Graph-based Context Management and Personalized User Recommendations in a Smart TV environment

Fotis Aisopos[i], Angelos Valsamis[ii], Alexandros Psychas[i], Andreas Menychtas[i] and Theodora Varvarigou[i]

[i]Distributed, Knowledge and Media Systems Group, National Technical University of Athens, Greece
{fotais, alps, ameny }@mail.ntua.gr, dora@telecom.ntua.gr

[ii]Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
ang.valsamis@gmail.com

Abstract. With the emergence of Smart TV and related interconnected devices, second screen solutions have rapidly appeared to provide more content for end-users and enrich their Smart TV experience. Given the various data and sources involved - videos, actors, social media and online databases- the aforementioned market poses great challenges concerning user context management and sophisticated recommendations that can be addressed to the end-users. This paper presents an innovative Context Management model and a related first and second screen recommendation service, based on a user graph analysis as well as collaborative filtering techniques in the context of a Dynamic Social & Media Content Syndication (SAM) platform. The model evaluation provided is based on datasets collected online, presenting a comparative analysis concerning efficiency and effectiveness of the current approach, and illustrating its added value.

Keywords: Smart TV Recommendations, Second Screen, Context Management, Graph analysis, Collaborative filtering

1. Introduction

A recent study conducted by Nielsen/yahoo, revealed that mobile devices are gradually employed more and more in parallel with TV usage. More specifically, 88% of the study group used their devices while watching TV, creating the so called second screen phenomenon[1]. Users comment or rate TV shows on Social Media and also search for related information about actors, places and all other sorts of information related to the show they are watching. This phenomenon is expected to grow exponentially, thus creating a huge impact on the way content is created and delivered, not only through regular broadcasting but also thought mobile devices.

Despite the growth of this phenomenon there are no second screen standards, protocols or ever common practices for end-users to discover and access additional information related to the consumed content. The lack of these technical attributes drives them to search intuitively for content in the social media or online search engines. Given the continuing growth of the internet content, a need arose for the creation of systems managing this content, in order to provide quality of service, content syndication, and recommendations for the users.

In the context of the SAM Project [1], a content delivery platform for syndicated data to be consumed in a contextualised social way through second screen devices has been provided. To this end, the former, out-dated model of users searching for the information they desire is replaced with a new

[1] Second Screen Society: http://www.2ndscreensociety.com/

approach, where information reaches users on their second screen using content syndication. This paper focuses on user interaction history with first and second screen and the related behavioral models applied in SAM, to form an innovative Context Management mechanism.

The Context Management approach presented is based on a No-SQL graph-based database. Thus, users, related media items (e.g. widgets appearing during a movie in second screen) as well as related actions are saved in the form of a graph, where graph analysis models and correlation techniques (collaborative filtering) are employed to properly assess the relevance of each media item for every user. As a result, a relevance rating list for each second screen user is produced, allowing personalized recommendations of multimedia and related assets during her interaction with SAM's generic dashboard, where first and second screen content is displayed.

More specifically, the current paper contributions are summarized below:

- Scalable and timely efficient user profiling and data analysis using an appropriate graph model (visualizing users, assets, interactions)
- Intelligent user Context Management model based on a combination of graph analysis and collaborative filtering
- Multi-level Smart TV user personalized recommendations via relevance rating approach for root assets (videos) in first screen as well as sub-assets (video-related sources of infomation) in second screen
- A comparative analysis of commonly used machine learning algorithms and clustering techniques, applied to Smart TV user context-based recommendations

The rest of the document is organized as follows: Section 1 is the current one and serves the purpose of the introduction. Section 2 presents related work on graph-based and collaborative filtering techniques, investigating the existing schemas for Smart TV user context modelling and recommendations. Section 3 introduces the SAM context management approach, while Section 4 elaborates on graph analysis, collaborative filtering and personalised recommendations. Finally, Section 5 analyzes the experimental setup and results and Section 6 presents the conclusions of the paper.

2. Related Work

Graph databases have been extensively used lately to optimize storage and processing of highly connected data. For example, authors in [2],[3],[4] provide insights into Neo4j graph database and its performance advantages, illustrating the various cases it is used, including recommender systems that apply “item-to-item” and “user-to-user” (i.e. collaborative filtering) correlation. Works illustrated in [5] and [6] employ a session-based temporal analysis and a knowledge graph analysis respectively to model users’ preferences and provide meaningful recommendations for online articles. The aforementioned papers make use of real online datasets to evaluate the performance of various graph processing algorithms as well as the efficiency of the examined solutions in terms of response time.

Demovic et al. [7] presented an interesting context-based graph recommendation approach, saving multimedia-related data in a graph structure and using Graph Traversal Algorithms to efficiently address user preferences. This work took into account explicit user "likes" for specific movies or genres, however it did not collect any other contextual or social data from the real dataset of users it involves. Focusing on Social TV Platforms, works in [8] and [9] highlight the concept of context management and analysis in the frame of social enabled content delivery to multi-screen devices. These papers present a novel solution for media content delivery, whose vision is based on the idea of fusing second screen and content syndication and on exploiting the advancements in the area of social media. Those form the basis of the current Context Management approach, in order to provide smart recommendations, in the frame of a Smart TV environment.

Recommender systems for eCommerce [10],[11] usually follow a personalised recommendation approach, based on users' clustering and correlation, as well as behavior factorization [12]. As expected, when it comes to TV-related personalised recommendations, a substantial amount of work has been performed, focusing on TV programs and movies' context evaluation. Krauss et al. [13] introduced personalized TV program recommendations based on users' viewing behavior and ratings, combining different data mining approaches (Content Based Filtering, Collaborative Filtering, Clustering, Association Rules and Support Vector Machines). A ten-fold cross validation over a user-generated dataset aggregated from the operation of the TV Predictor software resulted into a promising program prediction accuracy for the acquired set of users and ratings. Kim et al. [14] on the other hand, presented an automatic recommendation scheme based on a user clustering approach that did not require explicit ratings from TV viewers, but rather the watching history logs. The proposed rank model used a collaborative filtering technique, taking into account the watching times, to illustrate effectiveness with rich experimental results over a real usage history dataset.

Collaborative filtering techniques are frequently used by recommender systems [15] in several cases online, such as web services [16], products or movies [17] selection. Kwon and Hong [18] propose a personalized program recommender for smart TVs using memory-based collaborative filtering, employing a novel similarity method that is robust to cold-start conditions and faster than existing approaches. The evaluation uses an own-built crawling agent to retrieve movie reviews by real users and predict ratings for non-viewed programs. On the other hand, work in [19] proposes improvements to two of the most popular approaches to Collaborative Filtering, introducing a new neighborhood based model, as well as extensions to SVD-based latent factor models that allow improved accuracy by integrating implicit feedback into the model. Those improvements are evaluated with a very limited form of implicit feedback, available within the Netflix movie dataset.

SAM Context Management for Smart TVs attempts to progress beyond the state-of-the-art solutions presented above, by providing a personalized multi-level recommendation mechanism (applying both for first and second screen content), based on an efficient graph-based approach.

3. SAM Context Management Approach

3.1. Components Architecture and Data collected

SAM aims at the development of a context-centric middleware that acts supportively to its advanced federated social media delivery platform, providing open and standardised way of defining, characterising, discovering, syndicating and socially consuming media assets interactively. In the context of SAM's architecture, the generic components of Context Control are responsible for storing and managing context information used across the SAM Platform, as presented in Figure 1 [16].

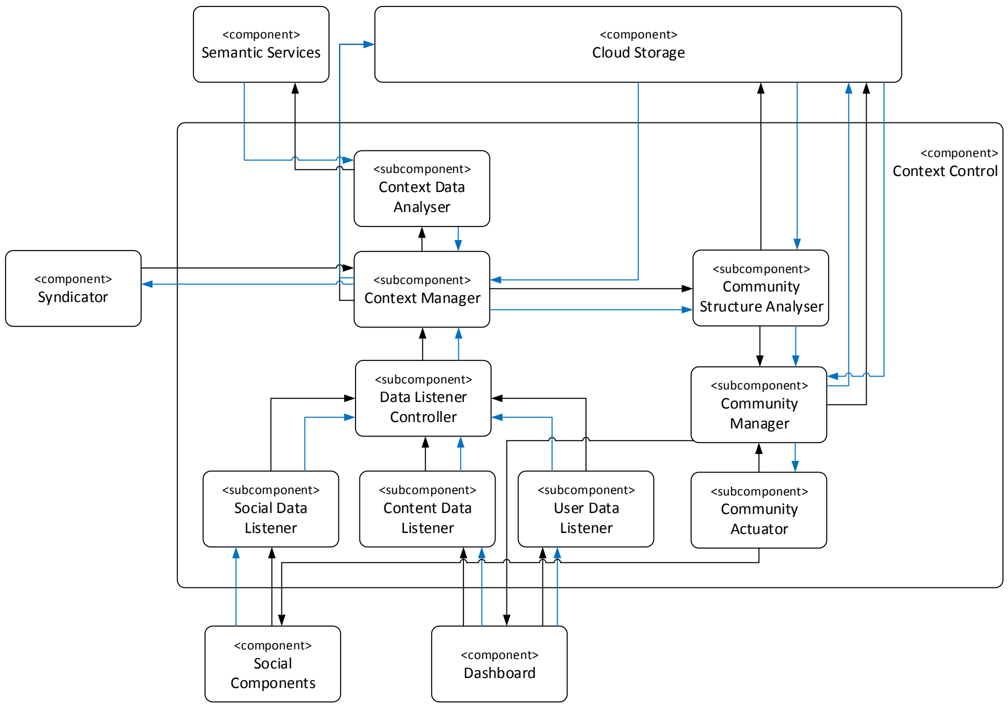
<component> Semantic Services
<component> Cloud Storage
<component> Context Control
<subcomponent> Context Data Analyser
<component> Syndicator
<subcomponent> Context Manager
<subcomponent> Community Structure Analyser
<subcomponent> Data Listener Controller
<subcomponent> Community Manager
<subcomponent> Social Data Listener
<subcomponent> Content Data Listener
<subcomponent> User Data Listener
<subcomponent> Community Actuator
<component> Social Components
<component> Dashboard

Figure 1: SAM Context Control Components

Figure 1 shows the subcomponents realising the Context Representation operations along with the connections between them and the interactions with other SAM components. The core component for context-related operations is the Context Manager component, collecting contextual information from Social Media, including SAM dynamic communities exposed by Community Structure Analyser, as well as the Syndicator and the Dashboard. The SAM Dashboard is connected to specific Data Listeners capturing and storing all interactions according to an extended W3C Social Web Working Group context model (the successor to the OpenSocial format).

The aforementioned Data Listeners (Social Data Listener, Content Data Listener, User Data Listener) are managed by a Data Listener Controller and forward to the Context Manager all data related to the Smart TV context, including Social Media posts on SAM multimedia and user interactions with first and second screen. The interactions which are useful for the Context Data Analyser are the ones illustrating user's relevance or satisfaction with the content provided, such "likes" or "clicks", as well as text comments or online posts that can be further evaluated with the support of Semantic Services for sentiment analysis, in order to enrich the user profile with contextual information.

In specific, the following user interaction items are collected from Generic Dashboard listeners and sent to SAM Context Management component to support the analysis:

Table 1: User interactions collected from Generic Dashboard

Root Asset Interactions	Widget Interactions
Consume Root Asset	Click Widget
Scroll Root Asset	Scroll Widget
Like Root Asset	Dismiss Widget (Close window)
Dislike Root Asset	Like Widget
Comment Root Asset	Dislike Widget
	Comment on Widget

3.2. SAM's Graph database

The Graph database created for SAM is composed of edges and nodes. As in common cases, nodes represent entities and edges represent relations between entities. Three types of node-entities have been defined: “Assets”, “Persons” and “Keywords”. Assets represent all kind of multimedia content in SAM (video, widgets, related information etc.), while keywords are a collection of nodes describing the tags an Asset can have. Finally the last type of node is Persons which is a representation of end-users of the SAM first and second screen. Every type of node has specific attributes, describing the information it enfolds. For instance, an Asset has identification, type, title, etc. and the Person has attributes such as name, identifier, etc.

As mentioned, nodes are connected to each other with edges called relationships. Assets can be connected with other Assets with the relationship “is root asset of”, signifying Assets that are widgets of a root Asset (movie).Assets are also connected with Keywords with the relationship “has keywords”. Likewise Assets are connected with Persons with a variety of relations. For Root Assets (movies), these relationships are “consumes” and “comments”. If a person has watched a movie or consumed related material then automatically an edge describing “consume” action is created in order to store this action. The same principle is applied to the “comment” relationship: if users express their opinion about an asset, the action is stored as “comment”. Other relationships are “like”, “dislike”, “click” and “scroll” which represent the connections between Persons and Assets. Edges have also attributes in order to enrich the information about the relationship between nodes. For example the edge “comment” contains information about the intensity of a comment, the type of the comment (if it is a negative or a positive comment) and the comment itself.

A screenshot instance of the SAM graph DB with some initial records can be seen in Figure 2:

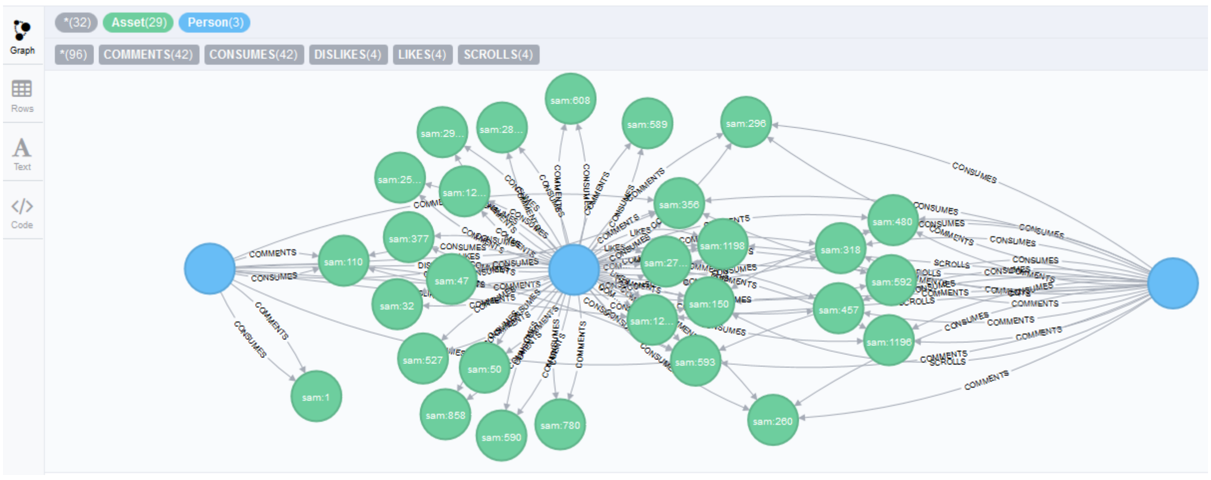

Figure 2: SAM Context Manager graph database example

Note that the same person can change her mind on an asset (e.g. dislike what she liked in the past). Thus, the authors decided to replace the explicit interactions (like/dislike, comment) with the latest one received from the SAM Dashboard, so that the relevance score to be calculated is efficient and properly updated.

4. Context Analysis and Recommendations

4.1. Graph analysis

A basic part of the analysis of the graph, is to apply some kind of "weights" to the lines connecting users and assets. Setting +1 and -1 as absolute values of relevancy and irrelevancy respectively, we apply those values to user-asset relations that explicitly show such a rating (like weights for +1, dislike weights for -1). On the other hand, comments on assets are saved along with their sentiment polarity and intensity (percentage of positivity or negativity), thus we can apply for positive comments a decimal weight, ranging from (0,+1] and for negative comments from [-1,0). Zero value obviously expresses neutrality.

However, consuming or scrolling a root asset also indicates some interest by the user. The same applies for clicking or scrolling a specific widget in second screen, while dismissing it before it automatically closes indicates lack of interest. To capture those implicit patterns, we need to make sure that they will not totally overlap the explicit ones already mentioned. For example, if a user has "liked" an asset, but on the other hand dismissed it early on, this implies a weaker "like" or "interest" relation. The approach that we follow to make sure the overall weight (sum of weights) is mainly defined by "likes" / "dislikes" and only partly affected by other interactions is to apply to the latest a weight of

$$w_i = \frac{p_i}{t-1} \qquad (1)$$

where p_i = polarity indication(+1,-1) and t = number of interaction types for this asset type. In this case, if an explicit interaction weight w_e is contradictory to implicit weights w_i, the overall weight

$$W = w_e + \sum w_i \tag{2}$$

will still bare the (now normalized) "polarity" of w_e.

Moreover, we want to collect indirect user relationships with an asset. In cases, for example, that a user has "liked" or commented positively for all widgets or keywords of a root asset (which may also exist in other videos as well), a strong indication of relevance to this root asset also exists. Similarly to the previous logic, we need to make sure that indirect relations to assets will not overlap a direct weight to it. Thus, for every rating to a connected asset/keyword we apply a weight of

$$w_x = \frac{r_x}{a + k + 1} \tag{3}$$

where r_x = rating of neighbouring node, a = number of neighbouring assets and k = number of keywords connected to the "under investigation" asset, and which were previously rated. Therefore, the overall relevance weight of a person for an asset now becomes:

$$W = w_e + \sum w_i + \sum w_x \Leftrightarrow W = w_e + \sum \frac{p_i}{t - 1} + \sum \frac{r_{xi}}{a + k + 1} \tag{4}$$

Running this process recursively to extract ratings of connected assets, we conclude to the following general algorithm for calculating the relevance of a SAM asset for a user node in the graph (keywords are also treated as assets in the following algorithm for simplicity purposes):

Pseudo-code of Context Management graph analysis to rate the relevance of an asset for user.

```
procedure AssetRelevance (user, asset, graph)

  RelevanceWeights we, wi, wx;
  InteractionList I;
  InteractionTypes types= Interaction.getTypes();
  t = types.length();

  if user.hasRated(asset) then
    we = user.getExplicitRating (asset);

  if user.hasInteracted(asset) then
    I = user.getInteractions(asset);
    foreach interaction ∈ I do
      wi = wi + interaction.getPolarity()/(t-1);
```

```
     A=asset.getNeighbours();
     n = A.length();
     foreach neighbour_asset ∈ A do
        rx = AssetRelevance (user, neighbour_asset, (graph-asset));
        wx = wx + rx /(n+1);

   return we + wi + wx;
```

To provide a rated list of assets to a user, based on her relevance to those, we need to calculate a user's node weights with existing assets. Thus, the algorithm shown above must be run for all assets in the graph which implies a complexity of O(n^2). However, given the fact that recursive calls only need to apply for depth 2 in order to make sense (when shortest path between assets equals 2 to bare some meaningful relevance), complexity is further reduced to O(n).

4.2. Collaborative filtering analysis

In cases of more "isolated" assets in the graph, when the user analyzed has not interacted with those or their neighbours (e.g. a new movie), it is obvious that the aforementioned analysis will not identify any meaningful relevance. In such cases, it was decided to use collaborative filtering among different users, in order to estimate the user relevance with the specific assets, based on her correlation with other users.

A most common approach used for collaborative filtering, having a dataset of simple numeric ratings [21], is using the Pearson Correlation Coefficient:

$$c_{au} = \frac{\sum_{i=1}^{h} (r_{ai} - \bar{r}_a) \times (r_{ui} - \bar{r}_u)}{\sqrt{\sum_{i=1}^{h} (r_{ai} - \bar{r}_a)^2 \times \sum_{i=1}^{h} (r_{ui} - \bar{r}_u)^2}} \qquad (5)$$

between users *a* and *u*, where in our case $h = |I_{au}|$ is the amount of assets having been rated by both users, r_{ai} is user *a*'s weight for asset *i* and $\bar{r}_a = average\ (r_{a1}, r_{a2}, \ldots, r_{ah})$.

Having calculated the correlation coefficients of a user with other users, collaborative filtering analysis can provide a prediction, rating her relevance with an asset *j*, based on other users' relevance for the specific asset and their correlation:

$$p_{aj} = \bar{r}_a + \frac{\sum_{u=1}^{g} (r_{uj} - \bar{r}_u) \times c_{au}}{\sum_{u=1}^{g} c_{au}} \qquad (6)$$

where *g* is the number of users having consumed j and p_{ai} is the predicted rating of relevance for user *a*.

4.3. Personalised recommendations

The results of the analysis processes described above for every user is two-fold:

- A rated list of root assets, consisted of pairs of videos and relevancy scores for the user
- A rated list of sub-assets of any root asset (appearing as widgets in second screen), consisted of pairs of widgets and relevancy scores for the user

Thus, personalized recommendations can be provided to first and second screen Syndicator component, to prioritize or even disappear irrelevant movies and related widgets of a movie upon consumption.

This results into the following two-level recommendation mechanism that provides:

- Smart recommendations of root assets (videos) to user
- Smart recommendations of second screen widgets to user, while watching a Smart TV program on first screen

5. Experiments and Evaluation

5.1. Dataset and Configuration

The presentation of the analysis methods above makes the performance advantages of the graph-based approach evident. For example, when retrieving for users having consumed a specific asset in the graph, Neo4j just returns the neighbours of the corresponding node, in contrast with the latency resulting from a respective SQL query. To acquire a meaningful and extended dataset, in order to form the SAM experimental Context Management graph, the authors decided to search online for available related data (users, movies, keywords, likes etc.).

Our experimental dataset is comprised of a big movie rating dataset found online [22], comprising a huge database of movies and user ratings, as well as keywords linked with those movies. To limit our scale and make a meaningful analysis, we selected the 30 most popular (in terms of number of ratings) movies and 5 most popular keywords for each. Thus the overall numbers of the initial dataset imported can be found in the following table:

Table 2: Initial dataset imported in Neo4j

Dataset metrics
users: 619
ratings: 7038
movies: 30
keywords: 98

The dataset imported was interpreted into the SAM logic, directly importing SAM users and assets (movies and keywords), and generating “comments” and likes/dislikes, based on the rating values.

Subsequently, it was split into training and testing sets (70% and 30% of the original rating respectively), with the first one to be fed into various data analysis algorithms and the later to be used as ground truth.

The graph analysis, supported by the Pearson Collaborative filtering, presented above was applied and compared with a k-nearest neighbours (K-NN) algorithm run over Neo4j[2], as well as various variations of machine learning algorithms (SVM, C4.5, MLP etc.) employed on Weka open-source software, version 3.7[3], inputting the initial dataset as an .arff file. All experiments were performed on a desktop machine with an Intel Core TM i5-3400 Processor, 2.80 GHz, 12GB of RAM memory, running 64-bit Windows 10 Pro N.

5.2. Experimental Results

In Table 3 an analytical report of accuracies and mean errors per approach is provided, for predicting users' relevance to the assets (movies) of the training set. The mean errors refer to the difference between the calculated values and the real original ratings, ranging from -1 to 1, which represent the current evaluation's ground truth.

It is worth noting that time comparison between algorithms running in batch mode that do not connect to a database and algorithms that return results on-demand, like the one we implement in SAM, is irrelevant. For example the Naive Bayes approach trains its model instantly but requires the full dataset available in memory. When considering our dataset of 7038 ratings this is feasible, however it is apparent that this is not a scalable solution. The motivation behind the decision to get metrics from state-of-the-art machine learning algorithms is mainly to evaluate the accuracy of our proposed algorithm.

Table 3: Accuracy and mean errors of SAM predictions, compared to State-of-the-Art machine learning techniques

	Mean absolute error	**Root mean squared error**	**Mean percentage error**
SVM with linear kernel	0.1099	0.3315	5.5%
C4.5 w/ 10 Bagging	0.1253	0.2688	6.3%
Best-first decision tree	0.1274	0.2626	6.4%
Logistic Regression	0.1269	0.2612	6.4%
LAC Lazy Associative Classifier	0.1306	0.2563	6.5%
Bayes Net with K2 search	0.1263	0.2559	6.3%
NaiveBayes	0.1283	0.2551	6.4%
Naive Bayes Tree	0.1334	0.2652	6.7%
MLP 100 neurons	0.1324	0.2632	6.6%
CNN 100 neurons	0.1269	0.2530	6.4%
CNN 1000 neurons	0.1202	0.2578	6.0%
Hoeffding Tree	0.1499	0.2684	7.5%

[2] https://neo4j.com/graphgist/8173017/
[3] http://www.cs.waikato.ac.nz/ml/weka/

Hidden Markov Model	0.1653	0.2875	8,3%
K-nearest neighbour on Neo4j	0.3226	0.4108	16,1%
SAM (graph + CollabFiltering)	**0.1312**	**0.2604**	**6.6%**

As can be observed in the experimental results, SAM's approach scores relatively well, taking into account the low performance cost introduced by the graph database, as well as its multilevel capabilities. The authors compared our graph-based analysis approach with variations of various known algorithms. In particular, we used Breiman's Bagging technique[4] with 10 iterations (C4.5) and also tried different neural network models (Convolutional Neural Network with 100 and 1000 neurons, Multi-layer Perceptron). Probabilistic models (Logistic Regression, Naive Bayes, Bayesian network with K2 as search algorithm) and simple tree implementations (C4.5, Best-first) were also tried and yielded competitive results. In terms of mean absolute error the SVM with a linear kernel was superior but with a weak root mean squared error, while when root mean squared error is considered the 100 neuron Convolution Neural Network produced best results.

Beyond comparing SAM Context Management approach with other machine learning techniques in terms of effectiveness, we also present a comparison of the two algorithms running on top of the graph database (SAM, K-NN) time-wise. Resulting from those algorithms, SAM's Context Management component exposed two recommendation web services respectively. Stress-testing our approach performance, as a commercial deployed service, we used JMeter software [23] to generate requests for all ratings of the testing set to those services. The response times of the first 1000 requests can be seen in the following diagram:

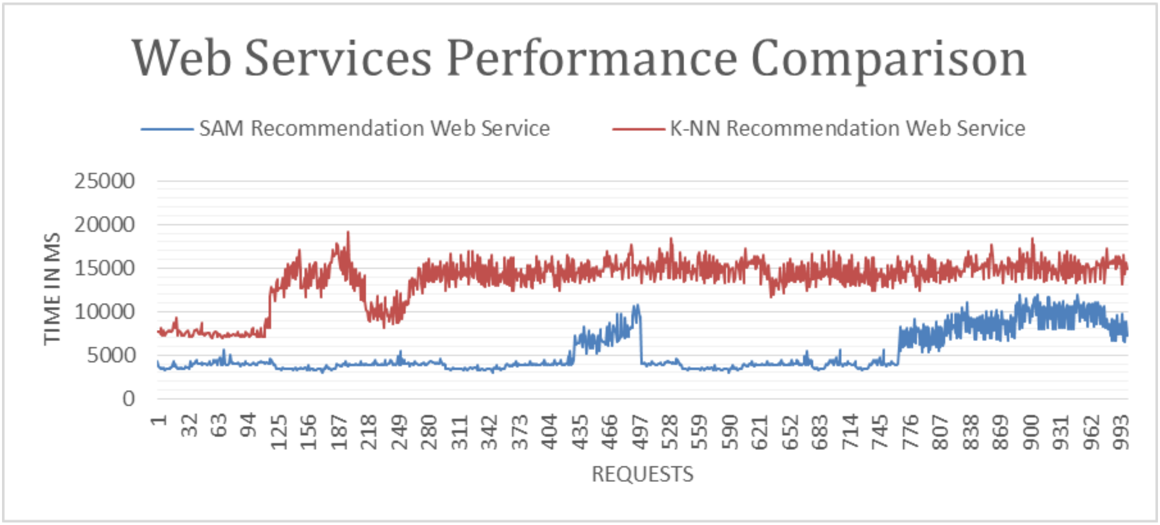

Figure 3: A time performance comparison between K-NN and SAM Recommendation Web Services

[4] http://machine-learning.martinsewell.com/ensembles/bagging/

SAM Recommendation Web Service evidently outperforms K-NN, one of the most popular clustering approaches for recommendations using graphs. Results above validate the current model's low complexity and illustrate its efficiency.

6. Conclusions

In this paper, we presented an efficient Context Management approach for Smart TV users, collecting context-related data and actions to provide personalized multi-level recommendations via a hybrid analysis method combining graph paths analysis and Pearson collaborative filtering. Experiments conducted were based in a real movie rating dataset found online and illustrated promising results, in terms of accuracy and performance. The full effectiveness of the current model, though, will be more evident with the addition of an extended user interaction dataset, which is expected to be aggregated from the SAM second screen listeners during the final trials of the project in the upcoming months.

In the future, the authors also plan to aggregate more and larger Smart TV-related datasets, and evaluate the current model end to end in both levels (first and second screen). The validity of the results could be better illustrated using more diverse datasets, in terms of user relevance scores. Thus, in the case of first screen recommendations, it would be desirable to have a training set, where ratings are distributed more widely, so that correlation techniques can generate more concrete and meaningful user clusters for that purpose.

7. Acknowledgment

This work has been supported by the SAM project and has been funded from the European Union's 7th Framework Programme for research, technological development and demonstration under grant agreement no 611312.

8. References

[1] Socialising Around Media (SAM) Project: Dynamic Social and Media Content Syndication for 2nd Screen, http://samproject.net/

[2] Vicknair, Chad, et al. "A comparison of a graph database and a relational database: a data provenance perspective." *Proceedings of the 48th annual Southeast regional conference*. ACM, 2010.

[3] Holzschuher, Florian, and René Peinl. "Performance of graph query languages: comparison of cypher, gremlin and native access in Neo4j." *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, 2013.

[4] J. J. Miller, "Graph Database Applications and Concepts with Neo4j." In Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA March 23rd-24th. 2013.

[5] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang and J. Sun, "Temporal recommendation on graphs via long-and short-term preference fusion," In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010.

[6] S. Kumar and M. Kulkarni, "Graph based techniques for user personalization of news streams," In Proceedings of the 6th ACM India Computing Convention, ACM, 2013.

[7] Demovic, Lubos, et al. "Movie recommendation based on graph traversal algorithms." Database and Expert Systems Applications (DEXA), 2013 24th International Workshop on. IEEE, 2013.

[8] A. Menychtas, D. Tomás, M. Tiemann, C. Santzaridou, A. Psychas, D. Kyriazis, J.V. Vidagany, S. Campbell, "Dynamic Social and Media Content Syndication for Second Screen", International Journal of Virtual Communities and Social Networking (IJVCSN), 2015 (accepted)

[9] C. Santzaridou, A. Menychtas, A. Psychas and T. Varvarigou, “Context Management and Analysis for Social TV Platforms”, eChallenges e-2015, 2015 (accepted)

[10] Sarwar, Badrul M., et al. "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering." *Proceedings of the fifth international conference on computer and information technology*. Vol. 1. 2002.

[11] Wei, Kangning, Jinghua Huang, and Shaohong Fu. "A survey of e-commerce recommender systems." *Service Systems and Service Management, 2007 International Conference on*. IEEE, 2007.

[12] Zhao, Zhe, et al. "Improving User Topic Interest Profiles by Behavior Factorization." Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015.

[13] C. Krauss, L. George and S. Arbanowski, "Tv predictor: personalized program recommendations to be displayed on smarttvs," in In Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, 2013.

[14] E. Kim, S. Pyo, E. Park and M. Kim, "An automatic recommendation scheme of TV program contents for (IP) TV personalization," 2011.

[15] Schafer, B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 291–324. Springer, Heidelberg (2007)

[16] Tserpes, Konstantinos, et al. "Service selection decision support in the Internet of services." Economics of Grids, Clouds, Systems, and Services. Springer Berlin Heidelberg, 2010. 16-33.

[17] Salter, James, and Nick Antonopoulos. "CinemaScreen recommender agent: combining collaborative and content-based filtering." Intelligent Systems, IEEE 21.1 (2006): 35-41.

[18] Kwon, Hyeong-Joon, and Kwang-Seok Hong. "Personalized smart TV program recommender based on collaborative filtering and a novel similarity method." *Consumer Electronics, IEEE Transactions on* 57.3 (2011): 1416-1423.

[19] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008.

[20] SAM deliverable D6.9.2 – Context Analysis & Dynamic Creation of Social Communities Public Report (second version), can be downloaded from: http://samproject.net/sam-community/

[21] Tserpes, Konstantinos, et al. "Service selection decision support in the Internet of services." Economics of Grids, Clouds, Systems, and Services. Springer Berlin Heidelberg, 2010. 16-33.

[22] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

[23] Apache JMeter, an open source Java application designed to load test functional behavior and measure performance, http://jmeter.apache.org/