UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

ISMAEL MESSIAS GOMES CARDOSO

VULCONT: A RECOMMENDER SYSTEM BASED ON CONTEXTS HISTORY ONTOLOGY

São Leopoldo
2017

Ismael Messias Gomes Cardoso

VULCONT: A RECOMMENDER SYSTEM BASED ON CONTEXTS HISTORY ONTOLOGY

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Advisor:
Prof. Dr. Jorge Luis Victória Barbosa

São Leopoldo
2017

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

Cardoso, Ismael Messias Gomes

Vulcont: A Recommender System based on Contexts History Ontology / Ismael Messias Gomes Cardoso — 2017.

81 f.: il.; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2017.

“Advisor: Prof. Dr. Jorge Luis Victória Barbosa, Unidade Acadêmica de Pesquisa e Pós-Graduação”.

1. Recommender Systems. 2. Ontology. 3. Contexts History. 4. Collaborative Filtering. I. Título.

CDU 004.89

Bibliotecária: Carla Maria Goulart de Moraes — CRB 10/1252

ACKNOWLEDGEMENTS

- Thanks to my family for unconditional support.
- Thanks to my friends for my comprehensible missing.
- Thanks to Prof. Barbosa to motivate me to be a better person everyday.
- Thanks to my colleagues Leandro D'Avila, Daniel Dupont, Fábio Dias, and Rodrigo Remor for all the teamwork.
- Thanks to my all workmates at SAP, specially Roger Stein for technical and non-technical mind-blowing conversations.
- Thanks to everyone that, directly or indirectly, helped me to reach this point and conclude this step of my journey.

"*Choice is the act of hesitation that we make before making a decision.*
It is a mental wobbling, and so we are always in a dither of doubt as to whether we are behaving the right way, doing the right thing, and so on and so forth, and lack a certain kind of self confidence.
And if you see you lack self confidence, you will make mistakes through sheer fumbling.
If you do have self confidence you may get carried away doing the entirely the wrong thing.
You have to regard yourself as a cloud, in the flesh, because you see clouds never makes mistakes.
Did you ever see a cloud that was misshapen?
Did you ever see a badly designed wave?
Heh, no they always do the right thing!
But if will treat yourself for awhile as a cloud, a wave, and realize that you can't make a mistake, whatever you do.
Because even if you do something that seems to be totally disastrous, it all come out in the wash somehow or other.
Then through this capacity you will develop a kind of confidence, and through confidence you will be able to trust your own intuition.
This is the middle way, of knowing it has nothing to do with your decision to do this or not, whether you decide you can't make a mistake or don't decide it, it is true anyway, that you are like cloud and water.
And through that realization without overcompensating in the other direction, you will come to the point of where you begin to be on good terms with your own being and to be able to trust your own brain".
Alan Watts

ABSTRACT

The use of recommender systems is already widespread. Everyday people are exposed to different items' offering that infer their interest and anticipate decisions. The context information (such as location, goals, and entities around a context) plays a key role in the recommendation's accuracy. Extending contexts snapshots into contexts histories enables that information to be exploit. It is possible to identify context's sequences, similar contexts histories and even predict future contexts. In this work we present Vulcont, a recommender system based on a contexts history ontology. Vulcont merges the benefits of ontology reasoning with contexts histories in order to measure contexts history similarity, based on semantic and ontology's properties provided by context's domain. Vulcont considers synonymous and classes' relations to measure similarity. After that, a collaborative filtering approach identifies sequences' frequency to identify potential items for recommendation. We evaluated and discussed the Vulcont's recommendation in four scenarios in an offline experiment, which presents Vulcont's recommendation power, due the exploit of semantic value of contexts history.

Keywords: Recommender Systems. Ontology. Contexts History. Collaborative Filtering.

RESUMO

O uso de sistemas de recomendação já é amplamente difundido. Diariamente pessoas são expostas a ofertas de itens que inferem seus interesses e antecipam decisões. As informações de contexto (como localização, objetivos, e entidades que cercam um contexto) tem um papel chave na acurácia da recomendação. Ampliando o uso de contextos para histórico de contextos, essa informação pode ser explorada ainda mais. É possível identificar sequências de contextos, similaridade entre histórico de contextos, e até prever contextos futuros. Neste trabalho é apresentado o Vulcont, um sistema de recomendação baseado numa ontologia de histórico de contextos. Vulcont une os benefícios do raciocínio da ontologia com o uso de histórico de contextos para quantificar a similaridade entre histórico de contextos, com base na semântica e outras propriedades da ontologia definidas pelo domínio do contexto. Vulcont considera sinônimos e relações de classes para calcular a similaridade. Por seguinte, um filtro colaborativo identifica a frequência de sequências para estimar items em potencial de recomendação. As recomendações do Vulcont foram avaliadas e discutidas em quatro cenários num experimento *offline*. O experimento apresentou o poder de recomendação do Vulcont, que é devido a exploração do valor semântico de histórico de contextos.

Palavras-chave: Sistemas de Recomendação. Ontologia. Histórico de Contextos. Filtragem Colaborativa.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API	Application Programming Interface
CBR	Case-Based Reasoning
CF	Collaborative Filtering
GUI	Graphical User Interface
JSON	JavaScript Object Notation
OMDb	Open Movie Database
OWL	Web Ontology Language
POI	Points Of Interest
RS	Recommender System
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

CONTENTS

1 INTRODUCTION

The world is currently in the *big data* era (CHEN; MAO; LIU, 2014). Information is gathered in large scale, and even in excess (HILBERT; LOPEZ, 2011). Among all this information, it is now more difficult to make choices or to decide between a huge variety of options. Some decades ago, the recommender systems (RSs) (BOBADILLA et al., 2013) arose as an option to solve that problem. Their purpose is to provide customized services in several different applications. Nowadays, the RSs are already a consolidated solution as information filtering systems. They have changed the way applications offer different services to end users. It is not hard to identify RSs in daily life, there are different computing areas which use them to provide personalized recommendations with a high level of quality and accuracy (JANNACH et al., 2009).

To create these personalized services, the RSs need to estimate the personal taste of users, and propose them a service or item which, based on some kind of knowledge, the RS considers to be relevant for the users' intentions. Often the RSs take advantage from user's context information gathered from interactions within the system. They use that information to predict user's goals and preferences. This approach is also known as *implicit data collecting* (ADOMAVICIUS; TUZHILIN, 2005) (OARD; KIM, 1998).

The use of *context information* (DEY, 2001) is widely spread in many applications from *ubiquitous computing* (ubicomp) (WEISER, 1999). Some of these applications use RSs to improve services offerings. According to Dey (2001), *context* is: "any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves". For example, the context information can be related to a physical context (location and time), environmental context (weather, luminosity and noise), personal context (social and group activities), application context (emails, web history), among others.

Contexts history information might be also used in some approaches of RSs, such as *collaborative filtering* (CF). This information can support RSs to estimate items of interest for a specific user based on previous experiences. The chronological collection of context information, known as *contexts history* (SMITH, 2008) (ROSA et al., 2015) (ROSA; BARBOSA; RIBEIRO, 2016), also called *trail* (LEVENE; PETERSON, 2002) (DRIVER; CLARKE, 2008) (SILVA et al., 2010) (CARDOSO et al., 2016) (BARBOSA et al., 2016), stores the user's activity among visited contexts. Using contexts history, RSs can recommend content and services based on the user's past experiences. Dey, Abowd and Salber (2001) already mentioned the importance of using contexts history for decision-making systems, which is the case of RSs. Also, Silva et al. (2010) used contexts histories to improve services offering in context-aware systems. In other words, the past activities of the user were stored as contexts history; such as performed activities, used applications, accessed contents, and any other possible information; so the applications used a source of information more complete.

With the evolution of RSs, new approaches were presented, and new techniques were developed in order the improve the efficiency of recommendations. In one of these approaches, the use of ontologies became an ally with the benefits of formal languages. Ontologies came up with the goal of representing a shared background knowledge for a specific domain (GRUBER, 1995). Adding ontologies in RSs allowed knowledge-based approaches to be combined with classical machine learning algorithms. So far, ontologies have been frequently used in RSs along with machine learning, statistical correlations, user profiling and domain specific heuristics (PAIVA; COSTA; SILVA, 2013).

In this work we propose Vulcont, a recommender system based on a contexts history ontology. Using ontology reasoning, Vulcont is able to explore the semantic value (taxonomy and synonymy) of contexts history, that compares the contexts history of an *Actor* (the entity which is the recommendation's target), with other contexts histories from a community. With that comparison, Vulcont can leverage contexts history similarity, identifying the most similar contexts histories that can provide *Objects* (recommendation's items) which are suggested to *Actors* as personalized recommendations, according to their preferences.

1.1 Motivation

In last decades, the data gathering and data flow have increased considerably. How to extract relevant information from a huge data volume is a question that motivated and still motivates studies from several areas, such as Bayesian networks, decision trees, support vector machines, neural networks, among others. These research fields that extract useful information from a huge data volume are been increasingly required.

As another subclass of information filtering systems, the recommender systems arise as a solution for the data volume problem. They came up in order to help people to handle high information flow, providing personalized recommendations of items and services for a potential interest. Among all RSs' approaches, we can highlight the CF which works based on information gathered from a users' community. This type of RS is widely used and had been studied over the last fifteen years.

Ontologies have been used in RSs since last decade. They already proved to be a valid approach whenever a knowledge needs to be shared, or to explore the use of semantic relations between concepts (KANG; CHOI, 2011). Therefore, different studies prove that ontologies do improve the recommendation power of RSs (NAUDET et al., 2008) (MORENO et al., 2013). Using context information we can leverage crucial data to provide personalized recommendations. With the semantic reasoner from ontologies, Vulcont can evaluate contexts history similarity in a more complete way. For example, in an application of movies' recommendation, the ontology can specify that genres “Terror” and “Horror”, or “Mystery” and “Thriller” are synonymous. Also, that “Anime” is a subgenre of “Animation”, as “Cyberpunk” is from “Sci-Fi”, or “Fairy Tail” from “Fantasy”.

1.2 Problems and Research Questions

In this proposed approach for RS, Vulcont uses contexts history ontology to leverage contexts history similarity exploring semantic relations. Hence, there are several questions that arise regarding this research. Some of these questions are:

- How to perform personalized recommendation based on contexts history ontology?
- How to calculate contexts history similarity from contexts history ontology?
- How effective is the use of ontology to model contexts histories and reasoning user's experiences from it?
- How to model an ontology to cover contexts history domain as generic as possible?
- How to specify the quality of recommendation in order to compare with common approaches?

In this work, we look for the answers of above questions. We would like to investigate the use of ontology for contexts history domain, and also its reasoning to explore contexts history knowledge as information source for personalized recommendations.

1.3 Objectives

This study aims to explore contexts history ontology to provide personalized recommendations. Vulcont uses ontology reasoning to calculate the similarity between two contexts histories, which is a source information to elaborate recommendations. So according to the contexts history that every user has, we are able to calculate the similarity with other contexts history from community. With the most similar ones, Vulcont can offer *Objects* as personalized recommendations. In order to achieve this major goal, we list the objectives as follows:

- To analyze the theoretical foundation of areas involved, clearly defining the concepts used in the study.
- To identify relevant related works, and compare them based on clear criteria.
- To model and implement the contexts history ontology, considering also reuse of existent ontologies
- To specify the recommendation system that using contexts history ontology is going to provide its recommendations
- To develop a functional prototype based on Vulcont model's definitions.
- To evaluate the developed prototype and validate Vulcont's purpose, considering possible improvements for future works.

1.4 Methodology

Initially, we researched about the use of ontologies in the area of recommender systems. We leverage the scenarios and advantages of using ontologies to generate recommendations. We researched the context-aware computing area to identify main studies related to user's contexts, and the use of contexts histories. Then we merged both areas, identifying study proposals of recommender systems based on contexts history.

After that, we searched about related works, elaborating the problems' definition. Once the problems were defined, we identified a clear possibility for contribution. In the sequence, we defined the contexts history ontology, as well as Vulcont's model. We developed and tested the contexts history ontology that is used by Vulcont. The following steps for the research were: to describe evaluation scenarios, to implement Vulcont's prototype, and then evaluate Vulcont's model.

1.5 Outline

This study is organized into 6 chapters. The second chapter presents background of concepts used in the work. Chapter three explores, describes, and compares related works and the main contribution of Vulcont. The fourth chapter presents the Vulcont's model, the contexts history ontology and its development. The fifth chapter describes the technical aspects of Vulcont's implementation, and reports and dissects its evaluation scenarios. Finally, chapter six concludes the work with the final considerations, compares Vulcont with related works, and points out aspects to be considered and improved in future works.

2 BACKGROUND AND BASIC CONCEPTS

In this chapter we are going to present several definitions and concepts from areas involved in this study. Firstly we will present the recommender system concept, its classifications and characteristics. Then we will introduce ontology, its definition, why to use it, and some classifications as well. Following, it is presented the concept of contexts history, also known in other studies as trails.

2.1 Recommender Systems

Every software tool or technique which has the major goal to provide items for a user is considered a recommendation system (MAHMOOD; RICCI, 2009) (RESNICK; VARIAN, 1997) (BURKE, 2007). Those suggestions can be related to many different decision-making process, for instance: Which item to buy? Which music to listen? Which news to read?

The generic term *item* directly refers to what the system recommends to the user. Therefore, the RS is usually designed considering the kind of item that is going to be recommended, such as movies, books, news, products, and so on. The items are recommended according also to the graphical user interface (GUI) and the main recommendation technique. All that to provide recommendations to users, helping their decision-making process.

From beginning, the RSs are designed for inexperienced users or users which have difficulty to evaluate a huge diversity of items that a website is able to provide (RESNICK; VARIAN, 1997). Nowadays, many web systems from e-commerce (Electronic commerce) are using RSs. One example is the website *Amazon.com*, which customizes its online store according to every user. Since the recommendations are dynamically generated, different users have different recommendations. However, still exists recommendations which are non-customized, and hence do not identify needs and preferences of user (SAMPAIO, 2006). This approach of recommendation is simple and widely used by magazines, newspapers, and news websites. A typical example is the Top 10 most viewed news. Although this approach is effective in certain circumstances, this non-customized recommendation is not attractive for studies in RS research field.

In order to elaborate customized recommendations, the RS requires information related to every user. Therefore, it is necessary the RS to manage users' profile (JANNACH et al., 2009). Although the existence of user profiles in RSs is totally essential, there are many approaches to consume that information, once it depends on recommendation technique used in the specific system. The user preferences can be obtained implicitly or explicitly, for instance monitoring the user's behavior, or explicitly asking the preferences and needs to the user.

In our daily life, it is common to consult other people opinions to help in decision-making situations (MAHMOOD; RICCI, 2009) (MCSHERRY; MIRONOV, 2009). For example, it is absolutely normal asking people for recommendation of musics, or books; in application jobs, you can use your recommendation letter to improve your chances to get the job; you can read a movie review to check if it does worth watching it.

Intending to imitate this specific behavior, the firsts RSs have algorithms that generate recommendations to a specific user based on users community. The items' recommendation is performed according to similarity of users' tastes. This approach is known as Collaborative Filtering (CF). It considers that if two users are similar, then future recommendations for these users will be relevant for both of them. So unevaluated or unknown items can be recommended to a user based on previous evaluation of another user with similar taste/opinion.

The researches in RS field are relatively new if compared with another classic system tools (such as databases and search tools). The RSs arose as a research field in the middle of 90s (GOLDBERG et al., 1992) (MAHMOOD; RICCI, 2009) (MCSHERRY; MIRONOV, 2009) (ANAND; MOBASHER, 2005). Since past years, the interest in this field is growing accordingly. For example, the Netflix, a service of online streaming of movies and TV series, promoted a competition to significantly optimize their RS. The reward for the winner was 1 million dollars (KOREN; BELL; VOLINSKY, 2009).

Among all recommendation approaches, we can highlight some of them, which have several studies in their areas along past years. This is the case for CF, recommendation based on content, based on knowledge, and hybrid approaches. It is also possible to highlight some new approaches, which is the case of RS based on context. The description of these approaches will be discussed in following subsections.

2.1.1 Collaborative Filtering

According to Jannach et al. (2009), the main idea in CF approaches is to explore historic information from users, or the opinion of a users community, in order to predict items that the user might be interested or can help user to achieve a certain goal. Nowadays, this kind of RS approach is already spread into several industry fields, mainly as a tool in commerce websites. This tool is able to customize the entire website according to user needs or preferences, promote additional items, and hence sales increasing.

Considering an academical perspective, this RS approach has been explored in a long term. Hence its advantages, performance, and limitations are already well-known. Along the years, several algorithms and techniques were proposed, implemented, and tested by simulation or real data cases. The purely CF approaches usually have an input matrix with user-item classification, and with that, two outputs can be generated: (1) a value, which indicates how much the user is willing to like the item; and (2) a list of several recommended items, for example a Top-10, which obviously exclude items already known by the user.

The CF is considered by many authors as one of main recommendation methods. This approach considers the user (actual and active) and the items' evaluations previously performed. Using this information as input, it is possible to identify similar users (nearest neighbor) which have similar past evaluations with the active user. Therefore, for every item which the user does not know, we can estimate the interest for that specific item according to nearest neighbors evaluation. The method has two assumptions: (1) users with similar interests in the past, will still have similar interests in future; and (2) the user preferences remain statics and consistent along the time.

The success of CF is real, so are the other challenges that arise with virtual commerce. In that particular scenario, it is complicated to identify in real time the nearest neighbors to perform recommendations. In this case, we use a different approach which is based on items. In that technique, it is possible to perform offline pre-processing, which allows recommendations from a large-scale matrix of evaluations (SARWAR et al., 2001). This different approach uses similarity of items, not of users as in CF.

There are several ways to get an item evaluation from the user. Explicitly asking the user is probably the most accurate one (JANNACH et al., 2009). In most cases, scales with 5 or 7 points are used. The scales approach has a low complexity once they can be converted into values and hence used in similarity calculations. The study of Cosley et al. (2003) explored the use of different evaluation scales and changes in the evaluation behavior of user. The scale's size is also important, which changes according to application area. One example is the study of Goldberg et al. (2001) which uses a scale from -10 to 10 in a RS for humorous domain. It is important to identify that the scales should change according to the application they are designed for, and the precision required. This way, it is possible to correctly evaluate without losing information related to user preferences. Some interesting topics related to scales, such as how the recommendation precision is affected by scale, or how to determine the optimal number of scales, are still open fields, as the study of Cosley et al. (2003) refers.

The main drawback of explicit evaluation is the additional effort that the user has to perform. Sometimes the user is not willing to provide it, or just refuses to do it. Therefore, the number of obtained evaluations will considerably decrease, which can also affect the quality of recommendations. At the same time, the study of Schafer et al. (2007) comments that this problem is not critical, once it is only necessary a few pioneer users. These pioneer users provide many items evaluations after starting the system, which will be stabilized after that.

In other hand, the implicit evaluations are collected by an application which is integrated with the RS. For example, if the user buys an item, this could be considered a positive evaluation. The system can also monitor the search history of user. If the user stays in website page long enough, the system could also estimate it as a positive feedback related to the item.

Even though the implicit evaluations do not interact with user, and the system can collect them constantly, it is impossible to guarantee that the behavior's understanding is correct. For instance, the user can buy a book, but it does not mean that he/she liked it; or the book could also had been bought to someone else. Since the evaluations are generated based on user's behavior, it is possible to use only the most consistent ones, based on certain criteria. Confirming this aspect, the same study from Schafer et al. (2007) reports that collecting implicit evaluations can even result in generated user models more accurate than explicit evaluations.

2.1.2 Based on content

RSs based on content use an approach to recommend similar items from the ones which the user showed interest in past. That way, the RS based on content compares the information from profile user, getting user's preferences and interests previously stored, based on item's attributes. After that, the system estimates similar items which the user could be interest, providing the recommendation (JANNACH et al., 2009).

The systems which provide recommendations based on content analyze a set of documents and/or items' descriptions which were already evaluated by the user, and then generate the user model/profile based on attributes of those items (MLADENIC, 1999). This generated profile represents the structured interests of the user, which will be posteriorly used by the RS to recommend new items. If the profile matches the user preferences, the information processing is significantly increased. For example, this information can be used to filter search results, determining if the user has interest in certain item, displaying only the interested ones.

Using the based on content approach, there are some advantages comparing with the collaborative approaches (RICCI; ROKACH; SHAPIRA, 2010):

- User independence - Recommendations based on content are able to generate user profile through evaluations provided by the user. While in CF, it is necessary several different users' evaluations so the system is able to provide recommendations.

- Transparency - Explanations of how the RS provides its recommendations can be explicit, showing the content which induced the recommendation of certain item. This can also indicate if the recommendation is trustful or not. In an opposite way, CF approach is considered a "black-box" once the explanation of a recommendation is not provided, since it is based on unknown users with similar interests.

- New item - RSs based on content are able to provide recommendation of an item, even if the item was never evaluated before. So this approach is not impacted by the problem of "first evaluator", which occurs in CF approach. The "first evaluator" problem occurs when the item was not evaluated, and therefore can not be recommended.

However, there are still some drawbacks to be considered in the approach based on content, as shows the study of Ricci, Rokach and Shapira (2010):

- Limited content analysis - The techniques used in recommendations based on content have a natural limit associated to the number and type of characteristics from items. Domain knowledge is frequently required. Hence, RS based on content can not properly recommend if the content does not have substantial information to differentiate items that user might have interest from the items where user is not interest at all.

- Excess of specialization - RSs based on content do not have a method to recommend unusual items. The systems suggests item which matches the profile user, then only similar items are recommended. Thereby, there is a tendency to the system to produce recommendations with a low innovation to the user.

- New user - A significantly evaluation quantity should be collected before the system comprehend the users preferences and interests, to then provide accurate recommendations. Therefore, in cases where the user has a few performed evaluations (such as a new user), the system will not be able to perform trustful suggestions.

For the problems reported above, there are studies which minimize them. For example, the study of *WordNet* (MILLER et al., 1990) explored the limited content analysis problem, while the study of Sheth and Maes (1993) checked the issue regarding the excess of specialization.

2.1.3 Based on knowledge

While RSs which use CF techniques rely on community evaluation as knowledge source for generating recommendations, RSs based on content use different sources of information to determine if the user would like certain item. Both systems have advantages when it comes to knowledge acquisition and maintenance, which have a low cost.

Both presented approaches have pros and cons, but there are also scenarios where these approaches will not have the desired performance. For example, some of these scenarios are the housing, cars and computers market. These items are not frequently bought by the same user. In such scenarios, a purely collaborative RS would not be able to recommend due the low number of available evaluations (BURKE, 2000). Furthermore, the time has a key role since past evaluations might already be outdated (mainly in computers scenario). Hence the item's evaluation can not help in RSs based on content. The use of specific requirements is also unusual in RSs that use CF or based on content approaches.

For that mentioned problems, there is new category of RSs, which is based on knowledge. The main advantage of these RSs is the fact they do not use evaluations as source for estimating recommendations. The recommendations are generated independently of individual evaluations of users. Instead, the RSs based on knowledge are focused on information filtering (KONSTAN et al., 1997) (PAZZANI, 1999), listing the most probable items that the user might be interested.

The recommendation process in RSs based on knowledge is highly interactive. This interactive aspect changed the interpretation of filtering systems into a broader one, which we define today as RS (BURKE, 2000). The recommendations performed by RSs based on knowledge rely on any knowledge sources which are not explored by CF or based on content approaches (BURKE, 2000) (FELFERNIG; BURKE, 2008).

There are basically two types of RSs based on knowledge: RSs based on restriction (FELFERNIG; BURKE, 2008) (FELFERNIG et al., 2006) (ZANKER; JESSENITSCHNIG; SCHMID,

2010) and RSs based on case (BRIDGE et al., 2005) (BURKE, 2000). Both approaches are similar in recommendation process, where the user specifies requirements and the system looks for items which matches the specification. If no item is found, the user should change his/her requirements. The difference between the two approaches rely on how the knowledge is provided. While the one based on case focus on similar items using different similarity measures, the RSs based on restriction depends on a specific set of defined restrictions.

In RSs based on restrictions, the recommended set of items is determined by items which fit the defined restrictions. In other way, RSs based on case use similarity to quantify the difference between two items, the one required by the user, and the possible items available for recommendations.

2.1.4 Based on context

The study of Ranganathan and Campbell (2003) defines context as "any information about circumstances, objects or conditions that surround the user, which are relevant for the interaction between user and ubiquitous environment". Therefore, the context becomes an additional information when compared to traditional information in user profiles (as demographics or interests), and can refer to information in several contexts such as physical (location, time), environmental (weather, light, sounds levels), informatives (stock exchange, games score), personal (health, humor, agenda, activities), social (group activities, social activities), applications (emails, web history), and systems (network traffic load, printer status) (RANGANATHAN; CAMPBELL, 2003).

Most of existent approaches in RSs are focused on recommend most relevant items to a specific user, not considering context information, such as time, location, or user's company. Hence, traditionally RSs handle applications with only two distinct entities with context association: users and items. However, in many applications (for instance tourism and adaptive web areas) considering only user and item might be enough. It is also important to consider context information in recommendation process to suggest items to user under specific circumstances.

To consider context information in RSs already is a requirement, inherited from ubiquitous computing. The work of Schilit, Adams and Want (1994) revealed the most important aspects to be considered: where you are?; who is with you?; which are the resources close to you?. The current user's location, user's companies, and the resources availability next to user, if properly explored, can increase recommendation accuracy in mobile applications. We can notice that the boundary between user model and context model is not quite defined. Particularly, differing user's interest in short or long term is a research point explored by studies related to user's profile modeling and personalization.

Among all the application domains, the use of RSs based on context is mainly related to areas such as mobile commerce, tourism, visiting guides, entertainment, and home computing. When analysing the distribution of works related to RSs based on context, clearly the tourism area has the majority of studies. However, is it not huge the quantity of applications which were evaluated in various research fields (JANNACH et al., 2009). Considering only the point of view of recommendation technology, in most practical cases the application only handles context awareness as a filter, where the inadequate items are removed according to context parameters (for instance, the location information). Consequently, this requires more studies related to RSs with context awareness.

2.2 Ontology

Over the past few years, the ontology's use has become widely spread. Several different areas are interested in its use and retrieve the best from its benefits. These areas such as web technologies, expert systems, and artificial intelligence are requiring the use of semantic meaning and relations to infer information from a huge knowledge base. The technology is currently in *big data* era, where information is abundant and even gathered in excess. The ontologies are now a relevant solution option when it comes to efficient and effective information gathering.

The recommender systems (RSs) has also emerged along the huge gathering of information. Recommender systems are now a vital solution, widely used in manifold areas that want a personalized offering of service. Simple examples are easily identified in normal day life: you access a shopping website, several items are offered; you want a movie to watch based on your personal taste, most relevant films are available to you; you need some tips to find best place for your vacations, you can count on many tourist systems that can offer you such information.

The use of ontology in RSs has started in last decade and until now it is a research field with high growth potential. This occurs due the current importance of recommender systems, which are widely used by big and small companies that want to offer a personalized service to every user; due the quantity of studies that approach new techniques and architectures for different domains; due problem's complexity, which remains open since there is not a best approach that fits all.

The term *ontology* has many different meanings. The term is originally related to philosophy, already been studied for over two thousand years, where it is a systematic account of existence. In computing science, the term became relevant in 90s, mainly related to Knowledge Acquisition (GUARINO; OBERLE; STAAB, 2009). In that time, Gruber (1995) set the basic definition as following: "An ontology is an explicit specification of a conceptualization". Later in 1997, Borst (1997) defined *ontology* as a "formal specification of a shared conceptualization". Finally in 1998, Studer, Benjamins and Fensel (1998) merged both definitions into "An ontology is a formal, explicit specification of a shared conceptualization". But still after that definition, the concept of every statement's element is ambiguous. So also the elements need their definitions as follows:

- Conceptualization - Gruber (1995) referred its definition from Genesereth and Nilsson (1987) which defined it as: "A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly".

- Formal and explicit specification - A explicit specification is related to specify the extension of every relation in the described world. The relation can not have an ambiguous comprehension, this is difficult since we humans do this specification implicitly. While the formal refers to expressions that a machine can read, so natural language is removed.
- Shared - Even though the conceptualization can not be exactly shared, since it depends on conceptualization of every individual, it is possible to share an approximation of conceptualization. A non-shared ontology will be less effective then a shared one, since the stakeholders need to know all primitive terms that the ontology represents.

In 1999, Amann and Fundulaki (1999) defined Ontology in a mathematical way, where the ontology is a triple O = (C, S, isa) where:

1. *C* is a set of classes, where every class refers to a set of real world objects, which are instances.
2. *S* is a set of slots, where every slot can refer to a property of class, or a relation between two classes.
3. *isa* is a set of inheritance relationships between classes. These relationships have subset semantics, and organize the classes in tree structures.

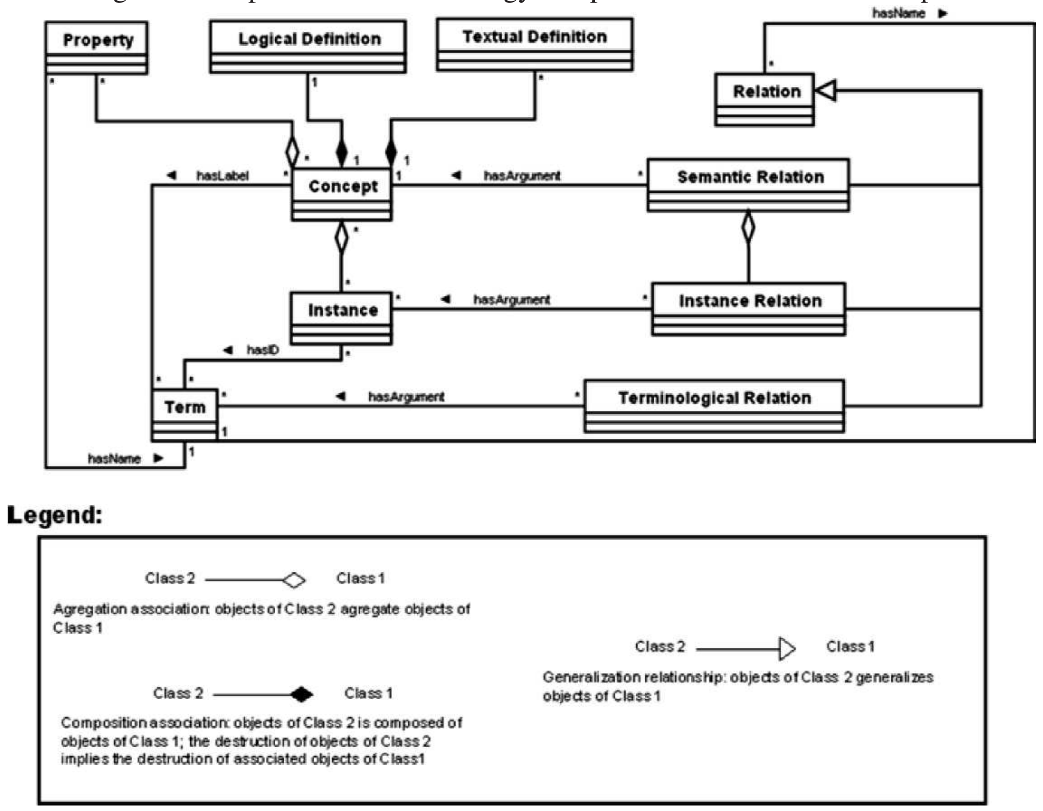
In 2001, Noy, McGuinness et al. (2001) defined ontology as a formal explicit description of concepts, or classes in a domain of discourse. The slots of classes are properties such as attributes and relation between two classes. There are also restrictions on slots (also called as facets or role restrictions), which maintain and guarantee the ontology's semantic integrity.

2.2.1 Ontologies Classification

According to the evolution of ontology's concept, there are still different variations of ontology. An ontology can represent different computer science objects according to needs (ROUSSEY et al., 2011). Every variation of ontology has a specific application for use, since their content differ. For instance, an ontology can be:

- a thesaurus for information retrieval
- a OWL (Web Ontology Language) model for use of linked-data

Figure 1 – Representation of ontology components and their relationships.


hasName ►
Property
Logical Definition
Textual Definition
Relation
◄ hasLabel
Concept
◄ hasArgument
Semantic Relation
Instance
◄ hasArgument
Instance Relation
◄ hasID
Term
◄ hasArgument
Terminological Relation
hasName ►
Legend:
Class 2 Class 1
Agregation association: objects of Class 2 agregate objects of Class 1
Class 2 Class 1
Composition association: objects of Class 2 is composed of objects of Class 1; the destruction of objects of Class 2 implies the destruction of associated objects of Class1
Class 2 Class 1
Generalization relationship: objects of Class 2 generalizes objects of Class 1

Source: (ROUSSEY et al., 2011)

- a XML (eXtensible Markup Language) schema for database contexts

There are specific works that focus on ontology's classification (LASSILA; MCGUINESS, 2001), (GÓMEZ-PÉREZ; FERNÁNDEZ-LÓPEZ; CORCHO, 2006), (BORGO, 2007), (ROUSSEY et al., 2011). These studies classify ontologies, but each one considers a different perspective. There is the ontology's classification based on expressivity and formality of language used (natural, formal, etc.), and also based on the scope of the objects' description.

The ontology's expressivity relies on the components used. These components can be concepts, properties, instances, axioms, among others. Roussey et al. (2011) represented these components as shown in Figure 1, where the ontology's classification is based on language expressivity. The expressivity can be a *Textual Definition* of a *Concept*, or a set of *Properties*, or even a *Logical Definition*, these three are known as intentional definitions. Also, the *Concept* can be defined by a set of *Instances*, called extensional definition. *Concepts*, *Instances* and *Properties* use *Terms* so humans can understand by just reading. The ontology's component are linked by *Relations*:

- *Semantic Relation* - Connect only *Concepts* together.

- *Instance Relation* - Connect only *Instances*. They are often instances of *Semantic Relations*, but not necessarily.

- *Terminological Relation* - Express the relationships that terms can have, for example synonyms, antonyms, etc.

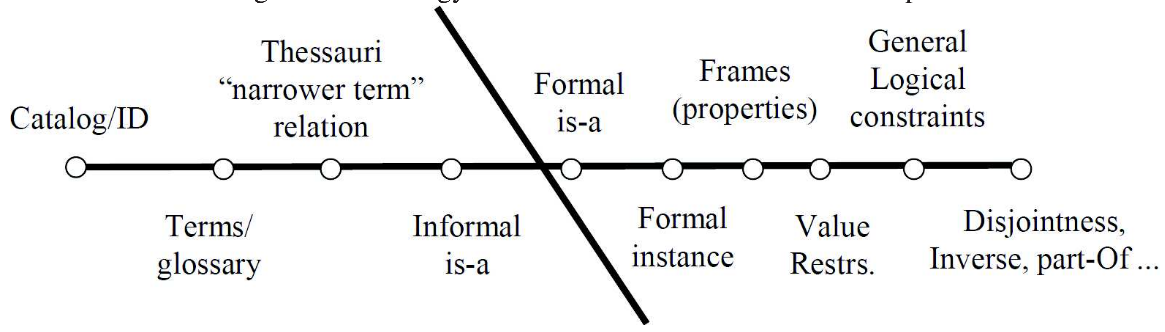
The work of Lassila and McGuiness (2001), also cited in Gómez-Pérez, Fernández-López and Corcho (2006), presents a classification of ontology based on expressivity and how the information is internally structured. The classification can be summarized as shows Figure 2. The main categories are listed as follows:

- *Catalogs* - The simplest notion of ontology can be a controlled vocabulary, a simple catalog with a finite list of terms.
- *Glossaries* - Beyond a list of terms, they also contain their meanings in natural language.
- *Thesauris* - They provide additional semantics in relations between terms, for example synonym relation. However, they do not use hierarchy structure.
- *Informal is-a hierarchies* - They share informal is-a hierarchies below a concept. Such hierarchies are not strictly subclasses or “is-a” relations.
- *Formal is-a hierarchies* - It contains subclasses relation. These type of hierarchy is necessary to exploit inheritance.
- *Formal is-a hierarchies with instances* - Beyond is-a hierarchy features, it also contains instances of the domain.
- *Frames* - These ontologies have classes with properties, that can be inherited according to formal is-a taxonomy.
- *Ontologies with value restriction* - Similar to frames, but they can have value’s restriction for a property.
- *Ontologies with logical constraints* - Most expressive ontologies, where it is possible to express first-order logic constraints among terms using expressive ontology languages.

According to the usage of these components, Roussey et al. (2011) presented four types of ontology as follows. The classification started from the less formal languages to the more formal one.

1. Information Ontologies - Composed of diagrams and sketches. They are used to organize and clarify ideas of certain domain. Only used by humans. Focus on concepts, instances and their relationships. Usually described by visual languages, so it can be easily comprehensible.

Figure 2 – Ontology classification based on structured scope.



Source: (LASSILA; MCGUINESS, 2001)

2. Linguistic Ontologies - They can be glossaries, dictionaries, controlled vocabularies, taxonomies, folksonomies, thesauri, or lexical databases. Focus on terms and their relationships. They are normally used to present and define the domain's vocabulary, but also it is the definition of a terminology agreement that a community has over certain domain. This is necessary in order to avoid ambiguity.

3. Software Ontologies - Focus on data storage and data manipulation. Concepts are defined by a set of properties, and also by relations between them. These ontologies provide conceptual schemata and models.

4. Formal Ontologies - They use formal logic to obtain clear semantics for concept's definition, clear motivations for differences between concepts, and strict rules regarding concepts and relationships definitions. Knowledge Bases are formal systems that use logical definitions to get the meaning of a domain's vocabulary. The logical definition of a concept is composed by logical formulas. These logical formulas (also called axioms) are combinations of concepts and semantic relations. Beside retrieval and storage of data, they also have data reasoning by inference engine due formal definitions and rules.

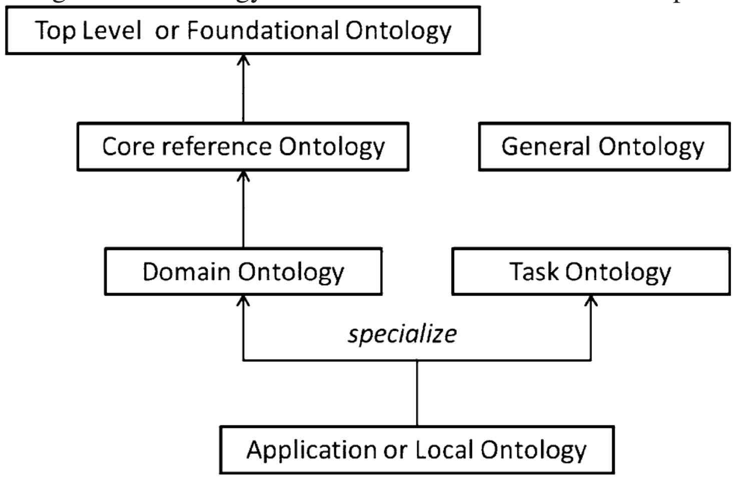
Another classification's criteria considers the scope that ontology is designed to cover. The narrowest scope is a Local Ontology, while the opposite are Foundational Ontologies, which can be interpreted as meta ontologies that describe concepts in highest level. Based on Roussey et al. (2011), the classification specifies six types of ontologies by their scope, as shown in Figure 3, and their definitions are the following:

1. Local/Application Ontologies - Specializations of Domain ontologies. Represents a specific model of domain for a single point of view.

2. Domain Ontologies - Represents a domain for a certain point of view. The view is defined by a group of people, so there is a consensus about its definition.

3. Task Ontologies - Represents the knowledge to achieve a generic task or activity. They are designed to solve problems associated to cross-domain tasks.

Figure 3 – Ontology classification based on content scope.



Source: (ROUSSEY et al., 2011)

4. Core Reference Ontologies - Standard ontology used by distinct users groups. Integrates different points of view for a specific domain, usually aggregates domain ontologies.
5. General Ontologies - These ontologies are not designed for specific domain. They have general knowledge regarding a huge area.
6. Top Level/Foundational Ontologies - Generic ontologies for very general concepts, even representation of world assumption. They usually define objects, relations, events, procedures, etc.

Based on the presented classifications of ontologies, the ontology used by Vulcont is a formal ontology. Vulcont's ontology uses formal logic to obtain semantic relations. These relations are expressed by strict rules or relations between concepts. They rely on domain knowledge, therefore the rules need to be defined or revised by a domain expert. Based on the scope, Vulcont's ontology is a domain ontology designed for recommendations.

2.3 Contexts History

According to Clarke and Driver (2004) a *trail* is a collection of locations with associated information in a recommended visitation order. The authors use the term *trail* to express the caption of daily user's activities, which will be used to adapt how mobile application with context awareness works. The mobile application can then consider the user history, as well as the previously visited locations. Applications with *trail awareness* use these sets of context information, such as the traffic in different highways, transportation facilities and weather conditions. Hence they are able to provide services as suggestion of alternates routes, or also manage logistics for transportation companies.

The work of Spence, Driver and Clarke (2005) extends their previous work purposing the analysis of history of recommended *trails*. *Trail histories* is how the authors call the information related to how the user used suggested *trails*. Including information regarding the original *trail* recommended by application, the reorganization of *trails* that happened, the current user's *trail*, and also additional information related to the context; all this information can indicate why the user did not follow the original recommended *trail*.

Posteriorly, the work Driver and Clarke (2008) defines *trail* as a set of activities to be performed by a user, similarly as a to-do list, with an organized sequence to be followed according context information and user's preference. The authors argue that the *trail* can be used as a generic model, attending requirements of time managing applications with context awareness. The *trails* can even overcome traditional techniques from this kind of application, that instead of using static lists, uses dynamically ordered lists according to user's context. From this, Driver and Clarke (2008) presents a framework able to provide functionalities to generate *trails* according to changes in current user's context.

An analysis of this data could support the applications' development of collaborative contexts. In the study, the authors define collaborative context as the data sharing between devices in ubiquitous computing environments. From the *trails history* analysis, it is possible to improve some features of this kind of application. Some examples are: To avoid the generation of unnecessary trails, the hypothesis verification, and also adaptation according to user's feedback.

In the doctors thesis of Smith (2008), storing user's activities along with context information generates *contexts history* visited by the user. These *contexts histories* can also be called *trails*, term used in work of Driver and Clarke (2008). The work Silva et al. (2010) proposes a *trail* management model which uses a service architecture available for client applications. In more recent studies, such as Li, Eickhoff and Vries (2012), *trails* are defined as a sequence of points of interest (POI) which the user daily have. The authors propose a prediction model, which identifies locations types which will compose the *trail* according to visited locations already stored in a *trail*.

All works mentioned in this section use a collection of events, ordered chronologically, and linked to identify an entity. The difference is the type of information stored in those sequences. The works store locations ((CLARKE; DRIVER, 2004), (LI; EICKHOFF; VRIES, 2012)), user activities ((CLARKE; DRIVER, 2004), (SMITH, 2008), (DRIVER; CLARKE, 2008)), or even generic *trails* ((SILVA et al., 2010)). The mentioned activities fit the description of Dey, Abowd and Salber (2001) for context's description. Since these *trails* are stored chronologically, they have information related to time (*Time category*). They are linked into identification entities (*Identity category*). The user's activity is linked to a *State/Situation category*, and concluding, the information related to user's location matches *Location category*. Therefore, the concept of *trail* can also be generalized as *contexts history*.

3 RELATED WORKS

In this chapter, we are going to analyze and compare the related works of Vulcont's study that were found. The main search criteria for related works was based on studies which involve recommender systems based on ontologies. Moreover, we filtered by recent works that used a hybrid approach in its recommendation system, since hybrid approaches can have a better performance with fewer of the drawbacks of a single approach (BURKE, 2002). The related works were evaluated based on ontology's usage and recommendation aspects. For the first topic, the scope reveals the ontology's wideness; how the ontology is created, if manually or in an automatic way; and if the ontology has user profiling. For the recommendation aspects, the discovery and recommendation algorithms are evaluated, and also if the recommender system uses contexts histories in its recommendations.

3.1 SigTur/E-Destination

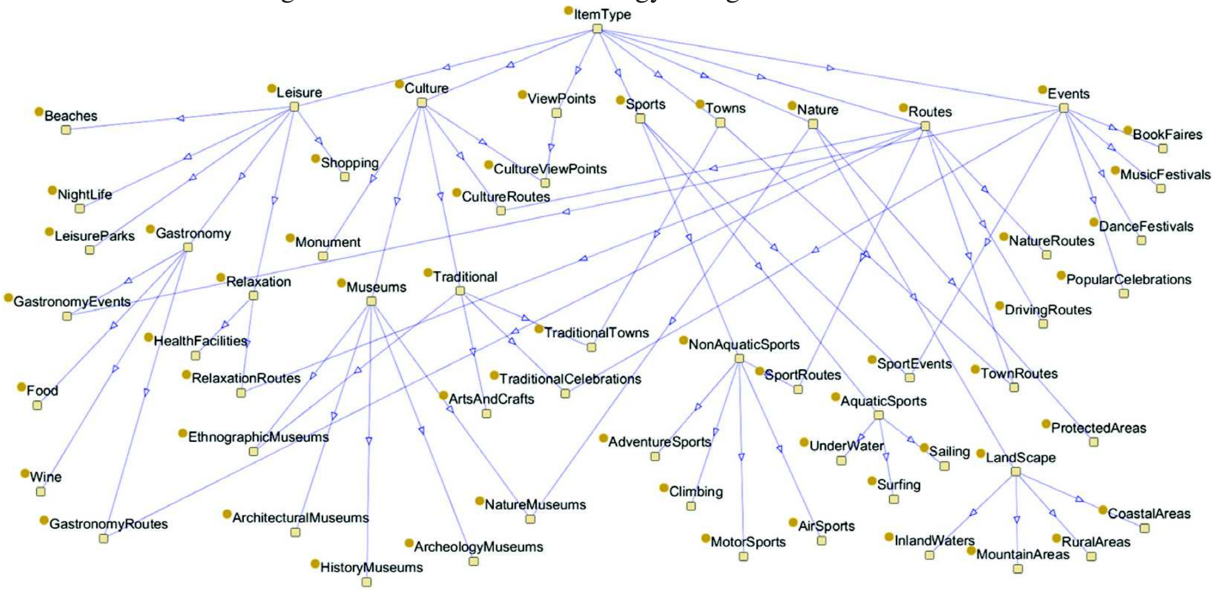
The work of Moreno et al. (2013) presents SigTur/E-Destination, which is a web recommender system for tourism in Spain, specifically for the region of Tarragona. The study mentions several ontologies created for tourism domain. The most recent was cDOTT ontology (BARTA et al., 2009), a core domain ontology which is based on Harmonise ontology and defines common ontology to support interoperability of agents in low-level operation. Harmonise (FODOR; WERTHNER, 2005) was one of first ontologies aimed for tourism, designed to cover four topics from the domain: attractions, events, food and drink, and accommodation.

The ontology created for the recommender system was designed according to territorial aspects of the "Camp de Tarragona and Terres de l'Ebre". The Figure 4 shows a partial view of ontology, with its classes and relationships. The ontology follows main concepts of thesaurus of the World Tourism Organization (UNWTO). The ontology's details level depends on set of activities in a particular area, where some concepts can be more explored than others. It contains up to 203 connected concepts, in 5 hierarchy levels. The proposed ontology is not a purely taxonomy, since it also contains multiple inheritance between concepts. It was developed using Protégé, represented in OWL Language, using Jena as Semantic Web Framework to manage the ontology and apply inference based on defined rules.

3.2 Ontology-Based Method for Personalized Recommendation

This work of Ge et al. (2012) proposes a recommender system that uses ontology in order to create domain and user's interest models. While the user ontology is built based on implicit and explicit data, the domain ontology is built from the merging of several local ontologies. This merging uses a matching method for concepts similarity based on semantic distance. The algorithm takes two concepts as input and computes a semantic similarity as output. These local

Figure 4 – Partial view of ontology in SigTur/E-Destination.



Source: (MORENO et al., 2013)

ontologies are built from three main types of sources: unstructured text documents, structured relational data sources, and semi-structured data sources in XML files.

For the structured case, the local ontologies are generated using RDB2OWL (ČERĀNS; BŪMANS, 2010) and XML2OWL (LACOSTE; SAWANT; ROY, 2011) mapping specification languages. Also, the mappings rules are defined according to schema and the query language. However, the study does not indicate what kind of rules can be extracted from data sources. Since the ontologies are created from different data sources, their domains can be considered as generic once it relies on data sources content.

For the recommendation, the system uses three ontologies: Reference ontology, that is used by the wrappers for filtering the results and discarding coordinates that does not relate to the terms in the query; Domain ontology, which is built from local ontologies; and User's interest ontology, generated by analyzing user's demographic characteristics (such as civil status, age or studies, identify the user in a social group or domain) and personal preferences information.

3.3 Generic Ontology for CBR-Based Recommender System

The work of Karim et al. (2014) approaches the use of a generic ontology for a recommendation based on Case-Based Reasoning (CBR) paradigm. As mentioned in the study, when developing a CBR system, the two main challenges are: (1) How to represent the cases, and (2) How to collect the knowledge that is source for reasoning process (BERGMANN; KOLODNER; PLAZA, 2005). They chose a generic ontology to represent the cases according to work Kolodner (1992). The modeled ontology was built to cover cases cross-applications. A *case* is composed by *problem*, *solution*, and *result*. To convert the generic ontology into application domain, the *components* are described according to domain knowledge. For that, they used two generic OWL properties in every component: GenericObjectProperty and GenericDataType-Property.

In order to get the most similar cases, two different types of similarity measure were used: (1) Local Similarity, that measures the similarity of each feature's value of case; (2) Global Similarity, that compares the new problem with the existing cases. For the CBR engine development, the jCOLIBRI Java framework (DÍAZ-AGUDO et al., 2007) was used. It provides a reusable and extensible architecture. The work did not perform any evaluation regarding the recommendation, since the focus was how to represent the case in a generic way, and also the integration with CBR system.

In a new work of Karim et al. (2015), the authors continued the research and presented a merge of CF with CBR recommendation for cross-domain applications. Differently from the first work, this new work presented a section to evaluate the provided recommendation. The evaluation used scenarios to analyze the generated recommendations according to three different user personal preferences. For future works, they proposed more complex scenarios, using recommendation of compounded items.

3.4 A Semantic Similarity Measure for Recommender Systems

The work of Lémdani et al. (2011) presented a semantic similarity that compares the relations between two individuals of an ontology, which can be used in recommender systems. This semantic similarity was designed to ontologies, regardless the domain of scope. They use a different approach and consider an item as being defined not only by its surrounding individuals (by means of properties), but also by neighborhood.

That similarity works iteratively as in the study Melnik, Garcia-Molina and Rahm (2002). However, in Lémdani et al. (2011), the similarity measure is adapted to generate recommendations. Beyond the recommendation itself, the similarity measure can also be used to enhance user profile management. This semantic similarity is also presented in work Lemdani et al. (2010). When comparing two instances, the relations are considered, and also other instances that are related to the pair which is being compared.

For the evaluation, 17 users explicitly evaluated the recommendation of items. The authors used two different domains, movies and papers. Initially every user evaluated 20 recommendations, with 1 to 5 stars. For each recommendation the user provided a feedback: like, dislike, or N/A. At the end, comparing with the classical recommendation (56,9%), the semantic similarity got a higher satisfaction (63,8%).

3.5 Multi-model Ontology-based Hybrid Recommender System in E-learning Domain

The work of Leyla et al. (2009) developed a hybrid recommender system, for e-learning domain, that merges two types of recommendation: content-based and rule-based. The first uses a domain ontology to represent learning materials. While the rule-based uses ontology to represent user's interest, so a cluster recommendation is added as an extra semantic recommendation.

The content-based recommends items (in this case Webpages) based on previous activities of users, in other words, their interests. While the rule-based recommends based on rules that precisely enable the recommended items to those with particular conditions (LEYLA et al., 2009). For the search engine, the authors used cosine-distance similarity that composes a query vector, and compares it with documents. For the cluster recommendation, a graph partitioning algorithm was developed from previous work Zhuhadar and Nasraoui (2008).

The work also approaches interest-based recommendations using rules. This recommendation allows the user to navigate through the semantic structure of his/her query, and select recommended terms. This approach is discussed in more details in previous work Zhuhadar, Nasraoui and Wyatt (2009).

For validation proposes, the authors used Top-n-Recall and Top-n-Precision to measure the effectiveness of re-ranking based on user's semantic profile. Ten different user profiles were selected for comparing the semantic search engine with the normal search. At conclusion, both personalization and semantic enrichment are potential elements to improve an E-learning Information Retrieval System.

3.6 Protus E-Learning

The study of Protus (PRogramming TUtoring System) (VESIN et al., 2013) proposes a new approach of RS in e-learning research field. The proposed RS is able to automatically adapt according to interests and knowledge level of every learner. The Protus RS is able to recognize different patterns and habits in learning style of users. The algorithm uses clustering techniques in order to group users with similar learning style. Then the habits and interests of users are analyzed by AprioriAll algorithm (PI-LIAN; TONG, 2005), which mines the sequence frequency. Finally, the system presents a recommendation based on rating of these frequent sequences. The Protus was tested in an experimental group, and the results were compared with a different group which learned same content in classical way.

The work explored the *learning styles*, a term which defines how the users are different when it comes to which kind of instructions or studies are more efficient for learning (PASHLER et al., 2009). Even though there are different learning styles models ((FELDER; SILVERMAN, 1988), (HONEY; MUMFORD, 1992), (KOLB, 1984), (PASK, 1976)), they all agree that each learner has personal tastes and preferences. Therefore, matching the learners preferences and how material and activities are offered to them will improve the learning adaptation (COFFIELD, 2004). The system was originally designed and implemented as a generic tutor system for distinct programming languages, the first implementation and testing was performed for a Java programming course (VESIN et al., 2008).

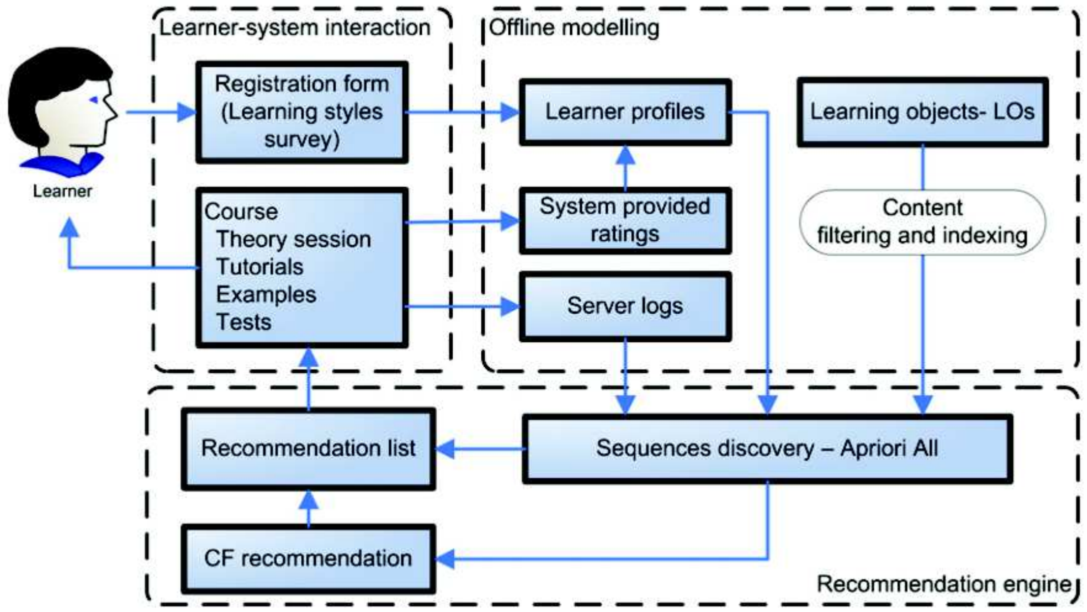
The Protus system is composed of five modules. The module's functions are described bellow:

- Domain Module - This module is in charge of storage all necessary learning material, tutorials, and tests. It basically defines how information content is structured.
- Learner Model - It is a collection formed by static and dynamic information about the learner. This information is used by Protus in order to predict learner's behavior, hence adapting the content according to learner's personal taste and needs.
- Application Module - It is responsible for performing the adaptation. It changes the content provided to learner based on learning style.
- Adaptation Module - It follows instructional directions determined by the *Application Module*. *Application* and *Adaptation modules* are separated because adding new clusters of content becomes easier. There are two ways to provide personalized content for learners: based on adaptive hypermedia, and the recommendation itself.
- Session Monitor - It is responsible for re-build the learner model along the user's session. It keeps track of learners' progress, correcting their errors, and redirect the session if necessary. After finishing the session, learners' preferences are properly stored on learner model. If the learner does not agree with the proposed learning style, it is possible to change that information at any time, directly editing the learning style.

Beyond the five modules already described, the system has a framework which determines and generates the recommendations. This framework is composed of following three modules:

- Learner-system interaction Module - It performs the first steps to built learner models. The information related to learner behavior is gathered here. This data is composed of sequential patterns, visited pages, test results, and earned grades. All content provided to the learner, such as theory sessions, tutorials, examples and tests; considers the input data for the processing.
- Offline Module - This module consumes learner models on-the-fly in order to generate learning style, which is based on learners' goals and profile. Once the learning style is defined for the learner, the content for recommendation is filtered taking in consideration an initial survey, status of course, learner's affiliation, gathered ratings, as well as ratings provided to the system.

Figure 5 – Recommendation component of Protus.



Source: (VESIN et al., 2013)

- Recommendation Engine - It generates the recommendation list based on filtered learning content and also discovered sequence patterns. For the output, a list of actions and courses are recommended. It will accordingly be used by learner-system interaction in a new session.

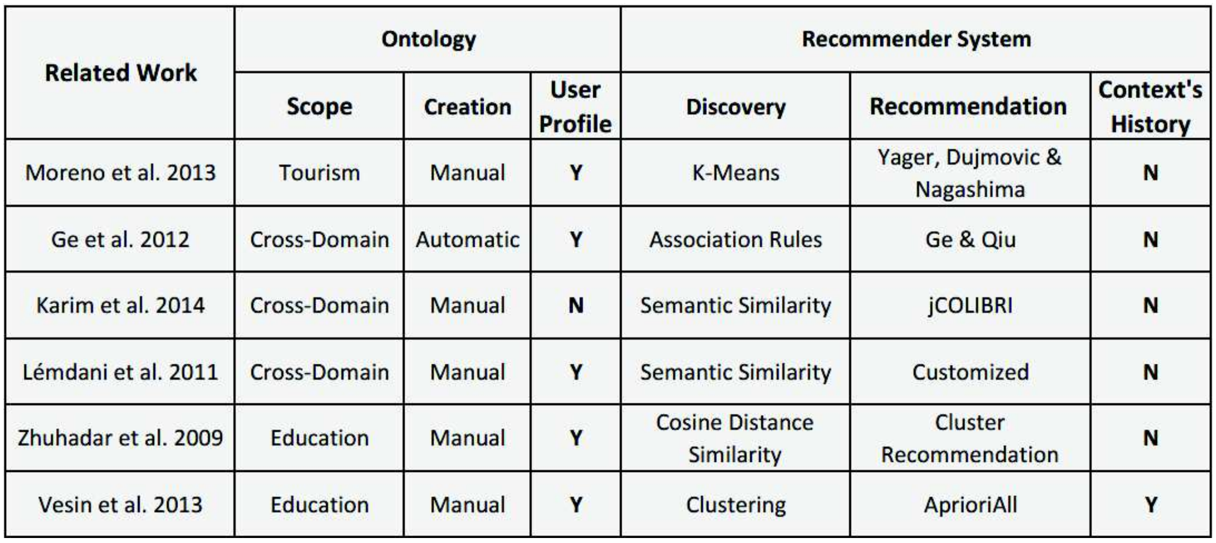
The recommendation in designed according to Figure 5. The new learner fills up a registration form for profile's creation. The profile contains static information (for example first name, last name, knowledge, preferences, etc.) and dynamic information (interests, dominant meaning words, behavior, among others.) The static information is editable. For new users, another survey is filled up in order to generate the first version of learning style. The learning style changes the way content is provided to the learner. In the recommendation system approach, the data is grouped by clustering techniques. So groups of learners are formed by their similar learning styles. The clusters are used as reference for choices in frequent sequences of learning activities. A rating is calculated by the system, creating a recommendation list.

3.7 Comparative Analysis of Related Works

Table 1 compares the related works based on ontology and recommendation aspects. For the ontology criteria, we compared the scope, creation method, and user profile usage. While for the recommendations the relevant aspects are the discovery approach, recommendation algorithm, and the usage of contexts histories.

For the domains scope of used ontology, Moreno et al. (2013) is focused on tourism, while Vesin et al. (2013) and Leyla et al. (2009) are focused on education. The remaining studies

Table 1 – Comparison of related works based on ontology and recommendation criteria.



Related Work	Ontology			Recommender System		
	Scope	Creation	User Profile	Discovery	Recommendation	Context's History
Moreno et al. 2013	Tourism	Manual	Y	K-Means	Yager, Dujmovic & Nagashima	N
Ge et al. 2012	Cross-Domain	Automatic	Y	Association Rules	Ge & Qiu	N
Karim et al. 2014	Cross-Domain	Manual	N	Semantic Similarity	jCOLIBRI	N
Lémdani et al. 2011	Cross-Domain	Manual	Y	Semantic Similarity	Customized	N
Zhuhadar et al. 2009	Education	Manual	Y	Cosine Distance Similarity	Cluster Recommendation	N
Vesin et al. 2013	Education	Manual	Y	Clustering	AprioriAll	Y

Source: Made by the author.

have a wider approach handling cross-domain ontologies. The creation of ontologies is usually performed manually by the related works. The exception is Ge et al. (2012) which creates the ontologies based on a source data. For the user profile usage, all related works have a profile handling except Karim et al. (2014).

Checking the recommendations aspects, for discovery criteria only Karim et al. (2014) and Lémdani et al. (2011) have a semantic similarity approach. While Moreno et al. (2013) uses K-means, Ge et al. (2012) has association rules. Leyla et al. (2009) has cosine distance similarity, and Vesin et al. (2013) utilizes a clustering method for its recommendation discovery. For the recommendation itself, every work has a different method to perform it. Moreno et al. (2013) uses studies Yager (1998) and Dujmović and Nagashima (2006); Ge et al. (2012) refers to its previous work Ge and Qiu (2008) for concept's semantic distance; Karim et al. (2014) uses jCOLIBRI (DÍAZ-AGUDO et al., 2007); Lémdani et al. (2011) uses a semantic similarity measure based on neighborhood; Leyla et al. (2009) has a clustering recommendation presented on Zhuhadar and Nasraoui (2008); and Vesin et al. (2013) uses Apriori All algorithm (PI-LIAN; TONG, 2005). For contexts history usage, the only study that has this approach is Vesin et al. (2013).

Based on the comparison, we identified that none of works has a cross-domain recommendation exploring the use of contexts histories. Therefore, in this work we propose Vulcont, whose main contribution is its approach for recommendations. Vulcont explores the benefits of ontology reasoning with the usage of contexts histories. It explores semantic relations (such as synonymous and subclasses) between instances in a contexts history ontology designed for recommendation. Vulcont calculates the similarity between the contexts history of an *Actor* with contexts histories from community, exploring semantic relations of *Contexts* to recommend *Ob-*

jects. This way, Vulcont is able to identify similar contexts histories, and finally recommend potential items to the *Actor*.

4 VULCONT MODEL

This chapter describes how Vulcont was modeled, and also provides details about its features. The first section details the contexts history ontology which was designed for Vulcont. The second section provides information regarding the architecture of Vulcont. Furthermore, we explain details about how Vulcont calculates the similarity of contexts histories and performs the recommendations.

4.1 Vulcont Contexts History Ontology

In last years, in computing research area, there were several advances in technologies for *Pervasive Computing* (SATYANARAYANAN, 2001). Among these acknowledged technologies, we can highlight the use of context information in different applications, also known as context-aware systems. The use of context information was a key factor for further technologies and studies in many different domains, such as education (WIEDMANN et al., 2016), logistics (OLIVEIRA et al., 2015), health (VIANNA; BARBOSA, 2014), and accessibility (CARDOSO et al., 2016).

The use of context information helped for the conceptualization of contexts history. The definition of contexts history, also known in other studies as *trails*, is provided in works of Driver and Clarke (2008) and Levene and Peterson (2002) as a historical collection of contexts' data within a certain period of time. A contexts history stores several contexts, in a chronological way. The location, performed action, and any other information considered relevant, are stored compounding a *contexts history element*, a kind of context "snapshot". A chronological collection of these elements compose then a *contexts history*. Using contexts history, we can explore context's patterns, similar contexts histories and even predict future contexts (ROSA; BARBOSA; RIBEIRO, 2016).

Besides context-aware systems, another relevant topic discussed in computing researches is regarding how the knowledge can be represented and stored, in order to be comprehensible by humans, but also in a computational way. In the *Formal Language* research field, the ontologies came up with shared conceptualizations of a specific domain, that are composed by different objects, such as entities, relations, functions, instances, among others.

Here we designed an ontology of contexts histories for recommendation. Different kinds of contexts can be represented by contexts history ontology, independently from the contained domain information. Hence, the contexts histories can be used for recognition of context's patterns, context's prediction and also items' recommendation. We searched for works involving context's representation and different perspectives to figure out how to design this ontology as widely as possible. The ontology should be able to represent many scenarios as possible, so there is no limitation about representing all required context's knowledge for certain domain.

4.1.1 Background for Contexts History Ontology

Before searching for works related to contexts history or context ontology modeling, we explored the widest definition of context information. Since the concept of context is simpler than contexts history (basically a chronological collection of contexts), we focused on context's definition, so the modeled ontology could cover most scenarios as possible where contexts are used in different domain's applications.

There are several works that define context information. The work of Gwizdka (2000) elaborates a compilation of different definitions for the context concept. For example, the definition of context-aware applications by Schilit, Adams and Want (1994), as a "software that examines and reacts to an individual's changing context". Furthermore, the same work lists three different kinds of context information: location, nearby people, and accessible computing devices.

There are also differences between internal and external context information. This distinction is defined in study of Schmidt, Beigl and Gellersen (1999), where the proposed model includes human factors, as well as physical environment. Generally, internal context information is used to describe user state, goals or tasks, user's work context, busyness, personal events, user's cognitive, emotional, and physical states. For external context, the environmental information is used. Such as location, proximity of other objects (people or devices), temperature, and time.

In work of Debes, Lewandowska and Seitz (2005), the definitions of context are listed based on several studies. Some of those works are Brown, Bovey and Chen (1997), Ryan, Pascoe and Morse (1998), and Schmidt et al. (1999). Nevertheless, the definition of Dey (2001) seems as the most appropriate and widely used by researchers community. Following his definition, context involves physical and conceptual status, this information is from interest to a specific entity. Dey (2001) states the following definition: "Context is any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects".

Following the conceptual idea of entities, Dey, Abowd and Salber (2001) defined three of them as *Places* (geographical spaces), *People* (individuals or groups), and *Things* (physical objects or software components). To describe these three entities, four categories were defined: *Identity*, *Location*, *Status*, and *Time*. In work of Dey et al. (1997), the necessary context's information is classified in three categories: *Physical context*, *Action-based context*, *Emotional context*.

Some studies such as Dey (1998), Schmidt et al. (1999), and Bornträger (2003) use user's status, emotional information, events, position, orientation, time, date, people and objects around the user, history information, user's preferences, communication's status, among others. It is complicated to list all necessary context information, since different applications have different requirements. The proposal is specifically to define the most generic context as possible. With that, the ontology can be modeled and implemented, representing the originally required contexts history knowledge for recommendation.

In order to cover a context as widest as possible, we reviewed the existing formal representations of contexts. One of the most known context formalizations is Heckmann (2006) which presents the SituationML. The SituationML is not a formal ontology, but a XML-based technological realization of the abstract model called SituationReports ans SituationalStatements. The “ML” at the end of the term stands for Markup Language. The SituationalStatements represents partial description of situations, such as user model entries, context information, or even low-level sensor data. The SituationReports is composed by several SituationalStatements. The SituationML organizes the information in a predefined hierarchy of meta-levels wrapped around the mainpart. The information in the meta-levels can be interpreted as a collection of slots or attributes arranged in fives layers, which are: *mainpart*, *situation*, *explanation*, *privacy*, and *administration*; as illustrated by Figure 6. This Figure also presents how data is stored in every layer, and how the SituationalStatements are organized compounding a SituationReport for an airport scenario.

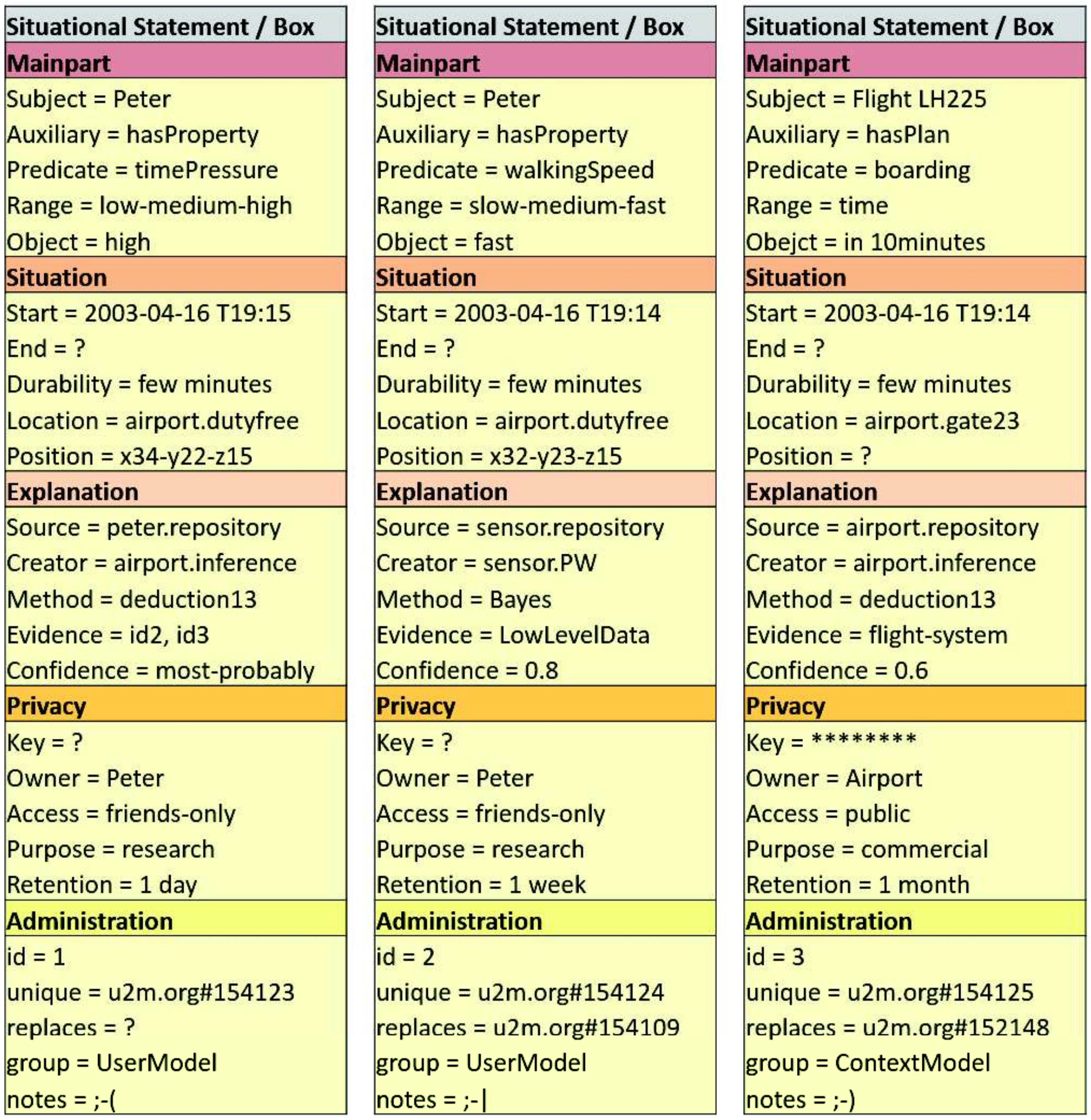
We also checked different ontology’s studies, such as GUMO Ontology (HECKMANN et al., 2005) and UbisWorld Ontology (STAHL; HECKMANN, 2004). GUMO Ontology has a similar structure to SituationML. Both use the schema “Subject - User Model Dimensions - Object”, where the User Model Dimensions has three parts: Auxiliary, Predicate, and Range. Different from the two contexts history representations presented, the UbisWorld can be used to represent parts of real world, for example an office, a shopping mall, an airport. It represents people, objects, locations, events, time, all of that with properties and features. UbisWorld is composed by partial ontologies rather than one for all aspects. The partial ontologies are the following: Physical, Spatial, Temporal, Activity, Situation, and Inference.

4.1.2 Contexts History Ontology Development

In order to have an ontology which represents contexts histories for recommendation in a most reliable way, there is certain knowledge that the ontology should be able to represent. In other words, there are queries (competency questions) which the represented knowledge in the ontology should be able to answer. These knowledge can be found in two different bases: terminological (T Box) composed by concepts definition and axioms; and assertional (A Box) with concepts and roles assertions. Specifying the competency questions, we defined them as the following list:

- What is a contexts history?
- What composes a contexts history?
- What is a context?

Figure 6 – SituationReport representing an airport scenario.



Situational Statement / Box	Situational Statement / Box	Situational Statement / Box
Mainpart	**Mainpart**	**Mainpart**
Subject = Peter	Subject = Peter	Subject = Flight LH225
Auxiliary = hasProperty	Auxiliary = hasProperty	Auxiliary = hasPlan
Predicate = timePressure	Predicate = walkingSpeed	Predicate = boarding
Range = low-medium-high	Range = slow-medium-fast	Range = time
Object = high	Object = fast	Obejct = in 10minutes
Situation	**Situation**	**Situation**
Start = 2003-04-16 T19:15	Start = 2003-04-16 T19:14	Start = 2003-04-16 T19:14
End = ?	End = ?	End = ?
Durability = few minutes	Durability = few minutes	Durability = few minutes
Location = airport.dutyfree	Location = airport.dutyfree	Location = airport.gate23
Position = x34-y22-z15	Position = x32-y23-z15	Position = ?
Explanation	**Explanation**	**Explanation**
Source = peter.repository	Source = sensor.repository	Source = airport.repository
Creator = airport.inference	Creator = sensor.PW	Creator = airport.inference
Method = deduction13	Method = Bayes	Method = deduction13
Evidence = id2, id3	Evidence = LowLevelData	Evidence = flight-system
Confidence = most-probably	Confidence = 0.8	Confidence = 0.6
Privacy	**Privacy**	**Privacy**
Key = ?	Key = ?	Key = ********
Owner = Peter	Owner = Peter	Owner = Airport
Access = friends-only	Access = friends-only	Access = public
Purpose = research	Purpose = research	Purpose = commercial
Retention = 1 day	Retention = 1 week	Retention = 1 month
Administration	**Administration**	**Administration**
id = 1	id = 2	id = 3
unique = u2m.org#154123	unique = u2m.org#154124	unique = u2m.org#154125
replaces = ?	replaces = u2m.org#154109	replaces = u2m.org#152148
group = UserModel	group = UserModel	group = ContextModel
notes = ;-(	notes = ;-|	notes = ;-)

Source: (HECKMANN, 2006)

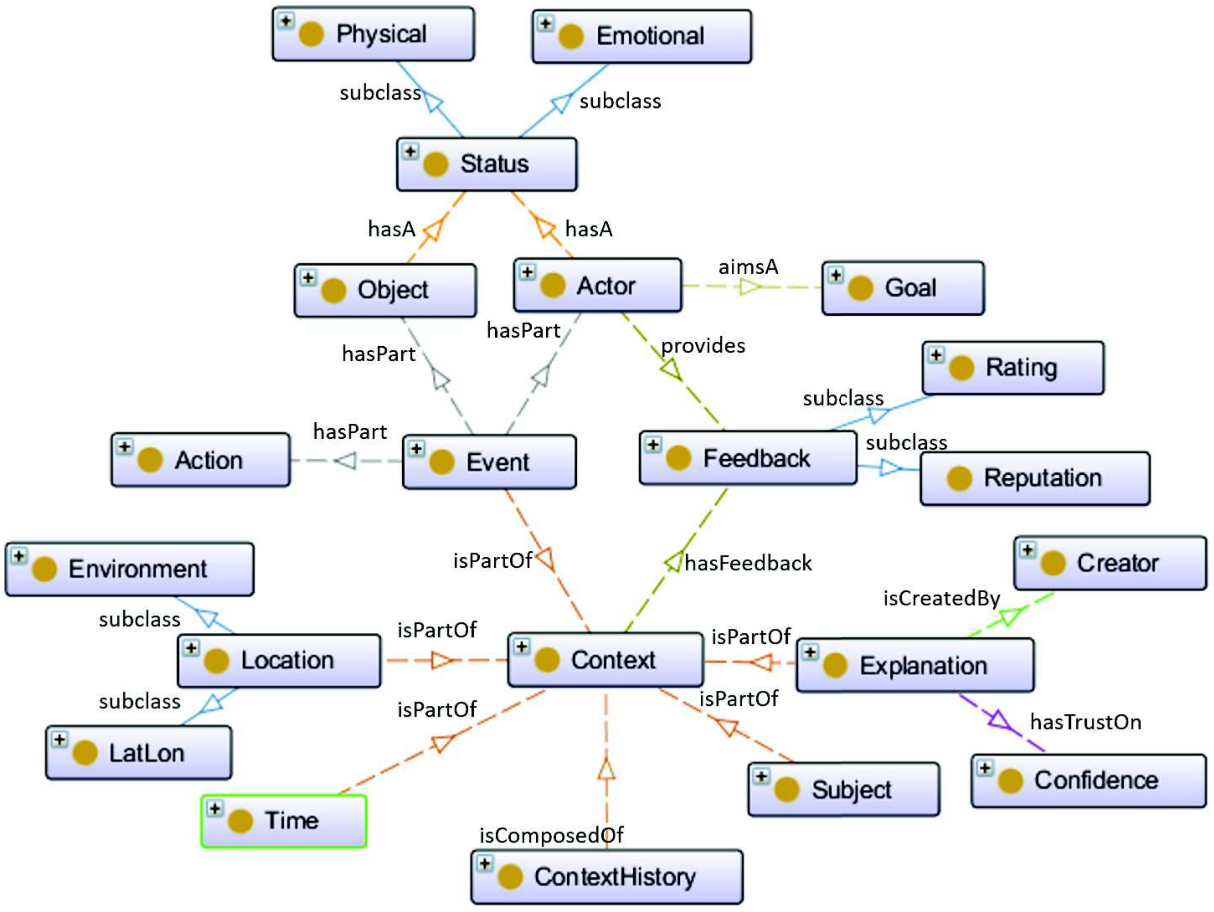
- What composes a context?
- What are the contexts associated to an actor?
- What are the contexts associated to a object?
- What action did the actor perform?
- What is the actor's goal by performing the action?
- Does the actor have a feedback after the action?
- Did the actor get closer to the goal by performing the action?
- What is the actor's status in a context?
- What is the object's status in a context?
- Who captured the context?
- Where the context where captured?
- When the context was captured?

With the context's definition as any relevant information which describes a situation of entities, and the definition of contexts history as a chronological collection of context's snapshots, we propose an ontology of *contexts histories* for recommendation. With that, the contexts histories and contexts can be modeled and implemented to become in fact an ontology which represents the required knowledge. The ontology was designed using Protégé (KNUBLAUCH et al., 2004), which is an open-source ontology editor.

Figure 7 presents a graphical representation of the developed ontology, where all classes and object's properties are shown. A brief description of every class and its properties are presented as follows:

- Contexts History - It is the main concept that the ontology represents. It has property *isComposedOf* linked with *Contexts*, since the contexts histories are a chronological collection of contexts.
- Context - The key concept of the ontology has a property *isPartOf*, so other dimensions are part of a *Context*. A *Context* has five optional dimensions: Time, Event, Location, Subject, Explanation; and also a *Feedback*.
- Time - Responsible for knowledge representation about when the context was captured.
- Subject - It represents the knowledge regarding the domain which is related to *Context*.

Figure 7 – Contexts history ontology for recommendation.



Source: Made by the author.

- Location - It represents knowledge regarding the local where context was captured. It has two subclasses: *LatLon* and *Environment*. The first one is focused on physical location, such as GPS coordinates, while the *Environment* represents a location through common labels, for example Home, Work, or any other semantic representation.

- Explanation - It is responsible for knowledge representation related to context's creator, linked through property *isCreatedBy*; and the level of confidence of that context's snapshot, using property *hasTrustOn*.

- Event - It represents the event performed in the *Context*. It is composed of three parts by property *hasPart*, where the parts are *Actor*, *Action*, and *Object*.

- Action - It represents what the *Actor* is performing. Ideally, the *Action* is the way to achieve the goals of the *Actor*.

- Object - It is in charge of representing an object or artifact which the *Actor* interacts with (within an *Action*). It might have a *Status* which complements the description of the *Object*.

- Actor - The *Actor* performs the *Action* on the *Object*. It is not necessarily a person, it can be a software agent or a representative entity. As well as the *Object*, it might have a *Status* connected by property *hasA*. The *Actor* may have also a *Goal*, which the *Action* will be oriented to achieve. The property between *Actor* and *Goal* is *aimsA*. The *Actor* can also provide a *Feedback*, with property *provides*, resulted from the *Action* performed on the *Context*. Therefore, the user profile management is composed by classes: *Actor*, *Goal*, *Status*, and *Feedback*.

- Status - It represents the states of *Actor* and *Object*, which can be divided into two distinct categories: *Physical* and *Emotional*.

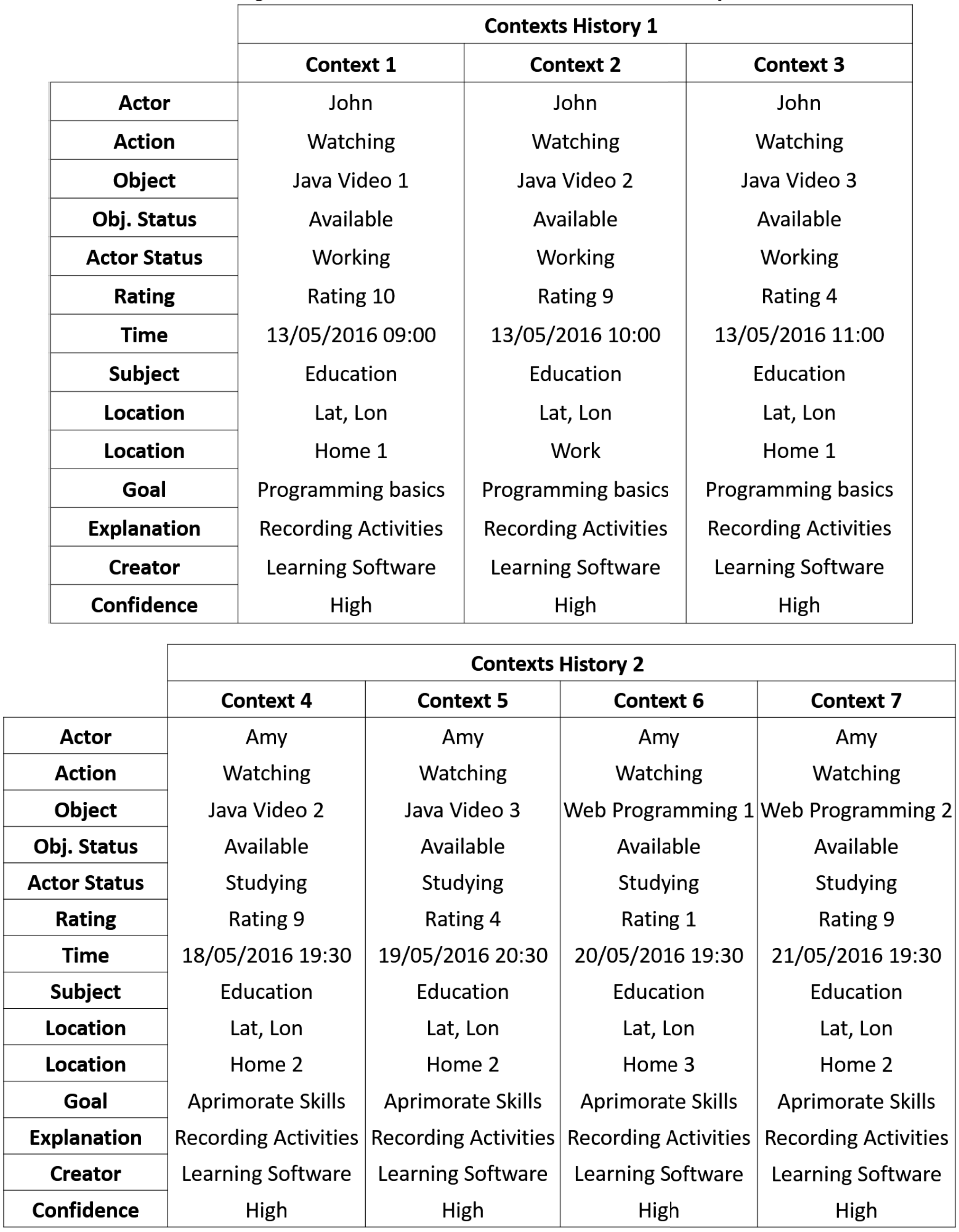
- Goal - This concept represents the goals, tasks, or objectives pursued by the *Actor* while performing a specific *Action*.

- Feedback - This class represents the knowledge provided by the *Actor* referring to a *Context*. The *Feedback* is related to the *Context* itself by property *hasFeedback*, where the *Action* performed will be evaluated by *Actor* according to the experience. The *Feedback* has two possible subclasses for measurement: *Rating* or *Reputation*.

- Creator - It represents the entity responsible for the capture of context's snapshots. It provides knowledge related to the context's source.

- Confidence - It is responsible for the confidence of context's source. Representing the reliability of context's information based on *Explanation* and *Creator*.

Figure 8 – Educational scenario with contexts history.



	Contexts History 1		
	Context 1	**Context 2**	**Context 3**
Actor	John	John	John
Action	Watching	Watching	Watching
Object	Java Video 1	Java Video 2	Java Video 3
Obj. Status	Available	Available	Available
Actor Status	Working	Working	Working
Rating	Rating 10	Rating 9	Rating 4
Time	13/05/2016 09:00	13/05/2016 10:00	13/05/2016 11:00
Subject	Education	Education	Education
Location	Lat, Lon	Lat, Lon	Lat, Lon
Location	Home 1	Work	Home 1
Goal	Programming basics	Programming basics	Programming basics
Explanation	Recording Activities	Recording Activities	Recording Activities
Creator	Learning Software	Learning Software	Learning Software
Confidence	High	High	High

	Contexts History 2			
	Context 4	**Context 5**	**Context 6**	**Context 7**
Actor	Amy	Amy	Amy	Amy
Action	Watching	Watching	Watching	Watching
Object	Java Video 2	Java Video 3	Web Programming 1	Web Programming 2
Obj. Status	Available	Available	Available	Available
Actor Status	Studying	Studying	Studying	Studying
Rating	Rating 9	Rating 4	Rating 1	Rating 9
Time	18/05/2016 19:30	19/05/2016 20:30	20/05/2016 19:30	21/05/2016 19:30
Subject	Education	Education	Education	Education
Location	Lat, Lon	Lat, Lon	Lat, Lon	Lat, Lon
Location	Home 2	Home 2	Home 3	Home 2
Goal	Aprimorate Skills	Aprimorate Skills	Aprimorate Skills	Aprimorate Skills
Explanation	Recording Activities	Recording Activities	Recording Activities	Recording Activities
Creator	Learning Software	Learning Software	Learning Software	Learning Software
Confidence	High	High	High	High

Source: Made by the author.

4.1.3 Ontology's Consistency Validation

For the consistency's validation of the modeled and developed contexts history ontology, we used a *shell* called PROWLOG (GLUZ, 2015) that, using PROLOG interpreter, is able to manage, change and perform queries on OWL ontologies. We created a small educational scenario, inserting knowledge through instances into the ontology. In this scenario, two contexts histories are representing interactions between students and learning objects. Figure 8 presents all knowledge inserted into the ontology in order to perform the first tests related to ontology's consistency. The students users are consulting learning objects in a sequence, and providing feedback about the content for every object. These actions, evaluations, and complementary information are all inserted into the ontology.

Based on the competency questions, we defined queries to retrieve the knowledge in the contexts history ontology. We modeled queries in order to answer every question. Based on the consistency validation, we can highlight some of the queries. For instance, the competency question: "What are the contexts associated to an actor?", we defined the following query that answers it. The query's result is precisely the desired knowledge:

```
select X where {X:'Context', Y:'Event', Z:'Actor', Y:isPartOf(X),
Y:hasPart(Z), Z:'ActorID'('01'^^xsd_integer)}.
```

Once we have all *Contexts* associated to an *Actor*, we can also request the *Status* of *Actor* (*Physical* or *Emotional*) during the *Context*. This is the answer for the competency question "What is the actor's status during a context?". So for example we select *Actor* "John", and among all his involved *Context*, the *Context* "2" in specific, we can obtain his *Status* as "Working" with following statement:

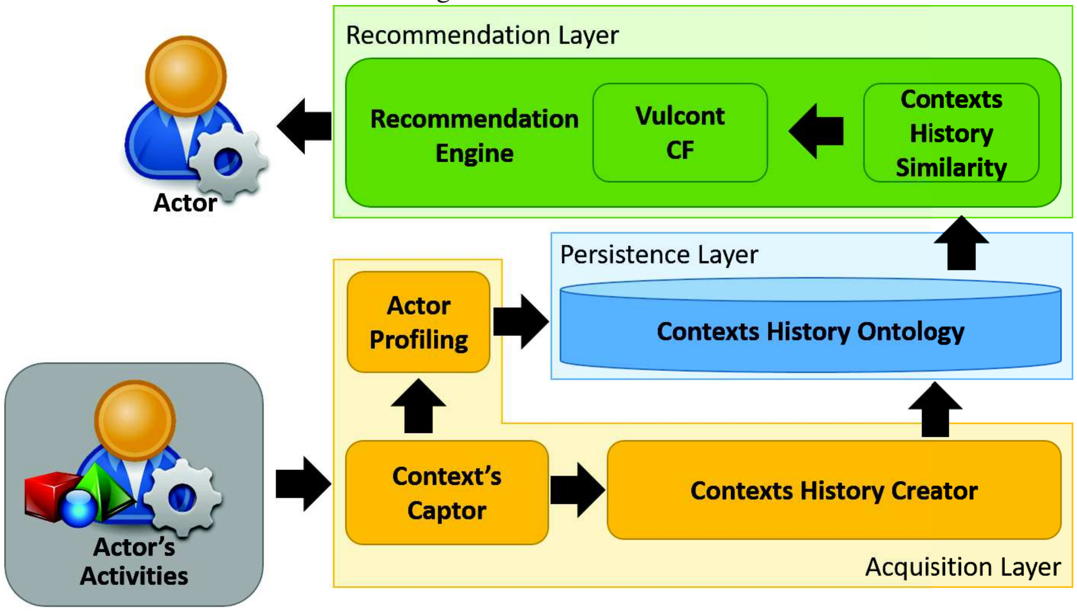
```
select X where {X:'Status', Y:'Object', Z:'Event', W:'Context', Z:isPartOf(W),
Z:hasPart(Y), Y:hasA(X), W:'ContextID'('01'^^xsd_integer)}.
```

After performing this testing to validate the ontology, we can guarantee that queries are able to return the necessary knowledge from ontology, and there is no internal inconsistency with classes and object properties. This validation scenario was created specifically for education domain. The ontology resulted in consistent returns from the 15 queries, confirming that the contexts history ontology is able to answer the competency questions.

4.2 Vulcont Architecture

In this section we describe the Vulcont's architecture, which is presented in Figure 9. The Vulcont performs recommendations based on similarity of contexts histories. The comparison includes semantic relations of instances from contexts history ontology. It recommends *Objects*

Figure 9 – Vulcont's architecture.



Source: Made by the author.

to an *Actor*. The *Object* is any entity that the *Actor* interacts with. The *Actor* is the entity which performs an *Action* in order to achieve a *Goal*, hence interacting with *Objects*. The *Actor* can be a user, a software agent, or any other entity that interacts with *Objects*. These interactions are then stored in a repository, compounding the contexts history ontology. The contexts histories are then used as source to calculate the similarity between two contexts histories. Finally, the *Objects*, which are unknown by the *Actor*, can be recommended once most similar contexts histories from community are identified.

The Vulcont analyzes the past contexts where the *Actor* was involved, comparing the *Contexts History* associated to the *Actor* with contexts histories from the repository. The similarity between two contexts histories is based on the similarity of the compounding *Contexts*. The similarity between two *Contexts* is calculated through the data properties, individually comparing each one. The data properties similarity is aggregated, resulting in a total similarity of *Contexts*. Then the mean of the similarity of *Contexts* defines the similarity between the two contexts histories.

The *Actor* has associated *Activities*, which are non-structured contexts histories. These *Activities* are the base for the profile of the *Actor*, where we can identify relevant information from *Actor*, such identities, goals, and status. This information is extracted by *Context's Captor* or explicitly provided by the *Actor*. The *Actor Profiling* is responsible for *Actor's* data manipulation, formalization, and insertion into repository *Contexts History Ontology*.

The *Activities* are collected by *Context's Captor*, which is the module in charge of the actions. It merges the *Events* of *Actor* with different context information, such as *Time*, *Location*, *Subject*, and *Explanation*. Once this information is collected, the *Activities* are then structured by the *Contexts History Creator*. The *Contexts History Creator* has as source the non-structured

Activities. The *Contexts History Creator* then harmonizes the source data, inserting it into *Contexts History Ontology*.

With the contexts history ontology, Vulcont consults the repository to identify similar contexts histories comparing to contexts history of a specific *Actor*. The *Contexts History Similarity* considers semantic relations of data properties from the instances of *Contexts*, as well as the similarity of numeric data properties, following the instance's similarity proposed in the study Andrejko and Bielikova (2009). Based on similar contexts histories, Vulcont can leverage *Objects* to recommend according to the received ratings and the position in contexts history, using its own collaborative filtering approach, the *Vulcont CF*. This selection of *Objects* is also performed inside the *Recommendation Engine*.

4.2.1 Vulcont Layers and Components

Vulcont has three main layers, each one with components within it. The following list will detail every layer's purpose and its respective components, providing information related to their scope, functionality and limitations.

- Acquisition Layer - This layer is responsible for data acquisition. The data is related to context's snapshots, and therefore, related to the contexts histories. The data is provided by the application which wants to use Vulcont's recommendations. This application can be from different domains once the contexts history ontology was designed to cover different scenarios and a wide range of context's information.

  Besides the context's information, another crucial data is related to the final target of recommendations: the *Actor*. Creating a profile for the *Actor* is important because we can formalize *Actor's* personal tastes and preferences. Vulcont should be able to personalize the recommendation, identifying goals, status, and positive previous experiences.

  The *Acquisition layer* is composed of three components:

  – Context's Captor - This component is in charge of collecting context's snapshots from source application. Along the contexts, the Actor's information is also collected, which will be used later by component *Actor Profiling* in order to create Actor's profile. The context's capture is often performed directly by the application, which logs the Actor's activities and the surroundings.

  – Actor Profiling - It manages the Actors' profiles, before inserting into respective classes in contexts history ontology. The Actor's information can be collected by the *Context's Captor*, but also explicitly provided by the Actor. As well as context's information, the application can already have the profiles, and therefore would be only necessary to propagate this data into *Contexts History Ontology*. In the ontology, the classes *Actor*, *Goal*, *Status*, and *Feedback* compound the Actor's profile.

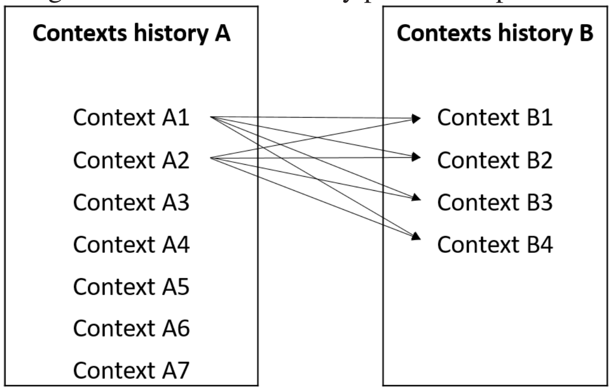
  - Contexts History Creator - This component is responsible for consolidate the context's snapshots, grouping them for the creation of contexts histories. The criteria of how context's snapshots are grouped to form a contexts history should be provided by application, since this is directly related to the domain's information. However, the collection should follow contexts history definition where the snapshots are chronologically organized.

- Persistence Layer - This layer is responsible for the data's persistence used by Vulcont. The data is stored into Vulcont's ontology for recommendation, which formalizes concept's representation. The *Contexts History Ontology* is responsible for the representation of contexts histories. This ontology stores all Actor's activities and the context information in that specific moment. The organizational criteria of these snapshots is also defined in the ontology. This ontology is then consulted by *Recommendation Engine* for recommendation's elaboration. Furthermore, the ontology also stores the knowledge related to Actor's profiles. The application's domain defines what information is relevant and should be used in a customized recommendation to an Actor. Additionally, all the semantic relations and ontology properties are considered when calculating contexts history similarity, according to the rules defined by the domain of *Objects*. The application must explicitly define the semantic rules while creating instances that will be used for similarity calculation.

- Recommendation Layer - This is the core layer of Vulcont, which is composed by three components. The layer is in charge of recommendations based on contexts histories. The input is provided by *Actors*, which is their contexts histories. The output is the recommendation of *Objects*, based on comparison between the contexts histories from community and the contexts history of the target *Actor*.

  - Recommendation Engine - The *Recommendation Engine* component orchestrates the recommendation. A certain *Actor* is selected as target of recommendation. The contexts history of the Actor is compared with contexts histories from community. The similarity is calculated by *Contexts History Similarity* component. Once the similar contexts histories are identified, the *Vulcont CF* component will leverage the relevant *Objects* to be recommended, according to data properties from ontology, as well as the feedback received in the contexts history.
  - Contexts History Similarity - This component compares and evaluates the similarity between two contexts histories. One from the *Actor* which will be the target of recommendations, and the other is a different contexts history from the ontology repository. All the *Contexts* that compose the contexts histories are compared with each other, as illustrated in Figure 10. In the illustrated example, since the "Contexts history A" is composed by 7 contexts and "Contexts history B" by 4, this will

Figure 10 – Contexts history partial comparison.



Source: Made by the author.

result in 28 contexts' comparisons (7 times 4). The final value for contexts history similarity is the mean of all contexts' comparisons. This approach estimates the typical similarity of contexts between the two compared contexts histories. The contexts' similarity is calculated comparing two context's instances. Every context's instance has different ontology properties, hence the comparison of instances depends on Object Properties and Data Properties. Using Object Properties such as "rdfs:subClassOf" and "owl:sameAs", it is possible to express semantic relations between the instances. This requires treating classes as instances, which can only be expressed in OWL Full.

When comparing literals of *number* or *date* type, they are mainly dependent on the application domain. A reasonable approach is presented by Equation 4.1, which is from the study of Andrejko and Bielikova (2009):

$$sim_{number}(x,y) = \begin{cases} 0.0 & \text{if } |x-y| \geq N \\ 1.0 - \frac{|x-y|}{N} & \text{otherwise} \end{cases} \tag{4.1}$$

where *x*, *y* denote a number, and *N* expresses the precision for this comparison. *N* denotes the unit which is still meaningful regarding what the number express. For example, a student's grade with range from 1 to 100, the respective *N* is 100. In another example for a different domain, in a job offer, a meaningful period is defined as one year based on a data property "Start date". In this case, *N* is 365 because the absolute value of distance between the dates is expressed by a unit of days.

When comparing literals of *string* type, a simple comparison using Java methods "equals" or "equalsIgnoreCase" does not provide a satisfactory result, also the result is a boolean instead of a similarity measure from 0 to 1. Therefore, we use Levenshtein similarity metric (PIETERSE; BLACK, 2015) to compare strings. This algorithm calculates the edit distance between two strings. For example, when comparing terms "book" and "cook", there is only one different letter, since the string's length is 4, the similarity is 0.75. In another example, the similarity between "smart" and "intelligent" is 0.09. This algorithm clearly ignores semantic relations. Therefore, Levenshtein similarity compares only literals. The semantic analysis of Vulcont is performed by the ontology reasoner according to relations specified directly in the ontology, according to the domain of data.

Comparing only literals is definitely not enough. The Object Properties should also be considered to compute the instances' similarity. Since the comparison is performed between *Contexts*, the instances that compound the *Contexts* are compared with each other, considering also the relations they might have. The *Context* can be compounded by five dimensions (*Location*, *Event*, *Time*, *Subject*, and *Explanation*), but it is not possible to have more than one instance for same dimension type, for example two *Locations*, or two *Events*. Therefore, when comparing two *Contexts*, Vulcont can compare their *Locations* and other dimensions always in pairs. Whether only one of compared *Contexts* has *Location*, then the similarity is 0. Also, the similarity between two instances follows:

∗ sim(x, y) $\in$ [0..1],

∗ sim(x, y) = 1 $\rightarrow$ x = y: objects are equal,

∗ sim(x, x) = 1: similarity is reflexive,

∗ sim(x, y) = sim(y, x): similarity is symmetric.

In the approach used by Vulcont, the similarity between two contexts' instances is the aggregation of similarities connected by properties. The aggregation is divided by the number of properties, to obtain a mean value of similarities. When comparing two instances, for example *InstA* and *InstB*, they have respective properties *A* = $\{property_1,..., property_n\}$ and *B* = $\{property_1,..., property_m\}$. Therefore, the total similarity will be measured by Equation 4.2. This equation was presented in the work of Andrejko and Bielikova (2009) which provides a solution for instances' comparison:

$$sim(InstA, InstB) = \frac{\sum_{i=1}^{|A \cup B|} PropertySim_i(elementA, elementB)}{|A \cup B|} \tag{4.2}$$

the elements *elementA* and *elementB* are instances connected by i^{th} property which is mutual between the two *Contexts*. If the property is not mutual, the similarity is 0. The similarity of all properties is aggregated and divided by the number of properties. Hence, all the properties have a same relevance.

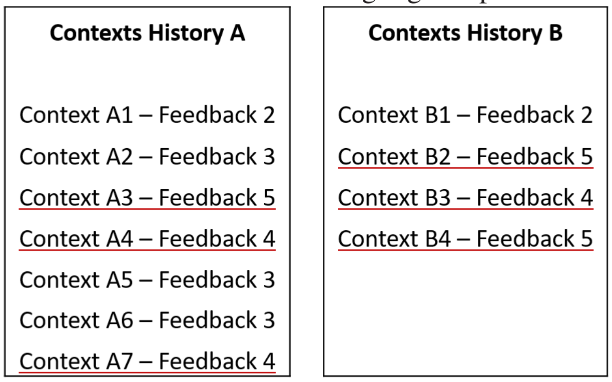
– Vulcont CF - Once the calculation of contexts history similarity was performed, this component identifies the possible items for recommendation. Vulcont CF analyzes individually the contexts histories with highest similarity based on the contexts history from the Actor which is target of recommendation. The number of selected contexts histories for analysis is based on the number of required items for recommendation. For instance, if five items are required, and the first contexts history (with the highest similarity) provides three *Objects*, the next most similar contexts history would be checked, until five *Objects* are selected. For the same scenario, if instead of five the requested number of items is three, only the first contexts history would be checked.

Vulcont CF checks the positive feedback in both *Contexts Histories*. The *Objects* with a good feedback are selected. Since these *Objects* were all compared by *Contexts History Similarity* component, it is already known the similarity between them. For example, Figure 11 shows two contexts histories. "Contexts history A" was generated by an *Actor* which is the target of recommendation, while "Contexts history B" is the most similar one. In the example, each *Context* has a *Feedback*, defined by the application as a 5 stars review system. In this case, the application considers that a positive feedback is 4 stars or more, as highlighted in Figure 11. Therefore, we have two lists of *Contexts* with positive feedback: *A3*, *A4*, and *A7* for "Contexts history A"; while *B2*, *B3*, and *B4* for "Contexts history B". These lists follow contexts history structure, organized by time where last contexts are the recent ones.

Once the sequences of *Contexts* with positive feedback are defined, Vulcont CF generates possible combinations of *Contexts* to compare with *Contexts Histories* from ontology repository. Following the example shown by Figure 11, there are three *Objects* to be recommended (from *Contexts* B2, B3 and B4). From the list of highlighted *Contexts* from "Contexts history A", *Context* A7 is the latest one. Therefore, Vulcont CF checks the occurrence of sequences in ontology repository: A7 then B2, A7 then B3, and A7 then B4. If the sequence exists, a longer sequence is checked, for example: A4 then A7 then B2, A4 then A7 then B3, or A4 then A7 then B4. A longer sequence is preferable, followed by number of occurrences, and then the mean of similarity over all contexts. This is the order of relevance for recommendation. This way we guarantee that even if there is no sequence at all, the recommendation will still be performed based on similarity. The sequence should follow a specific order of *Objects*. However, other *Objects* might exist among the sequence, so it does not require a straight sequence.

Vulcont CF handles exception scenarios as well. If any *Object* from "Contexts history B" is already known by "Contexts history A", that *Context* is discarded for recommendation. If there is no positive feedback in "Contexts history A" (the recommendation's target) the recommendation is performed based on contexts history similarity only, without considering positive feedback. When there is no positive feedback in "Contexts history B" (from ontology repository), this *Contexts History* is discarded for recommendation.

Figure 11 – Contexts histories with highlighted positive feedback.



Contexts History A	Contexts History B
Context A1 – Feedback 2	Context B1 – Feedback 2
Context A2 – Feedback 3	Context B2 – Feedback 5
Context A3 – Feedback 5	Context B3 – Feedback 4
Context A4 – Feedback 4	Context B4 – Feedback 5
Context A5 – Feedback 3	
Context A6 – Feedback 3	
Context A7 – Feedback 4	

Source: Made by the author.

5 EVALUATION EXPERIMENTS AND APPLICATION SCENARIOS

In this chapter we describe the implementation aspects for Vulcont. We detail how Vulcont's prototype was built for evaluation scenarios. Also, we explain how the Vulcont's evaluation was performed, describing the scenarios tested and the outputs obtained. We performed four different scenarios, exploring different aspects of Vulcont.

5.1 Implementation and Prototype Aspects

In this section we describe the technical aspects for the implementation of Vulcont. We implemented a functional prototype of Vulcont. As source of contexts histories, a dataset was used. Then recommendations are performed based on the similarity of contexts histories. Finally, Vulcont CF leverages items for recommendation based on sequences of *Objects* from ontology repository.

We implemented a prototype to validate Vulcont's model. The prototype was implemented using Java programming language. Java is widely used for integration with ontologies, where different APIs (Application Programming Interface) are available to facilitate the prototype's implementation. Some of these APIs are Apache Jena Ontology API (CARROLL et al., 2004), OWL API (MATTHEW; SEAN, 2011), and ProtégéOWL API (KNUBLAUCH et al., 2010).

In Vulcont's prototype we used Jena Ontology API to handle ontologies in Java, since it is one of most used APIs with an extensive documentation. For the persistence, we used TDB. TDB is a native high performance triple store. It is already an integrated Jena component for RDF storage and query, completely supporting Jena API. Therefore, TDB is accessed and managed directly via Jena API. This way, we are able to store triples designed by Vulcont's contexts history ontology. The information stored by TDB is retrieved using SPARQL (PRUD'HOMMEAUX; SEABORNE, 2008), which is the query language provided by the W3C (World Wide Web Consortium) for Semantic Web.

For evaluations purposes, we used the quite known dataset provided by MovieLens (HARPER; KONSTAN, 2015). This dataset contains over 20 million ratings of 27,000 movies evaluated by 138,000 users. The ratings evaluate movies with a 5 start review system, and half stars can also be used. In this case, we consider a positive feedback ratings "4.5" or "5.0". The file for ratings has four columns: userID, movieID, rating, and timestamp. They were mapped into classes *Actor*, *Object*, *Feedback*, and *Time* respectively. Since only the ratings does not provide a complete information for contextualization, we aggregated the ratings with a complete description of movies. The movies' description was obtained via OMDb (Open Movie Database) API (FRITZ, 2013). With that, we were able to retrieve a complete description from movies, such as actors, directors, genre, writer, metascore, rating, language, rated, and other complementary data. This information was also included in class *Object* from Vulcont's ontology. All the evaluations performed by a user were collected into a contexts history. Therefore, with this definition each user has a single contexts history.

Once the ontology structure was loaded into Jena API, we created and inserted instances with the data from MovieLens dataset and OMDb API. We performed the addition of properties in the ontology according to the data. The semantic value relies on context of data. In this case, we applied synonymous relation in instances using “owl:sameAs” property, and “rdfs:subClassOf” for classes’ structure. The Figure 12 shows the structure of genders of movies used in Vulcont’s prototype. There are main classes (left side) and subclasses (right side), and synonymous (separated by “/”, such as “Action / Adventure”, or “War / Battle”). If a movie is from gender “Adventure”, thus it is also from gender “Action”. But if another movie is from gender “Action”, the subclasses may not apply to it. This should be considered when comparing movies’ instances. The ontology reasoner interprets and understands genders “Action” and “Adventure” based on relation between them. In this Vulcont’s prototype, the similarity between these genders is 0.9, not identical but still high. The similarity between subclasses from a same main class is 0.8. Finally, the similarity between synonymous is 1 (instances are equal).

5.2 Evaluations Scenarios

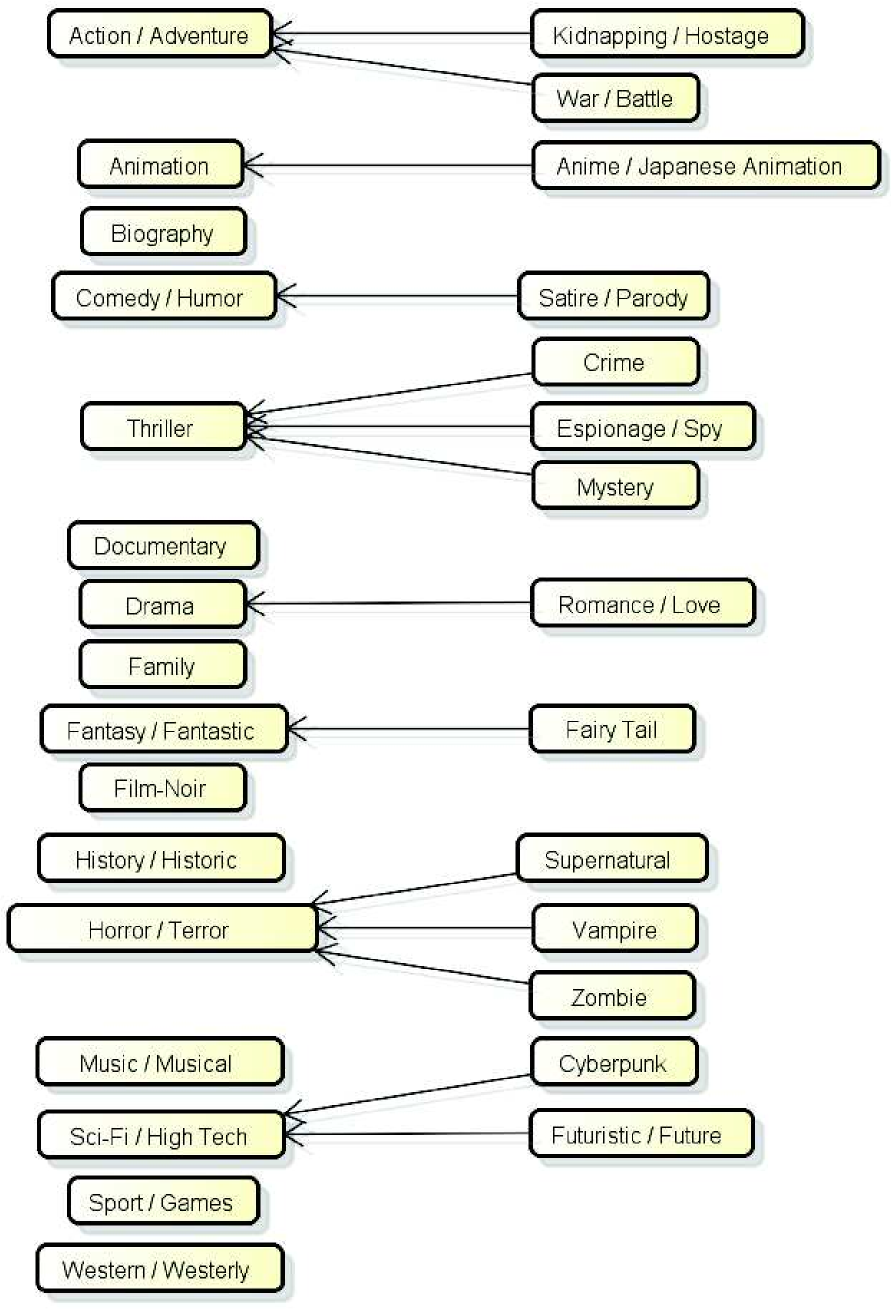
According to Ricci, Rokach and Shapira (2010), there are three main ways to evaluate recommendation systems: Offline experiment, User Studies, and Online experiment. For Vulcont’s evaluation, we decided to use the offline experiment approach since it is the only approach which does not require users, and can typically answer questions about the prediction power of evaluated algorithms. For testing the Vulcont’s prototype, we designed evaluation scenarios to discuss about the output of recommendations. This way we are able to understand how Vulcont performs its recommendations and validate them.

5.2.1 Scenario 1

In this first scenario, we randomly chose a user (ID 5) to perform a recommendation. The chosen user has a contexts history with 25 contexts, as presented in Table 2. We selected the initial 1000 users from repository to perform the similarity of contexts histories. From the 10 most similar contexts histories from repository, six movies were selected with a positive feedback, and after passing through component Vulcont CF, only three are in fact recommendable. There was one occurrence of duplicated *Objects* which were also known by the target of recommendation (Matrix, ID 2571), as well as another *Object* also already known (Whiplash, ID 112552).

When comparing a word cloud based on all genres from *Objects* in “Contexts History A” (Figure 13 (a)), with the word cloud generated from found movies during Vulcont CF process (Figure 13 (b)), we are able to see that genre “Drama” is present as main genre in both of them. Also, an important aspect to highlight is the relation between “Comedy” and “Humor”.

Figure 12 – Genders of Movies organized by taxonomy and synonymy.



Source: Made by the author.

Table 2 – Information regarding scenario 1.

Contexts History ID 5	
Movie's ID	
112183	4.0
112552	4.5
112556	4.5
114060	4.0
114180	1.5
114662	2.5
115170	3.5
115210	4.0
115617	3.0
115713	4.5
116797	4.0
119141	2.5
119145	3.5
119155	3.0
1203	4.0
120466	0.5
122882	4.5
129354	1.5
133419	3.5
2571	2.5
58559	4.0
6016	5.0
7502	4.5
79132	3.0
92259	4.5

Positive Feedback		Movie's Title
112552	4.5	Whiplash
112556	4.5	Gone Girl
115713	4.5	Ex Machina
122882	4.5	Mad Max: Fury Road
6016	5.0	City of God
7502	4.5	Band of Brothers
92259	4.5	The Intouchables

Found Movies
Matrix (2571)
Matrix (2571)
Whiplash (112552)
Boys Don't Cry (2908)
Election (2599)
There's Something About Mary (1923)

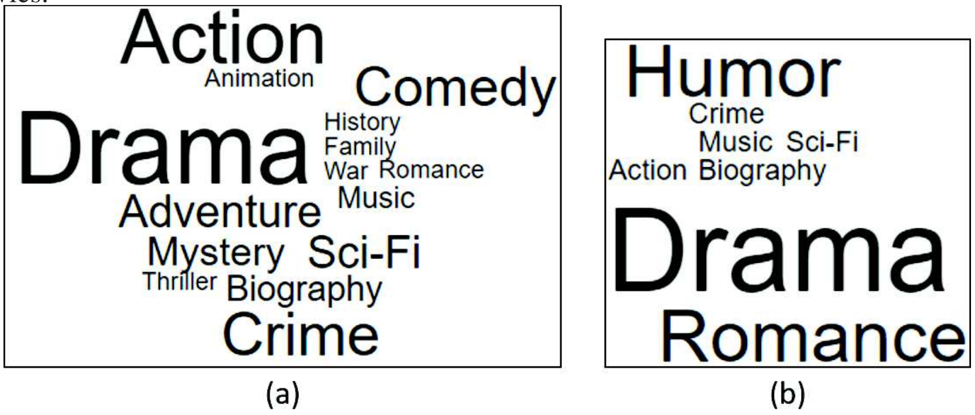
Source: Made by the author.

The word cloud does not consider semantic value, but the similarity measures are properly detecting that. Since "Comedy" and "Humor" are considered synonymous, they have a impact in the similarity, hence we can see the genres as Top 3 in the figures accordingly. This scenario showed that semantic relations are in fact being considered and impacting similarity analysis. This could only be achieved by ontology's usage. We focused on the movies' genres since the similarity of genres was the key data property, with highest similarity in the contexts history similarity.

5.2.2 Scenario 2

While the first scenario compared genders' frequency from all movies in contexts history with found movies, this second scenario compares genders' frequency between movies with positive feedback and recommended movies. For this scenario, we selected a random user (ID

Figure 13 – Scenario 1 - (a) Word cloud for genres in Contexts History A. (b) Word cloud for genres in found movies.



Source: Made by the author.

17731) and performed the recommendation comparing the contexts histories from 1000 random users from repository. We kept the number of compared contexts histories (one contexts history for one user). We defined the number of desired results as 5, therefore a Top 5 ranking will be returned with recommended movies ordered by relevance. This Top 5 contains only relevant *Objects*, where duplicated records and *Objects* already known by target of recommendation are already removed.

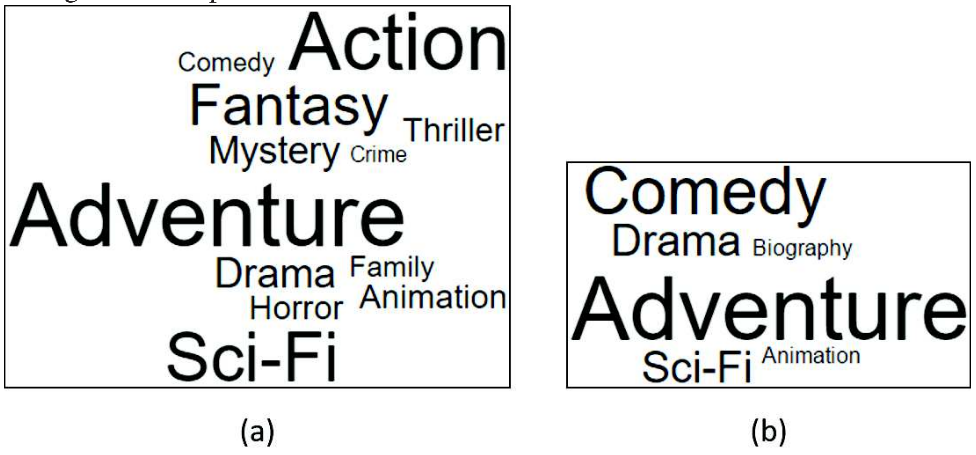
We created a word cloud presented in Figure 14 (a), which presents the genres by their frequency among all *Objects* with positive feedback which were evaluated in contexts of user 17731. We can see that “Adventure”, “Action”, and “Sci-Fi” are the main genres respectively. Even though “Adventure” and “Action” have a semantic relation considered by ontology reasoner, they were separately presented in word cloud.

The output of Vulcont CF is presented by Table 3 which presents the Top 5 *Objects* from recommendation. The record number 1 is provided by the contexts history with highest similarity. If we create a word cloud based on genres from Top 5 *Objects*, we will have it as shown in Figure 14 (b). For this word cloud in specific, we aggregated the occurrence of synonymous. Therefore even though there are entries for genre “Action” in Top 5, they were changed to “Adventure” to display its higher frequency.

5.2.3 Scenario 3

In this third scenario, we decided to hide some contexts from a contexts history of a random user (ID 229041). This way, the contexts history similarity will consider only contexts that are not hidden. Hence, we can evaluate if the recommendation is able to predict the hidden *Objects*, more importantly the ones with a positive feedback. The contexts history of user ID 229041 contains 980 past contexts. We decide the hide 25% of most recent *Object’s* evaluations. Therefore, we have a smaller *Contexts History* now with 735 *Contexts*. There are 245 hidden

Figure 14 – Scenario 2 - (a) Word cloud based on genres of positive feedback from user 17731. (b) Word cloud based on genres of Top 5 movies from recommendation.



Source: Made by the author.

Table 3 – Scenario 2 - Top 5 Objects recommended by its similarity and relevance.

Recommendation's sequence	Movie's ID	Movie's Genres
1	2394	Animation, Adventure, Biography
2	1391	Action, Comedy, Sci-Fi
3	356	Comedy, Drama
4	106002	Action, Sci-Fi
5	3052	Adventure, Comedy, Drama

Source: Made by the author.

Contexts, where the majority of them (93%) have a positive feedback.

We compared the *Contexts History* from user ID 229041 with 1000 contexts histories from repository, which were randomly selected. Once the 1000 similarity comparisons were performed, we selected the Top 5 most similar *Contexts Histories* to check the list of *Objects* with positive feedback. This list resulted in 3 records provided by 2 *Contexts Histories*, as Table 4 shows. The second *Object* "Independence Day" was in fact evaluated by user 229041 positively, and this *Object* was indeed part of one hidden context.

This analysis provide us some insights about the recommendation power of Vulcont. Even though the similarity analysis was performed using only 75% of contexts, for the first three items in recommendation, one certainly is relevant and has a positive feedback. The other two items were recommended based on positive feedback of users with high similarity comparing to user 229041. The *Object* 356 "Forrest Gump" is already positively known by user 229041, and its context was considered in similarity (it was not a hidden context). Hence, only "GoldenEye" is unknown, and therefore the user's feedback about this particular item is unknown.

Table 4 – Scenario 3 - List of movies with positive feedback from most similar contexts histories.

Contexts History ID	Movie's ID	Movie's title	Movie's genre
188996	10	GoldenEye	Action, Adventure, Thriller
188996	780	Independence Day	Action, Adventure, Sci-Fi
61435	356	Forrest Gump	Comedy, Drama

Source: Made by the author.

5.2.4 Scenario 4

This fourth scenario is similar to Scenario 3. We hid some *Contexts* from the *Contexts History* of an *Actor* (user ID 85612). This user has a *Contexts History* composed of 735 *Contexts*. Again, we hid 25% of most recent *Contexts*, resulting in a *Contexts History* with 551 elements, where 184 were hidden from contexts history similarity. However, differently from Scenario 3, we increased the number of random *Contexts Histories* from repository for comparison, from 1000 to 2000. Since more data is being considered in contexts history similarity, the more accurate the recommendation will be. After the contexts history similarity measurement, we obtained the list of most similar *Contexts Histories*. Before the Vulcont CF processing could filter the already known *Objects*, we checked the Top 5 recommended movies, shown by Table 5.

Table 5 – Scenario 4 - List of recommended movies before Vulcont CF.

Recommendation's sequence	Contexts History ID	Movie's ID	Feedback from Repository	Movie's Title	Movie's Genre	Known by Actor	Actor's Feedback
1	37875	1911	4.50	Doctor Dolittle	Comedy, Family, Fantasy	YES	4.50
2	34147	1721	5.00	Titanic	Drama, Romance	YES, but hidden	5.00
3	81875	1485	5.00	Liar Liar	Comedy, Fantasy	YES	5.00
4	176472	6565	4.50	Seabiscuit	Drama, History, Sport	NO	-
5	220293	105	5.00	The Bridges of Madison County	Drama, Romance	NO	-

Source: Made by the author.

As Table 5 refers, three of five recommended *Objects* are already known by the target *Actor*. Normally, after Vulcont CF processing, two of these movies would be removed. The already known movies are ID 1911 (Doctor Dolittle) and ID 1485 (Liar Liar). The other two movies "Seabiscuit" and "The Bridges of Madison County" are unknown by the user, therefore their feedback is also nonexistent.

As well as Scenario 3, this scenario provides insights about the prediction power of Vulcont's algorithms. We can see that Vulcont is able to identify *Context's* similarity, once the list of potential movies to be recommended has some *Objects* already known by the *Actor*. The recommendation list contains also the hidden movie “Titanic”, which certainly is an accurate recommendation. The known *Objects* are removed by Vulcont CF, so they will not reach the target’s of recommendation. However, these *Objects* are still relevant while calculating the contexts history similarity. Therefore, it is normal and even expected to have some known *Objects* in the list once we identified the most similar contexts histories and the *Objects* with a positive feedback.

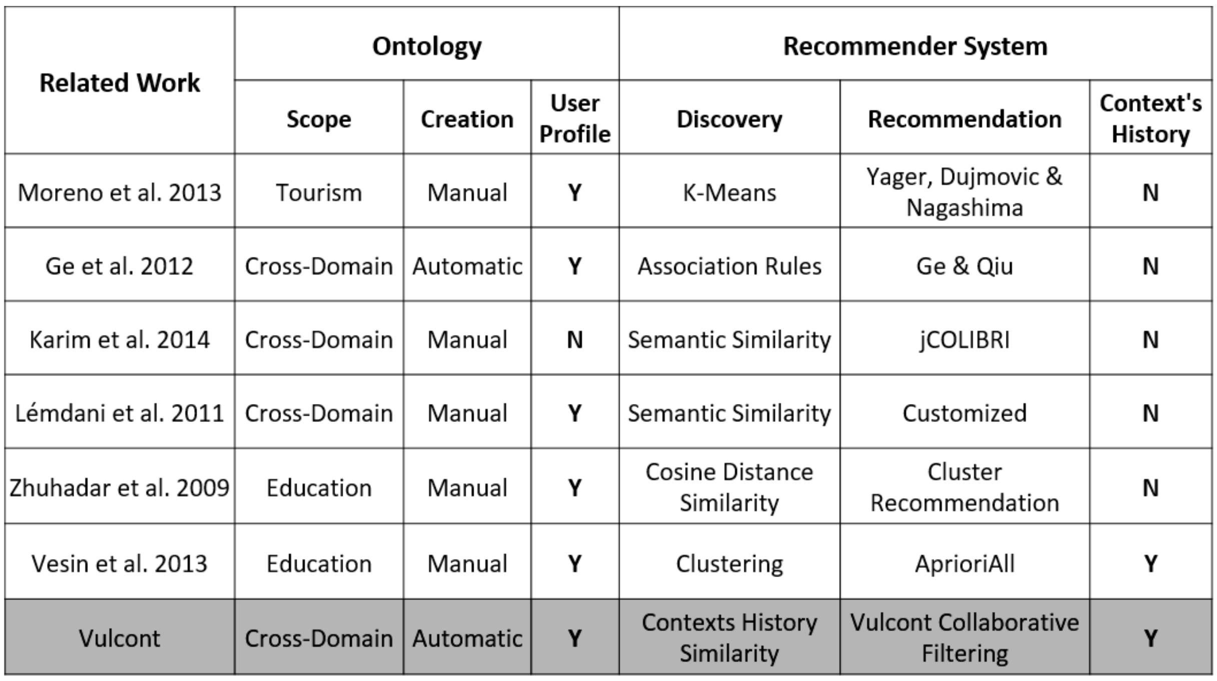
6 FINAL CONSIDERATIONS

In this work, we discussed the use of contexts history ontology in a recommender system. We proposed Vulcont's model and developed its prototype, which compares contexts histories, exploring semantic and ontology's properties, to identify most similar contexts histories. With that, Vulcont's recommends potential *Objects* to a target *Actor*. We also presented the contexts history ontology used by Vulcont for the recommendations. The Vulcont's ontology was implemented, and tested with an educational scenario.

As well as its ontology, we created four evaluational scenarios for Vulcont's prototype, using an offline approach according to Ricci, Rokach and Shapira (2010). Each scenario has its particularities and explores different aspects of Vulcont's recommendation. We commented about the results of every scenario, discussing about Vulcont's recommendation power, and also identifying possible aspects to be improved in future works. Lastly, based on the experiment with the offline approach, we identified that Vulcont's recommendations are working properly, providing reliable and meaningful recommendations. This occurs mainly due the use of semantic and relations properties in ontology, where its reasoner is able to consider these properties and return the required knowledge with queries.

6.1 Contributions

Table 6 – Comparison among related works and Vulcont.



Related Work	Ontology			Recommender System		
	Scope	Creation	User Profile	Discovery	Recommendation	Context's History
Moreno et al. 2013	Tourism	Manual	Y	K-Means	Yager, Dujmovic & Nagashima	N
Ge et al. 2012	Cross-Domain	Automatic	Y	Association Rules	Ge & Qiu	N
Karim et al. 2014	Cross-Domain	Manual	N	Semantic Similarity	jCOLIBRI	N
Lémdani et al. 2011	Cross-Domain	Manual	Y	Semantic Similarity	Customized	N
Zhuhadar et al. 2009	Education	Manual	Y	Cosine Distance Similarity	Cluster Recommendation	N
Vesin et al. 2013	Education	Manual	Y	Clustering	AprioriAll	Y
Vulcont	Cross-Domain	Automatic	Y	Contexts History Similarity	Vulcont Collaborative Filtering	Y

Source: Made by the author.

In this work, we detailed and compared six related works. Now we analyze Vulcont according to the same criteria applied in related works, as presented in Table 6. The Vulcont's ontology was designed for cross-domain recommendations. This means that Vulcont's ontology is able to represent context's information for recommendation regardless its domain. However, the collected contexts will be grouped to compound the contexts histories. In order to explore semantic and classes relations, these definitions are provided by the application, once this information relies on domain's rules. Once the context information is mapped into the ontology, its population is automatic using Jena Ontology API, so all the knowledge is represented in the ontology. Vulcont uses user's profile in ontology. This profile is compounded by concepts *Status*, *Goals*, *Feedback* and *Actor*. For the discover, the work explores contexts history similarity based on semantic relations and ontology's properties. The recommendation is performed based on an own implementation of collaborative filtering. Vulcont CF selects *Objects* based on existent *Context's* sequence in ontology repository (whenever possible), or simply by measurement of similarity.

The main contribution of Vulcont is the use of semantic relations and ontology's properties in a similarity measurement of *Contexts History*, which is a data's structure more complete when compared over single *Contexts*. We designed the ontology to support context information from different domains, but still aimed for recommendations. Vulcont was able to explore ontology reasoner, and use it in order to perform relevant recommendations considering contexts's sequences and similarity in its own collaborative filtering approach. In the end, Vulcont's recommendation is considered a hybrid approach since not only CF is considered to perform the recommendations.

6.2 Future works

Based on the performed evaluations for the four elaborated scenarios, we found aspects to be considered in future works, which can be improved and explored further. The first Vulcont's aspect to be considered in a future improvement is regarding the use of more ontology relations. Vulcont used taxonomy and synonymy for the movies' gender. Using other relations could expand and explore even more the use of ontology reasoner in the contexts history similarity. The Vulcont's prototype used synonymous relations between ontology's instances according to context's domain. Also, the organizational structure of concepts provided relations for classes and subclasses. However, different relations can still express further knowledge in the contexts history ontology, and this will potentially improve the performed recommendations.

For Vulcont's prototype, we created a local ontology for internal evaluations. However, it is totally viable to implement *web services* and expose Vulcont's ontology for remote management. The recommendations could be requested and consumed using JSON (JavaScript Object Notation) open source format. Also, Vulcont's recommendation engine can explore the use of distributed processing, since it is currently using serial processing for contexts history similarity.

For example in Scenario 4, the contexts history similarity measurement took around 15 hours. This was the longest processing time among all evaluation scenarios. The user (ID 85612) of Scenario 4 belongs to the 2% of users with greater number of evaluated *Contexts*. Since the contexts history similarity measurement is performed repetitively, and fits requirements of parallel processing to consistently reduce processing time.

For further analysis of Vulcont's recommendations, we can use different datasets from different domain to check the resulted recommendations. With that, we can compare results obtained with the experimental offline approach performed in this work, which used movies' recommendation in entertainment domain. Also, an online experiment with users with a well designed questionnaire can evaluate Vulcont in a more complete way, answering a wider set of questions compared to the offline approach.

REFERENCES

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. **IEEE Trans. Knowl. Data Eng.**, [S.l.], v. 17, n. 6, p. 734–749, jun 2005.

AMANN, B.; FUNDULAKI, I. Integrating Ontologies and Thesauri to Build RDF Schemas. In: **Research and Advanced Technology for Digital Libraries**. [S.l.]: Springer Science Business Media, 1999. p. 234–253.

ANAND, S. S.; MOBASHER, B. Intelligent Techniques for Web Personalization. In: **Lecture Notes in Computer Science**. [S.l.]: Springer Science Business Media, 2005. p. 1–36.

ANDREJKO, A.; BIELIKOVA, M. Comparing Instances of Ontological Concepts for Personalized Recommendation in Large Information Spaces. **Computing and Informatics**, [S.l.], v. 28, n. 4, p. 429–452, 2009.

BARBOSA, J. L. V.; MARTINS, C.; FRANCO, L. K.; BARBOSA, D. N. F. TrailTrade: a model for trail-aware commerce support. **Computers in Industry**, [S.l.], v. 80, p. 43–53, aug 2016.

BARTA, R.; FEILMAYR, C.; PRöLL, B.; GRüN, C.; WERTHNER, H. Covering the semantic space of tourism. In: WORKSHOP ON CONTEXT, INFORMATION AND ONTOLOGIES - CIAO 09, 1., 2009. **Proceedings...** Association for Computing Machinery (ACM), 2009.

BERGMANN, R.; KOLODNER, J.; PLAZA, E. Representation in case-based reasoning. **The Knowledge Engineering Review**, [S.l.], v. 20, n. 03, p. 209, sep 2005.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. **Knowledge-Based Systems**, [S.l.], v. 46, p. 109–132, jul 2013.

BORGO, S. How Formal Ontology can help Civil Engineers. In: **Ontologies for Urban Development**. [S.l.]: Springer Science Business Media, 2007. p. 37–45.

BORNTRÄGER, C. **Contextual Influence on the Usability of Different Media Types**. 2003. Tese (Doutorado em Ciência da Computação) — Diplomarbeit, Technische Universität Ilmenau, Fachgebiet Kommunikationsnetze, Ilmenau, 2003.

BORST, W. N. **Construction of engineering ontologies for knowledge sharing and reuse**. [S.l.: s.n.], 1997.

BRIDGE, D.; GÖKER, M. H.; MCGINTY, L.; SMYTH, B. Case-based recommender systems. **The Knowledge Engineering Review**, [S.l.], v. 20, n. 03, p. 315, sep 2005.

BROWN, P.; BOVEY, J.; CHEN, X. Context-aware applications: from the laboratory to the marketplace. **IEEE Pers. Commun.**, [S.l.], v. 4, n. 5, p. 58–64, 1997.

BURKE, R. Knowledge-based recommender systems. **Encyclopedia of library and information systems**, [S.l.], v. 69, n. Supplement 32, p. 175–186, 2000.

BURKE, R. Hybrid Recommender Systems: survey and experiments. **User Modeling and User-Adapted Interaction**, [S.l.], v. 12, n. 4, p. 331–370, 2002.

BURKE, R. Hybrid Web Recommender Systems. In: **The Adaptive Web**. [S.l.]: Springer Science Business Media, 2007. p. 377–408.

CARDOSO, I. G.; MOTA, B.; BARBOSA, J. L. V.; ROSA RIGHI, R. da. Vulcanus 2.0: a recommender system for accessibility. **CLEI electronic journal**, [S.l.], v. 19, n. 1, p. 6:1–6:24, apr 2016.

CARROLL, J. J.; DICKINSON, I.; DOLLIN, C.; REYNOLDS, D.; SEABORNE, A.; WILKINSON, K. Jena. In: WORLD WIDE WEB CONFERENCE ON ALTERNATE TRACK PAPERS & POSTERS - WWW ALT. 04, 13., 2004. **Proceedings. . .** Association for Computing Machinery (ACM), 2004.

ČERĀNS, K.; BŪMANS, G. RDB2OWL: a rdb-to-rdf/owl mapping specification language. In: INTERNATIONAL BALTIC CONFERENCE ON DATABASES AND INFORMATION SYSTEMS, 9., 2010. **Anais. . .** [S.l.: s.n.], 2010. p. 139–152.

CHEN, M.; MAO, S.; LIU, Y. Big Data: a survey. **Mobile Networks and Applications**, [S.l.], v. 19, n. 2, p. 171–209, jan 2014.

CLARKE, S.; DRIVER, C. Context-aware trails [mobile computing]. **Computer**, [S.l.], v. 37, n. 8, p. 97–99, aug 2004.

COFFIELD, F. **Should we be using learning styles? What research has to say in practice**. London: Learning and Skills Research Centre, 2004.

COSLEY, D.; LAM, S. K.; ALBERT, I.; KONSTAN, J. A.; RIEDL, J. Is seeing believing? In: HUMAN FACTORS IN COMPUTING SYSTEMS - CHI 03, 2003. **Proceedings. . .** Association for Computing Machinery (ACM), 2003.

DEBES, M.; LEWANDOWSKA, A.; SEITZ, J. Definition and Implementation of Context Information. In: KYAMAKYA, K.; JOBMANN, K.; KUCHENBECKER, H.-P.(HERAUSG.): JOINT SECOND WORKSHOP ON POSITIONING, NAVIGATION AND COMMUNICATION, 2005. **Anais. . .** [S.l.: s.n.], 2005. p. 63–68.

DEY, A.; ABOWD, G.; SALBER, D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. **Human-Comp. Interaction**, [S.l.], v. 16, n. 2, p. 97–166, dec 2001.

DEY, A. K. Context-aware computing: the cyberdesk project. In: AAAI 1998 SPRING SYMPOSIUM ON INTELLIGENT ENVIRONMENTS, 1998. **Proceedings. . .** [S.l.: s.n.], 1998. p. 51–54.

DEY, A. K. Understanding and Using Context. **Personal and Ubiquitous Computing**, [S.l.], v. 5, n. 1, p. 4–7, feb 2001.

DEY, A. K.; ABOWD, G.; PINKERTON, M.; WOOD, A. CyberDesk. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY - UIST 97, 10., 1997. **Proceedings. . .** Association for Computing Machinery (ACM), 1997.

DÍAZ-AGUDO, B.; GONZÁLEZ-CALERO, P. A.; RECIO-GARCÍA, J. A.; SÁNCHEZ-RUIZ-GRANADOS, A. A. Building CBR systems with jcolibri. **Science of Computer Programming**, [S.l.], v. 69, n. 1-3, p. 68–75, dec 2007.

DRIVER, C.; CLARKE, S. An application framework for mobile, context-aware trails. **Pervasive and Mobile Computing**, [S.l.], v. 4, n. 5, p. 719–736, oct 2008.

DUJMOVIĆ, J. J.; NAGASHIMA, H. LSP method and its use for evaluation of Java IDEs. **International Journal of Approximate Reasoning**, [S.l.], v. 41, n. 1, p. 3–22, jan 2006.

FELDER, R. M.; SILVERMAN, L. K. Learning and teaching styles in engineering education. **Engineering education**, [S.l.], v. 78, n. 7, p. 674–681, 1988.

FELFERNIG, A.; BURKE, R. Constraint-based recommender systems. In: ELECTRONIC COMMERCE - ICEC 08, 10., 2008. **Proceedings...** Association for Computing Machinery (ACM), 2008.

FELFERNIG, A.; FRIEDRICH, G.; JANNACH, D.; ZANKER, M. An Integrated Environment for the Development of Knowledge-Based Recommender Applications. **International Journal of Electronic Commerce**, [S.l.], v. 11, n. 2, p. 11–34, dec 2006.

FODOR, O.; WERTHNER, H. Harmonise: a step toward an interoperable e-tourism marketplace. **International Journal of Electronic Commerce**, [S.l.], v. 9, n. 2, p. 11–39, 2005.

FRITZ, B. **OMDb API**. URL: https://www.omdbapi.com/. Accessed on 31 January 2017, in **The Open Movie Database**.

GE, J.; CHEN, Z.; PENG, J.; LI, T. An ontology-based method for personalized recommendation. In: IEEE 11TH INTERNATIONAL CONFERENCE ON COGNITIVE INFORMATICS AND COGNITIVE COMPUTING, 2012., 2012. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2012.

GE, J.; QIU, Y. Concept Similarity Matching Based on Semantic Distance. In: FOURTH INTERNATIONAL CONFERENCE ON SEMANTICS, KNOWLEDGE AND GRID, 2008., 2008. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2008.

GENESERETH, M. R.; NILSSON, N. J. Logical Foundations of Artificial Intelligence. **Morgan Kaufmann**, [S.l.], 1987.

GLUZ, J. C. **PROWLOG Programação OWL em Prolog**. URL: http://obaa.unisinos.br/drupal7/?q=node/8. Accessed on 31 January 2017, in **Portal TAOS3**.

GOLDBERG, D.; NICHOLS, D.; OKI, B. M.; TERRY, D. Using collaborative filtering to weave an information tapestry. **Communications of the ACM**, [S.l.], v. 35, n. 12, p. 61–70, dec 1992.

GOLDBERG, K.; ROEDER, T.; GUPTA, D.; PERKINS, C. **Information Retrieval**, [S.l.], v. 4, n. 2, p. 133–151, 2001.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. **Ontological Engineering**: with examples from the areas of knowledge management, e-commerce and the semantic web. [S.l.]: Springer Science & Business Media, 2006.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? **International Journal of Human-Computer Studies**, [S.l.], v. 43, n. 5-6, p. 907–928, nov 1995.

GUARINO, N.; OBERLE, D.; STAAB, S. What Is an Ontology? In: **Handbook on Ontologies**. [S.l.]: Springer Science Business Media, 2009. p. 1–17.

GWIZDKA, J. What's in the context. In: COMPUTER HUMAN INTERACTION, 2000. **Anais...** [S.l.: s.n.], 2000. v. 2000.

HARPER, F. M.; KONSTAN, J. A. The MovieLens Datasets. **ACM Transactions on Interactive Intelligent Systems**, [S.l.], v. 5, n. 4, p. 1–19, dec 2015.

HECKMANN, D. **Ubiquitous User Modeling (Dissertations in Artificial Intelligence – Diski)**. [S.l.]: IOS Press, 2006.

HECKMANN, D.; SCHWARTZ, T.; BRANDHERM, B.; SCHMITZ, M.; WILAMOWITZ-MOELLENDORFF, M. von. Gumo – The General User Model Ontology. In: **User Modeling 2005**. [S.l.]: Springer Science Business Media, 2005. p. 428–432.

HILBERT, M.; LOPEZ, P. The Worlds Technological Capacity to Store, Communicate, and Compute Information. **Science**, [S.l.], v. 332, n. 6025, p. 60–65, feb 2011.

HONEY, P.; MUMFORD, A. **The Manual of Learning Styles**. [S.l.]: Peter Honey Publications, 1992.

JANNACH, D.; ZANKER, M.; FELFERNIG, A.; FRIEDRICH, G. **Recommender Systems**. [S.l.]: Cambridge University Press (CUP), 2009.

KANG, J.; CHOI, J. An Ontology-Based Recommendation System Using Long-Term and Short-Term Preferences. In: INTERNATIONAL CONFERENCE ON INFORMATION SCIENCE AND APPLICATIONS, 2011., 2011. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2011.

KARIM, J.; AUFAURE, M.-A.; CHIKY, R.; MANCENY, M. A generic ontology for CBR-based recommender systems. In: INTERNATIONAL WORKSHOP ON COMPUTATIONAL INTELLIGENCE FOR MULTIMEDIA UNDERSTANDING (IWCIM), 2014., 2014. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2014.

KARIM, J.; MANCENY, M.; CHIKY, R.; MANAGO, M.; AUFAURE, M.-A. Using collaborative filtering to enhance domain-independent CBR recommenders personalization. In: IEEE 9TH INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE (RCIS), 2015., 2015. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2015.

KNUBLAUCH, H. et al. **Protégé-OWL API Programmer's guide**. URL: http://protege.stanford.edu/plugins/owl/api/guide.html. Accessed on 31 January 2017, in **Protégé wiki**.

KNUBLAUCH, H.; FERGERSON, R. W.; NOY, N. F.; MUSEN, M. A. The Protégé OWL Plugin: an open development environment for semantic web applications. In: **The Semantic Web – ISWC 2004**. [S.l.]: Springer Science Business Media, 2004. p. 229–243.

KOLB, D. Individuality in learning and the concept of learning styles. **Experiential learning**, [S.l.], p. 61–98, 1984.

KOLODNER, J. L. An introduction to case-based reasoning. **Artificial Intelligence Review**, [S.l.], v. 6, n. 1, p. 3–34, 1992.

KONSTAN, J. A.; MILLER, B. N.; MALTZ, D.; HERLOCKER, J. L.; GORDON, L. R.; RIEDL, J. GroupLens: applying collaborative filtering to usenet news. **Communications of the ACM**, [S.l.], v. 40, n. 3, p. 77–87, mar 1997.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix Factorization Techniques for Recommender Systems. **Computer**, [S.l.], v. 42, n. 8, p. 30–37, aug 2009.

LACOSTE, D.; SAWANT, K. P.; ROY, S. An efficient XML to OWL converter. In: INDIA SOFTWARE ENGINEERING CONFERENCE ON - ISEC 11, 4., 2011. **Proceedings...** Association for Computing Machinery (ACM), 2011.

LASSILA, O.; MCGUINESS, D. The role of frame-based representation on the semantic Web, Rapport technique KSL-01-02. **Knowledge Systems Laboratory, Stanford University**, [S.l.], 2001.

LEMDANI, R.; BENNACER, N.; POLAILLON, G.; BOURDA, Y. A collaborative and semantic-based approach for recommender systems. In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS DESIGN AND APPLICATIONS, 2010., 2010. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2010.

LÉMDANI, R.; POLAILLON, G.; BENNACER, N.; BOURDA, Y. A semantic similarity measure for recommender systems. In: INTERNATIONAL CONFERENCE ON SEMANTIC SYSTEMS - I-SEMANTICS 11, 7., 2011. **Proceedings...** Association for Computing Machinery (ACM), 2011.

LEVENE, M.; PETERSON, D. Trail records and ampliative learning. **School of Computer Science and Information Systems, Birkbeck College, University of London, Research Report BBKCS-02-10**, [S.l.], 2002.

LEYLA; NASRAOUI, O.; WYATT, R.; ROMERO, E. Multi-model Ontology-Based Hybrid Recommender System in E-learning Domain. In: IEEE/WIC/ACM INTERNATIONAL JOINT CONFERENCE ON WEB INTELLIGENCE AND INTELLIGENT AGENT TECHNOLOGY, 2009., 2009. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2009.

LI, W.; EICKHOFF, C.; VRIES, A. P. de. Want a coffee? In: ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL - SIGIR 12, 35., 2012. **Proceedings...** Association for Computing Machinery (ACM), 2012.

MAHMOOD, T.; RICCI, F. Improving recommender systems with adaptive conversational strategies. In: ACM CONFERENCE ON HYPERTEXT AND HYPERMEDIA - HT 09, 20., 2009. **Proceedings...** Association for Computing Machinery (ACM), 2009.

MATTHEW, H.; SEAN, B. The OWL API: a java api for owl ontologies. **Semantic Web**, [S.l.], v. 2, n. 1, p. 11–21, 2011.

MCSHERRY, F.; MIRONOV, I. Differentially private recommender systems. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING - KDD 09, 15., 2009. **Proceedings...** Association for Computing Machinery (ACM), 2009.

MELNIK, S.; GARCIA-MOLINA, H.; RAHM, E. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 18., 2002. **Proceedings...** Institute of Electrical & Electronics Engineers (IEEE), 2002. p. 117–128.

MILLER, G. A.; BECKWITH, R.; FELLBAUM, C.; GROSS, D.; MILLER, K. J. Introduction to WordNet: an on-line lexical database. **International Journal of Lexicography**, [S.l.], v. 3, n. 4, p. 235–244, 1990.

MLADENIC, D. Text-learning and related intelligent agents: a survey. **IEEE Intell. Syst.**, [S.l.], v. 14, n. 4, p. 44–54, jul 1999.

MORENO, A.; VALLS, A.; ISERN, D.; MARIN, L.; BORRÀS, J. SigTur/E-Destination: ontology-based personalized recommendation of tourism and leisure activities. **Engineering Applications of Artificial Intelligence**, [S.l.], v. 26, n. 1, p. 633–651, jan 2013.

NAUDET, Y.; AGHASARYAN, A.; TOMS, Y.; SENOT, C. An Ontology-Based Profiling and Recommending System for Mobile TV. In: THIRD INTERNATIONAL WORKSHOP ON SEMANTIC MEDIA ADAPTATION AND PERSONALIZATION, 2008., 2008. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2008.

NOY, N. F.; MCGUINNESS, D. L. et al. **Ontology development 101**: a guide to creating your first ontology. [S.l.]: Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, Stanford, CA, 2001.

OARD, D. W.; KIM, J. Implicit feedback for recommender systems. In: AAAI WORKSHOP ON RECOMMENDER SYSTEMS, 1998. **Proceedings...** [S.l.: s.n.], 1998. p. 81–83.

OLIVEIRA, R. R.; CARDOSO, I. M.; BARBOSA, J. L.; COSTA, C. A. da; PRADO, M. P. An intelligent model for logistics management based on geofencing algorithms and RFID technology. **Expert Systems with Applications**, [S.l.], v. 42, n. 15-16, p. 6082–6097, sep 2015.

PAIVA, F. A. P. de; COSTA, J. A. F.; SILVA, C. R. M. A Hierarchical Architecture for Ontology-Based Recommender Systems. In: BRICS CONGRESS ON COMPUTATIONAL INTELLIGENCE AND 11TH BRAZILIAN CONGRESS ON COMPUTATIONAL INTELLIGENCE, 2013., 2013. **Anais...** Institute of Electrical & Electronics Engineers (IEEE), 2013.

PASHLER, H.; MCDANIEL, M.; ROHRER, D.; BJORK, R. Learning Styles: concepts and evidence. **Psychological Science in the Public Interest**, [S.l.], v. 9, n. 3, p. 105–119, dec 2009.

PASK, G. STYLES AND STRATEGIES OF LEARNING. **British Journal of Educational Psychology**, [S.l.], v. 46, n. 2, p. 128–148, jun 1976.

PAZZANI, M. J. **Artificial Intelligence Review**, [S.l.], v. 13, n. 5/6, p. 393–408, 1999.

PI-LIAN, H.; TONG, W. Web log mining by an improved aprioriall algorithm. **Engineering and Technology**, [S.l.], v. 4, n. 2005, p. 97–100, 2005.

PIETERSE, V.; BLACK, P. E. **"Levenshtein distance"**. URL: https://www.nist.gov/dads/HTML/Levenshtein.html. Accessed on 31 January 2017, in **Dictionary of Algorithms and Data Structures** [online].

PRUD'HOMMEAUX, E.; SEABORNE, A. **SPARQL Query Language for RDF**. URL: http://www.w3.org/TR/rdf-sparql-query. Accessed on 31 January 2017, in **W3C Recommendation**.

RANGANATHAN, A.; CAMPBELL, R. H. An infrastructure for context-awareness based on first order logic. **Personal and Ubiquitous Computing**, [S.l.], v. 7, n. 6, p. 353–364, dec 2003.

RESNICK, P.; VARIAN, H. R. Recommender systems. **Communications of the ACM**, [S.l.], v. 40, n. 3, p. 56–58, mar 1997.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to Recommender Systems Handbook. In: **Recommender Systems Handbook**. [S.l.]: Springer Science Business Media, 2010. p. 1–35.

ROSA, J. H.; BARBOSA, J. L. V.; KICH, M.; BRITO, L. A Multi-Temporal Context-aware System for Competences Management. **Int J Artif Intell Educ**, [S.l.], v. 25, n. 4, p. 455–492, jun 2015.

ROSA, J. H. da; BARBOSA, J. L.; RIBEIRO, G. D. ORACON: an adaptive model for context prediction. **Expert Systems with Applications**, [S.l.], v. 45, p. 56–70, mar 2016.

ROUSSEY, C.; PINET, F.; KANG, M. A.; CORCHO, O. An Introduction to Ontologies and Ontology Engineering. In: **Advanced Information and Knowledge Processing**. [S.l.]: Springer Science Business Media, 2011. p. 9–38.

RYAN, N. S.; PASCOE, J.; MORSE, D. R. Enhanced reality fieldwork: the context-aware archaeological assistant. In: COMPUTER APPLICATIONS IN ARCHAEOLOGY, 1998. **Anais. . .** [S.l.: s.n.], 1998.

SAMPAIO, I. A. **Apredizagem Ativa em Sistemas de Filtragem Colaborativa**. 2006. 82 p. Dissertação (Mestrado em Ciências da Computação) — Centro de Informática - Universidade Federal de Pernambuco, Recife, 2006.

SARWAR, B.; KARYPIS, G.; KONSTAN, J.; REIDL, J. Item-based collaborative filtering recommendation algorithms. In: WORLD WIDE WEB - WWW 01, 2001. **Proceedings. . .** Association for Computing Machinery (ACM), 2001.

SATYANARAYANAN, M. Pervasive computing: vision and challenges. **IEEE Pers. Commun.**, [S.l.], v. 8, n. 4, p. 10–17, 2001.

SCHAFER, J. B.; FRANKOWSKI, D.; HERLOCKER, J.; SEN, S. Collaborative Filtering Recommender Systems. In: **The Adaptive Web**. [S.l.]: Springer Science Business Media, 2007. p. 291–324.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-Aware Computing Applications. In: FIRST WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1994., 1994. **Anais. . .** Institute of Electrical & Electronics Engineers (IEEE), 1994.

SCHMIDT, A.; AIDOO, K. A.; TAKALUOMA, A.; TUOMELA, U.; LAERHOVEN, K. V.; VELDE, W. V. de. Advanced Interaction in Context. In: **Handheld and Ubiquitous Computing**. [S.l.]: Springer Science Business Media, 1999. p. 89–101.

SCHMIDT, A.; BEIGL, M.; GELLERSEN, H.-W. There is more to context than location. **Computers & Graphics**, [S.l.], v. 23, n. 6, p. 893–901, dec 1999.

SHETH, B.; MAES, P. Evolving agents for personalized information filtering. In: IEEE CONFERENCE ON ARTIFICIAL INTELLIGENCE FOR APPLICATIONS, 9., 1993. **Proceedings...** Institute of Electrical & Electronics Engineers (IEEE), 1993.

SILVA, J. M.; ROSA, J. H.; BARBOSA, J. L. V.; BARBOSA, D. N. F.; PALAZZO, L. A. M. Content distribution in trail-aware environments. **Journal of the Brazilian Computer Society**, [S.l.], v. 16, n. 3, p. 163–176, jul 2010.

SMITH, A. **Who controls the past controls the future-life annotation in principle and practice**. 2008. Tese (Doutorado em Ciência da Computação) — University of Southampton, 2008.

SPENCE, M.; DRIVER, C.; CLARKE, S. Sharing context history in mobile, context-aware trails-based applications. **COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP**, [S.l.], v. 577, p. 64, 2005.

STAHL, C.; HECKMANN, D. Using semantic web technology for ubiquitous hybrid location modeling. In: WORKSHOP ON UBIQUITOUS GIS IN CONJUNCTION WITH 12TH INTERNATIONAL CONFERENCE ON GEOINFORMATICS, GAVLE, 1., 2004. **Anais...** [S.l.: s.n.], 2004.

STUDER, R.; BENJAMINS, V.; FENSEL, D. Knowledge engineering: principles and methods. **Data & Knowledge Engineering**, [S.l.], v. 25, n. 1-2, p. 161–197, mar 1998.

VESIN, B.; IVANOVIC, M.; BUDIMAC, Z.; PRIBELA, I. Mile-Multifunctional Integrated Learning Environment. In: IADIS MULTI CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS MCCSIS 2008, AMSTERDAM, NETHERLANDS, 2008. **Anais...** [S.l.: s.n.], 2008. p. 104–108.

VESIN, B.; IVANOVIC, M.; KLASNJA-MILICEVIC, A.; BUDIMAC, Z. Ontology-based architecture with recommendation strategy in java tutoring system. **COMSIS Journal**, [S.l.], v. 10, n. 1, p. 237–261, 2013.

VIANNA, H. D.; BARBOSA, J. L. V. A Model for Ubiquitous Care of Noncommunicable Diseases. **IEEE Journal of Biomedical and Health Informatics**, [S.l.], v. 18, n. 5, p. 1597–1606, sep 2014.

WEISER, M. The computer for the 21 st century. **SIGMOBILE Mob. Comput. Commun. Rev.**, [S.l.], v. 3, n. 3, p. 3–11, jul 1999.

WIEDMANN, T.; BARBOSA, J. L. V.; RIGO, S. J.; BARBOSA, D. N. F. RecSim: a model for learning objects recommendation using similarity of sessions. , [S.l.], v. 22, n. 8, p. 1175–1200, aug 2016.
`http://www.jucs.org/jucs_22_8/recsim_a_model_for`.

YAGER, R. R. Quantifier guided aggregation using OWA operators. **International Journal of Intelligent Systems**, [S.l.], v. 11, n. 1, p. 49–73, dec 1998.

ZANKER, M.; JESSENITSCHNIG, M.; SCHMID, W. Preference reasoning with soft constraints in constraint-based recommender systems. **Constraints**, [S.l.], v. 15, n. 4, p. 574–595, jul 2010.

ZHUHADAR, L.; NASRAOUI, O. Semantic Information Retrieval for Personalized E-Learning. In: IEEE INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE, 2008., 2008. **Anais. . .** Institute of Electrical & Electronics Engineers (IEEE), 2008.

ZHUHADAR, L.; NASRAOUI, O.; WYATT, R. Dual Representation of the Semantic User Profile for Personalized Web Search in an Evolving Domain. In: AAAI SPRING SYMPOSIUM: SOCIAL SEMANTIC WEB: WHERE WEB 2.0 MEETS WEB 3.0, 2009. **Anais. . .** [S.l.: s.n.], 2009. p. 84–89.