

An Adaptive Contextual Recommender System: a Slow Intelligence Perspective

S.K. Chang, Duncan Yung
Computer Science Department
University of Pittsburgh
Pittsburgh, USA
{chang, duncanyung}@cs.pitt.edu

F. Colace, L. Greco, S. Lemma, M. Lombardi
DIEM
Università degli Studi di Salerno
Salerno, Italy
{fcolace, lgreco, slemma, mlombardi}@unisa.it

Abstract — This paper introduces an Adaptive Context Aware Recommender system based on the Slow Intelligence approach. The system is made available to the user as an adaptive mobile application, which allows a high degree of customization in recommending services and resources according to his/her current position and global profile. A case study applied to the town of Pittsburgh has been analyzed considering various users (with different profiles as visitors, students, professors) and an experimental campaign has been conducted obtaining interesting results.

Keywords — *Recommender System - Slow Intelligence Approach - Smart Adaptive System - Context-aware computing*

I. INTRODUCTION

Recommender Systems represent a meaningful response to the problem of information overload since the mid-1990s [28] when early works on this topic have been proposed. The aim of such systems is to predict user's preferences and make meaningful suggestions about items that could be of interest [29]. In literature, there are various approaches for recommending systems. In the content-based approach, the system recommends an item to a certain user relying on the ratings made by the user himself for similar items in the past [26]. In recent times, some improvements, such as a deeper user profile analysis [33] and the use of probabilistic methods [35], have been introduced together with some attempts to apply the content based approach to multimedia data [23, 18, 24]. However, a critical drawback of this approach is overspecialization, since the systems only recommend items similar to those already rated by the user. Another interesting approach is the collaborative filtering [1]. In this case, the recommendation is performed by filtering and evaluating items with respect to ratings from other users [33]. Ratings can be attributed in different ways and collected by explicitly asking users or implicitly tracking their actions [2]. Two basic methods, passive and active filtering, are exploited for filtering and recommending items together with nearest neighbor techniques [27, 21]. An important limitation of collaborative filtering systems is the cold start problem: situations in which a recommender is unable to make meaningful recommendations due to an initial lack of ratings.

A particular kind of collaborative approach is the collaborative competitive filtering that aims at learning user preferences by modeling the choice process in recommender systems [34]. Content-based filtering and collaborative filtering may be then combined in the so-called hybrid approach that helps to overcome limitations of each method [30]. In general, a recommendation strategy should be able to provide users with relevant information depending on the context [15, 19, 20] (i.e. user location, observed items, weather and environmental conditions, etc.) as in Context Aware Recommendation Systems. In the Contextual Pre-filtering techniques context information is used to initially select the set of relevant terms, while a classic recommender is used to predict ratings. In Contextual Post-filtering approaches context is used in the last step of the recommending process to contextualize the output of a traditional recommender. An important improvement for traditional recommender systems is in the possibility to embed social elements into a recommendation strategy [38]. In fact, the great increase of user-generated content in social networks, such as product reviews, tags, forum discussions and blogs, has been followed by a bunch of valuable user opinions, perspectives or tastes towards items or other users, that are useful to build enhanced user profiles. In such context, customer opinion summarization and sentiment analysis [39, 13] techniques represent effective improvement to traditional recommendation strategy, for example by not recommending items that receive many negative feedbacks [38]. Indeed, a lot of attention is nowadays being paid from vendors to consumer's voices because of the great influence they may have on the opinions and decisions of others [32] and some companies already provide several opinion mining services (e.g., Amazon, Epinions, etc.). In recent times, some works have been proposed to extend traditional collaborative filtering with the use of sentiment analysis techniques, thus providing effective improvement to system performances [22]: most of them make use of Part Of Speech (POS) tagging techniques and aim at refining standard collaborative filtering ranking outcomes in terms of numerical scales to take into account user community opinions. The work in [31] proposes a recommender system for movies that combines collaborative

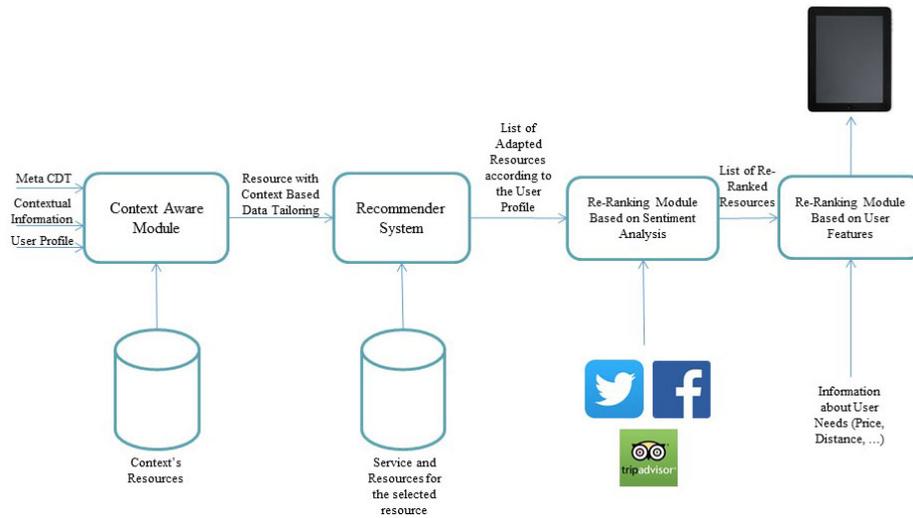


Figure 1 System architecture

filtering with sentiment: here sentiment classification is performed through both Naive Bayes classifier and unsupervised semantic orientation approach. Given this scenario, we propose a Recommender System based on the Slow Intelligence Approach [10]. The Slow Intelligence System is a general-purpose system characterized by being able to improve performance over time through a process involving enumeration, propagation, adaptation, elimination and concentration. A Slow Intelligence System continuously learns, searches for new solutions and propagates and shares its experience with other peers. Slow Intelligence Systems typically exhibit the following characteristics:

- Enumeration: In problem solving, different solutions are enumerated until the appropriate solution or solutions can be found.
- Propagation: The system is aware of its environment and constantly exchanges information with the environment. Through this constant information exchange, one SIS may propagate information to other (logically or physically adjacent) SISs.
- Adaptation: Solutions are enumerated and adapted to the environment. Sometimes adapted solutions are mutations that transcend enumerated solutions of the past.
- Elimination: Unsuitable solutions are eliminated, so that only suitable solutions are further considered.
- Concentration: Among the suitable solutions left, resources are further concentrated to only one (or at most a few) of the suitable solutions.

In the next paragraph, the general architecture of the recommender system is introduced and each module is described in details. In particular, the Context Aware Module and the Content Dimension Tree (CDT) approach is described. An example scenario and an experimental campaign close the paper.

II. A FRAMEWORK FOR CONTEXTUAL RECOMMENDATION BASED ON A SLOW INTELLIGENCE APPROACH

In this section, the System Architecture is introduced and described. Four modules compose the system: the context aware module, the recommender system module, Re-Ranking Module Based on Sentiment Analysis and the Re-Ranking Module Based on User Features. The architecture, depicted in figure 1, reflects the Slow Intelligence approach; in particular the context aware module and the recommender system module implement the adaptation and the enumeration phases, the Re-Ranking Module Based on Sentiment Analysis implements the propagation phase, the Re-Ranking Module Based on User Features implements the elimination phase. A “Contextual App” implements the concentration phase. In the following paragraphs more details about the various modules will be given.

A. The Enumeration and Adaptation stage

This stage aims at defining the user’s problems and the main strategies that have to be pursued for solving them. The combination of the Context Aware Module (CAM) and of the Recommender system can implement the Adaptation and the Enumeration phases. In the following paragraphs the two modules will be described:

1) The Context-aware module(CAM)

The purpose of this module is to provide a mechanism of dynamic and automatic invocation of services according to the context[11]. Since the purpose of this module is to deal with contextual changes that occur at runtime, there should be a mechanism that is concerned with the choice of the item to be invoked during the execution of the specific instance of the program, instead of associating a specific and concrete item to every activities in the design phase. Dynamic invocation of

items is implemented by context aware module configuration, in response to a user abstract request and according to the measured parameters at runtime. The concrete item to be invoked is chosen during the execution of the application. In general, we can divide the CAM module in two submodules: the High Level and the Low Level modules. The first one defines the problem and the resources or service at a high level of abstraction, while the second one gives the resources that contains concrete items or services.

The inputs of CAM are:

- Contextual Information: the user's position, which is collected from GPS sensor.
- Resources: all the resources that can furnished in each context. The resources are identified by the definition of a Point of Interest.
- User Profile: user's information, which are collected during the registration phase, e.g. user interests.
- Context Dimension Tree [3,4] is a model used to represent context in an extensible and orthogonal way, using the 5W-1H method. We realize a generic CDT, named Meta CDT, for all possible contexts and a specific CDT on the basis of the resource user choice.

For representing the scenario, the Context Dimension Tree (CDT) model shown in Fig.2 was used: the structure consists of two different types of nodes - the *black nodes*, which describe the context dimensions, i.e. the different points of view from which the system's situation can be observed, and the *white nodes* which describe the values that constitute the context; each node has a label with the name of the corresponding node. The CDT has a special white double circled node that represents the root of the tree. In addition, each leaf of the tree is a value node and they may feature parameters. The parameters are described by a white square and they are used as special filters helpful when a value has many instances.

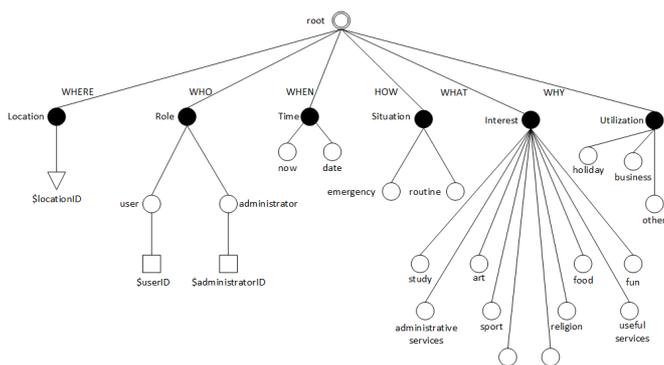


Figure 2 A Meta CDT example

This way of representing the context through the CDT allows the designer to characterize the relevant aspects of the considered scenario and to choose the dimensions and values of the tree in a correct way. It is important to underline how

the hierarchical structure of the CDT allows the description of the context with different levels of abstraction and granularity.

The output of CAM High Level is:

- Contextual Resources: all the resources that can furnished in a specific geographical area. User will choose a specific resource.

The input of CAM Low Level is:

- Resource: resource selected by user.

When the user chooses a resource, in CAM Low Level block, we know Location Dimension and we have a specific CDT with information associated to resource useful for representing current context and providing contextual services. The output of the system is a set of contextual items of current resource.

CAM module follows a methodology consists of three phases. Methodology has been realized in order to define all relevant contexts for the considered application, in order to provide contextual services managing database and performing reductions of their content based on the context.

- Design phase of contexts tree requires the design and use of CDT to represent and identify significant elements of context.
- Objective of definition phase of partial views is to identify each element of the context, then the value of each top dimension (child black node of root).
- Composition phase of views generates the global view associated with each context, which is made from union of partial views. The objective is to obtain a valid and specific query for the current context.
Then, it will be possible to interrogate the system in order to obtain the corresponding contextual items.

2) Recommender System

The developed recommender system is a content/service recommender system able to recommend a list of objects (contents or services), given the user profile and the contextual resource to which the objects belong. It can also dynamically update user profile. The resource is used for customizing our general recommender system to a particular, domain specific, recommender system. A resource can be a Cinema, a Museum, etc. The CAM sends to recommender system an identifier of the resource and through it the recommender system can recover, from a configuration database, some parameters that allow personalization of general recommender system in a domain specific way. In particular, every domain of interest has a specific database storing local informations and the similarity files, on objects of the domain, are generated once from a sub-system created ad hoc, as discussed later. CAM sends to recommender system also the user identifier. In fact, CAM maintains an high level user profile, with many interests for different domains, the recommender system instead maintains a low level user profile with specific

interests for the domain. These last interests are obviously a subset of the first. The proposed recommender system is actually an hybrid recommender system that has common features with Content-based and Collaborative systems. In particular it considers:

- User Similarity;
- Object Similarity;
- Users behaviour in the system;
- Users history.

We will now discuss each aspect in more detail.

User Similarity

User profile is used for clustering users and then creating groups of similar users. The clustering here is based on Jaccard Similarity [37] performed on user's interests for a specific domain. The interests for each user are represented through a binary vector. User similarity is a first important aspect to compute since it is fundamental for the calculation of the *global browsing matrix*, whose elements contain the ratio of the number of times object o_i has been accessed by any user immediately after o_j to the number of times any object in O has been accessed immediately after o_j .

Object Similarity

Object similarity is based on high-level information of the contents/services to recommend. A sub-system computes similarity indexes between objects through Wu-Palmer's metric[2]. For every domain there is a taxonomy, describing the most important features of the objects, that allows objects' comparison. The used metric allows to compute a similarity matrix (similarity files that are generated off-line). The object similarity is very important not only for the evaluation of the candidates but also for the location of the set of good candidates for recommendation.

The proposed recommender system is however independent from the way of computing the object similarity, so the sub-system designed, using taxonomies, is only one of the possible ways for computing object similarity.

Users behaviour in the system and Users history

The recommender system uses different structures to maintain users behaviour and users history. In particular, the following matrices are defined:

- local Connection matrix. Its generic element is defined as the number of times the object o_i has been selected by user u_i immediately after o_j .
- global Connection matrix. Its generic element is defined as the number of times object o_i has been selected by any similar user immediately after o_j .

User's history and behaviour are stored and retained in the local Connection matrix, where the occurrences and the order of what user prefers are stored. The global Connection matrix, instead, is used to retain dynamically the general behaviour of the similar users in the system. User profile is updated through the observation of the behaviour of the user in the system and by inferring the interests associated with chosen objects. Practically we analyze the local Connection matrix and when a generic element goes over a certain threshold, we analyze the related objects. From these objects, the related interests are extracted and user profile is updated. So we obtain implicit feedback from user's behaviour in the system. Now that we have a complete view of the system, we can summarize the process of the recommendation. The CAM module enumerates some recommendable objects to the recommender module starting from the user position. For each object, the most similar objects are obtained from the similarity matrix and the system adds all the objects o_i that have been selected by user in two steps after o_j , the 'init' objects. On this set a ranking vector is calculated, based on global browsing matrix and similarity matrix through Power Method's invocation. Finally from the ranked list of objects, obtained after this step, the first 50 objects are selected as recommendations for the current user.

B. Propagation stage

This module aims to involve the experience of other peers in the resolution of the problems. In particular, various policies can be defined (for example: the system can consider only peers that have profiles similar to the user, or with similar interests). For the selection of point of interests and resources, we consider only those having good reviews from users with similar profiles. This result can be obtained by the use of a module based on the sentiment analysis approach. In particular, this module can work on reviews that can be collected from Tripadvisor, Facebook or Twitter.

At the end of this stage a list of re-ranked resources/services is obtained.

1) Re-Ranking Module Based on Sentiment Analysis

This subsection describes the proposed methodology for the sentiments' extraction from user comments/reviews and its integration in the proposed recommendation strategy.

In particular, the used sentiment extraction technique is an improvement of the approach presented by some of the authors in a previous work [5], where the Latent Dirichlet Allocation (LDA) has been adopted for mining the sentiment inside documents. In our view, the knowledge within a set of documents can be represented in a compact fashion by the use of a complex structure: the Mixed Graph of Terms (mGT). This graph contains the most discriminative words and the

probabilistic links between them. More in details, we define a structure made of weighted word pairs, which has proven to be effective for sentiment classification problems as well as text categorization and query expansion problems [7, 8, 9]. The main reason of such discriminative power is that LDA-based topic modeling is essentially an effective conceptual clustering process and it helps discover semantically rich concepts describing the respective affective relationships. Using these semantically rich concepts, that contain more useful relationship indicators to identify the sentiment from messages, it is possible to accurately discover more latent relationships and make fewer errors in the predictions. The mGT is built starting from a set of comments belonging to a well-defined knowledge domain and manually labeled according to the sentiment expressed within them. In this way the mGT contains words (and their probabilistic relationships) which are representative of a certain sentiment for that knowledge domain. The LDA approach allows to obtain an effective graph by using only few documents. A mGT graph includes two kinds of nodes: the *aggregate roots* nodes, defined as the words whose occurrence is most implied by the occurrence of all other words in the training corpus, and the *aggregate* nodes, defined as the words most related to aggregate roots nodes from a probabilistic point of view.

In [5] the LDA approach and the mGT formalism have been used for the detection of sentiment in tweets. The approach aims at using the mGT, obtained by LDA based analysis of tweets, as a filter for the classification of the sentiment in a tweet. The sentiment extraction is obtained by a comparison between document and the *mixed graph of terms* according to the following algorithm:

• *Input of the algorithm:*

- A set of comments, reviews about items or social posts;
- The sentiment oriented mixed graphs of terms mGT+ and mGT- obtained analyzing the (positively and negatively) training comments;
- An annotated lexicon L.

• *Output of the algorithm:*

- The average probabilities P + and P - which express the probability that a sentiment, extracted from the set of comments or posts, is “positive” or “negative”.

• *Description of the main steps:*

1. For each word in the mGT+ and the mGT- their synonymous are retrieved through the annotated lexicon L.
2. For each comment c_i the probabilities $P_{c_i}^+$ and $P_{c_i}^-$ are determined as:

$$P_{c_i}^{+/-} = \frac{(A + B + C + D)}{4}$$

A being the ratio between the sum of occurrences in the comment of words that are Aggregate Root Nodes and the

total number of the Aggregate Root Nodes in the (positive/negative) mGT; B the ratio between the sum of occurrences in the document of words that are Aggregate Nodes and the total number of the Aggregate Nodes in the (positive/negative) mGT; C the ratio between the sum of the co-occurrence probabilities of Aggregate Root Nodes pairs that are in the document and the sum of all the co-occurrence probabilities of Aggregate Root Nodes pairs in the (positive/negative) mGT; D the ratio between the sum of the co-occurrence probabilities of Aggregate Nodes pairs that are in the document and the sum of all the co-occurrence probabilities of Aggregate Nodes pairs that are in the (positive/negative) mGT; 3. For each item the probabilities P + and P - are determined as:

$$P^+ = \sum_i \frac{P_{c_i}^+}{num_of_comments}$$

$$P^- = \sum_i \frac{P_{c_i}^-}{num_of_comments}$$

C. Elimination

This module aims to find the best resource/service according to some contextual features that characterize the user. In this phase, we propose to maintain a global resource/service quality table that keeps tracks of the quality of all resources/services in different aspects (e.g. service quality, price, transportation etc.) and a top-k function for each user so as to reflect his/her own preferences. Based on the quality table and the top-k function, each resource/service input of this module is ordered based on its top-k score. Furthermore, the user is asked to offer feedback after visiting the resources/services. Then, the top-k function is refined based on the feedback. More details will be presented below.

1) Global Resource Quality Table

The global resource/service quality table contains scores of all resources/services in all pre-defined aspects. It reflects the quality of resources/services. The value of each aspects is the median of all feedbacks from all users. Median Voter Theorem states that setting the value to be the medians of n feedbacks can satisfy most people in the population, where n is the population size. We believe that a value that can satisfy most people can truly reflect the quality of a resource. Hence, when the number of feedbacks increase along with time, the global resource/service quality table can gradually reflect the true quality of resources/services. For example, the global resource/service quality table below contains 3 resources and 4 aspects (score of each aspect ranges from [0-10]). The input order is not part of the table. It is given as the input of this module.

Resource	Service Quality (SQ)	Price (P)	Transportation (T)	Content (C)	Input Order (IO)
Carnegie Museum	9	10	1	5	2
Botanic Garden	9	5	9	2	1
Cathedral of Learning	2	3	8	7	3

Table 1: Global Resource/Service Quality Table

2) Top-k Function and Top-k Score

The top-k function is a personalized function which reflects the preference of a user. It takes into account all aspects in the global resource/service quality table and the input order. Hence, the number of coefficients of the top-k function is the number of aspects in the global resource quality table plus 1 and the sum of all coefficient equals to 1. Initially, all coefficients have the same value.

For example:

$$f = a \times SQ + b \times P + c \times T + d \times C + e \times IO_Score$$

where a, b, c, d, e are coefficients and $a + b + c + d + e = 1$.
For a new user, we assume that $a = b = c = d = e = 0.2$.

The top-k score of a resource/service is computed based on the top-k function, every aspect of the resource/service, and the input order of the resource. The input order score (IO_Score) is computed based on Definition D1.

Definition D1 [Input Order Score]: IO_Score is defined as:

$$IO_score = \begin{cases} 10 - (i - 1) & \text{for } |R| \leq 10 \\ 10 - \left\lfloor \frac{10(i-1)}{|R|} \right\rfloor & \text{for } |R| > 10 \end{cases}$$

where $|R|$ is the size of resource/service input and i is the input order (IO in Table 1).

All input resources/services are ordered base on their top-k scores (high to low) and top-k resources/services are returned to users.

3) Update of Top-k Function based on User Feedback

After the top-k resources/services are sent to the user, the user is asked to offer feedback to the system. The user can choose to offer feedback of any aspects of any resource/service to this module. Based on the feedback, this module computes the difference of each aspect between offered feedback and values in the global resource/service quality table and average the difference for each aspect. We propose to increase/decrease the top-k coefficient based on the average difference. We first recomputed the value of aspects that needed to be refined using this formula- $\frac{\text{avg.diff}}{10} \times \text{coefficient}$. Then, we normalize all coefficients so as to make their sum to be 1.

D. Concentration

This module aims to shows the outputs obtained from the other modules of the system. In particular, by this module the user can interact with the personalized services and resources. In this scenario, the concentration module has been built as a mobile contextual app that collects the outputs of the others modules and offers them by a friendly interface. The main feature of this app is that for each user it shows the ability to change services and contents depending on the context in which he/she is located. The app can also send information to the other modules that can update their information and refine the selected contents and services.

III. EXAMPLE SCENARIO

In this paragraph an example of how the proposed approach works is provided. Luca is a researcher and he is going to Pittsburgh for a two days business trip. During his spare time in Pittsburgh, he wants to visit some places. He has on his smartphone the contextual app that implements the Adaptive Contextual Recommender System approach. When he arrives at Pittsburgh, the app collects his current location and sends this information to the server. According to his profile, the Context Aware Module (CAM) retrieves a list of possible domains of interest from the context's database. In other words, the Context Aware Module selects a set of the possible domains that can be interesting for the user in a certain context. In this case, for example, if Luca has interests in nature and history the system will furnish resources that are linked to these domains. If Luca selects "Nature" the system will furnish all the resources that are related to this domain (e.g. Parks, Museums, Historical Buildings). At this point Luca can select Museum and the Recommender Module returns a list of resources or services related to this topic (e.g. Carnegie Museum of Natural History, the Heinz History Center, a booking web-site and so on). Luca can reorder the list of resources by the use of the sentiment analysis module. In this case the resources and the services selected by the recommender system can be re-ordered according to the sentiment retrieved in internet about them. In particular, the sentiment analysis module collects posts related to the selected resources from the famous social network "trip advisor" or from their official web sites. The SIS module receives the list of recommendations from the sentiment analysis module and a top-k function of Luca is used to re-rank the result of all recommendations from all domains. The top-k function reflects Luca's needs and preferences, and it is updated gradually based on Luca's feedbacks. The top-k score is computed based on Luca's current top-k function and a global resource score table, which is updated according to users' feedback of recommendations. Given the list of recommendations, Luca's top-k function $y = 0.8 \times \text{worth_visit} + 0.2 \times \text{price}$ reasonable, and the global resource score table, the SIS module computes the top-k

scores of all resources in all domains and return recommendations to Luca.

IV. EXPERIMENTAL RESULTS

For the experimental stage, 50 user profiles have been considered. Each user profile is defined as a vector of interests and can be dynamically updated according to user choices or feedbacks. The following set of possible interests has been considered: study, sport, courses, administrative services, transport, religion, food, useful services, fun. In our experimental campaign, we assumed profiles to contain at most three main interests. We identified about 126 geo-localized resources and services in Pittsburgh area, grouped in 29 points of interests such as: *Cathedral of Learning, Sennott Square, Restaurants area, Petersen Events Center, The Pitt Shop, Barco Law Library, Holland Hall, Carnegie Museums, Carnegie Mellon University, Phipps Conservatory* and so on. Depending on user profile and position, the CAM module and the Recommender System module provided a set of ranked results corresponding to recommended services or resources for each user. This ranking has been first refined by the sentiment analysis module and then the top-50 results for 50 users were given as input for the Elimination module. We assumed the global resource/service quality table to contain 4 features - *service quality, price, transportation, and content*, and they were all initialized to 7/10. Every features of the top-k function of all users was initialized to 0.2 (There are actually 5 features- *service quality, price, transportation, content, and input order*). Before we generated the top-5 result for all 50 users, we modeled the evolution process of the SIS module:

1. We defined a set of high quality services.
2. We randomly generated feedback from a user.
3. The randomly generated value was discount by 30% if the service was not in the high quality service set.
4. Then, the global resource/service quality table and top-k function of the user were updated based on the feedback.
5. We did this for all 50 users and repeat 2-4 for 30 times for all 50 users.

By doing that, we could model the randomness of user feedback while still penalizing poor quality services/resources. After that, top-5 results for 50 users were obtained by using results from recommender module and the current global service quality table and top-k functions in the Elimination module.

Relevance assessment was made by 150 students from the University of Salerno grouped in 50 sets (one for each profile): each student in a group assigned a binary relevance level to each of the top-5 retrieved results for the given profile; in this way, the relevance of each item was assessed through a majority vote rule. Once relevance levels have been assigned to the retrieved results, information retrieval performance measures were used to assess the quality of system's output. In particular, precision@5 (Fig.3), average precision and standard deviation on precision values for different profiles were calculated as shown in Table 2.

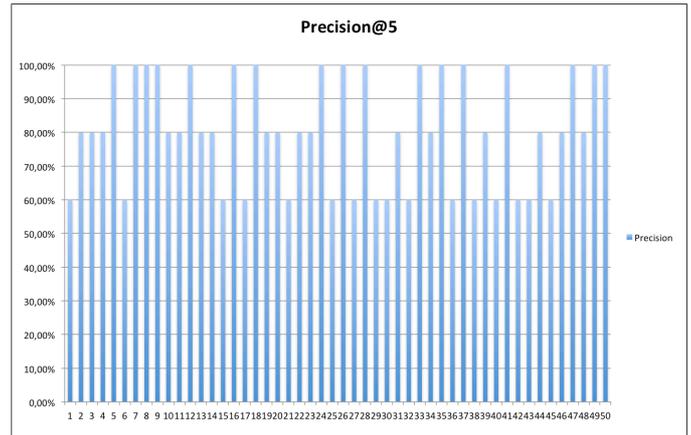


Figure 3 Precision@5 for each user profile

Average Precision@5	Standard Deviation
80,40%	16,41%

Table 2 Global performance evaluation

V. CONCLUSIONS

In this paper, an original approach to recommendation has been introduced. In particular, the proposed system is based on the Slow Intelligence Approach and integrates methodologies as the context aware approach and the sentiment analysis. The CDT formalism has been adopted for the context representation and a real case has been investigated developing a Contextual App for the Pittsburgh city. The results obtained by the experimental campaign are satisfying and show the good perspective of this kind of approach. Further developments involve the application of the proposed approach in various contexts and an improvement of the recommender approach according to an effective collaboration approach thanks to a closest integration with the most important social networks.

REFERENCES

- [1] Gediminas Adomavicius and Alexander Tuzhilin. *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*. IEEE Transactions on Knowledge and Data Engineering, 17:734–749, 2005.
- [2] Massimiliano Albanese, Antonio d’Acierno, Vincenzo Moscato, Fabio Persia, and Antonio Picariello. *A multimedia recommender system*. ACM Trans. Internet Technol., 13(1):3:1–3:32, November 2013.
- [3] Bolchini, C., Schreiber, F. A., and Tanca, L. *A methodology for very small database design*. Information Systems, 32(1):61–82, March 2007.
- [4] Cristiana Bolchini, Carlo Curino, Fabio A. Schreiber, Letizia Tanca: *Context integration for mobile data tailoring*. SEBD 2006: 48-55.
- [5] Francesco Colace, Massimo De Santo, and Luca Greco. *A probabilistic approach to tweets’ sentiment classification*. In Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on, pages 37–42, 2013.

- [6] Francesco Colace, Massimo De Santo, and Luca Greco. *E-learning and personalized learning path: A proposal based on the adaptive educational hypermedia system*. International Journal of Emerging Technologies in Learning (iJET), 9(2):pp-9, 2014.
- [7] Francesco Colace, Massimo De Santo, and Luca Greco. *An adaptive product configurator based on slow intelligence approach*. Int. J. Metadata Semant. Ontologies, 9(2):128–137, April 2014.
- [8] Francesco Colace, Massimo De Santo, Luca Greco, and Paolo Napoletano. *Text classification using a few labeled examples*. Computers in Human Behavior, (0):–, 2014.
- [9] Francesco Colace, Massimo De Santo, Luca Greco, and Paolo Napoletano. *Weighted word pairs for query expansion*. Inf. Process. Manage., 51(1):179–193, 2015.
- [10] "Francesco Colace, Massimo De Santo"(2011). *A Network Management System Based on Ontology and Slow Intelligence System*. INTERNATIONAL JOURNAL OF SMART HOME. Vol. 5-3. Pag.25-38 ISSN:1975-4094.
- [11] Dey, A. K. *Understanding and Using Context*. Personal and Ubiquitous Computing 5, 1 (2001), 4-7.
- [12] Dey, A. K., and Abowd, G. D. *CybreMinder: A Context-Aware System for Supporting Reminders*. In Proc. HUC '00 (2000), pp. 172-186.
- [13] Xiaowen Ding, Bing Liu, and Philip S. Yu. *A holistic lexicon-based approach to opinion mining*. In Proceedings of the 2008 International Conference on Web Search and Data Mining, WSDM '08, pages 231–240, New York, NY, USA, 2008. ACM.
- [14] Ruihai Dong, Michael P. O'Mahony, Markus Schaal, Kevin McCarthy, and Barry Smyth. *Sentimental product recommendation*. In Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pages 411–414, New York, NY, USA, 2013. ACM.
- [15] Paul Dourish. *What we talk about when we talk about context*. Personal and ubiquitous computing, 8(1):19–30, 2004.
- [16] Andrea Esuli and Fabrizio Sebastiani. *Sentiwordnet: A publicly available lexical resource for opinion mining*. In In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC 2006, pages 417–422, 2006.
- [17] Gayatree Ganu, Yogesh Kakodkar, and Am'elie Marian. *Improving the quality of predictions using textual information in online user reviews*. Inf. Syst., 38(1):1–15, March 2013.
- [18] Yoshinori Hijikata, Kazuhiro Iwahama, and Shogo Nishida. *Content-based music filtering system with editable user profile*. In Proceedings of the 2006 ACM symposium on Applied computing, SAC '06, pages 1050–1057, New York, NY, USA, 2006. ACM.
- [19] Katerina Kabassi. *Personalisation systems for cultural tourism*. In Multimedia services in intelligent environments, pages 101–111. Springer, 2013.
- [20] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. *Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering*. In Proceedings of the fourth ACM conference on Recommender systems, pages 79–86. ACM, 2010.
- [21] Yehuda Koren. *Factorization meets the neighborhood: a multifaceted collaborative filtering model*. In Proceedings of the 14th ACM SIGKDD International conference on Knowledge discovery and data mining, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.
- [22] Cane WK Leung, Stephen CF Chan, and Fu-lai Chung. *Integrating collaborative filtering and sentiment analysis: A rating inference approach*. In Proceedings of The ECAI 2006 Workshop on Recommender Systems, pages 62–66. Citeseer, 2006.
- [23] Veronica Maidel, Peretz Shoval, Bracha Shapira, and Meirav Taieb-Maimon. *Evaluation of an ontology-content based filtering method for a personalized newspaper*. In Proceedings of the 2008 ACM conference on Recommender systems, RecSys '08, pages 91–98, New York, NY, USA, 2008. ACM.
- [24] Katarzyna Musial, Krzysztof Juszczyszyn, and Przemyslaw Kazienko. *Ontology-based recommendation in multimedia sharing systems*. System Science, 34:97–106, 2008.
- [25] Nikolaos Pappas and Andrei Popescu-Belis. *Sentiment analysis of user comments for one-class collaborative filtering over ted talks*. In Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, pages 773–776, New York, NY, USA, 2013. ACM.
- [26] Michael Pazzani and Daniel Billsus. *Content-Based Recommendation Systems*. pages 325–341. 2007.
- [27] Naren Ramakrishnan, Benjamin J. Keller, Batul J. Mirza, Ananth Y. Grama, and George Karypis. *Privacy risks in recommender systems*. IEEE Internet Computing, 5:54–62, November 2001.
- [28] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. *GroupLens: An open architecture for collaborative filtering of netnews*. pages 175–186. ACM Press, 1994.
- [29] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.
- [30] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. *Methods and metrics for cold-start recommendations*. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.
- [31] Vivek Kumar Singh, Mousumi Mukherjee, and Ghanshyam Kumar Mehta. *Combining collaborative filtering and sentiment classification for improved movie recommendations*. In Chattrakul Sombatheera, Arun Agarwal, Siba K. Udgata, and Kittichai Lavangnananda, editors, MI- WAI, volume 7080 of Lecture Notes in Computer Science, pages 38–50. Springer, 2011.
- [32] Johann Stan, Fabrice Muhlenbach, Christine Largeron, et al. *Recommender systems using social network analysis: Challenges and future trends*. Encyclopedia of Social Network Analysis and Mining, pages 1–22, 2014.
- [33] Xiaoyuan Su and Taghi Khoshgoftaar. *A survey of collaborative filtering techniques*. Advances in Artificial Intelligence, 2009, 2009.
- [34] Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng. *Collaborative competitive filtering: learning recommender using context of user choice*. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pages 295–304. ACM, 2011.
- [35] Hilmi Yildirim and Mukkai S. Krishnamoorthy. *A random walk method for alleviating the sparsity problem in collaborative filtering*. In Proceedings of the 2008 ACM conference on Recommender systems, RecSys '08, pages 131–138, New York, NY, USA, 2008. ACM.
- [36] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit S Dhillon. *Parallel matrix factorization for recommender systems*. Knowledge and Information Systems, pages 1–27, 2013.
- [37] Zheng, Nan, and Qiudan Li. *A recommender system based on tag and time information for social tagging systems*. Expert Systems with Applications 38.4 (2011): 4575-4587.
- [38] Xujuan Zhou, Yue Xu, Yuefeng Li, Audun Josang, and Clive Cox. *The state-of-the-art in personalized recommender systems for social networking*. Artif. Intell. Rev., 37(2):119–132, February 2012.
- [39] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. *Movie review mining and summarization*. In Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM '06, pages 43–50, New York, NY, USA, 2006. ACM.