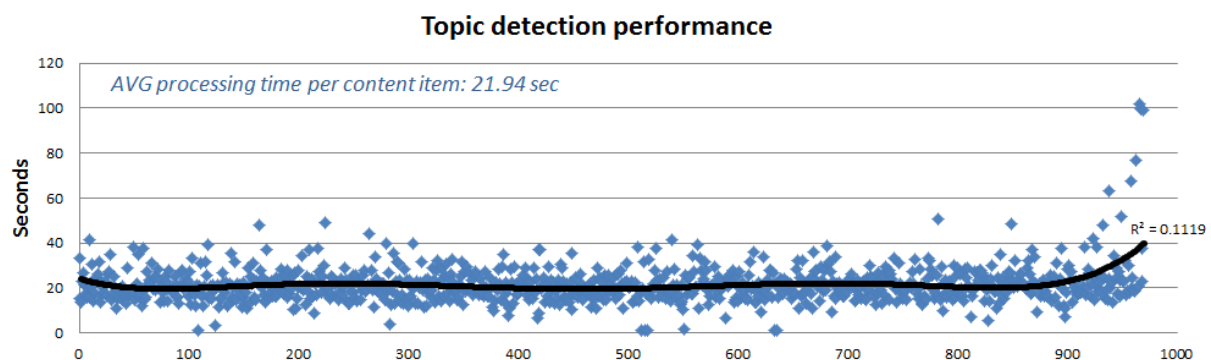


Figure 28: LiFR's time performance for topic detection. Points denote each content item's processing time. The line shows the polynomial trendline (order of 6) of the data points.

¹⁸ Due to time restrictions, fuzzyDL was terminated for the 1000 instances case after it exceeded 12 hours of processing without rendering results.

The experiments presented in this section verify the efficiency of LiFR as a lightweight reasoner, both as compared to other prominent fuzzy reasoners, as well as in executing real-world tasks and within the LinkedTV context. Therefore, it fortifies the capacity of LiFR to be used as the underlying reasoning service even in limited-resource devices¹⁹, thus enabling it to run on the user client so as to preserve user privacy in the personalisation setting of LinkedTV.

7.1.2 Bringing recommendations to the general workflow

When personalized content and/or concept delivery is opted, the LinkedTV Player uses requests to the platform API with the “personalized” flag and gets the personalization data produced by the LiFR-based reasoner from the LinkedTV Platform.

The workflow is specified in deliverable D5.6, section 2.6, and its summary is as follows. The LinkedTV platform provides the recommendation candidates set (along with its semantic descriptions/annotations) and the ID of the current user. The recommendation module retrieves the profile of the current user based on his ID, while the candidate content annotations/entities are exposed to LUMO, as described in D2.6, ch.5. Once the recommendations are produced based on this information, they are passed to the LinkedTV platform via the recommenders’ RESTful services (cf. D4.5, ch. 3.2.3). The player receives the information from the LinkedTV Platform as specified in deliverable D5.6, section 2.6.

In deliverable D3.7, there are more details about the way the player plans to take into account the personalization results offered by WP4.

¹⁹ Resource-constrained devices (e.g. smartphones, tablets, set-top boxes) have memory limitations. The lowest memory limitation is that of first-generation smartphones, which had a heap size limitation of 16MB and in which LiFR can perform, while newer versions can exceed 64MB. In tablets, the heap size nowadays amounts to an average 256MB (commonly, much less or more is possible, depending on the device), in which LiFR performs, while the compared reasoners exceed for more complex problems. Source: <http://stackoverflow.com/questions/5350465/android-heap-size-on-different-phones-devices-and-os-versions>

8 The Experimental Technology

The core technology displayed in Figure 1 can be completed by optional experimental technologies. The optional modules on explicit user interaction and knowledge base (LUME and LUMOPedia module in Figure 1) have been developed as an experimental branch in the LinkedTV workflow. Two possible merits can be derived from an explicit approach.

First, the user has immediate access to his or her preferences, so that he or she can change the cues that lead to the ranking and/or filtering of the links presented, thus feeling more involved within the overall process.

Second, the ontology items do not need to be as tied to other linked-open-data ontologies such as yago or dbpedia in order to synchronise the mappings of the media fragment annotation, but can be formulated in an even more intuitive way. The explicit user interaction workflow has been erected over the course of the year and been brought to a stable release at a time where not all necessary APIs from the other work packages were already established, leading to several software interfaces where internal short cut solutions were preferred (e.g., using existing web player technology in order to show a change in video ranking).

All steps in the explicit user interaction workflow have their own APIs as explained in the following Section, so that they can be integrated as a module into the core workflow. However, as the need to focus on predominant technology was suggested in the second year's review but also became apparent from remaining time/person resources, it was decided to maintain the explicit user interaction branch in working order but not further interact with other services. In this way, it is possible to evaluate in the upcoming D4.7 how much interaction is needed to produce a user model that satisfies scenario persona demands if (a) his or her preferences are stored directly as weights within the explicit ontology, or if (b) a likely user reaction to a given set of videos is simulated by anticipating his or her player / kinect interaction. Beyond that, hybrid solutions such as a pre-filtering using one recommendation system in order to speed up the second, or by presenting the implicitly learned user preference rule in such a way that it is convenient for the user to adapt them explicitly as well, would be interesting perspectives beyond the LinkedTV project duration.

An optional personal recommender (based on LSF) was also set up and it is described in Chapter 11. This one is used with the experimental branch (LUMOPedia, LUME, LSF) in order to be able to test this experimental branch before the core recommender is ready.

9 Experimental Reference Knowledge

9.1 LUMOPedia

In order to provide a more consistent and coherent knowledge base to the Personal Recommender (see chapter 11), we have further developed the LUMOPedia which was originally described in D4.4 section 6.1.1. An AJAX powered web-based frontend for the exploration and collaborative modification of the knowledge base has been. In the backend, in order to improve the reasoning performance, the whole knowledge base is materialised in a relational database²⁰ and in-database analytics have been applied.

Note that, while the description of the core ontology LUMO-v2 builds upon aspects already elaborated in detail in D4.2 and D4.4, the optional LUMOPedia was covered only briefly (6.1.1 in D4.4) and is thus in need of more extensive explanation.

9.1.1 Design considerations

During the new development of LUMOPedia, we have decided to use a dedicated temporal-aware relational schema to tackle the following common issues with Semantic Reasoning and Linked Open Data (LOD):

- Semantic noise and ambiguity inside of LOD
- Missing unified and standard class taxonomy
- Both OWL and OWL2 do not scale well for large data sets
- RDF triple stores suffer from the join performance

9.1.1.1 Dedicated Temporal-aware Relational Schema

We decided to develop our own relation schema to store large scale knowledge bases like LUMOPedia inside of a relational database. The reason not to use existing RDF triple serialization frameworks like Sesame [BRO03] or Jena [MCB02] is that we want to fully control the low-level details to achieve maximum reasoning performance. With this schema, a subset of OWL expressivity can be provided. We highlight some of the features in the following list – it is however by no means complete:

- Classes with internal IDs, official names, alias, mapping to other LOD classes.
- Relations between classes. Multiple inheritances are supported. This results in a Directed Acyclic Graph (DAG) for the class taxonomy.

²⁰ we decided to embrace the relational database instead of the triple stores and graph databases - which are getting more and more attentions in NOSQL scenarios - is based on the considerations of efficiency, scalability, availability (both know-how and technical systems) and ecosystems.

- Instances with internal IDs, ranking, mapping to other LOD instances.
- Properties with domain and range. The range can be other instances or primitive values like integer or date.
- Relations containing the extracted facts from other LOD data set with temporal valid time information.
- Deductive inference rules.

All relations modelled in this schema are temporal-aware. For example, the fact “Obama is president” is not always true. This is actually only valid in a given time interval. Yago2 [HOF13] tries to address similar problems. However no efficient temporal reasoning approach is proposed. With the development of temporal databases [SNO99], all the temporal reasoning tasks can be reduced to RDBMS which is capable to efficiently process temporal queries over hundreds of millions of records using certain index structures [KAU13]. Traversing the class taxonomy is a common operation in semantic reasoning tasks. Typical queries like “give me all politicians who are born in France” can be translated into side-effect free recursive SQL queries and further processed with transparent parallelisation and indexing inside of the RDBMS. This can greatly reduce the complexity of the developing a semantic reasoner and it works for all transitive properties. Inverse properties can be defined within this schema. Relations of the inverse properties can be derived at runtime. To improve the performance however, in the current implementation “Materialized Views” are used to store all intermediate results with indexing to accelerate the query processing. The database schema hosting the LUMOPedia knowledge base is illustrated in Figure 29.

9.1.1.2 Reasoning with Open World and Closed World Assumptions

Description Logics [BAA03] which is the theory foundation of OWL [MCG04, MOT09] employs Open World Assumption (OWA) [RUS10] for its reasoning. Other semantic systems, like Frames-based Protege [NOY00] however, use Closed World Assumption (CWA) [REI77] to drive the underlying inference engine. In general OWA is more powerful but does not scale well comparing to CWA. The inference engine for LUMOPedia uses both assumptions for different tasks to maximize the reasoning performance. Temporal facts like “Obama is a president” are not stored explicitly as a relation. Instead a rule is defined to specify which kind of person is a president, e.g.

$$\text{President}(X)@T \equiv \text{Person}(X) \wedge \text{hasProfession}(X, \text{president})@T$$

With this approach we can inference the knowledge base to get all presidents at any given time. The combination of CWA and OWA is well suited to solve the temporal-aware reasoning issues.

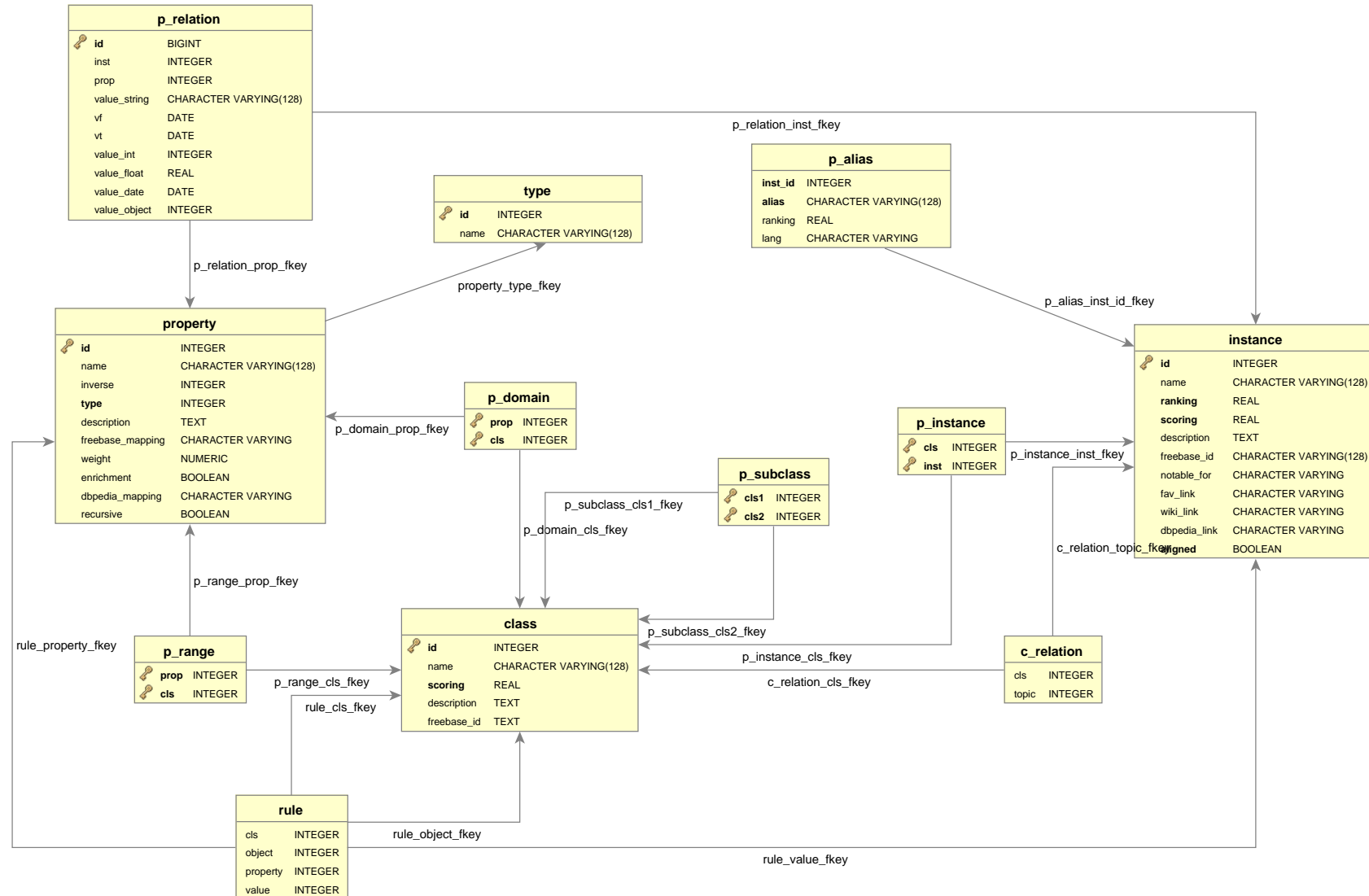


Figure 29: Relational database schema hosting the LUMOPedia knowledge base

9.1.1.3 Unified Ontology for User and Content Modelling

An ontology [GRU93] together with its instance space form a well-structured knowledge base. Ontologies shape the conceptual space of an application domain. Various upper-level ontologies [MAT06, NIL01] and domain-specific ontologies [RAI07, FOAF, MOVO] have been proposed to cover different kinds of modelling tasks. In LUMOPedia a unified ontology for user and content modelling (LUMO²¹, see chapter 4) is integrated to support the semantic recommendation. LUMO contains a lightweight class taxonomy and property definitions covering most aspects of user modelling and content annotation. The whole ontology is developed as OWL axioms at first and then imported into LUMOPedia through an importer utility program. Currently LUMOPedia has 14 149 OWL axioms including 562 class assertions. In the following, a sample of the ontology is illustrated as OWL axioms:

```
. . .
OWL : SubClassOf(Organisation,OWL : Thing)
OWL : SubClassOf(AdvisoryBoard,Organisation)
OWL : SubClassOf(Highway, Street)
OWL : DataPropertyRange(age, xsd : integer)
OWL : ObjectPropertyRange(livesAt, location)
OWL : ObjectPropertyRange(subTopic, topic)
OWL : ClassAssertion(topic, sport)
OWL : ObjectPropertyAssertion(subTopic, Golf, sport)
. . .
```

To fully leverage the enormous data available in LOD, the concepts/classes in LOD are mapped to the classes in LUMO. This is a semi-automatic process. We used different technologies developed in the ontology alignment community [OAE]. In the end some of them are manually calibrated based on their semantic meanings. After the classes are mapped, the property assertions in other LOD data sets are imported automatically. This process is very time-consuming and we have done it in an incremental way, i.e. only those facts which are relevant for generating the recommendations are imported. If new media contents are ready for recommendation, during the analysis phase, new facts from other LOD data set will be imported automatically. This keeps the knowledge base lightweight and always consistent with the contents which need to be recommended.

9.1.2 Statistics of the LUMOPedia knowledge base

There are plenty of information in the LUMOPedia knowledge base. Some of them are manually curated for the semantic recommendation systems while the rest are imported from

²¹ Currently, only a subset of the whole LUMO ontology is imported and aligned with the LUMOPedia knowledge base

other LOD dataset with low semantic confidence. The whole statistics about this knowledge base is summarised in Table 15.

Table 15: Statistics of the LUMOPedia knowledge base

| | |
|----------------------|------|
| Classes | 562 |
| Instances | 7969 |
| Relations | 5618 |
| Relation definitions | 29 |
| Topics | 147 |
| Persons | 522 |
| Locations | 1314 |
| Organisations | 428 |
| Events | 25 |

For the manually curated instances, the distribution of the number of curated instances is illustrated in Figure 30. The X-axis is the number of references times of an instance. For instance, there are almost 80 unique curated instances which are used three times as property domains.

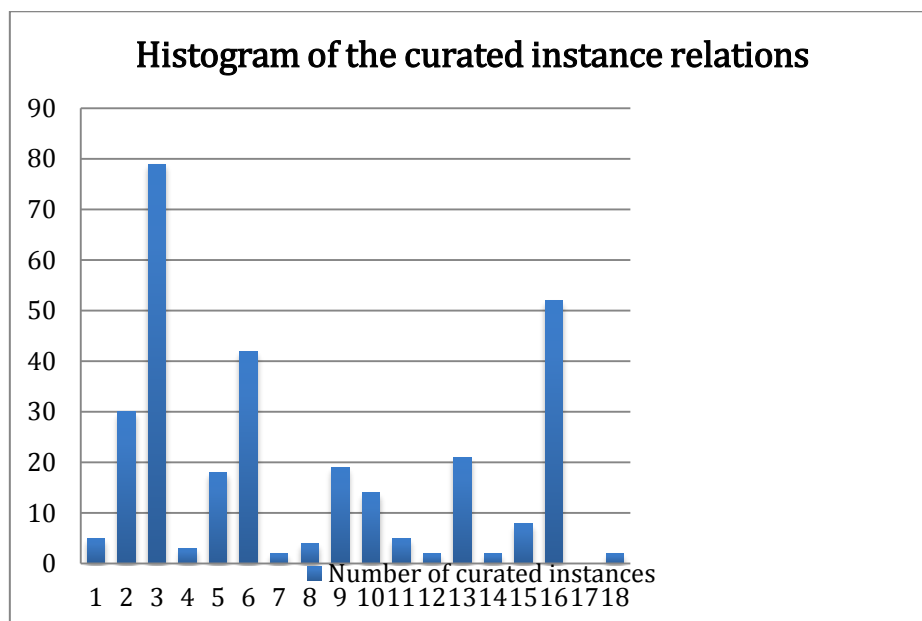


Figure 30: Histogram of the curated instance relations

9.1.3 LUMOPedia Browser as the frontend

In order to provide an easy-to-use user interface for our partners to explore and collaboratively manage the LUMOPedia knowledge base, we have developed a web-based frontend “LUMOPedia Browser”. A screenshot of this frontend is given in Figure 31. Currently this web application is hosted on the Fraunhofer server and can be evaluated by visiting the URL:

<http://linkedtv1.iais.fraunhofer.de:8080/kiwi-war/>

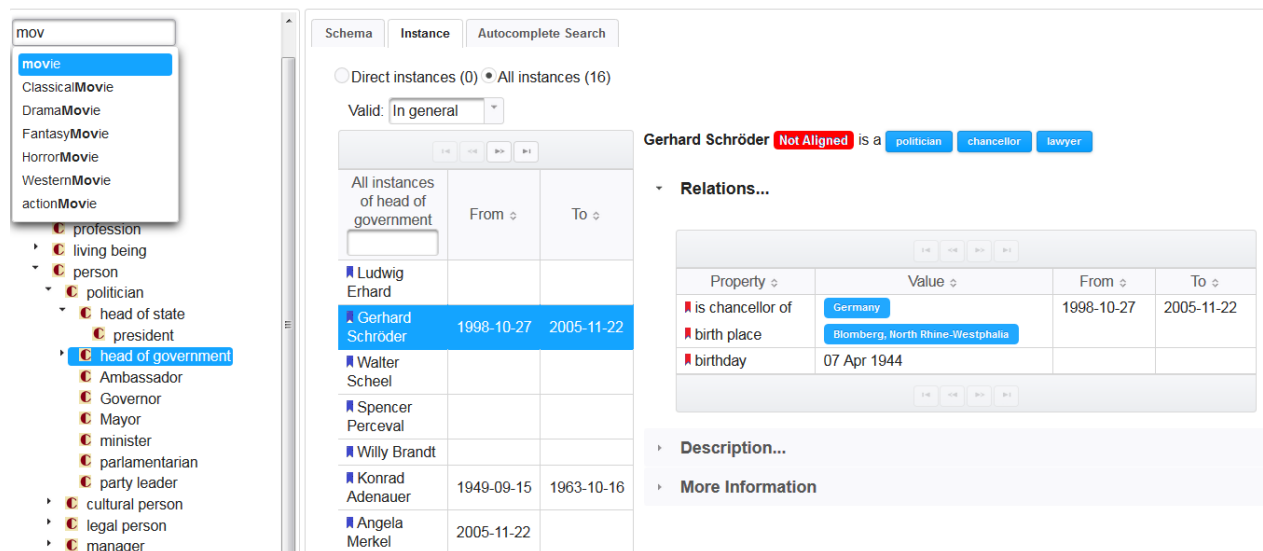


Figure 31: LUMOPedia Browser - the web-based frontend of the LUMOPedia knowledge base

The frontend contains mainly three parts which are elaborated in the following subsections:

- The class taxonomy
- The schema and property definitions
- And the instances with temporal constraints

9.1.3.1 The class taxonomy

Classes in an ontology form a Directed Acyclic Graph (DAG). In LUMOPedia Browser, it is visualised as a tree with multiple inheritance support, i.e. if a class is the parent class of multiple subclasses, then it will appear more than once in the tree. Subclasses are indented in the tree-based visualisation (see the left most part of Figure 29 and Figure 30). All the subclasses can be recursively expanded or collapsed with a single mouse click.

To enable a fast navigation in the class taxonomy, class name based auto-completion is supported to search a class (see the top left dropdown box in Figure 31).

9.1.3.2 The schema and property definitions

Properties can be asserted to a set of classes. If a class in the “class taxonomy” part is selected, the properties which are associated with this class are listed on the right side of the frontend under the tab “Schema” (see Figure 32). These properties can be classified into four categories:

- The directly/explicitly associated properties using the class as the domain. For example the property “has actor/actress” for the class “movie” as depicted in Figure 32.
- The inherited properties using the class as the domain. For example, the properties “deals With”, “has Language”, etc. which are actually associated to the parent classes (the “media item” class in this case) of the class “movie”.

- The directly/explicitly associated properties using the class as the range. For example the property “starred in” for the class “movie” as depicted in Figure 32.
- The inherited properties using the class as the range. For example, the properties “author of”, “treated by”, etc. which are actually associated to the parent classes of the class “movie”.

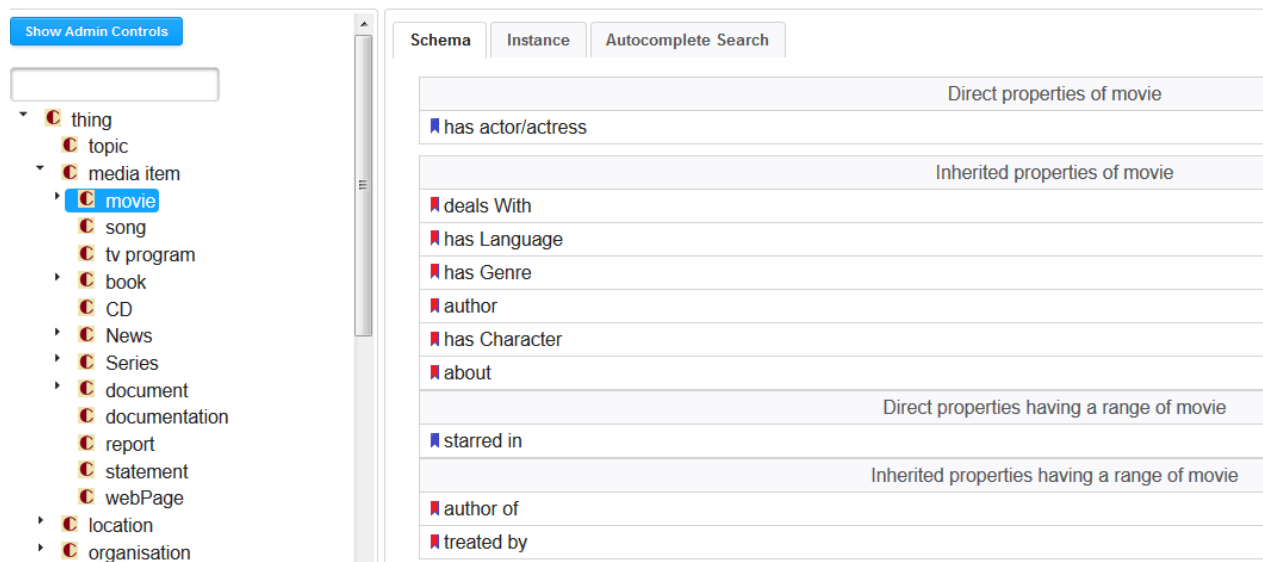


Figure 32: The defined and inherited properties for the class "movie"

9.1.3.3 The instances with temporal constraints

The instances of a selected class are listed on the right side of the frontend under the tab “Instance”. This includes the direct instances and indirect instances. Two radio buttons are supplied in the user interface to choose one of them. The “Valid” dropdown box provides four options to let the user specify a valid time window to see those instances which are valid during this given time period. Currently only four predefined time windows are provided: Now, Past Decade, Past Century, and in general.

Each instance has a valid time period. Currently the temporal information is imported automatically from other LOD datasets like Freebase. However, it is possible to provide certain widgets on the GUI to curate this information directly by the partners.

Once an instance is selected by the user, the corresponding classes (one instance can have multiple classes), the relations (relationships between this instance and other instances or primitive data), and other information like the Wikipedia URL, freebase ID, etc. are listed on the right side of the LUMOPedia Browser (see Figure 31).

9.1.4 Backend with JavaEE and PostgreSQL

The whole LUMOPedia Browser application is written as a Java Enterprise Application with the web profile. It uses the JSF/PrimeFaces as the frontend framework and EJB, CDI, JPA as the backend. The whole architecture is illustrated in Figure 33 as a UML deployment diagram.

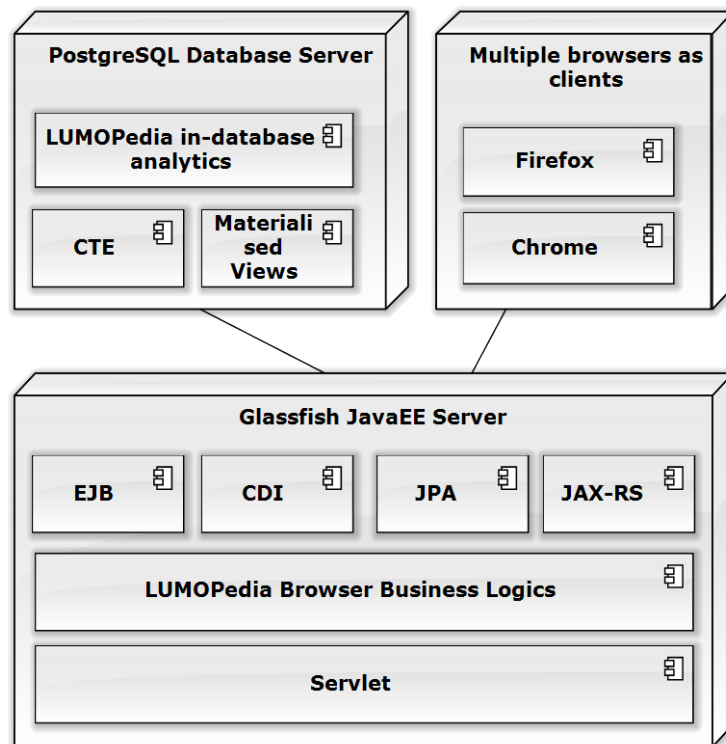


Figure 33: Architecture of the LUMOPedia Browser application

In this new version of LUMOPedia, we have completely re-designed the backend. One of the most significant changes is the application of a relational database to replace the Virtuoso triple store. We have heavily used various in-database techniques to reduce the development complexity and improve the system runtime performance.

For example, in order to calculate the transitive closure of all the instances of a given class, the original solution was calculating them recursively in the application server, i.e. the Glassfish Server. This however requires multiple communication overheads between the application server and the database server. Besides that the SQL queries needed to be parsed multiple times, the query plan must be generated multiple times as well. All these overheads are eliminated in the new design by using one of the in-database techniques – Common Table Expressions (CTE). This modification has significantly improved the performance of the system, decreased the reaction time and reduced the development efforts since most codes are declarative SQL queries now instead of the imperative Java programs.

One of the CTE queries used in the backend to calculate all the instances (direct and indirect) of a given class is given below:

```

CREATE OR REPLACE VIEW kiwi.all_instances AS WITH RECURSIVE
transitive_closure AS
(
    SELECT
        p_subclass.cls1 , p_subclass.cls2 , 1 AS level
    FROM kiwi.p_subclass

    UNION
  
```

```

        SELECT
            t2.cls1 , t1.cls2 , t2.level+1
        FROM kiwi.p_subclass t1
        JOIN transitive_closure t2
            ON t1.cls1=t2.cls2
    )
    SELECT
        t1.cls1 AS id1 , t2.inst AS id2 , t3.name AS cls , t4.name AS
inst , t1.level
    FROM transitive_closure t1
    JOIN kiwi.p_instance t2
        ON t1.cls2=t2.cls
    JOIN kiwi.class t3
        ON t1.cls1=t3.id
    JOIN kiwi.instance t4
        ON t2.inst=t4.id

    UNION
    SELECT
        t1.cls AS id1 , t1.inst AS id2 , t3.name AS cls , t4.name AS
inst , 0 AS level
    FROM kiwi.p_instance t1
    JOIN kiwi.class t3
        ON t1.cls=t3.id
    JOIN kiwi.instance t4
        ON t1.inst=t4.id
;

```

It uses recursive CTE to construct a temporary table and use that table to recursively generate the results table.

9.1.5 Summary

The LUMOPedia knowledge base contains both frontend and backend facilities designed in such a way that it can be used in a modular fashion, i.e., by accessing the provided API functionality. This also means that, for testing, it additionally provides an environment where curated input can be plugged in and visualized on different services as presented via the screenshots. The frontend provides an easy-to-use interface to explore and manage the knowledge while the backend provides various reasoning services in an efficient and scalable manner. Heavily using in-database techniques ensures the system performance and reduces the development efforts significantly.

10 Experimental Explicit User Interaction

The personalization and contextualization part of the LinkedTV ecosystem provides an extra module (LUME) to enable the user to explicitly manage their user models. It is developed as an HTML5 single page application with cutting-edge web technologies to provide maximum system portability and user experience. It is a further development of the LUME editor described in D4.4 section 6.2.

10.1 LUME: the user profile editor

The LinkedTV User Model Editor (LUME) is the further development of the original LUME described in D4.4 section 6.2. The basic functionality “user model management” remains unchanged. However, in order to provide better user experience and ensures seamless integration with the new version of LUMOPedia (see Chapter 9) and the video player, both the frontend and the backend have been redesigned.

10.1.1 System requirements

The detailed system requirements have been elaborated in D4.4 section 6.2. In this section, we will briefly give the core requirements for LUME:

- A user model contains a set of user interests. A user interest consists of an entity (denoted by an IRI) and a weight. The value of the weight is a real number between 0 and 1. It can be -1 denoting that user X dislikes the entity denoted by the IRI.
- Each user interest can be associated with a set of constraints like ‘politician comes from Germany’. Each constraint contains a relation and a value. The type of the value depends on the specification of the relation. Adding constraints to an interest is however optional.
- Users can explicitly define the contexts they want to use to structure their interests. Based on our experiences however, manage the contexts in a manual fashion is difficult for normal users. Therefore in the new version of LUME, only a root context is provided. It is however easy to extend the context support if necessary.

LUME uses the LUMO ontology and the LUMOPedia knowledge base as an information backbone to provide users sufficient information to model their user interests. A revised architecture diagram is illustrated in Figure 34 as a UML deployment diagram.

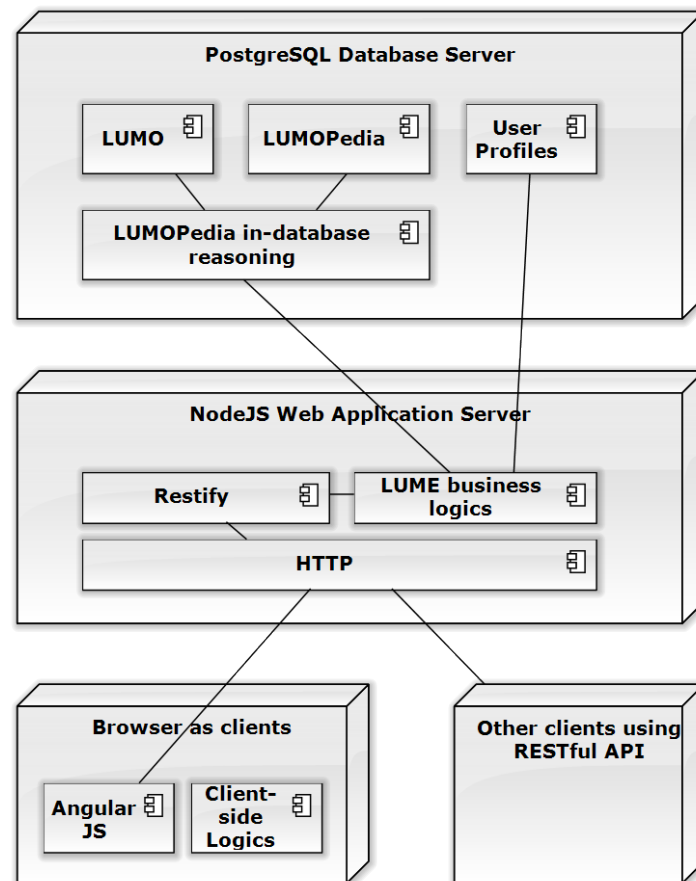


Figure 34: The revised architecture of LUME

10.1.2 HTML5-based frontend design

The new frontend is an HTML5 single page web application. Full CRUD (Create, Read, Update, Delete) operations are supported to manage the user profiles. One screenshot is given in Figure 35. We have chosen the client MVW (Model-View-Whatever) framework AngularJS²² to implement the frontend. This will make the integration with other system components like the video player easier since the single page web application can be embedded in other components with a small amount of efforts. The application can be visited with the URL:

<http://linkedtv1.iais.fraunhofer.de:8888/personal-recommender/#/um/11>

For security reasons, open access is not foreseen for the public. Instead, a custom user account can be created with little effort by contacting the team at Fraunhofer IAIS (<name>.<surname>@iais.fraunhofer.de).

²² <https://angularjs.org/>

10.1.2.1 Manage user models

The user model (UM) consists of a set of user model entries (UME). Each UME has following types of information:

- ID – it is used internally to identify a UME. The ID is not directly visible to the user. It must be distinct.
- Activated – the user can manually activate or deactivate a UME by clicking a checkbox in the user interface. This influences the recommendation results. In the frontend, if a UME is deactivated, it is marked with a lower opacity value (see the second UME in Figure 35. This provides a holistic view about all the user models.
- Name – a readable text which provides more human readable information about the entity in the UME. The “Name” must not be unique (instead we use an internal ID). Besides that, normally there are several names for an entity, e.g. Chancellor Merkel, Angela Merkel, etc. We choose to use an official name for the UME. If a UME is not a single instance but with relation constraints or conjunctions, then the name will be automatically concatenated with “and” or other predefined delimiters.
- Description – this provides detailed information about this entity used in the UME. The descriptions are in natural language and extracted from other LOD datasets. The visibility of this data can be switched on and off by the user through a single click.
- Weight – it denotes how the user likes this UME or dislikes it. We have provided an easy to use widget to let the user interactively and asynchronously change the weight value without reloading the whole page. This needs AJAX support of the browser and the RESTful server of the server side. Disliked UMEs are highlighted with red background colours. One example is the fourth UME – Barack Obama – in Figure 35.

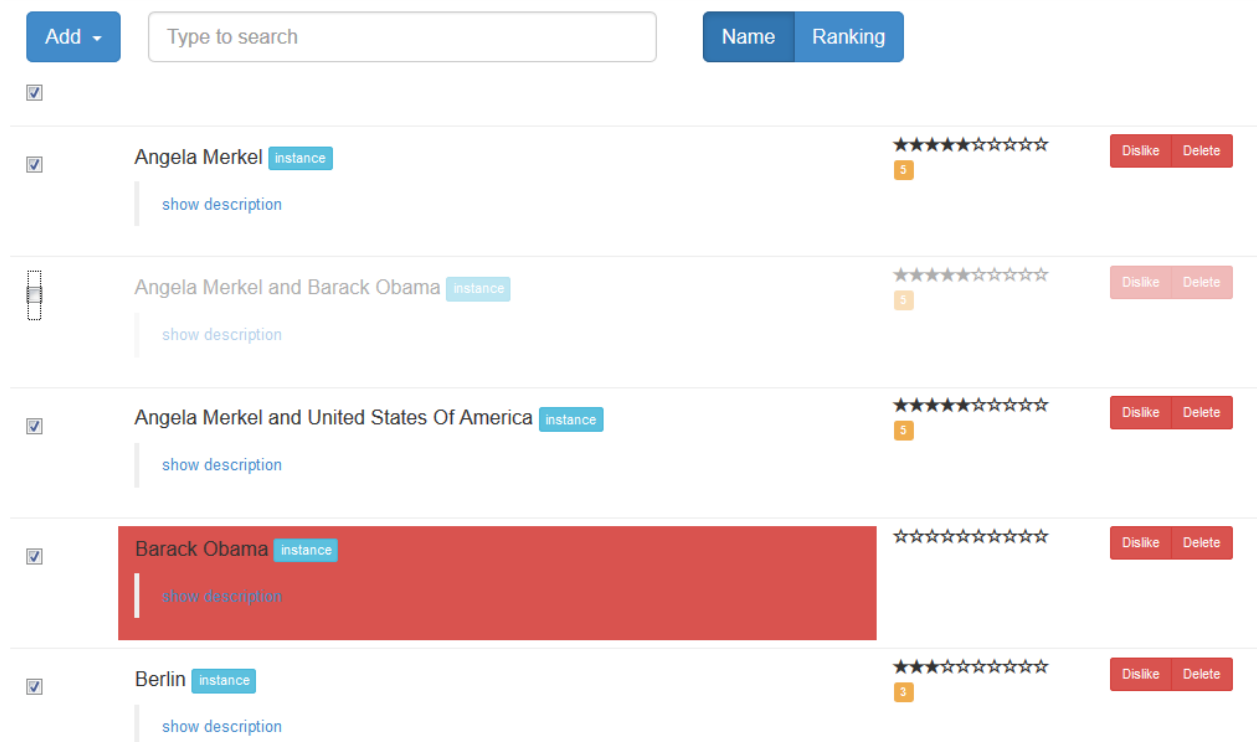


Figure 35: A screenshot of the LUME user profile editor

To add a UME, simply click the “Add” button on the top left of LUME. Two methods are provided to add a UME:

- Structured search under the guidance of autocomplete
- Fully natural language based

Structured search with autocomplete can guide the user to generate a UME by searching in the LUMO ontology and the LUMOPedia knowledge base. The user needs to give the first several letters of the name he/she is interested in. If the LUMO and the LUMOPedia knowledge base have modelled this information in a structured way, it will try to suggest the available entities. For example, if the user gives “ang” in the search box, and since LUMO and LUMOPedia has modelled two semantic entities named “Angela Merkel” and “Angst” which starts with “ang”, those two will be suggested to the user (see Figure 36). After the user selected one, more detailed information will be extracted from the knowledge base and displayed to the user (see Figure 36).

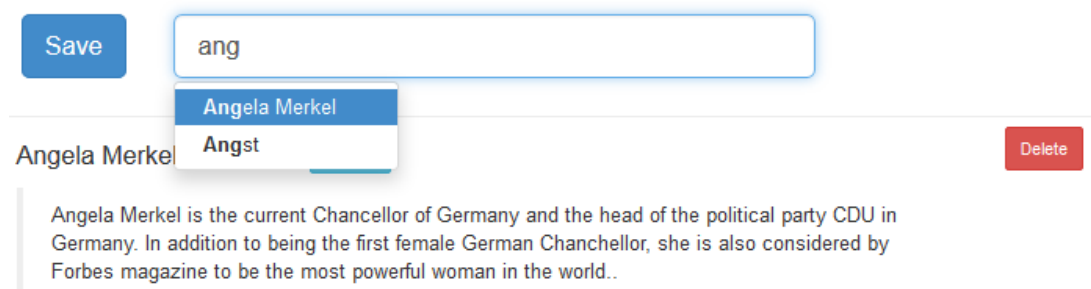


Figure 36: Add an instance as a UME in LUME

The interface will become complicated if the user wants to create UME with constraints. Currently only classes can be assigned constraints. Constraints are based on the properties defined in the LUMO ontology. In this use case however, the Closed World Assumptions are applied to limit the “range” candidates of a given property. For example, after the user has chosen to use the class “politician” as the base entity, and decided to constraint this class with the relation “birth place”. Since the “range” of the property “birth place” is of type “location”, i.e. only instances of the class “location” can be added to the value (see Figure 37). The autocompletion for the letter “h” is limited to five candidates which are all “locations” in the LUMOPedia knowledge base. This feature improves the consistency of the user model and the underlying knowledge base.

The screenshot shows a web interface for creating a User Model Entity (UME). At the top, there is a 'Save' button and a text input field containing 'What are you interested in?'. Below this, a section for 'politician (class)' is visible, with a 'Delete' button and an 'Add relation' button. Two dropdown menus are shown: 'has spouse' and 'birth place'. The 'has spouse' dropdown is set to 'Annette Schavan'. The 'birth place' dropdown is set to 'h', which has triggered an autocomplete list showing 'Hamburg', 'Hanover', 'Hesse', 'Hagen', and 'Halle'. Below the dropdowns, there is a 'Contact: rued' field and a 'Frau' field.

Figure 37: Add a class with constraints as a UME in LUME

The natural language processing (NLP) based solution relies on the services provided by TextRazor²³. The results of TextRazor will be aligned at runtime with the LUMO ontology and LUMOPedia to provide more semantic information. It provides other means for the user to express their interests. To use it, just give the sentence which describes the interest and press the “Process” button. For example, the sentence “german politician in spain” will provide model suggestion with three entities “Germany”, “politician” and “Spain” and a handful of the relations (see Figure 38) which are aligned with the knowledge base.

The screenshot shows the NLP interface. At the top, there is a text input field containing 'german politician in spain' and a 'Process' button. Below this, a section titled 'Which ones do you mean?' displays the results. It shows three entities: 'Germany', 'politician', and 'Spain'. Below these entities, there are several relations listed in green boxes: 'politician birth place Germany', 'politician lives at Germany', 'politician leads Germany', 'politician is member of Germany', 'politician birth place Spain', 'politician lives at Spain', 'politician is member of Spain', and 'politician leads Spain'.

Figure 38: Natural Language interface in LUME

²³ <https://www.textrazor.com/>

In order to eliminate the semantic misunderstandings and consolidate the model generation, the user needs to choose and confirm the “structured” user profiles. For example, by clicking on one of the model constraints (in green), the class “politician” is automatically used as the entity and the constraints are added automatically as well Figure 39.

The screenshot shows the LUME interface. At the top, there is a search bar with the text "What are you interested in?" and a blue "Process" button. Below the search bar, a dropdown menu is open with the title "Which ones do you mean?". It contains several green buttons with text: "politician birth place Germany", "politician lives at Germany", "politician leads Germany", "politician is member of Germany", "politician birth place Spain", "politician lives at Spain", "politician is member of Spain", and "politician leads Spain". Below this, the class "politician" is shown with a blue "class" label and a red "Delete" button. Underneath, the attributes "birth place" and "lives at" are listed, both with the value "Spain".

Figure 39: Eliminate the semantic misunderstanding by selecting the structured information

To remove a UME in LUME, simply click the “delete” button which is on the same row with the UME. A confirmation popup dialog will be displayed to avoid any unintended delete operations (see Figure 40).

The screenshot shows a confirmation popup dialog. The dialog has a title "Sure?" and a message "You want delete Berlin?". At the bottom, there are two buttons: a green "Ok" button and a red "Cancel" button. In the background, the LUME interface is visible, showing a list of user model entries. The first entry is "Angela Merkel" with a blue "instance" label and a "show description" button. The second entry is "Angela Merkel and E" with a blue "instance" label and a "show description" button. To the right of the list, there are two rows of five stars each, with "Dislike" and "Delete" buttons next to them.

Figure 40: The confirmation popup dialog for the deletion of a UME

To enable the fast navigate in a large set of user model entries, filtering and sorting can be used. We provide a search box on the top of LUME. If the user types any letters in the search box, the UME list will be instantly filtered and refreshed (see Figure 41, the list is now filtered by the pattern “ang”). Similarly, the whole list of the UMEs can be sorted by the name or by the ranking/weight. By clicking one of the buttons (“Name” and “Ranking”), the list will be filtered based on the client-side logics. No interaction with the server is needed. This provides the maximum reaction performance and minimised the network delay.

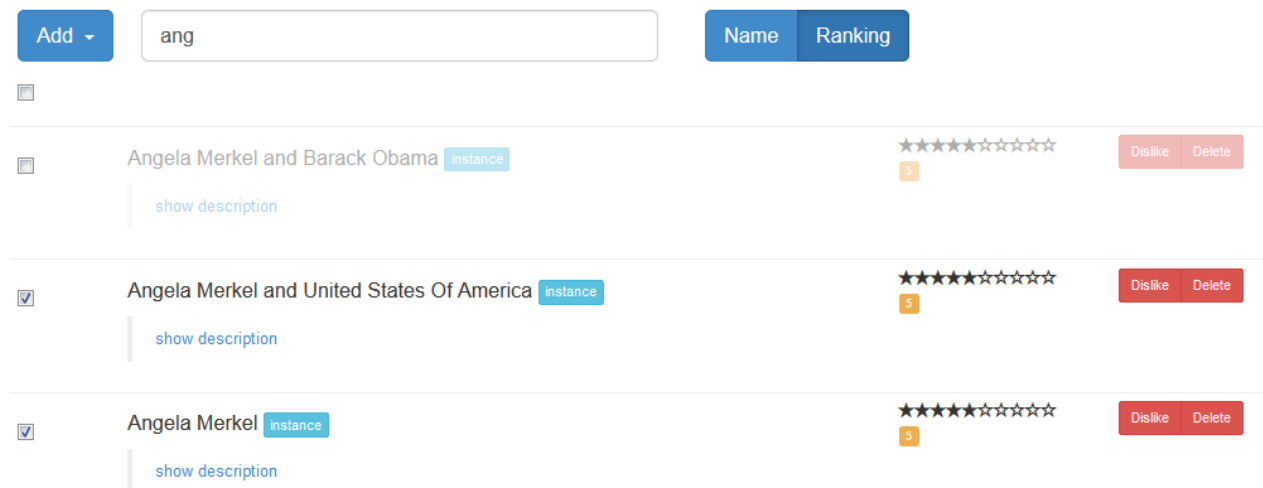


Figure 41: Fast filtering and ranking the UME list

10.1.3 NodeJS powered service layer

As illustrated in Figure 34, the application server is based on NodeJS²⁴ which is a server-side JavaScript runtime environment based on Google V8 engine²⁵. The decision to replace the original Java-based server layer with the NodeJS-based services is based on the following considerations:

- Most of our time-consuming logics are pushed into the DBMS to minimise the network latency and reduce the imperative Java code. Therefore we need a “thin” service layer. Comparing to the NodeJS-based solutions, the Java service stack is far too heavy and overkilled in this case.
- Direct JSON support. We have extensively used JSON for the web services. JSON is intrinsically part of the JavaScript language. There is little overhead to serialise and de-serialise the JSON object. Java has good support for JSON encoding, however for our purpose, Server-Side JavaScript is much more lightweight and more suitable.
- Functional-based Non-blocking paradigm. Both Java and Server-Side JavaScript requires a virtual machine as the runtime environment. Functions in JavaScript are however first-class citizens and the non-blocking call-backs are more suitable for services-oriented applications.

The service layer in the current version of LUME is much “thinner” than the one described in D4.4 section 6.2. The development itself is agile and test-driven which has greatly reduced the development efforts and guaranteed the quality of the service. All the services are pro-

²⁴ <http://nodejs.org/>

²⁵ <https://code.google.com/p/v8/>

tected with HTTP Basic Authentication which provides a certain level of security of user privacies. To summarise, all the services implemented in this layer²⁶ is listed in Table 16.

Table 16: The list of all RESTful services implemented in LUME service layer

| URL | HTTP Verb | Description |
|------------------------------|--------------|---------------------------------|
| /pr/user/:user_id | GET/POST/DEL | Get, create and delete a user |
| /pr/user | GET | Get the list of all users |
| /pr/ume | POST | Create a new UME |
| /pr/ume/:id/weight | PUT | Update the weight of a UME |
| /pr/ume/:id/active | PUT | Activate or deactivate a UME |
| /pr/um/:id/active | PUT | Activate or deactivate all UMEs |
| /pr/ume/:id | GET/DEL | Get or delete a UME |
| /kiwi/suggestion | GET | Get suggests from the KB |
| /kiwi/entity/:id/description | GET | Get the description of a UME |
| /pr/um/:ctx_id | GET | Get all UMs of a given context |
| /pr/class/:cid/relation | GET | Get relations of a given class |

10.1.4 Data Management with PostgreSQL

All the user models / profiles are stored in the relation database PostgreSQL with the schema illustrated in Figure 42. Foreign keys are defined to ensure the consistency of the whole dataset. The database schema is strong normalised to reduce possible redundancies. To clearly separate the logics of data manipulation from the client side logics, most of the data-related operations are encapsulated inside of the database and exposed to the application server/service layer through database user defined functions (UDFs). This design methodology has greatly simplified the development of the service layer and kept it easy to be extended and maintained.

For example, in order to expose the user model entries of a given context, we have defined a UDF `ltv.get_user_model_conj` which accepts a parameter the `context id` and returns the list of all UMEs of the given context. This design hides the whole complexity of the SQL query and makes the database schema transparent to the application logics. That means, in the future, if the database schema needs to be changed, the application code however does not needs any modification. This design de-couples both software systems.

²⁶ These services are however not intended for other partners. They are designed to be consumed only by the LUME editor.

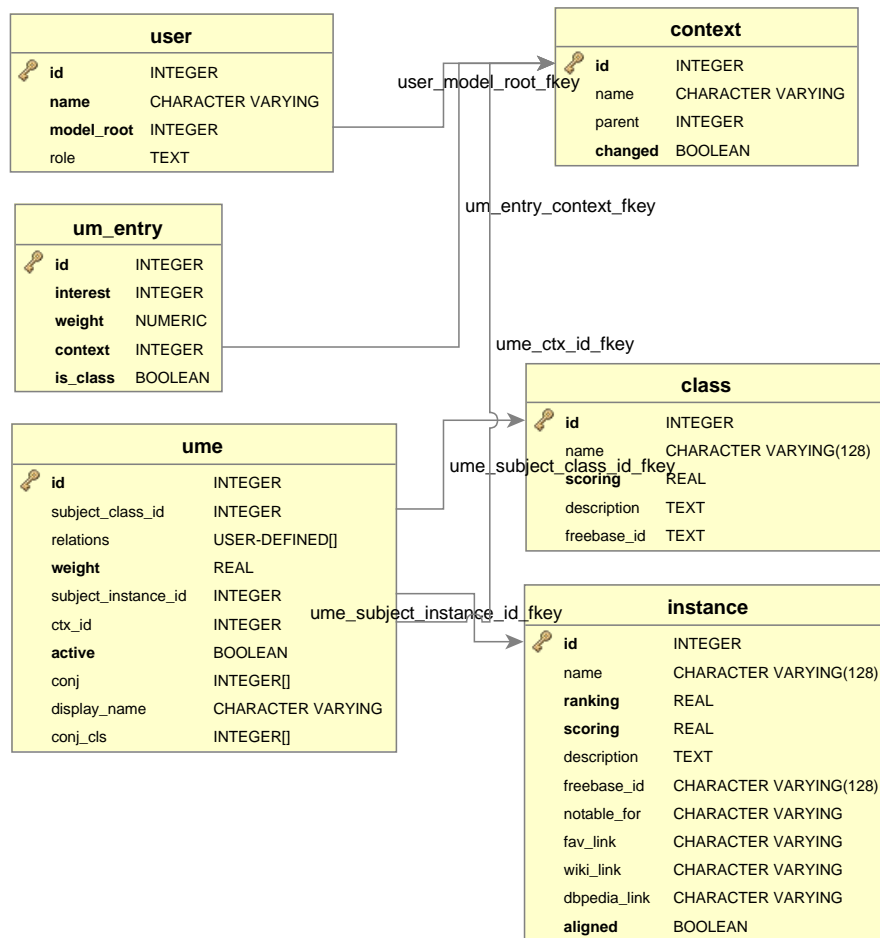


Figure 42: The database schema for storing the user models

10.1.5 Summary

LUME is the frontend user interface for the users to manage their user profiles in an intuitive and easy-to-use way. It is developed as an HTML5 single page application which provides systems based on cutting-edge client-side MVW technologies. On the backend, the NodeJS-powered service layer provides a thin and easy-to-maintain software system. The data storage layer based on the relational database provides a robust and reliable infrastructure to manage the user profiles.

11 Experimental Recommendation

Effective and fast recommendation generation plays a central role in large-scale content consumption systems like online video platform and distributed advertising systems. Utilising the semantics hidden in the contents is an attractive and challenging task for both academia and industry. In LinkedTV, in addition to LiFR already detailed in previous sections, the optional personal recommender was also implemented (see Figure 2, LSF module). Personal Recommender is basically a smaller extension to the LSF described in D4.5 section 3.1. Since this is an experimental branch only, the main changes with regards to D4.5 were made on the interface level and development of the graph matching concept. Further, we conducted preliminary tests with a video base containing more than 2500 videos from the LinkedTV context to test recommendation and scalable performance.

11.1 Personal Recommender

Personal Recommender is largely based on the LSF described in D4.5 section 3.1, and features a graph matching concept, performing the whole matching process inside of the relational DBMS. It contains a semantic recommendation engine focusing on accuracy, efficiency and scalability. The in-database recommendation engine leveraging various database features - indexing, clustering, partitioning, materialized views, etc. - to provide efficient reasoning on top of the structured data available in the Linked Open Data cloud.

11.1.1 System design

Basically we use similar system design architecture as the LUME application. The UML deployment diagram is illustrated in Figure 43. PostgreSQL powers the database storage and in-database analytics. NodeJS delivers the core infrastructure for the server layer. AngularJS is responsible for the client CRUD logics.

11.1.1.1 Incrementally Building the Knowledge Base

The first step to generate the recommendations is to incrementally construct the knowledge base based on the available recommended media items – in our case it is the videos. The most two important considerations to establish such a knowledge base are 1) it should be comprehensive and consistent enough for the targeted domain and 2) it can be accessed efficiently and should scale up to hundreds of millions of facts for complex reasoning tasks. This is the construction of the LUMOPedia knowledge base which has been described in Chapter 9.

Till now, the knowledge base consists of about 8000 instance assertions interrelated with around 6000 property assertions (more statistics see the Table 15). These assertions are incrementally imported from Linked Open Data (Freebase, Yago, DBPedia, etc.) automatically based on the LUMO ontology. This knowledge base is materialised in a relational database based on the schema introduced in Chapter 9. The reasoning performance scales very

well. Typical reasoning tasks like “All politicians from France whose birthday is between 1940 and 1990” can deliver results within several milliseconds.

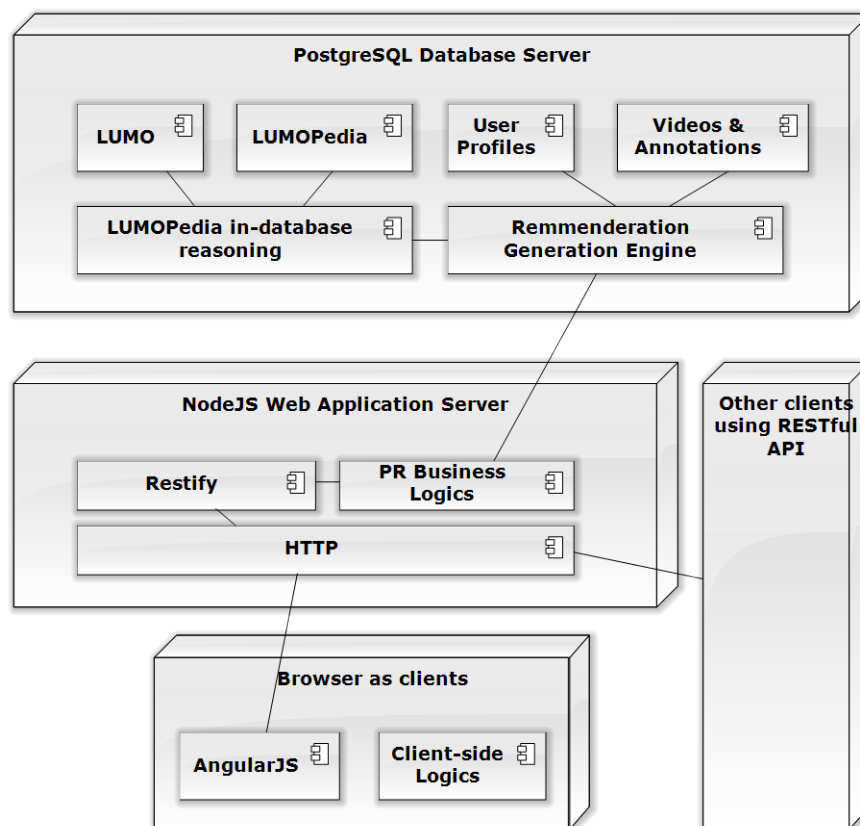


Figure 43: UML deployment diagram of the Personal Recommender

11.1.1.2 Materialise the Semantics via Enrichment

To enable fast matching between user models and media content the semantic closure of media item annotations are pre-calculated and indexed in Personal Recommender. We call this process of personalized enrichment only “enrichment” in a scope of simplification, but this enrichment is different from the one in WP2. The enrichment process is deterministic, i.e. as long as the media items do not change and the knowledge base keeps static, the enrichment results for any annotation are always the same. Constantly recalculating the enrichment is not necessary. On the other side, the enrichment process is quite time-consuming: in our test case with 60,000 annotations it takes hours to fully generate the semantic closures and materialise them into the database. Pre-calculating and indexing the enrichment is a key factor for the performance improvement.

Within the knowledge base, the enrichment for each instance is computed as the recursive set of parent classes (shown in Figure 44 in grey) weighted by the branching distance in addition to the set of related instances (shown in Figure 44 in blue). Figure 44a represents a subset of the LUMOPedia knowledge base where circles represent classes and squares represent instances. The goal in this example is to enrich the red instance with the help of Personal Recommender-based ontological reasoning. In Figure 44b, all the parent classes of the specified instance are recursively considered till the class ontology root. This is done by in-

voicing one of the Personal Recommender reasoning services. In Figure 44c, all the other instances (blue) in the knowledge base related to the specified instance are considered by invoking another Personal Recommender reasoning service. The final enrichment result for the given instance is shown in Figure 44d.

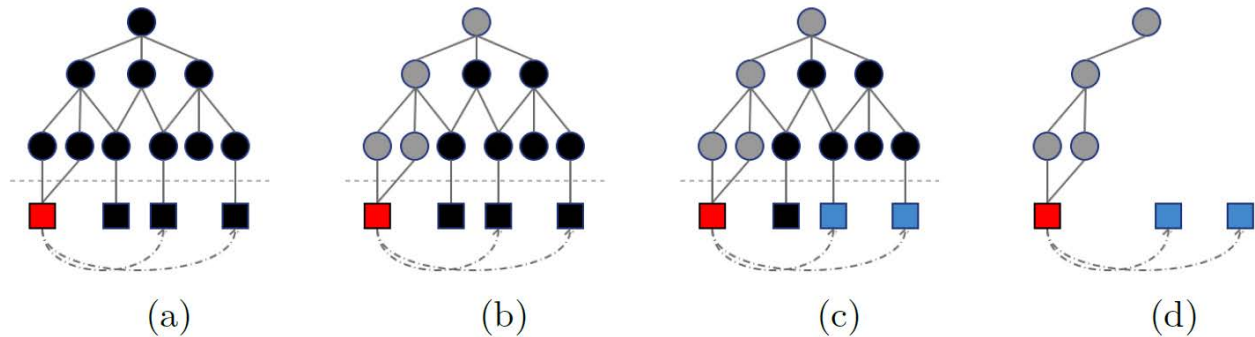


Figure 44: The illustration of the enrichment process for an instance

11.1.1.3 Semantic Recommendation Generation

The recommendation list is generated by semantically comparing the similarity between user models and the media items. The LUMOPedia knowledge base is used as a “semantic bridge” which connects both sides not only by the literal similarity (like “cat” and “cats”) but also the underlying meanings like (“president” and “Obama”). In the end, the media items with higher rankings are recommended to the user.

The ranking of a given media item in the recommendation list is generated by correlating the interest weight of each user model entity to all the annotations of the media item. Different similarity functions are defined in Personal Recommender. The recommender system can use any of them or any combination of these similarity functions to generate the final recommendations. One example of the similarity function to calculate the ranking of a media item M for a given user U is defined as

$$\text{Ranking}(M, U) = \sum_{i=1}^n \max_{j=1}^m (W_U^{e_i} * C_M^{a_j})$$

where $W_U^{e_i}$ is the weight of the i -th user model entry in the given user model U ; and $C_M^{a_j}$ is the confidence of the j -th annotation for the given media item. Based on the relational schema introduced in section 3.2, we can simply write the following SQL queries in a declarative fashion to generate the top 10 rankings for the user “Peter”:

```
WITH t AS (
  SELECT ume.id, ema.media, max(ume.weight * ema.confidence) ranking
  FROM user_model_entry ume INNER JOIN extended_media_annotation ema
  USING entity_id
  WHERE ume.user = 'Peter'
  GROUP BY ume.id, ema.media
)
SELECT t.media, sum(ranking) ranking
FROM t
```

```
GROUP BY id
ORDER BY ranking DESC
FETCH FIRST 10 ROWS ONLY
```

11.1.1.4 The sample video base

In order to test the performance and effectiveness of Personal Recommender, we have got a sample video base with Speech-To-Text annotations. This video base contains over 2500 videos. To map the texts to the LOD entities, the TextRazor annotation web services²⁷ are used. We have implemented a small Java-based utility tool to consume the service and batch process the Speech-To-Text outputs. In the end, we get over 40 000 video annotations which are mapped to over 8 500 entities in LUMOPedia knowledge base. The top 30 LUMOPedia entities which are used most frequently in this sample video base are illustrated in Figure 45. One remark to this diagram is that the entity “oder” (second one from the left) is actually a false positive introduced by TextRazor. The semantics of this entity is actually a river in Germany, however in the annotations this is not the case.

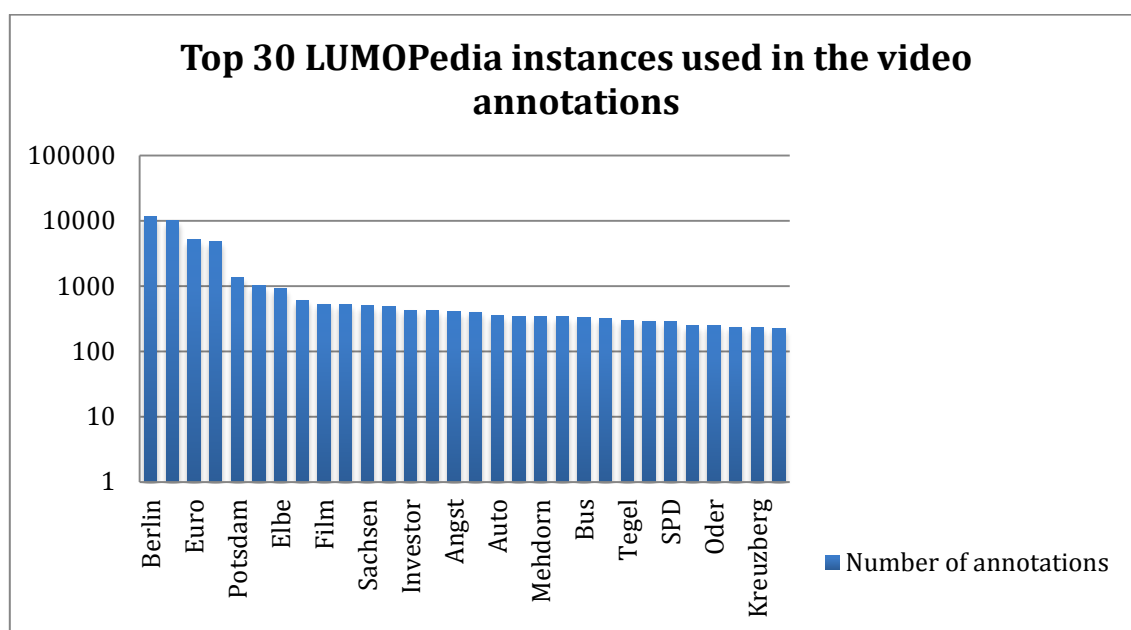


Figure 45: Top 30 LUMOPedia instances used in the video annotations

11.1.1.5 Related web contents

Delivering related web contents based on the currently watched video frames is one feature in Personal Recommended. It is also one of features highlighted by the LinkedTV ecosystem. Personal Recommender currently has implemented the interface to Flickr²⁸, Wikipedia²⁹ and

²⁷ <https://api.textrazor.com/>

²⁸ <https://www.flickr.com/services/api/>

Twitter³⁰. To ease the development, we have used the Popcorn.JS³¹ to interact with the HTML5 videos and extract the related web contents on the fly.

11.1.2 Personal Recommender – the prototype frontend

The prototype frontend of Personal Recommender is a HTML5-based single page application. It integrated a simplified video player to test the recommendation functionalities. As illustrated in Figure 43, NodeJS powers the server side and PostgreSQL is used as the data storage and reasoning facility. One screenshot of the Personal Recommender frontend is given in Figure 46 and the application can be visited with the URL:

<http://linkedtv1.iais.fraunhofer.de:8888/personal-recommender/>

The video base can be accessed by visiting the following URL:

<http://linkedtv1.iais.fraunhofer.de:8888/personal-recommender/#/vb>

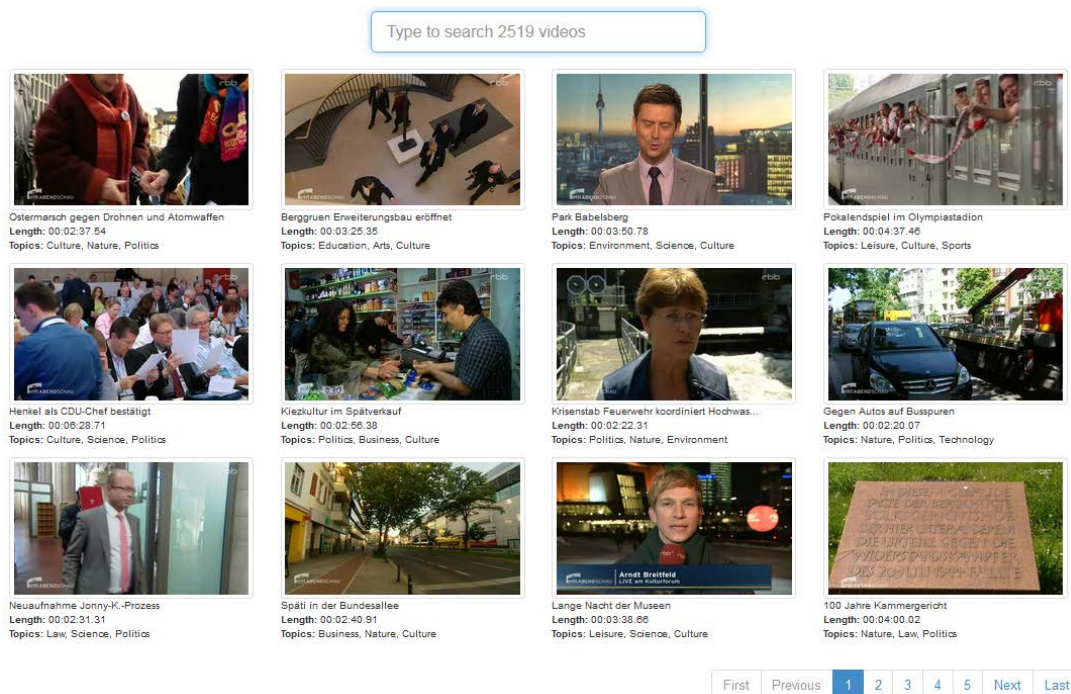


Figure 46: One screenshot of the video base displayed in the Personal Recommender frontend

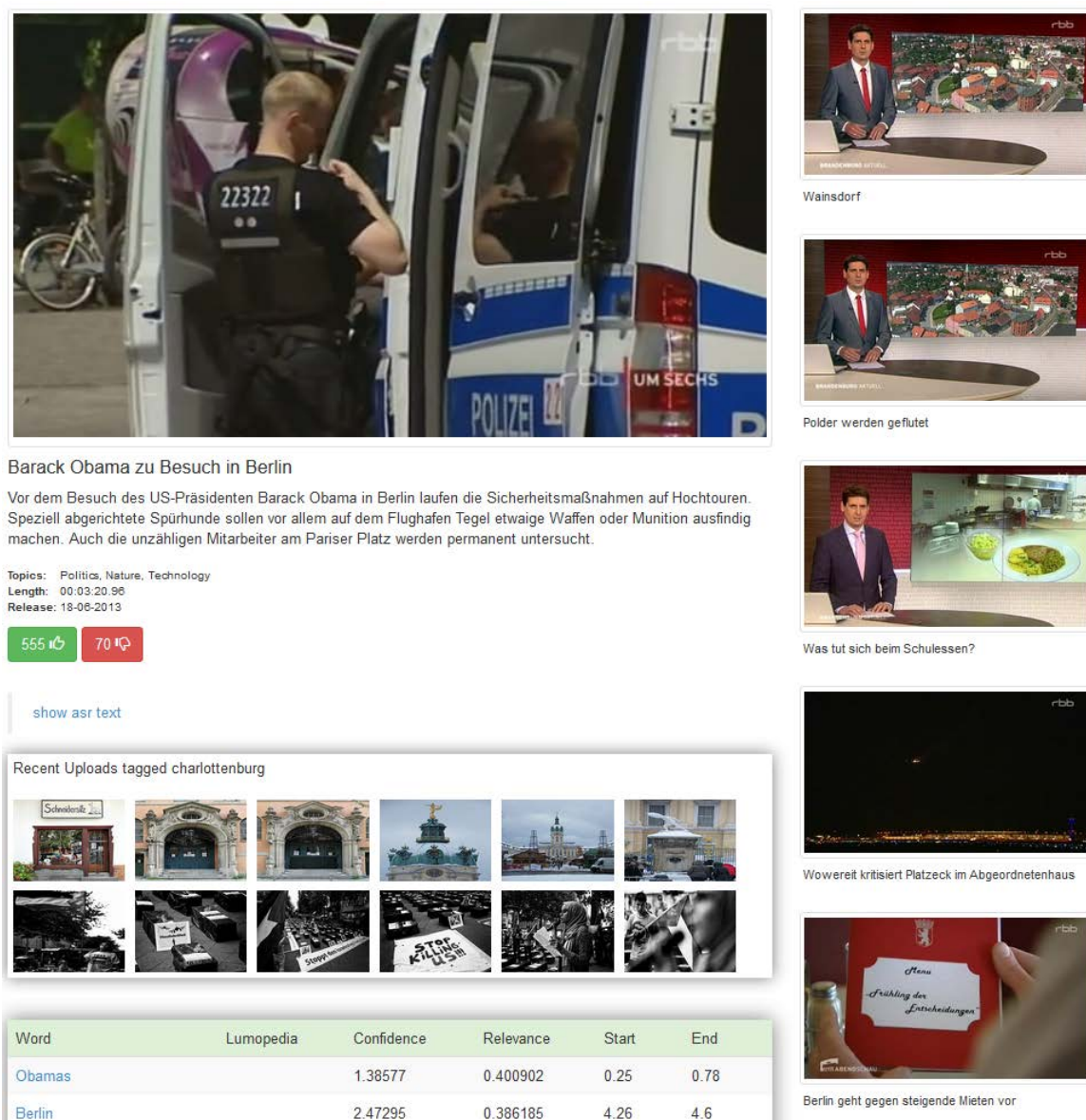
One screenshot of the video base is depicted in Figure 46. There are currently 2519 videos and interactive full-text search is supported to explore the video base. Pagination is provided at the bottom with each page 12 videos. The title, the length and the topics are displayed directly under the poster image of the video.

²⁹ http://www.mediawiki.org/wiki/API:Main_page

³⁰ <https://dev.twitter.com/docs/api>

³¹ <http://popcornjs.org/>

Depends on the user model created in the LUME editor (see section 10.1), the video recommendation list is generated on the fly with the in-database analytic techniques. Therefore no caching mechanism is needed now for the current video base.



The screenshot displays a video player interface with a main video and several related content thumbnails on the right. The main video is titled "Barack Obama zu Besuch in Berlin" and shows a police officer in a vest with the number 22322. The thumbnails include news segments from rbb (Radio Berlin Brandenburg) with titles like "Wainsdorf", "Polder werden geflutet", "Was tut sich beim Schussen?", "Wovoreit kritisiert Platzzeit im Abgeordnetenhaus", and "Berlin geht gegen steigende Mieten vor".

Below the main video, there is a section titled "Recent Uploads tagged charlottenburg" showing a grid of images. Below this is a table with the following data:

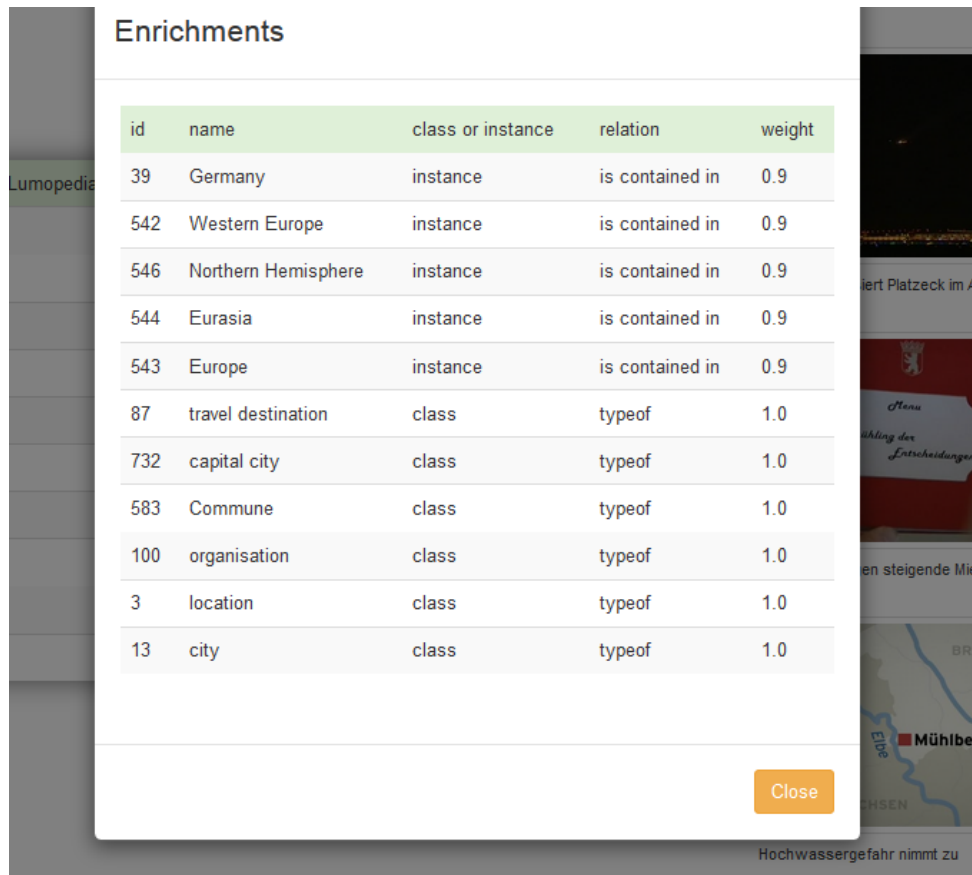
| Word | Lumopedia | Confidence | Relevance | Start | End |
|--------|-----------|------------|-----------|-------|------|
| Obamas | | 1.38577 | 0.400902 | 0.25 | 0.78 |
| Berlin | | 2.47295 | 0.386185 | 4.26 | 4.6 |

Figure 47: One screen shot of the Personal Recommender

Related web contents are displayed during the video plays. At each time stamp, the client side logics checks if there are related web contents registered. If yes, it tries to connect to the server like Twitter – with CORS³² enabled - and extract the needed information on the fly. There are some delays but still acceptable. For example, in Figure 47 the images about “Charlottenburg” are fetched from the Flickr server and displayed in the frontend – all of these are synchronised with the video playing.

³² Cross Origin Resource Sharing (CORS) is a standard technique in the web to enable browser-based distributed resource and service consumption.

On the right side of the video currently playing are the related videos (see Figure 47) which are generated based on the current watched video and the user model. Below the related web content is a list of personalised video entities. This is more for debugging purpose. After one click on these entities, the enrichments which are introduced earlier are displayed in a popup dialog (see e.g. Figure 48 shows the enrichments of the entity “Berlin” in the LUMO-Pedia knowledge base). The list contains the internal ID, the name, the type, the connected relation/property and the weight of the enrichments.



| id | name | class or instance | relation | weight |
|-----|---------------------|-------------------|-----------------|--------|
| 39 | Germany | instance | is contained in | 0.9 |
| 542 | Western Europe | instance | is contained in | 0.9 |
| 546 | Northern Hemisphere | instance | is contained in | 0.9 |
| 544 | Eurasia | instance | is contained in | 0.9 |
| 543 | Europe | instance | is contained in | 0.9 |
| 87 | travel destination | class | typeof | 1.0 |
| 732 | capital city | class | typeof | 1.0 |
| 583 | Commune | class | typeof | 1.0 |
| 100 | organisation | class | typeof | 1.0 |
| 3 | location | class | typeof | 1.0 |
| 13 | city | class | typeof | 1.0 |

Figure 48: The enrichments of the entity "Berlin"

11.1.3 The RESTful web services

In order to support the inter-operability between software components, and in particular with the LinkedTV platform and the other personalization and contextualisation services, Personal Recommender provides RESTful web services to access the contexts and user models.

The following table give more details about the RESTful web services:

| POST a user | |
|---------------------|---|
| Description: | Insert a new user and automatically create a new context for this user. |
| Pattern: | /pr/user/{user_id} |
| HTTP method: | POST |

| | |
|------------------------|---|
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | curl -X POST 'http://linkedtv1.iais.fraunhofer.de:8888/pr/user/4711' -u {username:password} |
| Output: | <pre>{ "msg": "ok" }</pre> |

| GET a user | |
|------------------------|--|
| Description: | Get the list of all contexts of the given user. |
| Pattern: | /pr/user/{user_id} |
| HTTP method: | GET |
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | curl -X GET 'http://linkedtv1.iais.fraunhofer.de:8888/pr/user/4711' -u {username:password} |
| Output: | <pre>[{ "context_id": 1008 }]</pre> |

| DELETE a user | |
|------------------------|---|
| Description: | Delete the given user. |
| Pattern: | /pr/user/{user_id} |
| HTTP method: | DELETE |
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | curl -X DELETE 'http://linkedtv1.iais.fraunhofer.de:8888/pr/user/4711' -u {username:password} |
| Output: | <pre>{ "msg": "ok" }</pre> |

| GET recommended videos | |
|------------------------|--|
| Description: | Get the list of all recommended videos of the given context. |

| | |
|------------------------|--|
| Pattern: | /pr/rcm/{context_id} |
| HTTP method: | GET |
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | curl -X GET 'http://linkedtv1.iais.fraunhofer.de:8888/pr/rcm/1007' -u {username:password} |
| Output: | <pre>[{ "video_id": 814, "video_title": "aktuell_20130307_cebit_m_16_9_512x288", "video_label": "Sch\u00fclerergebnistag auf der CeBIT", "video_summary": "Ein Besuch auf der gr\u00f6\u00dftesten Computermesser...", "video_score": 0.93 }, ...]</pre> |

| GET related videos | |
|------------------------|---|
| Description: | Get the list of all related videos of the given context and video. |
| Pattern: | /pr/related/{context_id}/{video_id} |
| HTTP method: | GET |
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | curl -X GET 'http://linkedtv1.iais.fraunhofer.de:8888/pr/related/1007/814' -u {username:password} |
| Output: | <pre>[{ "video_id": 783, "video_title": "aktuell_20130304_cebit_m_16_9_512x288" "video_label": "Brandenburger auf der CeBIT", "video_summary": "Dienstag \u00f6ffnet die CeBIT in Hannover...", "video_score": 0.45 }, ...]</pre> |

| GET media fragments | |
|---------------------|--|
| Description: | Get the list of all entities with additional information (media fragments) of the given video. |
| Pattern: | /pr/mfa/{video_id} |
| HTTP method: | GET |

| | |
|------------------------|--|
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | <code>curl -X GET 'http://linkedtv1.iais.fraunhofer.de:8888/pr/mfa/814' -u {username:password}</code> |
| Output: | <pre>[{ "entity_id": 13852, "entity_label": "Hannover", "wiki_link": "http://en.wikipedia.org/wiki/Hanover", "wiki_name": "Hanover", "entity_confidence": 2.07348, "entity_relevance": 0.197577, "entity_time_start": 7.59, "entity_time_stop": 8.2, }, ...]</pre> |

| GET freebase id | |
|------------------------|--|
| Description: | Get the freebase id and the wikipedia link of the given kiwi id. |
| Pattern: | <code>/pr/fid/{kiwi_id}</code> |
| HTTP method: | GET |
| Content-Type: | JSON |
| Authentication: | You need an account for the Personal Recommender from Fraunhofer. |
| Example: | <code>curl -X GET 'http://linkedtv1.iais.fraunhofer.de:8888/pr/fid/319' -u {username:password}</code> |
| Output: | <pre>[{ "freebase_id": "m.03pbf", "wiki_link": "http://en.wikipedia.org/wiki/Hanover" }]</pre> |

11.1.4 Summary

Personal Recommender is a prototype implementation of the graph matching based recommendation methods and finalizes the implementation of LSF introduced in D4.5. It improves the system accuracy and performance by manually curating the knowledge base and heavily using the in-database analytic techniques. The thin service layer design makes the system more maintainable and easier for further extensions.

12 Conclusions & Future Work

The deliverable shows both an extended framework for contextualization and personalization (Figure 2) which is available for conceptual testing and a core framework which is in process of implementation (Figure 1). This pipeline is final and the communication between the different bricks are defined and already implemented or in process of implementation. The way the different modules work is also finalized and the entire framework is able to provide contextualized features and personalization to the two personalization-aware scenarios described in chapter 2. In addition, the contextual features are already used in the third artistic scenario which is “context-based” only.

After finishing the last implementation testing steps, the pipeline will be ready to be tested in different situations with two different scenarios. The results of those tests will be detailed in the next deliverable (D4.7) which deals with the system validation.

13 Bibliography

- [BAA03] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
- [BIE13] Johannes Hoffart, Fabian Suchanek, Klaus Berberich, Gerhard Weikum: YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. Special is-sue of the Artificial Intelligence Journal, 2013
- [BRO03] Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: An architecture for storing and querying rdf data and schema information. In: Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W. (eds.) Spinning the Semantic Web. pp. 197–222. MIT Press (2003)
- [EUR] Europeana database : <http://www.europeana.eu/>
- [FAC] Face Tracking. Microsoft Developer Network. <http://msdn.microsoft.com/en-us/library/ji130970.aspx>
- [FOAF] FOAF: Friend-of-a-friend Project (March 2014), <http://www.foaf-project.org/>
- [GON13] Gonzalez-Jorge, H., Riveiro, B., Vazquez-Fernandez, E., Martínez-Sánchez, J., & Arias, P. (2013). Metrological evaluation of microsoft kinect and asus xtion sensors. Measurement, 46(6), 1800-1806.
- [GRU93] Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2), 199–220 (Jun 1993), <http://linkinghub.elsevier.com/retrieve/doi/10.1006/knac.1993.1008>
- [HAW05] Hawkins, R.P., Pingree, S., Hitchon, J., Radler, B., Gorham, B.W., Kahlor, L., Gilligan, E., Serlin, R.C., Schmidt, T., Kannaovakun, P., Kolbeins, G.H. 2005. What Produces Television Attention and Attention Style? Genre, Situation, and Individual Differences as Predictors. Human Commun. Research 31(1), 162–187
- [HOF13] J. Hoffart, F. M. Suchanek, K. Berberich, G. Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence, Volume 194, January 2013, Pages 28–61 <http://dx.doi.org/10.1016/j.artint.2012.06.001>
- [KAU13] Kaufmann, M., Manjili, A.A., Vagenas, P., Fischer, P.M., Kossmann, D., Färber, F., May, N.: Timeline index: a unified data structure for processing queries on temporal data in sap hana. In: Ross, K.A., Srivastava, D., Papadias, D. (eds.) SIGMOD Conference. pp. 1173–1184. ACM (2013)
- [KIN10] Microsoft. Kinect sensor. <http://www.xbox.com/kinect>
- [KUC13] Kuchař, J. Kliegr T. GAIN: web service for user tracking and preference learning – a SMART TV use case. In Proceedings of the 7th ACM Recommender Systems Conference (RecSys 2013), Hong Kong, China. October 2013.
- [KUC14] Kuchař, J. Kliegr T. InBeat: News Recommender System as a Service @ CLEF-NEWSREEL'14. To appear.
- [KUL00] Kuleshov, V., Precup, D. Algorithms for the multi-armed bandit problem. Journal of Machine Learning Research. 2000
- [LEH14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, Christian Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. To appear in the Semantic Web Journal (2014).

- [MAT06] Matuszek, C., Cabral, J., Witbrock, M.J., DeOliveira, J.: An introduction to the syntax and content of cyc. In: AAAI Spring Symposium: Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering. pp. 44–49. AAAI (2006)
- [MCB02] McBride, B.: Jena: A semantic web toolkit. IEEE Internet Computing 6(6), 55–59 (2002)
- [MCG04] McGuinness, D.L., Van Harmelen, F., et al.: Owl web ontology language overview. W3C recommendation 10(2004-03), 10 (2004)
- [MOT09] Motik, B., Patel-Schneider, P.F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al.: Owl 2 web ontology language: Structural specification and functional-style syntax. W3C recommendation 27, 17 (2009)
- [MOVO] Ontology, M.: IntelLEO Movies Ontology (March 2014), <http://www.intelleo.eu/ontologies/vw/movies/spec/>
- [NIL01] Niles, I., Pease, A.: Towards a standard upper ontology. In: FOIS. pp. 2–9 (2001)
- [NOY00] Noy, N.F., Fergerson, R.W., Musen, M.A.: The knowledge model of protégé-2000: Combining interoperability and flexibility. In: Dieng, R., Corby, O. (eds.) EKAW. Lecture Notes in Computer Science, vol. 1937, pp. 17–32. Springer (2000)
- [OAE] Initiative, O.A.E.: Ontology Alignment (March 2014), <http://oaei.ontologymatching.org/>
- [QUA] Qualisys. Products and services based on optical motion capture. <http://www.qualisys.com/>
- [RAI07] Raimond, Y., Abdallah, S.A., Sandler, M.B., Giasson, F.: The music ontology. In: Dixon, S., Bainbridge, D., Typke, R. (eds.) ISMIR. pp. 417–422. Austrian Computer Society (2007)
- [REI77] Reiter, R.: On closed world data bases. In: Logic and Data Bases. pp. 55–76 (1977)
- [RUS10] Russell, S.J., Norvig, P.: Artificial Intelligence - A Modern Approach (3. internat. ed.). Pearson Education (2010)
- [SNO99] Snodgrass, R.T.: Developing Time-Oriented Database Applications in SQL. Morgan Kaufmann (1999).
- [TSA14a] D. Tsatsou, V. Mezaris, "LUMO: The LinkedTV User Model Ontology", Proc. Extended Semantic Web Conference (ESWC'14), poster track, Anissaras, Crete, Greece, May 2014.
- [TSA14b] D. Tsatsou, S. Dasiopoulou, I. Kompatsiaris, V. Mezaris, "LiFR: A Lightweight Fuzzy DL Reasoner", Proc. Extended Semantic Web Conference (ESWC'14), poster track, Anissaras, Crete, Greece, May 2014.
- [WYR] Wyrembelski, Adam. "Detection of the selected, basic emotions based on face expression using Kinect."