Università degli studi di MODENA e REGGIO EMILIA
Facoltà di Ingegneria "Enzo Ferrari"

Corso di Laurea Specialistica in Ingegneria Informatica

A SEMANTIC RECOMMENDATION SYSTEM FOR MOBILE DEVICES: DESIGN AND DEVELOPMENT

UN RECOMMENDATION SYSTEM SEMANTICO PER DISPOSITIVI MOBILI: DESIGN E SVILUPPO

Relatore:
Chiar.mo Prof.
Sonia Bergamaschi

Candidato:
Elena Parmiggiani

Correlatore:
Prof. Monica Divitini
Basit Ahmed Khan

Anno Accademico 2009/2010

Keywords:
Ontology
MobileLearning
ContextAwareness
RecommendationSystem

Contents

List of Figures

Italian abstract

Il presente lavoro di tesi è stato realizzato presso il *Department of Computer and Information Science*, parte della *Norwegian University of Science and Technology* (NTNU) di Trondheim, in Norvegia, e si colloca all'interno del progetto FABULA[1].

Scopo di FABULA è trovare e sviluppare soluzioni innovative per sistemi mobili di *e-learning*, fornendo supporto per l'apprendimento informale basato su attività che richiedono la collaborazione tra gli utenti che si incontrano in modo casuale e imprevisto (*serendipitous interaction*) all'interno di un ambiente di tipo cittadino. In questo modo, il contesto in cui l'utente e l'applicazione sono immersi acquisisce un ruolo da protagonista per il processo di apprendimento (*situated learning*). Di conseguenza, l'applicazione deve essere progettata come *context aware*.

In generale, la progettazione di un framework di tipo *context aware* consiste di quattro passaggi:

1. in primo luogo, le informazioni riguardanti il contesto e provenienti da sorgenti eterogenee devono essere raccolte e convertite in un formato comune;

2. in secondo luogo, le informazioni devono essere strutturate attraverso una rappresentazione generale, ad esempio con l'uso di una o più ontologie;

3. successivamente, i dati devono essere filtrati in base a regole specifiche al fine di decidere cosa è rilevante in un determinato contesto;

4. infine, i contenuti ottenuti grazie ai tre passaggi precedenti devono essere presentati o consigliati (*recommended*) all'utente.

In questa tesi vengono proposti approcci per realizzare il secondo e il quarto passaggio. I principali scopi, pertanto, consistono in:

(a) utilizzare tecniche di natura semantica, e in particolare le ontologie, per ottenere un modello del contesto per un'applicazione come FABULA;

(b) giustificare e sviluppare un sistema di *recommendation* (in inglese, *Recommender System*, da qui in avanti RS) di contenuti rilevanti all'utente, attraverso il design e la realizzazione di un algoritmo specifico.

Un'accurata analisi dello stato dell'arte presente in letteratura è stata necessaria a monte della fase di progettazione.

[1] www.fabula.idi.ntnu.no

Per quanto riguarda la modellazione del contesto, la defizione basilare è quella fornita da Dey e Abowd [10], secondo cui il contesto è l'insieme di ogni informazione che possa essere usata per caratterizzare una situazione: le categorie più sfruttate sono il luogo, l'identità e lo stato dell'utente. Successivamente, è stata realizzata una dissertazione basata su punti di vista tipici di diverse discipline riguardo alle possibilità di dividere il contesto in categorie predefinite. Tuttavia, un problema pendente rimane quello dell'assegnazione di un valore di rilevanza, o un peso, alle informazioni provenienti dal contesto. Secondo autori come Dourish [11], è impossibile assegnare una rilevaza a priori, ma l'importanza di ogni elemento viene stabilita durante l'interazione tra l'utente e il mondo circostante. Quindi, come modellare un contesto in continua evoluzione? Dopo l'analisi di diversi approcci, si è giunti alla conclusione che un modello basato su un ontologia in OWL-DL [14] sia la soluzione migliore, in quanto fornisce la possibilità di rappresentare il contesto come un tipo di conoscenza e di attuare un processo di *reasoning* grazie alle logiche descrittive. Le categorie, o dimensioni, del constesto individuate per gli scopi di questo lavoro sono: l'utente, i luoghi, le attività e i servizi.

In particolare, per rappresentare le informazioni riguardanti l'utente, è stata realizzata un'ontolgia specifica, collegata a quella principale, estentendo l'ontologia FOAF[2] (Friend-Of-A-Friend), e basata sulla divisione dei dati tra ciò che proviene da un profilo esplicitamente realizzato dall'utente su FABULA e su quanto implicitamente raccolto: le sorgenti sfruttate per quest'ultimo tipo di dati sono composte principalmente dalla storia di quanto fatto dall'utente quando è online (attività svolte, luoghi visitati) e da un suo eventuale profilo su un Social Network. A scopo di esempio, è stato preso in esame Facebook.

La rappresentazione dell'informazione riguardante i luoghi è partita invece dall'infrastruttura teorica proposta da Rossitto [7]: qui un luogo viene introdotto come *una nozione che descrive un dato ambiente attraverso le esperienze e le attività delle persone al suo interno.* Pertanto, ogni ambiente ha sia una denotazione fisica, che corrisponde alla sua descrizione meramente geografica (e in questo caso viene denominato spazio, o *space*), sia una connotazione (detta appunto luogo, o *place*) che cambia da persona a persona ed emerge dalle attività, dagli interessi e dalle esperienze del singolo. In altre parole, a partire da uno spazio geografico comune, una sovrastruttura viene costruita utilizzando le informazioni esplicite ed implicite riguardo all'utente. Al fine di fornire una struttura all'ambiente fisico, la città è stata suddivisa in cosiddette *learning places*, ossia aree (rappresentabili da rettangoli le cui coordinate degli spigoli sono predefinite) all'interno delle quali sono identificate una serie di *learning opportunities*, o opportunità di apprendimento (musei, monumenti, o luoghi di interesse generale), che vengono presentate e suggerite all'utente in base ai suoi interessi e alla sua attività all'interno della città. Sempre secondo Rossitto [7], ci sono vari modelli concettuali con cui rappresentare le connotazioni assunte da un luogo: nel caso di FABULA, si è deciso di pensare al significato di un luogo (sia una *learning place* sia una *learning opportunity*) come all'intersezione di tre livelli: storico (la storia

[2]http://xmlns.com/foaf/spec/

passata di quel luogo), culturale (le possibilità di apprendere nozioni in genere riguardo al luogo) e sociale (la possibilità di stabilire nuove relazioni sociali). Ad ogni luogo viene inizialmente assegnato un voto per rappresentare il grado di interesse per ciascuno di questi livelli. Poi, dopo averlo visitato, anche l'utente può lasciare un proprio giudizio.

La categoria delle attività è strettamente connessa ai luoghi, in quanto un attività viene resa disponibile in una determinata *learning place* o in prossimità di una *learning opportunity*. Le attività sono costituite da una sequenza di *task*, o compiti, che l'utente deve portare a termine. Possono essere solitarie (si pensi, ad esempio, a un quiz a risposta multipla con domande di carattere generale sulla cattedrale di Trondheim), oppure di gruppo (classico è l'esempio del gioco della caccia al tesoro). Per quest'ultimo tipo di attività sono necessari due o più utenti interessati, e durante lo svolgimento viene formata una *social configuration*, cioè si viene a costituire un gruppo di utenti online che costruiscono in modo imprevisto una rete di interazioni: è la dimostrazione di una struttura sociale che nasce da un'attività di apprendimento. Anche alle attività viene assegnato un grado di interesse dal punto di vista storico, culturale e sociale.

I servizi, infine, vengono resi disponibili durante lo svolgimento di una data attività in un determinato luogo.

Tale struttura è stata prima elaborata con un meta-modello realizzato tramite un *class diagram*, e poi tradotta in ontologia con l'ausilio di Protégé[3]. Si veda l'appendice A per una dettagliata descrizione di questo software.

Anche al fine di realizzare un RS per presentare all'utente contenuti rilevanti è stato necessario partire da un'accurata analisi dello stato dell'arte. Sono stati analizzati i pregi e i difetti dei diversi tipi di RS esistenti e si è cercato di combinarne gli aspetti positivi. Un algoritmo è stato progettato per consigliare all'utente un elemento di una delle quattro categorie individuate: un amico, un luogo, un'attività o un servizio. La parte riguardante la raccomandazione di un amico è stata implementata in Java al fine di portare a termine una serie di test e valutare l'impatto delle soluzioni adottate.

La racommandazione di un amico u_a ad un utente u_b è stata realizzata attraverso il confronto tra le informazioni riguardanti u_a e u_b. Tali informazioni sono memorizzate in formati e sorgenti diversi, per cui è stato prima necessario radunarle e rappresentarle in un formato confrontabile. Si tratta del profilo esplicitamente creato dall'utente su FABULA, dei log delle attività svolte e dei luoghi visitati e dell'eventuale profilo su Facebook. La fase di confronto utilizza tecniche sia di natura statistica, sia di natura semantica: in questo secondo caso viene considerata la similarità di significato tra i termini raccolti analizzata per mezzo di WordNet[4] (si veda l'appendice A per una descrizione di WordNet e delle misure di similarità disponbili in letteratura). Da subito è apparso evidente come fosse necessario assegnare un peso, o un livello di affidabilità, agli utenti al momento di confrontare i dati provenienti dal loro profilo Facebook: è stato

[3] http://protege.stanford.edu/

[4] wordnet.princeton.edu/

così introdotto un meccanismo di penalizzazione (all'aumentare del fattore di penalizzazione, diminuisce l'affidabilità di un utente) basato su cinque livelli per utenti che siano stati o troppo o troppo poco attivi, se confrontati con il comportamento medio degli utenti FABULA su Facebook. Il comportamento medio è stato rappresentato come la media degli elementi di diverso tipo che gli utenti hanno: il computo è stato realizzato dal punto di vista dell'utente per cui la raccomandazione viene fatta, pertanto dipende dai privilegi di accesso che questi ha verso i profili degli altri utenti. I gradi di scostamento dal comportamento medio sono stati ottenuti per mezzo della deviazione standard.

Il suggerimento di un luogo o un'attività avviene invece con lo stesso tipo di procedura, poichè i profili di attività e luoghi presentano formalmente la stessa struttura: una descrizione in testo libero, una serie di *tag* utlizzati per etichettare un elemento e un voto al livello di interesse di quell'elemento da un punto di vista culturale, sociale e storico. Per questo scopo è stato fatto ampio uso di tecniche di *information retrieval*. I profili degli elementi costituiscono i documenti, mentre quelli degli utenti sono le query. La racommandazione è ottenuta con due metodi:

- con il primo metodo, i documenti (siano le attività oppure i luoghi) vengono clusterizzati in base al loro grado di similarità. All'utente vengono poi raccomandati gli elementi che si trovano negli stessi cluster di quelli che sono già presenti nella sua storia passata su FABULA. Un peso maggiore viene assegnato agli elementi che si trovano nei cluster in cui vi è una maggiore presenza di elementi appartenenti alla storia dell'utente;

- l'altro metodo è basato sul computo della similarità di tali documenti con la query: in questo caso si utilizza il *vector model*. In questo caso, un peso ad ogni elemento viene assegnato utilizzando l'opinione di un set di *vicini* u_k all'utente u_a per il quale viene fatta la raccomandazione, identificati come gli utenti a lui più simili trovati con la procedura descritta in precedenza per consigliare una nuova amicizia. Tale opinione consiste nel giudizio dato da u_k al grado di interesse sociale, storico e culturale di un elemento, moltiplicato per il grado di correlazione di u_k con u_a e per il grado di similarità tra il documento e u_a.

Da ultima, la raccomandazione di un servizio discende direttamente da quella di luoghi e attività, in quanto, come detto, un servizio è disponibile nel corso di un'attività e solo in determinati luoghi.

Questo lavoro è in gran parte formato da una fase di progettazione. Tuttavia, una serie di test è stata portata avanti per permettere la valutazione delle scelte fatte nella realizzazione della parte di algoritmo che si occupa di raccomandare nuovi amici. Un dataset è stato costruito da zero per l'occasione, formato da diciassette profili di utenti su FABULA, undici dei quali aventi anche un profilo su Facebook, e di profili di attività e luoghi.

A livello implementativo è stato utilizzato il linguaggio Java. L'interfaccia con l'ontologia di contesto è stata realizzata tramite il software Jena[5] (si veda

[5]http://jena.sourceforge.net/

l'appendice A per una descrizione del software). Data la natura eterogenea (file di testo, file XML, tabelle di database, etc.) delle sorgenti, è stato inotre necessario realizzare un set di funzioni per convertire i dati in un formato comune. La raccolta dei dati dal profilo Facebook ha richiesto invece una gestione separata e la creazione di un'interfaccia specifica basata sulle *GraphAPI*[6].

Successivamente, in fase di test, si è visto come:

- aggiungere il confronto dei dati presi da Facebook, quando disponibili, aumenta la correlazione media tra gli utenti dell'11%, raggiungendo picchi del 41%;

- introdurre un fattore di penalizzazione su cinque livelli per un utente basandosi su quanto questi si discosta dal comportamento generale degli utenti del sistema riduce la correlazione media tra utenti del 1.90%. Tuttavia, in questo caso è necessario analizzare i sigoli valori e quello che emerge è che vi sono poche percentuali molto alte, che vengono abbassate quando considerate in media per la presenza di molti valori nulli;

- il computo della correlazione tra le informazioni basata sul significato dei termini aumenta la correlazione media quasi del 7%, in quanto tale tecnica consente di reperire concetti simili (si pensi ad esempio a due utenti che rispettivamente indicano di essere interessati a *nuoto* e a *piscina*) la cui similarità non sarebbe però individuata con un confronto semplicemente lessicale.

Molti sono i temi interessanti che si sono manifestati nel corso di questo lavoro e che meriterebbero maggiore approfondimento. In futuro, sarebbe interessante investigare diverse metodologie per affrontare gli stesssi problemi. Per quanto concerne la modellazione del contesto, sarebbe interessante ricercare un'ontologia specifica per descrivere l'ambiente come sovrapposizione di spazio e luogo, atta ad estendere l'ontologia di base allo stesso modo di quella realizzata per l'utente.

Riguardo invece alla realizzazione del sistema di raccomandazione, si potrebbero sperimentare approcci diversi nei seguenti aspetti:

- interfacce con altri Social Network contententi informazioni di carattere diverso rispetto a Facebook;

- differenti misure della similarità semantica tra concetti, che facciano ad esempio uso del loro contenuto informativo piuttosto che della sola distanza nella gerarchia `is-a` di WordNet;

- per quanto concerne il processo di penalizzazione degli utenti, i risultati non si possono definire completamente soddisfacenti: il metodo qui proposto può essere modificato, ad esempio adattando un algoritmo come il Clarke Tax [16], il quale è pensato per assegnare una tassa all'utente in base a quanto la sua presenza influisce sul resto del gruppo.

[6] http://developers.facebook.com

In conclusione, è bene ricordare come questa algoritmo sia stato realizzato in modo volutamente indipendente dalla piattaforma. Essendo comunque l'architettura di FABULA basata su un sistema multi-agente, in futuro è auspicabile la realizzazione di un *recommender agent* specifico che implementi l'algoritmo progettato per il presente lavoro.

CHAPTER 1

Introduction

1.1 Motivation

In recent years, the development of wireless networks has made it possible to design e-learning systems able to take into account both pedagogical and technical aspects and, in addition, to actively participate to the learning process carried on by a user. Individual users have new possibilities to learn from each other and from the environment in which they are located. This leads us to the notion of *situated learning*, which has been introduced in [23] as the outcome of interaction with the social and the physical environment.

First of all, in order to understand what the environment is, an accurate modeling of the context surrounding the user and the interactive application is required. It is particularly important for those mobile handheld applications where the user's context is changing rapidly. In addition, it is fundamental to notice that a good mobile application cannot ask the users to provide the necessary information: it is likely that most users do not know which information is potentially relevant.

In order to achieve this result, the application should be designed as *context aware*. The most important task therefore deals with understanding what pieces of information need to be collected in order to properly design the situation where both the application and the user are moving. Many approaches in literature have been proposed to solve the problem of context representation. Thus, given a theoretical background, it is necessary to apply the ideas to a practical design of the context that can be used as a source for the application to chose the most appropriate behavior.

Once a design of the current context is made available, such semantically complex representations need thus to be presented to the user in a way that suits his interests, needs and purposes. That is the reason why a Recommender System (RS) is required for suggesting content and items to the user, who otherwise would not be able to discover by himself through the classical search techniques [39].

The consequence of these requirements is the collection of large amount of data, whose meaning has to be understood and exploited by an efficient

Information Retrieval (IR) system: in fact, Semantic Web technologies act as the best tool to define a representation and a proper selection of data [39].

Context becomes then both the source to chose which categories are to recommend and the ground where pieces of information are retrieved and whose importance is decided based on the ongoing situation surrounding the user.

1.2 Contribution

In general, the design of a context-aware framework involves at least four steps:

1. At first, raw information coming from different heterogeneous sources in the surrounding context has to be collected and turned into a common format. Usually, wrappers are used for this purpose;
2. On a second stage, this information has to be gathered and processed into a general representation, e.g. by means of one or more template ontologies;
3. On a third level, contextual information has to be filtered according to specific rules aimed to decide if a particular piece of information is necessary or interesting for the user;
4. Finally, the content resulting from the previous step has to be presented or recommended to the user.

In this work, I will present approaches to deal with the second and the fourth point.

The two main goals are:

- to exploit ontologies and semantic technologies in order to obtain a model of context for a mobile learning application;
- to justify and develop a set of semantic-based context-aware recommendation services to suggest relevant pieces of information to a user. An algorithm has been written for this purpose.

An overall ontology-based framework has been realized to represent the different categories, or dimensions, of the context. Out of the several and sometimes striking approaches found in literature, the dimensions identified are: the users, the places, the activities that a user can be engaged in and the services available. In order to describe the user, an inner ontology was developed extending the FOAF (Friend-Of-A-Friend) vocabulary[1] and grounded on the division of the information available between what

[1] http://xmlns.com/foaf/spec/

is explicitly stated by the user and what is implicitly retrievable: this latter category was to be carefully investigated and the sources exploited were mainly based on the history of the user's behavior and his profile (if present) on a Social Network. Facebook[2] was identified as the most widely used one and used as an example in this work.

Afterwards, a new algorithm has been designed and implemented in order to recommend to the user an item belonging to one of the identified dimensions of context. The novelty of this approach lays in the generality and reusability for different application fields of the methods developed and in the combination of solutions belonging to content-based and collaborative filtering RS available in literature.

When matching the pieces of information collected from the explicit and implicit sources about the user, useful ideas found in literature have been adapted and mixed together. First of all, the content coming from a user's profile is matched with the one from another user's profile both statistically and semantically. The semantic analysis of word similarity has been carried on by means of WordNet[3], a powerful tool to browse the relationships between concepts. The statistical analysis was mainly keyword-based, aimed to compute how many items the two users have in common out of the total amount of items each of them had.

A way to add a weight to such matching values is then proposed, in the form of different levels of penalization for a user having a behavior that is too different from that of the other users of the application: it is the case of a user having too many or too few items if compared to the other users, which means that he has been too much or too little present online. The general behavior of the system is represented with the mean value of items the users have, and the penalty factor assigned to a user is then directly proportional to his distance from the mean value, based on the standard deviation.

The different levels of penalization can be looked at as a way to represent the trustworthiness of a user inside the FABULA Social Network.

Different tools are thus used to recommend a place or an activity to a user, by exploiting IR techniques: two ways (a cluster-based one and a similarity-based one) are adopted to obtain a recommendation, aimed at combining solutions typical of different RS and at overtaking their main drawbacks.

The last dimensions of context, services, has been left as the last one because it has been ontologically designed as the result of two different pieces of information: the activity that requires a given service and the place where the service is available.

To conclude, in this work I decided not to deal with the still open issue of relevance assignment to sources of context. I will assume that a *relevance engine* is made available inside the FABULA system, aimed to compute the

[2] www.facebook.com

[3] wordnet.princeton.edu/

weight, or the importance, of each type of information used to obtain the recommendation values.

1.3 The FABULA project

This work has been carried on at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, as part of the FABULA[4] project. FABULA stands for *FremrAgende By for Undervisning og LAering* (*Seamless networks for transforming the city into an arena for learning*).

Figure 1.1: The FABULA logo

Aim of the project is to construct a (web-) services based e-learning system for mobile networks, and to support informal collaborative mobile learning activities (i.e. it supports learning which happens inside a learning group whose members can communicate with each other) in a city wide context. The target is a collection of individual users who carry sophisticated mobile devices with them. Such a system is intelligent because it is aware of the context of the learner and adopts its behavior accordingly. Intelligence is built over ontological reasoning and semantic interoperation. Moreover, FABULA is able to recommend learning content to the user and to tailor different learning activities based on the information collected about the context.

1.4 Results overview

The work done for this thesis is made of two bigger parts: the design of a model of context for a mobile learning application and the realization of a recommendation algorithm based on this model. The main focus is set on the design phase, but a series of tests has been carried on so to evaluate the impact of the choices made when realizing the algorithm to recommend a friend to a user of the FABULA application. In particular, we will see that:

– retrieving data from a Social Network (Facebook in this case) increases the average correlation of more than 10%, reaching peaks of around 40%;

[4] www.fabula.idi.ntnu.no

- adding a semantic matching procedure to compute the correlation among the profile of two users improves the correlation up to the 5%;
- having a penalization factor based on how many items the user has in Facebook if compared to the average behavior of the FABULA users decreases the matching value of 1.9% on average, but in this case, owing to the nature of this procedure, the effect must be investigated for every single case.

1.5 Report outline

The structure of the present thesis is the following: chapter 2 will provide the reader with an overview of the background where the project has been developed, i.e. the description of the working settings and of the concepts at the core of the choices made. The architecture of the FABULA application is also presented. In chapter 3, a discussion about the related work in the field of context awareness and of context representation on the one hand and of the approaches to realizing a recommendation system on the other hand is undertaken. The theories and the methods identified here will then be used and revised in chapter 4 in order to design a model for the context of the application, and in chapter 5, where a description of how the algorithm at the heart of the recommendation system developed is given. Chapter 6 will finally give an evaluation of the test phase: after giving an example scenario, the impact of the solution adopted is analyzed. Chapter 7 contains the conclusions about the work done and the suggestions for the future work.

CHAPTER 2

FABULA: an overview

2.1 Working scenario

The FABULA project considers a learning process that takes place in a city-wide contextual space. The city used as a setting is Trondheim, Norway. A wireless network is available indoor and outdoor in the city center and in the most important tourist attractions.

A dataset of seventeen users, sixteen places in Trondheim and a tenth of activities has been created to test the algorithm developed.

The users Seventeen users of different ages and nationalities and with different purposes are walking around Trondheim. Each of them is carrying a smartphone with the FABULA application running. They all have a profile in the FABULA system, where they explicitly inserted some information about them: name, sex, age, home country, e-mail address, languages spoken, level of education, lists of interests, etc. They also set a few preferences about whether they want to be visible to other users, to receive recommendations about other people or items, to be sent communications, etc. For each smartphone, a user agent is running, identified by a unique AID and sending information about the current geographical position of the user. Each user has already become friend with some other users: more active users already have a list of four or five friends inside the FABULA network.

Eleven users have also a profile on Facebook, the most widely used social network nowadays, and granted FABULA the permission to access their personal information stored there. These users have been all active on Facebook, and have already set a list of interests that can be added to the ones available in the FABULA profile to understand what their hobbies are and what interests they have. They have also tagged a list of items as interesting (by choosing the *I like it* option on an item's page) and have become friends with other people inside the Facebook social network. Among their friends on Facebook, there might be some FABULA member, too.

Each user has been a member of FABULA since some time, so lists of the activities he has been involved in and the places he has visited are stored together with the information coming from Facebook in order to build an implicit profile.

All of the FABULA and the Facebook profiles where created from scratch, by simulating the behavior of real users.

The places Sixteen interesting places have been identified inside the city of Trondheim where the users might be involved in a learning activity. Owing to its university, Trondheim is a city with one fifth of the population made by students, so many attractions deal with the university and the student's social life: for example the student society's building Studentersamfundet, the main building of the Norwegian University of Science and Technology, and so on. Other types of places include the cathedral, botanical gardens, etc. Figure 2.1 shows a map of the city center of Trondheim. Each place's profile is filled with a description in free text, a set of geographical coordinates bounding the area where the attraction is located and a rating for each of the levels of interest of a place from the historical, social and cultural point of view. Each place is also labeled with one or more tags, that can be either pre-defined or user-defined.

The activities A set of activities also is made available for the users. Each activity is associated with a place and is made of a collection of tasks that have to be accomplished in order to complete the activity. As for the places, each activity is given a value ranging from 1 to 10 to describe the level of interest from a social, an historical and a cultural point of view. Tagging is available also for the activities.

2.2 Core concepts

2.2.1 Spaces and places

As I wrote in the introduction, FABULA is strongly tied to the notion of *situated learning*, introduced in [23] as the outcome of interaction with the social and the physical environment: learning is situated in a given context and the acquisition of knowledge by a user is strongly intertwined to that specific context.

As far as the physical environment is concerned, the FABULA strongly refers to the theory described in [7]: in her thesis, Rossitto introduces *place as a notion that describes a given environment by encompassing people's experiences and activities within it.* Each environment has therefore both a physical description, corresponding to the geographical information about it (it is a *space*), and it is also invested with a meaning that is different from

Figure 2.1: Map of the central area of Trondheim

user to user, encompassing and emerging from his activities, experiences and interests. This latter lens through which we look at a space is called *place*.

Such a theoretical background is fundamental for an application like FABULA, as the city is both the subject and the object of the interaction, because the learning carried on by the user is about the city and takes place inside the city. The city, if intended as a *space*, has to be invested with a meaning (it becomes a *place*), based on the pieces of information collected about each user to be considered. That is to say that, out of a common geographical space, a superstructure is built, the place, by using the user's explicit and implicit profiles available for the single user as the bricks to construct it.

The model of context designed for this thesis tries to give an application to these concepts.

2.2.2 Collaborative learning and serendipitous interaction

The main purpose of the FABULA project is to find and develop novel solutions for promoting a city-wide learning process inside a mobile network. To achieve this result, social collaboration has a key role. Forms of collaboration between peers start in a spontaneous way and might evolve towards more complex group-based ones. From the perspective of the social environment, the application should be able to foster *serendipitous interaction* (i.e., occurring in an unplanned and unexpected way) among individual users [5] who carry sophisticated mobile devices with them. Awareness about each other's surrounding context and the proper representation of this acquired knowledge is the mechanism to successfully find new collaborators and therefore enhance the learning process.

In their paper about the role of technology in city-wide collaborative learning, Canova Calori and Divitini [20] identify the contribution of a mobile application as a key in supporting, in particular, the performance of shared tasks and the social matching and networking. They provide scenarios to enhance their thesis. For the first point, it is highlighted how a city can become the location for carrying out a shared task among a group, or a social configuration, of people that are distributed. In the second case, it is demonstrated how the social aspects of collaborative learning in a city-wide scenario can be enriched by either reinforcing the sensation of connectedness among friends and by *promoting interaction between strangers based on affinity* [20]. The recommendation system developed as part of this work also aims to make a contribution to this last aspect.

2.3 The FABULA architecture

In the landscape of existing e-learning platforms, it is easy to identify some common aspects [24]:

- Services are always layered as in a stack and each of them provides support to the upper level;
- A digital repository is used to store the result of the learning process;
- The users interacts directly with fine grained application services at higher levels of the stack;
- Common services provide essential functionalities that the user is not exposed to;
- The infrastructure consists of the backbone services dealing with HTTP, XML, SOAP, etc.

However, such system are mainly focused on content delivery, and are hardly meant to support informal learning. Two important features need to be added [24]:

1. In most situations, the user is unaware of the learning possibilities surrounding him/her: therefore, there is a need for proactive behavior of the learning system;
2. Most of the frameworks do not provide sufficient support for real time collaboration among users and situatedness of the learning activity;

The project FABULA tries to gather all of the listed characteristics. From a general point of view, FABULA system can be divided into two main sub-parts:

1. Service architecture, dealing with the services/functionalities provided by the system;
2. Multi-agent system, responsible for the active role of the system during the learning process.

2.3.1 FABULA Service Architecture

The FABULA system uses services as the basic building blocks. Those services are divided into logical layers. This layering is based on the functionalities of the services and helps to separate them based on their role during the system execution. There are two points of view to look at the system:

- Passive services perspective: it considers the system as a collection of services; thus, a passive service is a functional unit of the system, which is described, published, searched and invoked on demand;

- Active services perspective: a passive service becomes active when it is hosted by a software agent, that controls when the service(s) is(are) accessed; each agent is autonomous and may be proactive, and cooperate with other agents in order to increase the learning outcome. FABULA multi-agent system (F-MAS) operates at all layers previously mentioned.

Passive perspective Services are layered into three different levels, from the highest to the lowest:

- Application specific learning services layer: this layer includes the services that allow the user to edit, update, manipulate learning artifacts and that are dependent on the learning applications. Categories of services in this layer are:
  - Authoring support services;
  - Collaboration support services (to support performance of shared tasks, social networking, etc.);
  - Communication services;
  - Application specific services (they can be added based on the content of application).
- Basic learning services layer: this layer uses the functionalities of the services in the lower level and supports those on the upper level at run time by using Service Access Points (SAP). Services at this layer maintain heterogeneity and autonomy in the system. Categories are:
  - Community management services (meant to manage different structures of user communities, for example recommendation about social connection);
  - Application composition services;
  - User management services;
  - Event management services (necessary abstraction to deal with learning applications).
- Resource management service layer: this level handle the data stored in and accessed from the repositories. Categories of services are:
  - Discovery services (provide information about services at the upper level);
  - Security services (identity and permission management of user and its agent);
  - Information retrieval services;
  - Metadata and semantic annotation services (they work over the ontology repository and are needed for the semantic reasoning with the concepts of the FABULA ontology).

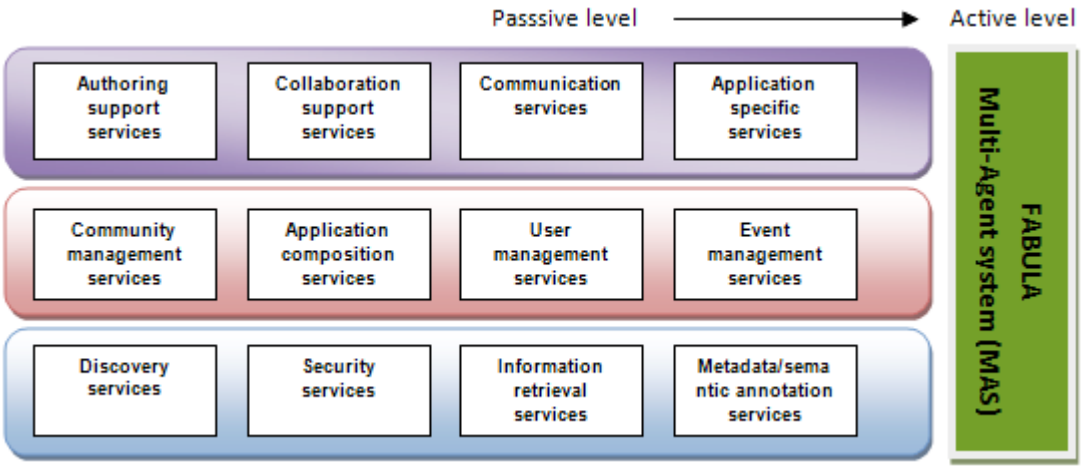

Figure 2.2: The FABULA system architecture [24]

Active perspective: the AGORA framework AGORA (Agent Oriented Resource mAnagement) is a multi-agent framework, which is general enough to be adoptable to a number of situation and domains [24] [25]. Furthermore, it is extendable: it provides the basic functionalities which can be enriched using low level tools if needed. In addition, AGORA framework allows software agents to understand both FIPA (Foundation for Intelligent Physical Agent) and W3C standards, so to avoid gateway translators that may become a bottleneck and a single point of failure [25]. AGORA framework is build upon the JADE middleware, a fully FIPA compliant agent development environment. AGORA framework makes it possible to simplify the modeling of all the dynamic behaviors of agents at execution time, including communication, coordination and negotiation. For each one of these acts, a cooperative work point is mapped into an AGORA node. This node can be considered as a database containing information about locators of AGORA Managers and descriptions and locators of services [25]. Inside of an AGORA node we can find three default agents (AGORA Managers):

- the Manager Agent, responsible for registration and un-registration of other AGORAs and agents (Registered Agents), for maintaining the information about the node up to date and for providing access to the AGORA services; it is a model-based reflex agent;
- the Coordinator Agent, which ensures that all participants of the activity follow the rules prescribed in the workflow;
- the Negotiator Agent, which implements logic of conflict resolution, using by default the Contract-Net protocol. Both the Negotiator Agent and the Coordinator Agent are goal-oriented proactive agents.

Registered Agents are thus the external agents that want to use the functionalities provided by the AGORA and that communicate with the AGORA

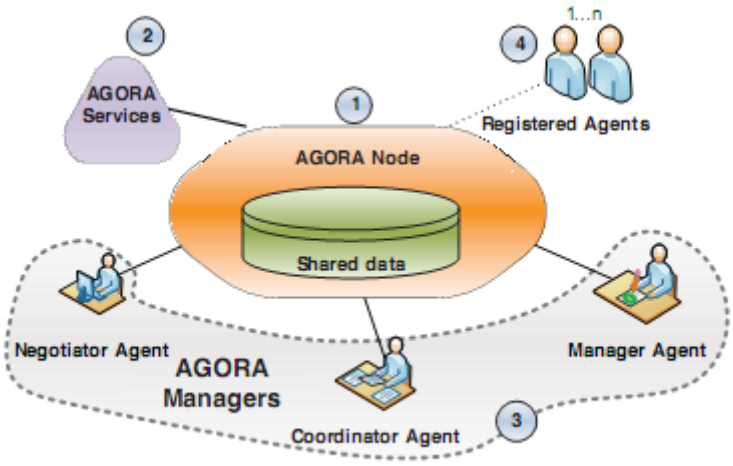

Figure 2.3: Scheme of an AGORA NODE [25]

Managers through message passing. They are only requested to understand ACL or SOAP protocols. AGORA nodes can be combined together and each AGORA can create a child AGORA, by following an incremental fashion and building a parent-child graph. Parent-child relationships support either a static approach to search resources or a peer-to-peer one, where each AGORA node is a peer in the system. Any AGORA can register itself to any other AGORA as required. As a result, all the services and functionalities of agents in the system are mapped to a respective AGORA node.

CHAPTER 3

Related work

3.1 Background theories on context modeling

Many mainstream researchers in the human-computer interaction (HCI) field have been exploring the forms of interactions that can be achieved when a computer technology is integrated in the user's everyday life. This kind of research has been labeled with many different names, from ubiquitous computing to context-aware computing, from pervasive computing to embodied interaction, and so forth. While examining a number of approaches to context-aware system design, Chalmers [6] presents an extreme view aimed to distinguish what he calls *context-aware and ubiquitous computing* from *computer supported collaborative learning* (CSCW). On the one hand, in this paper, CSCW is said to be focused on the intersubjective aspects of context, constructed in and through the dynamic of each individual's social interaction; on the other hand, context-aware and ubiquitous computing is presented as concentrated on the computational representation of context by combining different senses and media, rather than the social construction of context through interaction.

In order to better understand what context should the application be aware of, the first step is giving a definition of context. Many researchers in literature have attempted to do so, mainly by enumerating examples or synonyms. Some consider context to be the user's environment, others consider it to be the application environment, making the mistake to think of the application and the user as two disconnected worlds. A good definition is the one by Dey and Abowd [10]:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*

As a result, if a piece of information can be used to characterize the situation of a participant in an interaction, then that information is context. As the

two authors state, context is typically made up of the location, the identity and the state of the people involved and computational and physical objects.

3.1.1 Categories of context

From a practical point of view, we now need to classify the information that make up the context. It is interesting to underline the so-called *pragmatic* approach presented by Ekbia and Maguitman [12] and Dewey [9]. They provide a notion of context divided into two components:

- *Background*, both spatial and temporal, which is ubiquitous in all thinking: it does not come into explicit purview and is taken for granted. On one hand, the spatial aspect has to do with the contemporary aspects in which the interaction emerges; the temporal aspect, on the other hand, is intellectual as well as existential. The former deals with the social and cultural settings where the individual is. The latter is *part of the material means that contribute to the possibility of a thought process*;
- *Selective interest*, which conditions the subject matter of thinking.

As a consequence, the main pieces of the background puzzle are time and space. What the application senses is a state at a given time in a given space. According to Zibetti and Tijus [41], a state includes a temporal location and is generally defined as the set of static properties of the external and physical objects of a given situation. Bradley and Dunlop [2] propose a multidisciplinary approach to investigate the contextual information. They argue that, in order to maximize usability, users and their heterogeneous interactions need to be placed at the center of a design process. In particular, the authors analyze the definitions proposed by linguistics, computer science and psychology.

The linguistic approach Let us begin with the studies in the area of linguistics: a categorization of contextual information used within a communication act according to Bunt [3] might be:

1. *Linguistic*: properties of the surrounding linguistic, textual or spoken, material;
2. *Semantic*: object properties and relations relevant to the task;
3. *Physical*: physical circumstances/environment in which the interaction takes place;
4. *Social*: type of interactive situation and user's roles;
5. *Cognitive*: the participants' beliefs, intentions, plans and other attitudes.

Connolly [8] distinguishes between the *linguistic* and *situational context*, and provides an interesting notion of *intertext*, as the concept that, to interpret a part of one text, information from some other text may be required. As a result, situational context refers to the pertinent aspects of the environment that are nontextual in nature. The task of the mobile application designer is therefore to decide what intertext to provide. The notion of context is of great interest in the area of computer science. Recurring categories to gather pieces of contextual information are:

- User's location and environment (where);
- Identities of nearby people and object and changes (who and what you are with + temporal data necessary);
- Time (when);
- User's emotional state and focus of attention (activity/who).

Dey and Abowd [10] state that there are certain types (or categories) of context that are, in practice, more important than others. These are the primary context type:

- Who's' identity;
- Where's' location;
- When's' time;
- What's' activity.

The aim of this classification is to determine why the situation is occurring (i.e., to characterize the situation), also by inferring the secondary context types, that can be indexed by primary context types because they are attributes of the entity with primary context (e.g., obtaining the email address after retrieving the identity of a person). In this way, context can be used and organized.

The psychological approach From a psychological perspective, Bradley and Dunlop report a classification of information to conceptualize what is context and what is not, based on Smith [36]:

- *Focal information*: it is directly on the focus of attention;
- *Contextual information*: it is processed outside the focus of attention;
- *Meaningful context*: it refers to verbal/semantic material (what is said and what it means);
- *Incidental context*: it just happens to be present. Perceiving some incidental objects from the environment may change the way to solve a problem.

The interactions between the user and the application provide four possible combinations of what they both know about the user's intentions, leading to four areas for four different design issues:

- *Explicit*: user's intentions known to both;
- *Inferred*: application infers (knows) info about the user's implicit intentions;
- *Implicit*: intentions known to the user, unknown to the application;
- *Hidden*: unknown to both.

The reason for this classification is that it is dangerous to center the primary focus on the application when defining context. It is necessary for the system to become aware of user's requirements. According to Dey and Abowd [10]:

> *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.*

The authors also give a categorization for features that context-aware applications may support:

1. Presentation of information and services to a user;
2. Automatic execution of a service;
3. Tagging of context to information for later retrieval.

Let us get back to the first differentiation listed between background and selective interest. If these two categories are referred to the agent (human or software), it is clear that the attention of the agent is at any time focused on a limited region of the domain knowledge, i.e. to pieces of knowledge having to do with that specific application domain. Contextual elements useful for the agent's task are only a portion of that region. In order to refer to where the attention is focused, contextual knowledge involved needs to be referable: that is where the need for a context ontology stands. It is a way to model the context in which the user and the application are moving.

Relevance Öztürk and Aamodt [34] propose a difference between internal and external context from the agent's point of view:

- *Internal context*: it represents the agent's state of mind, emerges while the agent solves a problem and captures goals, interests and information needs in the process of problem solving. In my opinion, this definition of internal context corresponds to the previously recalled categories of existential temporal background and selective interests;
- *External context*: it is the situation, that is to say the ground facts existing in a certain situation, contextual facts pertaining a given situation. It is quite similar to the mentioned notion of spatial background. It is related to the environment and the target.

Both external and internal contexts are then divided into an *interactive* and an *independent* context, based on what is directly affecting the occurring interaction and what is not. It is possible to consider meaningful context as the one that can directly interact with the user or the application, so to become the interactive context. On the other hand, interactive context has a lot in common with incidental context, as it only happens to be present, without affecting the current activity.

This definitions introduce the notions of relevance and focusing. In fact, it might be useful to assign a priority, or a relevance value, to pieces of information within the user's internal context and update those values based on the external context, because not everything that is internal knowledge is useful. The notion of relevance, from a computational point of view, results in pre-selecting from the search space the information that is either useful for certain goals, or is within certain precision limits [12]. It is possible to roughly outline the process leading to the choice of aspects of a concept to actively process. Internal context determines the user's tasks. Those tasks impose a perspective which implies the focus of attention, captured in terms of part of core-domain, external and case-based knowledge. Concepts are therefore represented as a set of features, i.e. things to take into account when performing a given task: based on the task in which a context is presented, only a particular set of features is activated. That is why it is important to assign a relevance value to features. However, objects and people pass from being relevant or irrelevant to the interaction taking place during its existence. Here stands a problem of understanding how and when they become relevant or irrelevant: maybe the classification provided by psychology might be helpful.

Ontological Commitment In addition to relevance assignment, another issue deals with the correct understanding of actions occurring in the sensed environment. The interpretation of a sensed event is constrained by the data actually available to the interpreter, the mobile application in this case. This happens for a human observer, too: he or she is lead to a certain interpretation of what is happening nearby by what he or she sees or hears. Zibetti and Tijus [41] recall the notion of *ontological commitment*: what is seen is interpreted to be an instance of the extension of a concept. Different observers of the same situation might have a same or different ontological commitment: having the same ontological commitment provide the observers with a shared interpretation of events. Obviously, the richer the contextual information is, the easier it will be to correctly interpret a given situation. Another interesting aspect in Zibetti and Tijus's work [41] resides in the classification of the internal hierarchical levels of action attribution. Tree levels are identified:

1. *The realization level*;
2. *The semantic level*;

3. *The pragmatic level.*

For instance, the act of moving one's finger (realization level) coincides with the switching on the light (semantic level) in order to alarm a burglar (pragmatic level). However, the ontological commitment is based on which types of contextual information are collected. The most important dimensions to capture, according to Zibetti and Tijus, in order to result in a shared understanding, are the temporal and the spatial contexts, that overlaps to the previously listed category of background. It is also important to notice that the higher the level of interpretation is, the more contextual information an agent needs. From another point of view, the higher the level of interpretation, the higher the risk for different commitments from two different agents.

The design process according to Bradley and Dunlop By borrowing ideas from various disciplines like linguistics, psychology and computer science, Bradley and Dunlop, in their proposed model of context [2], argue that the starting points for the design process should be:

- *Differentiation between user's and application's world*: need to distinguish what is focal to the user and to the application and what is contextual. Contextual world can be broken down into four main dimensions, common to the user and the application: task context (relationships, benefits, constraints), physical context (environmental location), social context (surrounding people), cognitive context (short and long-term memory abilities, preferences, opinions, etc making reference to a user profile and to sensed information) and temporal context (time is generally important and gives the situation a meaning, based on the application destination). Application's context is underlined as important, too: it is about the capabilities and limitations of both the application (hardware) and the sources (e.g., a web-based server) from which data are derived;
- *Separation between meaningful and incidental dimensions of context*: it is important to notice that incidental occurrences in the contextual world are normally unrelated to the user's primary high-level goal. The application must be able to decide whether it is necessary to sense also incidental (focal and contextual) information or not. Contextual data can be used to infer future user's intentions of which the user is currently unaware;
- *User and application's processes*: user's cognitive goals are continuously re-shaped by his perception of meaningful and contextual worlds (see task, physical, cognitive and temporal contexts). Goals are used to carry out focal actions. Focal interactions shape the context within which each future interaction takes place and contributes to the user's construction of future goals. It might be important to analyze focal

interactions to shape the context and the user's future goals, as user is influenced by the dimensions contextual world.

The authors consequently identify the processes related to the user and to the application.

The user process steps can be the following:

1. User has a goal that is broken down into a series of low-level goals;
2. User selects meaningful and relevant aspects of the external contextual environment perceived with his senses together with the temporal context. I.e., among all the meaningful pieces of information, he discards what is not relevant. He is influenced by external contextual environment;
3. Information interpretation through user's cognitive context, leaving a meaningful representation of the external world (compared with past acquired knowledge);
4. Decision making about the most appropriate goal to undertake a meaningful focal action;
5. The result is the formation of experience and knowledge.

As far as the application is concerned, the tools providing the necessary context abstraction are:

- *Widgets*: these are sensor abstractions (wrappers) concealing details of how sensing and interpretation of the environment occur. They provide an interface to automatically deliver information to the interested components or services. Perhaps, it may be useful to insert a parser as an intermediate stage between widget and aggregators to collect syntactic pieces of info about textual data;
- *Aggregators*: they store multiple pieces of low level info logically related and stored in a common repository;
- *Interpreters*: they are able to abstract low-level data to a higher level.

First of all, data are aggregated and logically stored under common repositories relating to the main context categories, for example: user (U), task (T), physical (P), Social (S), application (A). For each application design, it is possible to choose what types of context are actually useful. Aggregated sensed data contain past, future and present dimensions. Application's incidental world also contains a resource discovery, where tools in the environment are discovered, if inferred to be relevant, and downloaded, in order to support both meaningful or incidental app processes.

Secondly, a matrix is built:

At each stage, based on the rules derived from the past, relevant aggregated sensed data are chosen for the subsequent stage, as in a stack layering. I.e., at each stage, for each kind of application, only some dimensions of relevant

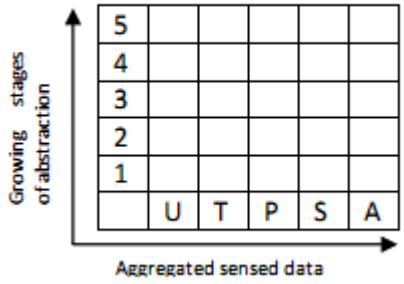

Figure 3.1: Aggregated sensed data at different levels of abstraction

aggregated data are collected, and then a path through the matrix is built, from the first stage to the last. The path through the matrix is then passed to the focal application service selection. Four forms of focal services are detected:

- Task-specific information (inferred or requested) presented to the user via user interface. This service has a lot in common with the presentation of information and services step in [10];
- Information is not presented to the user but the application executes a focal service as directed by the user (e.g., augmenting reality). Also this step overlaps the tagging of context in [10];
- The application infers information about a course of action without the user's intervention and acts consequently. See automatic execution of a service in [10];
- An application service is executed to facilitate other application services.

3.1.2 Context as a dynamic construct

In the previous section, I have analyzed some of the approaches to context modeling grounded on definitions that tend to emphasize objective features that can be tracked and recorded relatively easily, whereas they gloss over aspects of the user experience such as subjectively perceived features and the way that past experience may influence current activity. This critique was carried forward, among others, by Chalmers [6] and Greenberg [17], mainly against the definition and toolkit proposed by Dey and Abowd [10]. In his paper, in particular, Greenberg states that context should be considered as a dynamic construct, viewed over a period of time, episodes of use, social interaction, internal goals and local influences. Its idea is grounded on the consideration that it may be difficult for the system designer to determine a priori which pieces of information are needed to accurately determine a contextual state within a set of pre-determined contextual states.

It is easy to observe that external things, such as the people or the physical environment, are relatively simple to capture, whereas, on the contrary, internal things (individuals' interests in that given contextual situation, their history of interaction, their current goals and everything that is not directly inferred) are extremely difficult to capture. In addition, Greenberg underlines that, even if two contextual situation look almost the same at a first glance, the desired action or response in one case may be completely different from the other because, for instance, a different series of events led to that situation. By recalling Heidegger's hermeneutic philosophy, Chalmers describes experiences and theories in which *human activity is treated as an ongoing temporal process of language and interpretation, rather than a series of separable perceptions, each of which frames and fixes the world as a set of symbols and signs.* Here the question is: what does this mean?

Embodied interaction: context as a feature of interaction In order to try to understand what this obscure statement says, let us turn our attention to the ideas presented by Dourish [11], in a paper that dates back to 2003. It is possible to list two different uses of context in ubiquitous computing systems:

- In the first case, a system encodes context along with pieces of information so that it can be later used as a retrieval cue;
- In a second kind of approach, context is designed to dynamically adapt the behavior and the response of the system to patterns of use.

Dourish embraces the second approach and presents an interesting perspective on HCI, that he calls *Embodied interaction*, aimed to gather issues coming from both CSCW and context-aware computing. According to him:

> *The essential feature of embodied interaction is the idea, as illustrated above, of allowing users to negotiate and evolve system of practice and meaning in the course of their interaction with information systems (...) Embodiment is not about physical reality, but rather about availability for engagement. The embodied-interaction perspective is concerned with the way in which the meaningfulness of artifacts arises out of their use within systems of practice.*

By analyzing this definition, we get to understand that context has a computational meaning if treated inside an interaction process, that is to say that the real importance of context is not what it as an a-priori description of the surrounding physical world, but the role he plays in interaction. What makes information meaningful is therefore an outcome of the activity carried on by the application and the user.

Dourish therefore argues that context is an *interactional problem*, rather than a *representation problem*. The representational account of context states that the problem with context modeling consists of a collection of

any information that can be used to characterize the situation of an entity [10]. The allusion to the theories by Dey and Abowd is clear. Such an approach is based on a few assumptions: context is a stable (meaning that the relevance assignment process can be made only once) form of information that can be defined in advance, and it is clearly separated from activity (i.e., the content). The interactional approach to context previously mentioned provides an alternative view that overturns these assumption. Rather than being considered as information, according to Dourish, context is a relational property between objects and activities. This means that some piece of the context surrounding the application and the user just is or is not relevant, based on the activity that is being carried on. This also means that context is particular to a given activity or action and relevant settings. Relevance indeed emerges out of interaction, *determined in the moment and in the doing*. In addition, contextual features and their scope are defined dynamically as the action goes on and more piece of information are collected. As a result, context and activity cannot be separated but have to be analyzed together.

This discussion highlights the importance of the designer's role in addressing the issue of embodiment: he should develop a system that is able to let the user create and communicate the meaning, by managing the assignment of a meaning to an object inside the user's everyday activity. Anyway, as Chalmers [6] points out, the designer cannot of course predetermine the user's activity and interpretations, but he cannot even be totally uninvolved in that process. The designer has no choice but to influence the way that users manage coupling.

3.2 Approaches to context representation

As far as the modeling of the context for a system providing learning application like FABULA is concerned, we need to move a step forward towards practice.

What we have learned in the previous analysis is that the general functions of a context-aware application should not be interwined with the definition and evaluation of pieces of information regarding context, which are prone to many and sudden changes. The approach we are following is aimed to realize a good context information modeling formalism that reduces the complexity of the applications run and that improves their maintainability and evolvability. After that, we can proceed with the actual implementation and insert the data. However, maintaining and evaluating contextual information is expensive, so their re-use and sharing between context-aware applications must be taken into account from the beginning.

Let us first take a quick look at a number of generic (i.e., suitable for any kind of application) context models and the requirements they should fulfill.

General requirements In this paragraph, we present a list of the features that a context model and its corresponding knowledge managing system should have, according to Bettini et al. [13]:

1. *Heterogeneity and mobility*: when designing a context-aware application that has to collect pieces of information about the surrounding context, of whatever nature and format, we have to deal with a large variety of sources, from low-level sensor to information provided by the user in some profile and from a database to a geographic map, mainly differing in their update rate and in the semantic level in which they provide data to the upper level. In addition, many context-aware applications like FABULA are also mobile: this means that contextual information provisioning must be dynamically adaptable to the changing environment. It is important to notice that an important role in this case is played by location and spatial layout;
2. *Relationships and dependencies*: it often happens that some sort of relationship between two different type of context exists. For instance, a change to the value of one property may affect the values of other properties;
3. *Timeliness*: a context-aware application must need to access past states, i.e. context history. The management of such information could become difficult has the number of updates sometimes is very high, depending of the domain of the application;
4. *Imperfection*: there is a need to model also the contextual information quality in order to support reasoning about context;
5. *Reasoning*: the two main techniques that context-aware application must support are consistency verification of the model and context reasoning;
6. *Usability of modeling formalism*;
7. *Efficient context provisioning*: in case the application deals with large models and numerous data objects, suitable access path to efficiently access information must be provided. Such paths represent the context dimensions along which the application select context information, and are referred to as primary context and used to access secondary context through indexing techniques. Activity of user is a commonly used primary context attribute. Of course, this requirement of data organization recalls the theories by [2] I previously described.

In their paper, Bettini et. al. [13] present three modern approaches for context modeling that best fit the previously listed requirements.

Object-role based models of context information Object-role based modeling approaches, and in particular fact-based context modeling approach, were born as a result of the effort to build models that were formal enough to support query processing and reasoning. Of course, they have

their root in database modeling techniques. The support for reasoning is based on CML (Context Modeling Language), a language based on ORM (Object-Role Modeling), which was developed for conceptual modeling of databases. CML provides the designer with a graphical notation able to support the formal specification of the context requirements for a context-aware application. CML uses the formality of ORM to support the evaluation of simple assertion as well as SQL-like queries. One of the most interesting features of CML is the ability of supporting querying over uncertain information: it is possible to state if an assertion is true or possibly true in case of so-called "alternatives" constructs.

A grammar for CML was developed in order to capture more complex condition by assertions and to formulate high-level abstractions of context (so-called "*situations*"), expressed using a novel form of predicate logic.

It is therefore possible to build a graphical schema of the context representing a given *situation* by means of fact types, in which a given object type plays a role. We can consider fact types as relationship between two entities (objects).

Even though this model looks quite interesting for general purpose context-aware applications, one of its major drawback comes to the fore when a hierarchical structure is needed, or one particular dimension of context is dominant with respect to the other (I am thinking, for example, of space) from the perspective of querying. In this case, perhaps another representation is more appropriate.

Spatial models of context information When looking back at Dey and Abowd's definition of context [10], it becomes clear that space is a central aspect among all context entities. In their definition, the authors mention the place is one of the entities that are "considedered relevant to the interaction between the user and the application". In an interaction-based context-aware application, a modeling of vicinity is required and space is well suited to organize and efficiently access context information.

Spatial context models are, in general, fact-based models where context information is organized by physical location. Such location might be predefined (if the entity the user is interacting with is static) or might be obtained by positioning systems. Two kinds of coordinate systems are supported by positioning systems:

1. Geometric coordinates: points and areas are represented in a metric space, like the WGS84 coordinates of GPS (latitude, longitude and elevation above sea level). This representation allow distance calculation and nearest neighbor queries;
2. Symbolic coordinates: these kinds of coordinates are represented by an identifier, for example a room number, a ID of a cell phone and so on.

Spatial context models can be layered in four semantic levels of growing abstraction along the tiers of spatial ontologies proposed by Frank [15]: in the lower layer, we have the ontology describing physical reality. The assumption is that for each couple of a property in the world and a given point in time-space, there is a single value. At the top of this stack, in the fourth and higher tier, rules are finally modeled so to be used by human or software agents for deduction. Of course, this tier is generally built into database query languages, applications or reasoning engines of knowledge-based systems. Ontology-based models of context information that I describe later typically cover the whole stack up to this layer.

According to Bettini and his colleagues [13], spatial context models allow reasoning about the spatial relationships of objects, covering the inclusion in a distinct area or range and the distance to other entities: is that user in the same (pre-designed) place as me? How far is he? Usual spatial queries regard the retrieval of the position of other objects (or users), the retrieval of other objects in a given range and the discovery of which ones among these are the nearest neighbors. When the amount of spatial context information becomes to large to manage, a good solution might be to introduce a preselection phase of relevant pieces of context information to reduce the size of the knowledge base in order to speed up the reasoning process.

Ontology-based models of context information A third way to model context in the paper by Bettini et al. [13] is following an approach based on ontologies[1]. Context can be seen as a specific kind of knowledge, that can be handled through appropriate tools for representation and reasoning. Past research already highlighted that the analysis of the tradeoff between expressiveness and complexity of reasoning led to choose description logics among others logic-based formalisms [14]. If we represent the context in terms of classes and relationships, the subset of OWL[2] admitting automatic reasoning, i.e. OWL-DL [14] is the description logic we are looking for. The main purpose why it useful to exploit the representation and reasoning of these logics is that they allow us to recognize that a particular set of instances of basic context data and their relationships actually reveals the presence of a more abstract context characterization: for instance, a user's activity can be automatically recognizable. This is maybe an interesting way to model the "context as a dynamic construct", as we said in the previous sections.

OWL-DL can be therefore used to model a particular domain with *classes*, *individuals*, individuals' characteristics (*datatype properties*) and relationships among individuals (*object properties*). OWL-DL then provides operators to build complex description of classes and properties out of their elementary descriptions.

[1] http://en.wikipedia.org/wiki/Ontology_(information_science)
[2] http://www.w3.org/TR/owl-ref/

In particular, this approach to context modeling is particularly interesting for the purposes of this thesis since ontologies are well suited for knowledge sharing (think of two software agents and their communication) since they provide a formal specification of the semantics of context data. It is also possible to obtain more expressive ontology languages by extending OWL-DL with rules.

According to the paper of Bettini al., [13], the two major drawbacks of this modeling approach are:

- little support for modeling temporal aspects in ontologies is given;
- OWL-DL does not include very expressive constructors that would be helpful for modeling complex domains, such as a user's activity, like the relation of composition or the role-value-maps). Computational capability of OWL-DL is already expensive and such relationships would lead to serious performance issues, but an intelligent use of such constructs together with an efficient ontology pruning would be helpful.

3.2.1 The use of ontologies for context modeling

In the previous section, I overviewed a few relevant approaches to represent the context, as presented in [13]. This categorization is interesting for the development of a context model for FABULA: it leads us to think of developing an ad-hoc context ontology to represent the context surrounding the user's learning process, because we think that ontologies are the right approach to addressing the problem of context modeling in a mobile learning application leaning on a multi-agent system like FABULA. As a matter of fact, ontologies are nowadays the best solution to support knowledge sharing, context reasoning and interoperability for ubiquitous applications.

As Bettini and his colleagues [13] point out, an interesting approach is the one proposed in SOUPA [19], an ontology for modeling context in pervasive environments, and in CONON [38], for smart home environments. The SOUPA ontology is based on two different but related sets of ontologies: the SOUPA Core, defining a general and shared vocabulary, and the SOUPA Extension, aimed to extend the general ontology by defining additional domain-specific knowledge. The strategy applied is to borrow terms from well-known ontologies, without directly importing them [19], in order to enhance the reasoning performances. In particular, among the referred ontologies, we find the Friend-Of-A-Friend ontology (FOAF)[3], used to express the user's profile and his social connection. This approach is quite similar to the one adopted in CONON ontology, where the context model is divided between what is general and shared by every component (upper ontology), and what is domain specific and therefore a collection of ontology sets are made available to detail the feature of each sub-domain (e.g., home-domain ontology, office-domain ontology, etc.).

[3] http://xmlns.com/foaf/spec/

Two interesting approaches were applied to CHIP [39] and SMARTMUSEUM [21]. Both of these works are aimed to develop ontology-based recommendation services for museum collections. Nonetheless, they are also suitable for city-wide context. Indeed, the architecture of SMARTMUSEM has been motivated over two different scenarios [21]: either a user visiting a museum (inside scenario, based on RFID tags), or a user walking around the city (outside scenario, based on GPS information) and looking for interesting sites to visit. A user profile is built in three different RDF coded segments, out of the user ability information, his preferences, interests and the visit history. The recommendation service realized works with a user interest profile and the current context profile. As in happens in FABULA, the initial pieces of information about the interests of the user can be set up by means of a web interface with keyword selection over the whole vocabulary available (static profile construction phase). Afterwards, the user profile is updated according to the activity of the user (dynamic phase). For instance, when the user tags an item, than a RDF triple is built and attached to the context the tagging belongs to. As a result, the user profile is set of profile items pi in the form of RDF triples: $pi =< t, ct, w >$, where t is a triple, ct is the context of the triple and w is the weight for that triple inside that context. The context is profiled by collecting the relevant pieces of information about the situation surrounding the user, e.g. his location when the recommendation occurs. Spatial restriction are stored as triples inside the profile.

The approach proposed in CHIP [39] to create a user model is also very interesting for us. The scenario is the same as for SMARTMUSEUM. As it happens in other works, the user profile is built by specializing the FOAF ontology: here, the focus is set on the *foaf:Person* and *foaf:OnlineAccount* classes, and the *foaf:holdsAccount* property connecting those two classes [39]. A user in CHIP is a subclass of the *foaf:OnlineAccount* class (see figure 3.2). Each time a user rated an item, an instance of a specific CHIP class called *chip:RatedRelation* is built to connect the user to the rating value and to the rated object. Such rated object is an instances of a class coming from the metadata vocabularies expressed in RDF Schema. Restrictions over properties are expressed in OWL.

3.2.2 Summary

Let us now summarize the most important issues overviewed in the previous section, and how they are relevant for this work.

The milestone in the model of context developed as part of this thesis is the definition by Dey and Abowd [10], that defines the context as a collection of any piece of information that can be used to characterize a situation: most widely used information is made of the location, the identity and the state of the user. Above this basic information, it was necessary for the sake of this work to learn how to model the whole available information

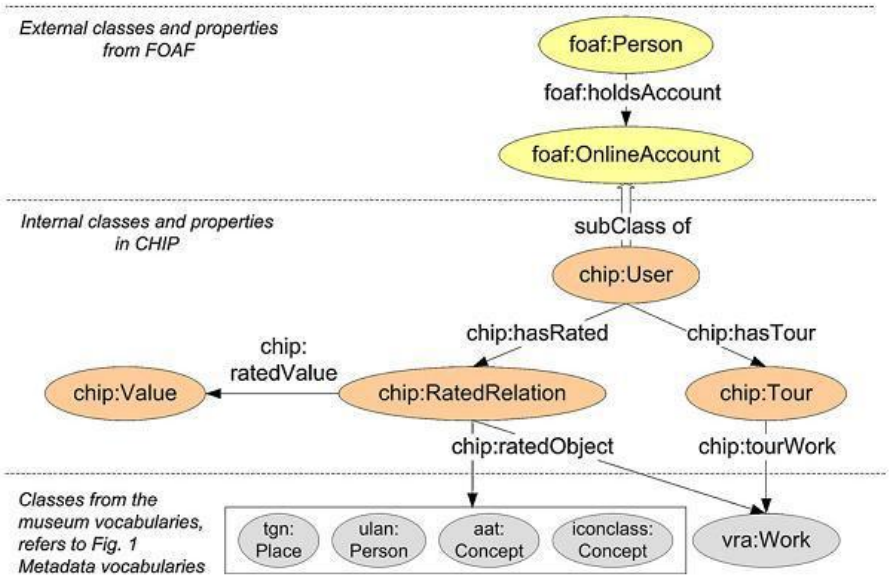

Figure 3.2: Main classes and properties in the CHIP User Model [39]

and to identify a set of categories to be expanded one by one. Categorizations based on different areas of study have been then analyzed. The most important ones answer the wh-questions and are: the personal information about the user (who), the information about the environment (where, i.e. the place), the temporal information (when) and the activity carried on by the user (what, i.e. the activity). According to Dey and Abowd [10], these are the primary context types, out of which context can be organized and other secondary pieces of information can be inferred. The distinction by Bradley and Dunlop [2] between what is explicit and what is implicit in the interaction between the user and the application then helped the organization of the information about the user.

Another important topic is relevance assignment. As said in [34], from a computational perspective it might be useful to assign a priori a relevance value, or a weight, to the incoming pieces of information, according to the goals and tasks of the user and the application. Nonetheless, this process might cause the loss of important information that passes from being irrelevant to relevant based on a set of unpredictable causes while the learning process is occurring. As a result, relevance assignment is still an open issue in literature.

Other researchers, like for instance Dourish [11], have instead argued that the context and its relevancy cannot be defined a priori, but its real importance emerges while the interaction between the user and the surrounding world is being carried on. So the question became: how to model such ever-changing context? Three approaches to context representation have been studied: object-role based models, spatial models and ontology-based models: for the purpose of the present work, the last category seemed to

be the one that best fit the requirements outlined by Dourish [11]. By representing the context as a specific kind of knowledge through an ontology and performing a reasoning based on description logics, we can achieve the representation of the context that evolves during the interaction and does not need to be defined a priori by the designer. In addition, the theory presented by Dourish was the key to model the user's activity as a sequence of simple tasks to be accomplished. The dynamics of the interaction with the physical world and has then been represented as a social configuration, or group, that is the outcome of a learning activity carried on by more than one user. Such modeling process will be further described in the next chapter.

3.3 Context-aware Recommendation Systems

Once the representation of the whole context is ready, we now have to face a further challenge: how to deal with the potentially huge amount and diversity of information coming from many heterogeneous sources? How to present this information to the user in a way that fits his personal interests and aims, tailored on his preferences? The answer is brought us by Recommendation Systems (RS). As stated in section 2.3.1, the FABULA platform from a passive perspective uses services as its building blocks and organizes them into three layers: application specific learning services, basic learning services and resource management services [24]. The service provided by RS is a good example of an application specific learning service.

In literature and on the Web many use cases to exploit RS are available and applications range from e-commerce to movie of music recommendation. I will now quickly overview the main types of techniques available.

3.3.1 Content-based RS

The approach of content-based RS is to compare the content retrieved about the user (his profile, interests, etc.) with the content available about one item [33]. The output of a content-based algorithm is a relevancy value of that item for the user, compared to the items the user has been interested in before. A good example of content-based RS is *Culture Sampo*[4], a Finnish system that uses tags of artworks in a museum-wide scenario. According to [33], one of the main problems in developing a content-based RS is finding a representation of the documents: the most commonly used approach is by a vector with the most relevant words inside that document. It is anyway straightforward to see that the two main drawbacks of this approach are:

1. that the user needs to explicitly state a model of his prefences;

[4]http://www.kulttuurisampo.fi/?lang=en

2. the system misses to find new and perhaps interesting items, because it only relates to what the user listed.

3.3.2 Collaborative Filtering RS

A solution that successfully addresses the drawbacks of content-based RS is represented by Collaborative-Filtering (CF) RS. This is also the most widely spread recommendation technique [1]. The idea behind this approach is to provide a recommendation or a prediction for an item based on the opinions of users that are similar to the user the item has to be recommended to. The information about the other users' opinion can be obtained either out of what the user explicitly stated or by means of some implicit techniques. Given a set of m users $U = \{u_1, u_2, ..., u_m\}$ and a set of n items $I = \{i_1, i_2, ..., i_n\}$, $u_a \in U$ is the active user for whom an item likeliness has to be computed by either a prediction or a recommendation value. Prediction is a numerical value expressing the likeliness of item $i_i \notin I_{ua}$ for u_a, whereas recommendation is a list of the top-N items i $\notin I_{ua}$ that u_a is assumed to like the most.

As figure 3.3 shows, the input of a CF algorithm is a user-item matrix with a rating value in each cell. The output can be either the prediction of the interest of a given user u_a for an item i_i, or a list of the top-N items for u_a.

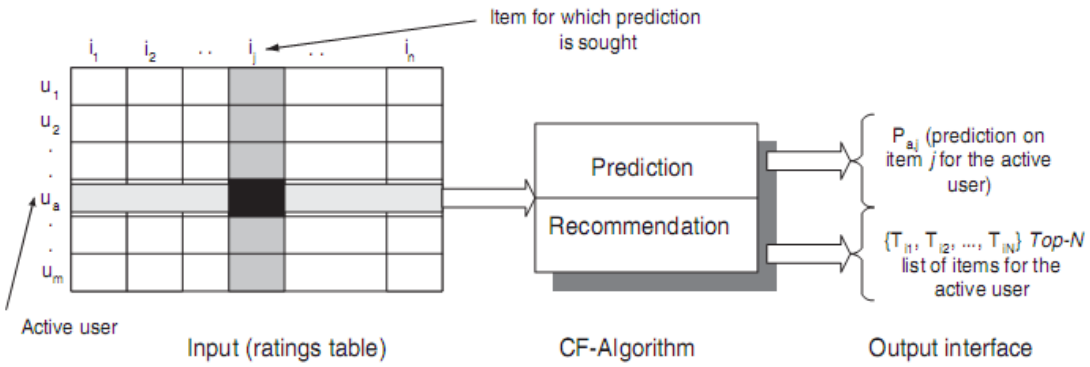

Figure 3.3: The Collaborative Filtering process as presented in [1]

According to researchers, the CF algorithm in literature can be divided into two principal categories [1]: *memory-based*(or *user-based* or *nearest-neighbor*) and *model-based*(or *item-based*).

Memory-based CF algorithms: they use statistical techniques in order to find a a set of neighbor users for user u_a, out of those users having a history of agreeing with the target user when rating an item. Once the neighborhood is formed, the result of these algorithms is a prediction or a list of recommendations for the active user. The predicted vote of the active user for one item is a weighted sum of other users' votes, where

the weights considered can be the distance, the correlation or rather the similarity between each user and the active user. Algorithm can therefore be distinguished based on the way they obtain these weights. For instance, in the field of information retrieval, an interesting approach is to treat a document (i.e., the information about one user or one item) as a vector of n word frequencies, and the similarity between two documents is given by the cosine of the angle formed by these two vectors on the n-D space.

Model-based CF algorithms: they are instead focused on developing a model of user ratings: they compute the expected value of a user prediction, given his ratings on other items [1]. Machine learning techniques used for this purpose encompass Bayesian networks (by formulating probabilistic models), clustering (by gathering similar users in the same class and estimating the likelihood of a user to be in a particular class C and the conditioned probability of ratings) and rule-based algorithms (by discovering associations between co-rated items).

Such CF algorithms, however, suffer from a number of limitations:

1. Sparsity problem: many commercial RS (e.g, Amazon, eBay,...) have large sets of data available, the number of rated items but the active users is less than 1% of the total items. On the other hand, in some real situation like when a user is visiting a city for the first time, it is likely that we find no user with some rating in common.
2. Cold start problem: it is not possible to compute the similarity between two users when at least one of them is new inside the system and has no history of past activity available. At this point, the utility of a content-based RS comes to the fore to try to fight this problem.
3. Scalability: if the RS is centralized, the load on the server increases with the growing number of users and, at the same time, such CF algorithms are not able to scale well as the dataset becomes larger. As a result, accuracy in recommendation or prediction decreases.

3.3.3 Hybrid RS

The previous overview of algorithms for RS shed a light on the need to combine the positive aspects of both content-based and CF RS. Many ways of achieving this result have been proposed, and the most interesting one is the *Collaboration via content* proposed in in [33]. Here, a content-based approach is exploited to find the similarities among users by browsing their profiles, where the meaningless items have been previously eliminated. This step is useful so to avoid considering two users to be similar with a number of irrelevant words in common and having a low weight for the whole analysis. Then, the similarity value between two profiles is computed. Afterwards, as it happens in CF, the prediction made is determined by a weighted average of all user's prediction for that given item using the correlation between profiles as the weight [33].

3.4 Approaches to RS realization

Two of the most interesting solutions to the realization of recommendation services from my point of view are those proposed for the SMARTMUSEUM [21] and CHIP [39] systems, which I already mentioned in section 3.2.1 while talking about the way they handle ontologies to construct profiles. Such systems are meant to provide the user with a recommendation of items he might be interested in.

The SMARTMUSEUM project is able to combine collaborative and content-based recommendations [21]. As previously recalled, in SMARTMUSEUM the content filtering is based both on the statically defined information and on the dynamically constructed activity history, which compose the user profile. Each item in the user profile is a triple $pi =< t, ct, w >$, where t is again a triple, ct is the context of the triple and w is the weight for that triple inside that context.

What is also worth mentioning is what is considered as context. The very first important piece to collect is the location (one of the main types of context identified in the previous sections) of the user in the moment when the recommendation is requested, by means of a triple to store in context profile. Spatial restriction can be, in the case of an outside scenario, a bounding box of WGS84 coordinates.

The weight w is based on the current context surrounding the user, and represents the maximum likelihood of the profile item in that context. The mathematical procedure is described in [21], and makes use of the Laplace smoothing to give a non-null weight to the observed triples that do not match the user's current context. Each triple can be expanded to other triples by means of a query where the Wu-Palmer [40] similarity measure is set as smaller than 0.5. The reason of this passage is to expand each triple to all the subsumers and to a reasonable amount of super classes in the lexical hierarchy. This approach looks like an interesting solution to the problem of relevance assignment, though it is not completely fit for the representation of context developed for this thesis.

The weighted triples are then used to query over the knowledge base, that is composed of triples that are indexed and combined by means of subsumption reasoning. The result of this procedure is a normalized vector space of documents. The recommendation value for a given item is therefore the result of the cosine similarity measure between the triples coming from the user's profile and the documents vectors.

In CHIP [39], a semantic-oriented content-based recommendation system is instead proposed: a so-called *belief value* is computed to predict the user's interest in other artworks and topics, by taking into consideration the semantic relationship involving the artwork the user has already rated and the related topic.

These last two solutions have been exploited inside this work when developing the algorithm to recommend items (e.g., a place or an activity) to a user. As it happens for the already rated items in CHIP, the history of activities done and places visited is considered here. Items are represented as vectors of words according to the vector model of information retrieval, and then clusterized based on their similarity values calculated with the cosine similarity measure. The user is then recommended new items belonging to the cluster where most of his past items belong to. In addition, as it happens in SMARTMUSEUM, the recommendation can be obtained by means of the cosine similarity between the user (as the query) and the items (represented as normalized vectors).

If we now move in the direction of recommendation systems developed specifically for context-aware mobile learning, useful ideas can be borrowed from [27], a work developed inside the FABULA project. Here, a framework is presented, where user profile attributes are exploited to recommend a set of suitable collaborators to a user.

$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ u_n \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & \cdots & \cdots & u_n \end{array} \\ \begin{bmatrix} s_{11} & s_{12} & s_{13} & \cdots & \cdots & s_{1n} \\ s_{21} & s_{22} & s_{23} & \cdots & \cdots & s_{2n} \\ s_{31} & s_{32} & s_{33} & \cdots & \cdots & s_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ s_{n1} & s_{n2} & s_{n3} & \cdots & \cdots & s_{nn} \end{bmatrix} \end{array}.$$

$$s_{ij} = \begin{cases} -1 & i = j \\ s(i,j) & u_i, u_j \text{ do not have collaboration relationship. } 0 \le s(i,j) < 1 \\ 1 & u_i, u_j \text{ have collaboration relationship.} \end{cases}$$

Figure 3.4: The user-user matrix according to [27]

The attributes coming from the user's profile are organized in five categories and weighted according to their relevancy. The recommendation mechanism relies on a user-user matrix *s* (see figure 3.4), where each cell *s(i,j)* is the score between user i and user j. This score is the weighted mean of a set of values that are the result of the matching of the data from the five types of attributes. Such attributes mainly overlap the categories of context we already identified: the user's personal information and interests, the location, the activities done (weighted over time: more recent activities are more relevant) and the user's ability to maintain existing relationships (that can be seen as a way to calculate the trustworthiness of a user). Such a representation of the user's profile is not far from the one developed for the present work. The last attribute, in particular, is interesting for us because it represents the number of current friends of a given user out of the number of friends of the user who has most. The same idea can be applied to other kinds of items, like places or activities, in order to give a higher weight to the user who has more friends, even though a solution to contrast people

having too many sparse items compared to the majority has to be found. As far as the matching based on the location of the user is concerned, it is also useful to notice that in [27] the smaller the distance between two users, the higher the recommendation score. A distance threshold is also preset in order to avoid matching with far away users or items. Weighting the recommendation based on the distance of the user from the item to be recommended might be one of the solutions to counteract the problem of assigning a priori a relevance value to contextual items.

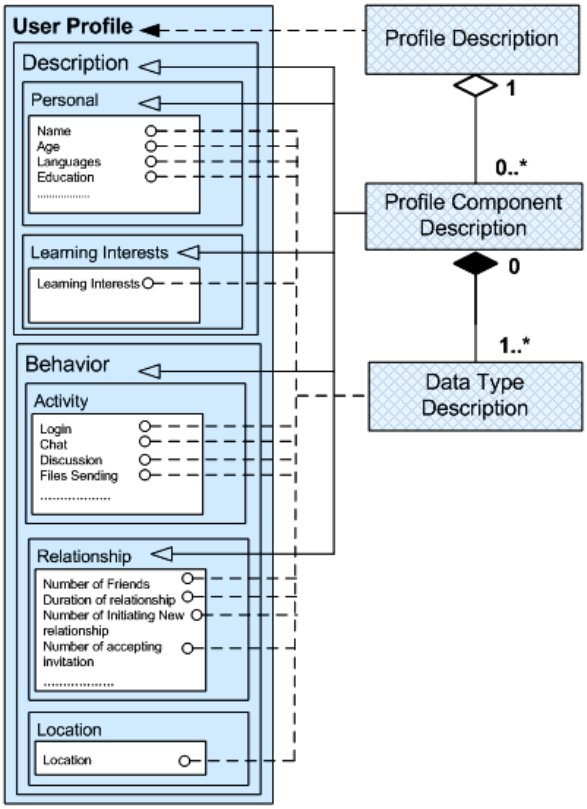

Figure 3.5: The profile as considered in [27]

CHAPTER 4

Context meta-model design

In this section, I will describe the process of designing a meta-model for the contextual information available for an application like FABULA and the way this has been translated into an ontology. A theoretical background of each step is given throughout the whole section.

4.1 The meta-model design

Our choices were grounded on the theories existing in literature, as seen in chapter 3. The definition by Dey and Abowd [10] is the basis of the context model developed. We were lead by their idea that *Context is any information that can be used to characterize the situation of an entity.* Nonetheless, a further step needs to be taken in order to understand what is actually contextual information and what is not. As Dourish points out [11], it is not easy to determine a priori a set of relevant pieces of information (I already outlined the concept of relevancy in chapter 3), and, above all, it is even more difficult to design the internal things that are important for a given situation, for instance the individuals' interests in that given context, their history of interaction, their current goals and whatever can be meaningful for the current ongoing situation. We therefore wanted to model both the explicit and the implicit information about the user, so to obtain an explicit and an implicit profile: this distinction was based on Bradley and Dunlop's paper [2], where explicit information is described as *the user's intention known to both him and the application*, whereas implicit information is *known to the user but not to the application*, that needs a way to retrieve it and prune what is not relevant.

The first thing to do was to clearly identify the top dimensions (i.e., categories) of context that were important for the city-based scenario the application is thought for.

A useful preamble to do so has been to realize a meta-model of the complete context for the FABULA application's purposes, by attempting to gather all the possible pieces of contextual information that could be available in a city-wide scenario.

The meta-model has been developed by means of a class diagram[1]. Each class represents a concept that can be described by a list of attributes.

In literature, many examples of context modeling for a mobile learning application are available. Approaches presented in the SMARTMUSEUM [21] and CHIP [39] projects are thought for museum collections, but are also suitable for outside scenarios [21] like the one FABULA is designed for. Interesting exploitation of ontologies to model and reason about the context are in [19] and [38].

When representing the context, we also decided not to take into consideration the information about the mobile device: the reason is that we did not want to ground our work on hardware limitations, so to make the model developed suitable for different kind of applications and independent of a set of hardware requirements. In addition, for the sake of this report, we deliberately decided to skip the issues dealing with privacy, but they are definitely important for the work done.

4.1.1 Modeling the user class

Since the FABULA application is meant for people walking around the city engaging some form of learning, the first two categories to model were the users and the city itself.

The process of modeling the information about the users was influenced by the idea that it is possible to divide the information about one person connected to FABULA between what is explicit and what is implicit. The former kind of context is composed of what is explicitly stated by the user, for example a profile the user is invited to create when registering to the FABULA application. The latter kind is instead to be retrieved by the application without the user being bothered and is meant to avoid redundancy in the information building. It is therefore to be constructed out of a profile of the user on a social network (e.g., Facebook[2], Twitter[3], MySpace[4], etc,), on his chat logs on FABULA and on the history of the learning sessions he was engaged in (e.g., list of the activities and of the places visited). The decision to have an interface with a social network is enhanced by the growing availability of social-networks connect services launched by the major social-networking sites such as Facebook or MySpace. As a result, membership is increased by providing content from a variety of sources in a seamless manner [31]. For the sake of example, we created a connection with the user's profile on Facebook, which today is to any extent the most commonly used social network.

As evident in figure 4.1, the explicit information about the user are collected in:

[1] http://en.wikipedia.org/wiki/Class_diagram
[2] http://www.facebook.com/
[3] http://twitter.com/
[4] www.myspace.com/

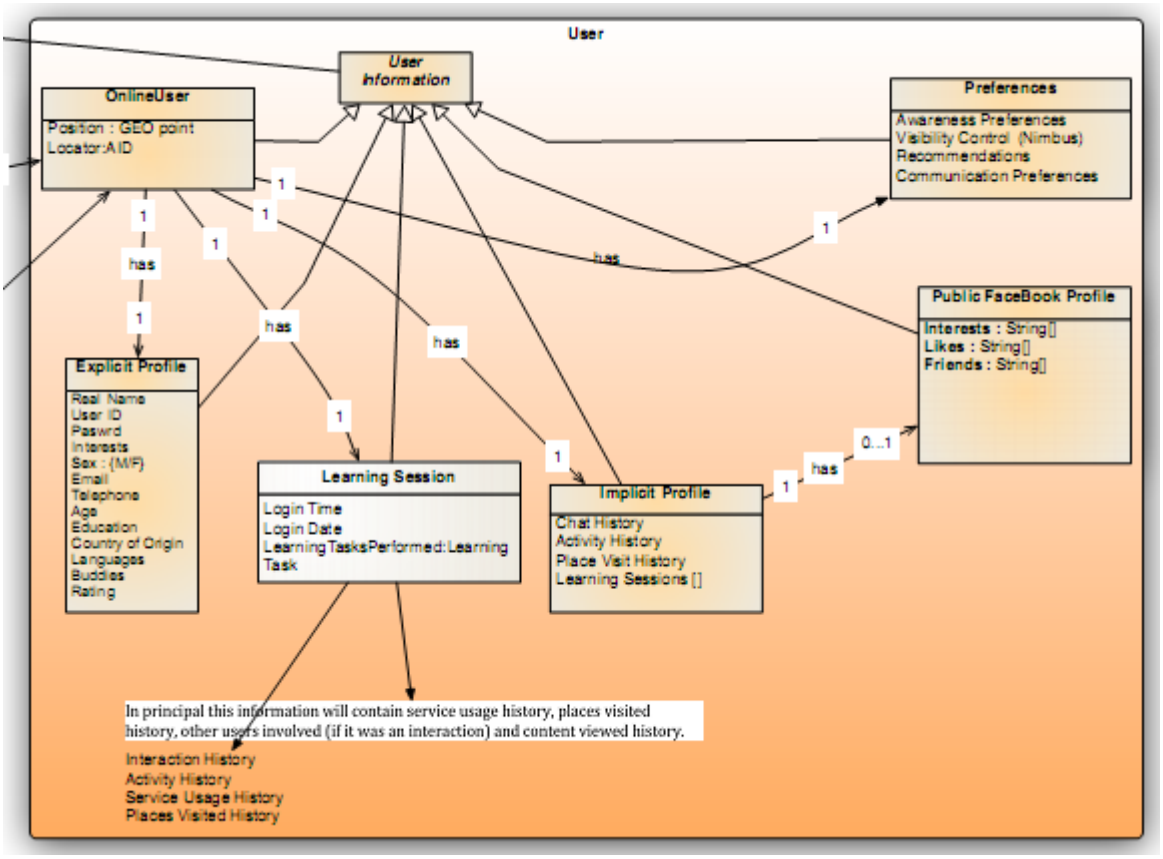

Figure 4.1: A snapshot of the developed meta-model representing the information about the user

- the *Explicit Profile* class from a profile that the user is asked to create when he registers to the FABULA application for the first time;
- the position of the user and the identity of the AGORA User Agent responsible for his device in the *OnlineUser* class;
- the user's preferences about whether being visible, receiving recommendations from the system and communication modalities in the *Preferences*.
- the learning session an online user is engaged in, by saving login date and time and the task sequence performed (*Learning Session* class).

By means of the information collected in the *Learning Session* class, it is possible to store the history of the activities the user did, the places he visited and the logs of the chat sessions he started: all these items are represented as attributes of the *Implicit Profile* class. If a user has granted the access to his Facebook profile, then this class can be associated to at most one *Public Facebook Profile* class.

At this point, the question is: what information should we pick from the Facebook profile of a user? The most important items, in our opinion, are:

- the friends, in order to see if two FABULA users are friends on Facebook but not on FABULA;
- the items on Facebook the user tagged with an *I like it* label;
- the user's interests, to enrich the interests list on the FABULA profile with further topics.

All of these items are wrapped into arrays of strings that become attributes of the *Public Facebook Profile* class.

4.1.2 Modeling the place class

Another fundamental aspect that is necessary to determine is the setting where the situation occurs as far as a mobile learning application like FABULA is concerned.

To achieve the representation of a place as described in section 2.2, the city of Trondheim is divided into a number of learning zones, that is to say areas where a collection of interesting opportunities to engage a learning session where identified. An example is presented in figure 4.2. Each learning opportunity is then labeled with a value rating the three main aspects of the type of learning experience a user can live there. According to the analysis carried out by Rossitto [7], *a place emerges from the interplay of different dimensions*. We adapted this approach to our purposes and identified three levels describing the experience that the user can live in a place. To each level, a rating value is assigned:

1. Social level: it represents the possibilities to engage social relationships in a place. It implies the presence and awareness of others, based on their GPS position and their visibility preferences;
2. Historical level: this level encompasses the past of a place;
3. Cultural level: at this level, possibilities to learn about general cultural peculiarities of one place.

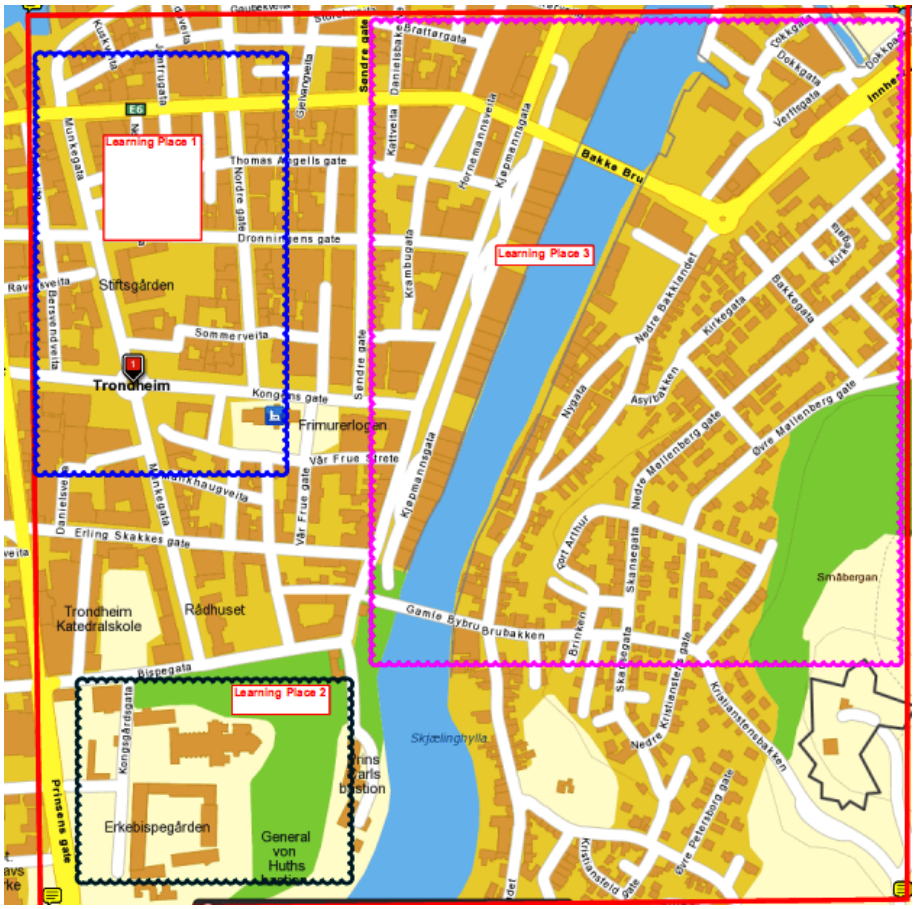

Figure 4.2: The division of the citycentre of the Norwegian city of Trondheim into three learning places

In figure 4.3 a schema of the city is given: a learning opportunity, an area and the city itself are all described by a few common attributes inside the *Learning Area* class, e.g. the geographical coordinates of the four corners delimiting their perimeter. Also available are a description given in free text and a set of tags, both pre or user-defined, that might be useful in order to cluster similar kinds of areas. A different class called *Learning Experience* is used to rate the level of the historical, social and cultural experience that can be lived in any given area.

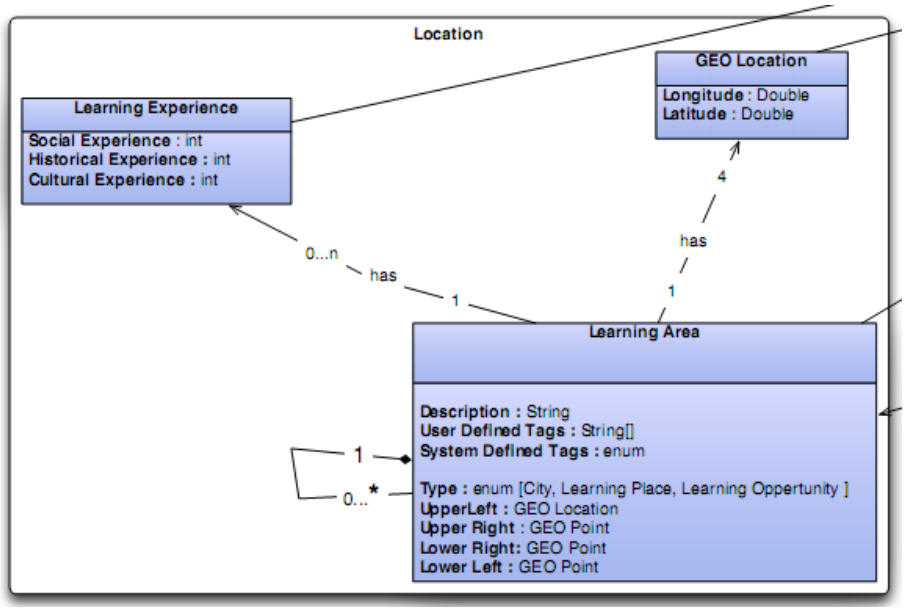

Figure 4.3: A snapshot of the developed meta-model representing the model of space

4.1.3 Modeling the social configuration

As outlined in section 2.2, another aspect that a mobile e-learning application is about is the serendipitous interaction among people[5], i.e. the social connections that are build in an unplanned, random way and destroyed while a user is engaged in a learning process inside the city. The result of such interaction gives birth to a group, or a social configuration, modeled with the *Group* class in figure 4.4. The reason is that a social structure is the outcome of an interaction-based learning activity that presents a collection of tasks to a group of online users: a group of user, in our model, exists as long as there is some activity (made of a collection of tasks) they are engaged in all together.

4.1.4 Modeling the activity class

To achieve the modeling of a social configuration, a category that is important to model is the activity the user can be engaged in. In our model of context, each activity, or *Composed Learning Application* as in figure 4.5, is made up of a sorted list of tasks that the user (in case it is an *Individual Learning Application*) or the group of users (if it is a *Group Learning Application*) has to accomplished. So far, a few types of tasks have been identified for FABULA, e.g. *Multiple Choice Task* (the user has to answer a multiple-choice question) and *Be At Place Task* (the user has to reach a given point in the city). Even the tasks are rated with a cultural, historical and social level of learning, as it happens for a learning area. At this point,

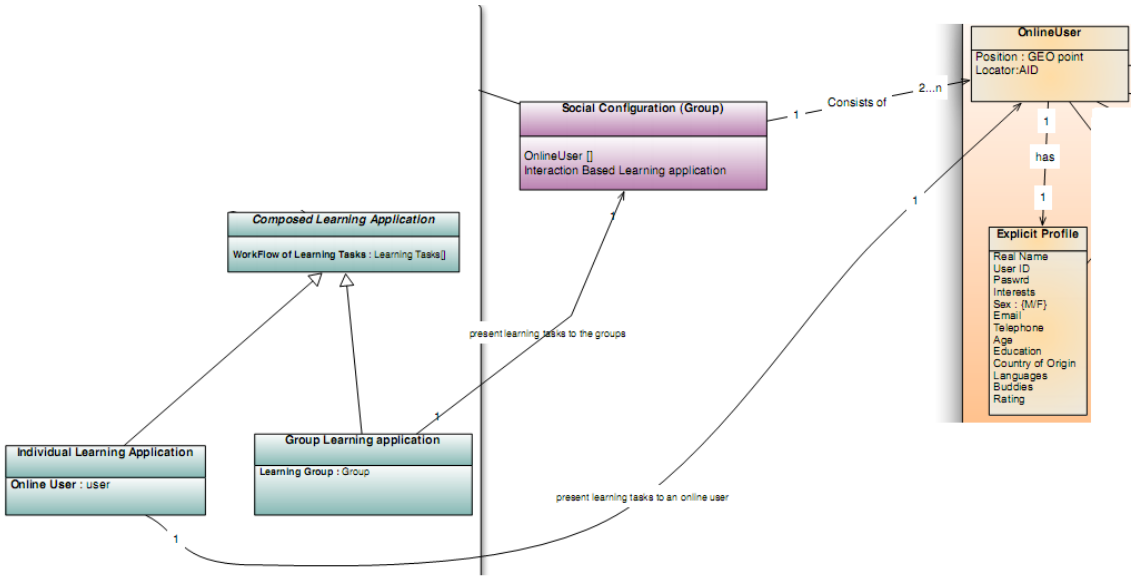

Figure 4.4: A snapshot of the meta-model where a group learning application presents a set of tasks to accomplish to a group of online users, who constitute a social configuration for this purpose

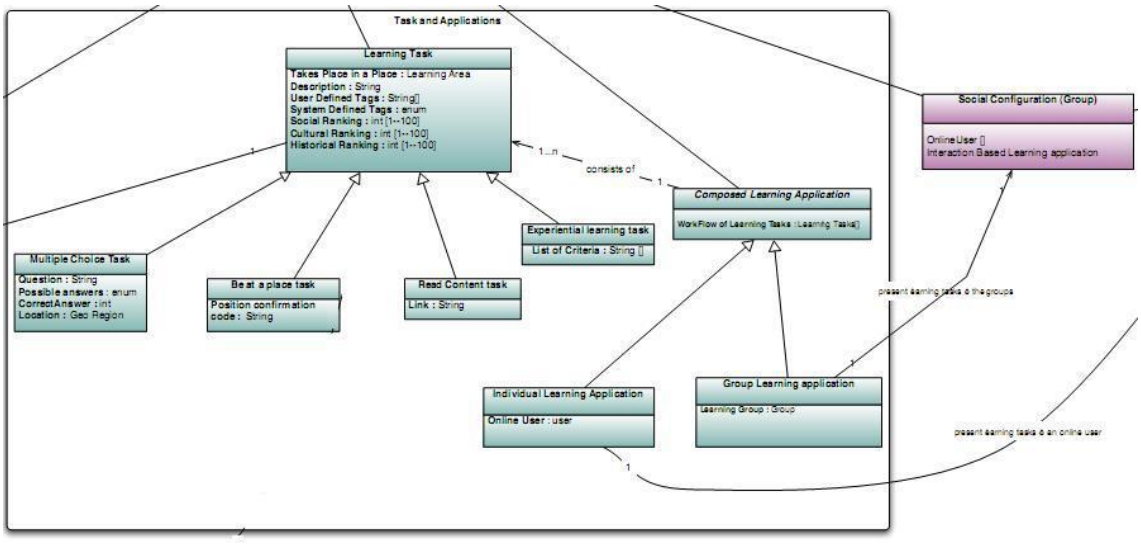

Figure 4.5: A snapshot of the developed meta-model representing an activity as a set of tasks

it is important to underline that such tasks become available only in some given learning areas of the city, so they are closely connected to the spatial context.

4.1.5 Modeling the services

The last main category to be represented inside the developed meta-model is about the services. To represent a service, both the notion of place and that of activity are necessary:

- a service is required to accomplish a learning activity, that is made up of a collection of tasks;
- a service is available in a given environment, and this brings us to the notion of location awareness of one service and to that of situated learning presented in section 1.

4.2 The translation of the meta-model into an ontology

Once a first version of the meta-model representing the contextual information was ready, it was necessary to move a step forward towards making this information easy to be used and shared. The criteria identified by Bettini et al. [13] listed in the previous chapter identified a set of requirements that a context model and its context management system should fulfill. Context is considered as a kind of knowledge that is dynamically changing, so it seems natural to look in the direction of systems able to enhance knowledge representation and reasoning. Ontologies based on description logics have by no means emerged as a strong tool to achieve this result. The subset of the OWL language admitting automatic reasoning (i.e. OWL-DL [14]) is the description logic we chose.

As a result, the model was translated into an ontology, that is general enough to admit merging with other ontologies representing each dimension of context.

The tool used to build the ontology is Protégé 4.1[5], that was the best solution to conceal powerful reasoning and graphical simplicity. See appendix A for a detailed description of the software.

4.2.1 The modeling approach

As I previously described, AGORA is a multi-agent framework built upon the JADE middleware, a fully FIPA compliant agent development environment. It is important to remember this when choosing a methodology for

[5]http://protege.stanford.edu/

modeling information, in order to build each agent's knowledge model and let an agent pose a query or communicate its internal knowledge to another. The agent knowledge model is based on the *behavioral architecture*, where agent's actions can be grouped in terms of tasks to be accomplished.

The modeling approach we decided to follow is grounded on the CommonKADS[6] methodology to support structured knowledge engineering [18]. CommonKADS is a model suite independent of any particular modeling tool, taking different perspectives on a knowledge-intensive task. It is divided into two main parts [29]:

- A knowledge model based on other three main models:
  - **The Organizational or Environmental Model**: it models the environment of an application and the actors (agents) involved.
  - **The Actor Model**: it models the actors performing tasks (as it is in the behavioral architecture) and actions. An actor can be either a human or a software agent, or another entity performing actions monitored by the system. The actor's context modeling is important since it defines the current actor's environment and resources resulting from accomplished goals.
  - **The Task Model**: it models tasks, activities, process and actions relevant to actors. An important property of a task is a Domain, with a relation useful for updating the actor's context and resources.
- A design of the system.

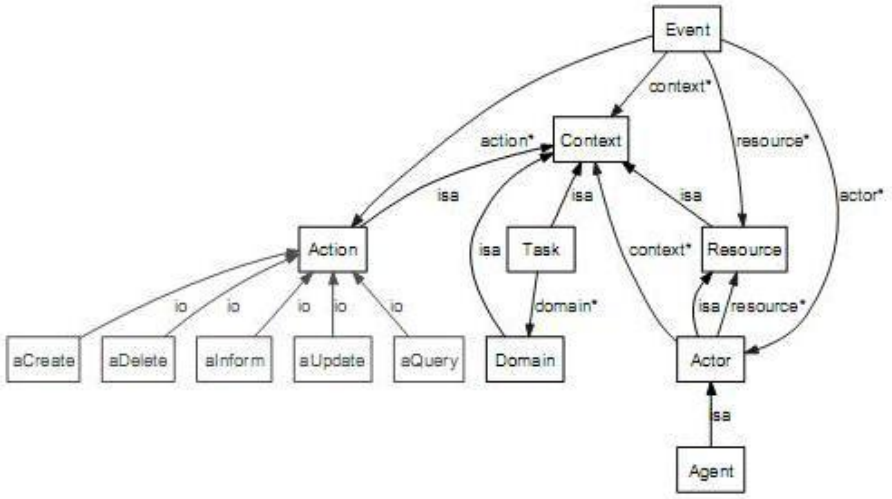

Figure 4.6: The basic ontology for knowledge modeling proposed for the AgentOWL library

[6]http://www.commonkads.uva.nl/

A knowledge model that is generic and suitable for applications with discrete, fully observable environments where environment changes can be captured by events is the one proposed for the AgentOWL library [29]. Here, the model is described using description logics. All agents are expected to share the same ontology, based on five main elements: *Resources*, *Actions*, *Actors*, *Context* and *Events*.

Such a model was taken as a starting point to build the context ontology.

4.2.2 The context ontology

As it happens for the SOUPA ontology described in section 3.2.1, the developed context ontology consists of two distinctive but related subsets of ontologies: the basic ontology and the extension ontologies specific of one dimension of the context to be accurately described. For the time being, the only dimension whose extension has been completed is the user.

The `is-a` hierarchy of the basic FABULA ontology is depicted in figure 4.9 and aims to fix the overall structure of the context surrounding the user and the application inside the city. The three main classes are `Event`, `Resource` and `Action`. An `Action` is one of the actions that the AGORA agents can do. The `Resource` is used to describe the actors involved in the application and the environment where they live. The former elements are modeled by means of the `Actor` class, that is further specified in the `Agent` class, that subsumes the different types of agents involved in the AGORA system.

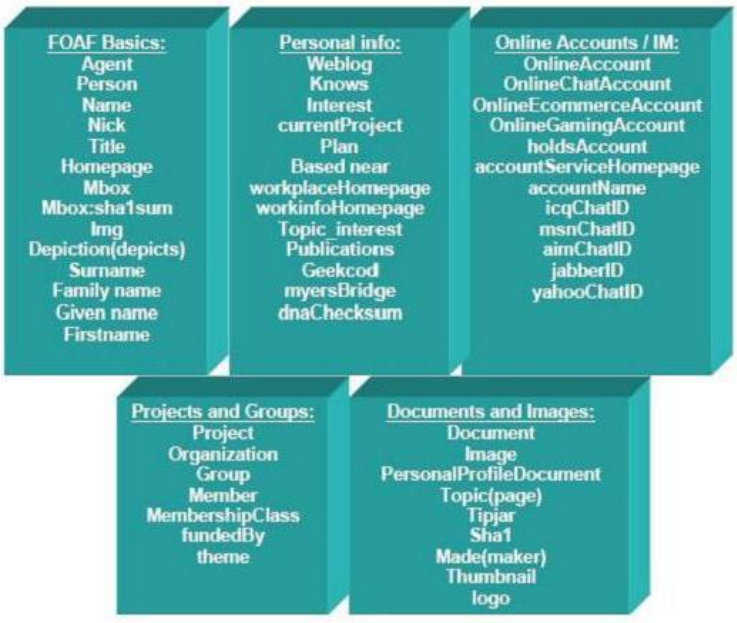

Figure 4.7: The FOAF vocabulary structure

The `FabulaConcept` class is instead used to model the environment, or the context. Concepts that, at a meta-model level, inherited from a top level

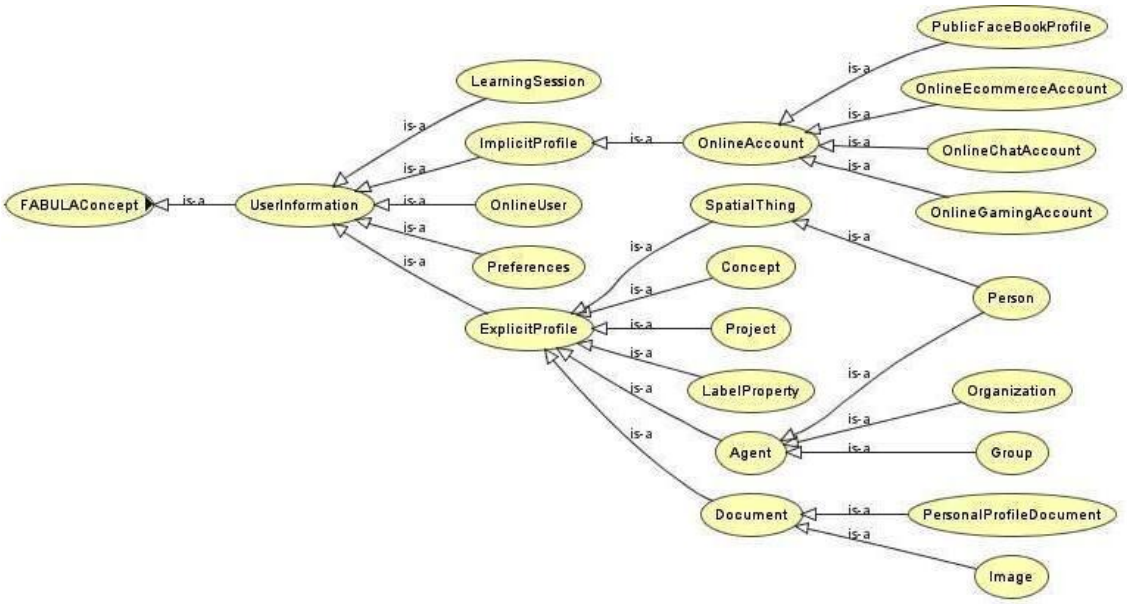

Figure 4.8: The User ontology

concept `FABULA_System_Entity`, now have been translated into subclasses of this class. This part of the ontology is the core of the context modeling that this work wants to achieve.

In order to model the relationships existing between users, our purpose is to give a weight to the already existing ones, whose trustworthiness is already ensured. The `foaf:knows` object property allows the interlink between to user's profile in the FABULA network.

The user dimension has been represented with one extension ontology specific for this category. The representation format chosen to depict the user information is the FOAF vocabulary[7], that has become the de-facto standard for this purpose since its introduction in 2001. It is a lightweight and open solution to build a reusable profiling enriched with the relationships the user already has inside all the social networks available in the web. The structure of this vocabulary is shown in figure 4.7[8].

Figure 4.8 shows instead how the FOAF classes have been used to realize the is-a structure of the user ontology.

If a user has a profile on a social network and the user grants the permission to access it, then his personal information must be updated accordingly: one use can have one or more accounts that are here subclasses of `OnlineAccount`. In the case of a Facebook profile, a user is connected to the `PublicFacebookProfile` class through the `hasFBProfile` object property.

[7]http://www.foaf-project.org/
[8]http://xmlns.com/foaf/spec/

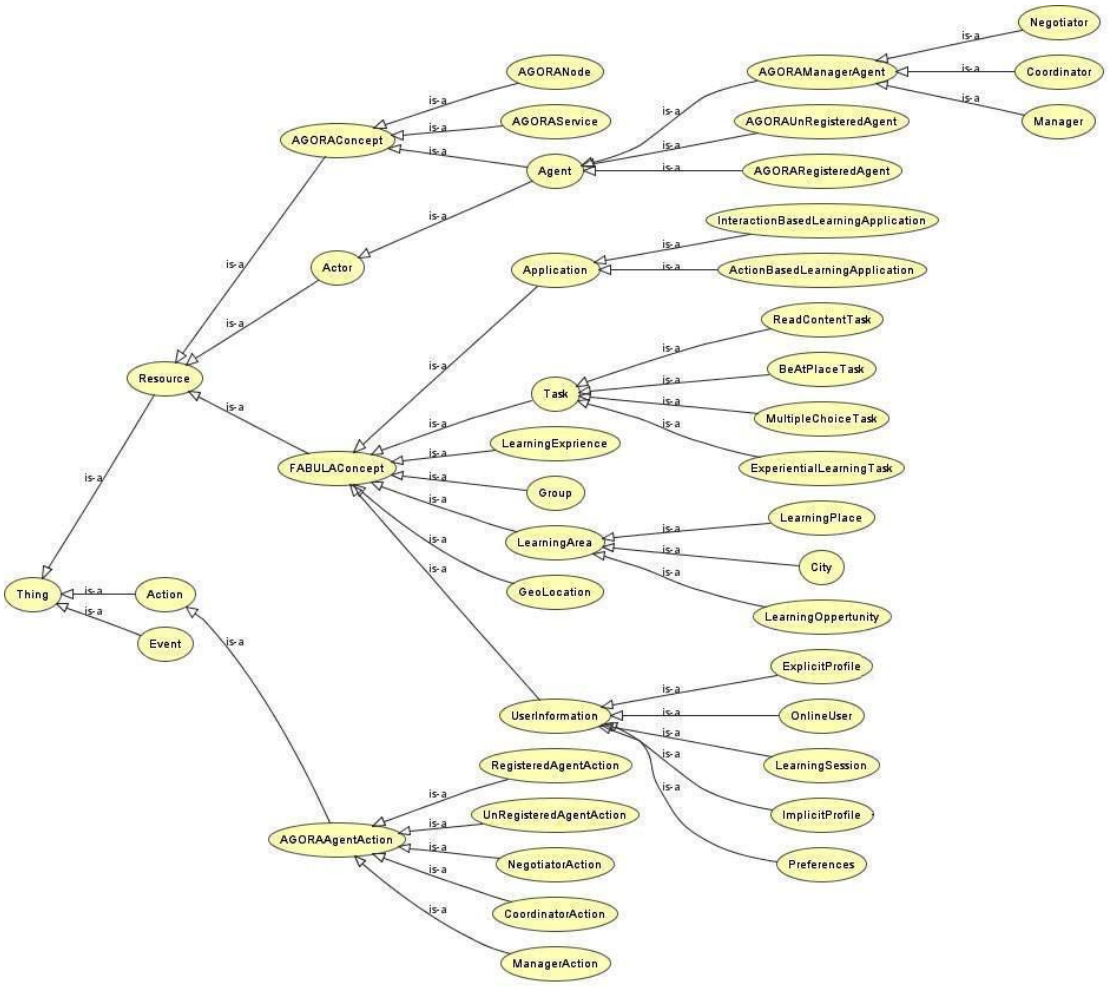

Figure 4.9: Structure of the basic FABULA ontology developed following the CommonKADS approach

CHAPTER 5

Design and implementation of the recommendation algorithm

In the previous chapters, I have described the realization of a model for the city-based contextual information available for a mobile learning application, by also grounding the choices made on a literature review.

The design of a context-aware recommendation system is now required in order to present the information to the user and to suggest him interesting items belonging to one category of the context: another user (that can be seen as a collaborator or a friend), a place, an activity or a service.

The potentially available information organized through the described ontology needs to be filtered according to specific rules, in order to match the user's profile with another item's profile and to make a recommendation. To do so, it is necessary to assign a relevance value to each piece of information, or attribute, coming from different sources. It is a very difficult process because, even though on the one hand it is impossible to state a priori what matters and what does not in a given situation, on the other hand it is almost impossible to leave to the application a decision that has to be based on many and often unpredictable factors.

Approaches proposed to overcome this obstacle and calculate weights for the attributes are the following:

1. to ask a set of pedagogy experts to assign a priori the weights for different attributes;
2. to ask the users to input the importance of each attribute;
3. to use feedbacks: at first, some likely values are assigned to different attributes. The the results of the recommendation (if it is accepted by the user or not) help to adust the appropriate weight;
4. to adopt a hybrid solution combining two or three of the previous points.

None of these solutions completely fits the requirements of the present work. Furthermore, designing a detailed relevance algorithm is out of the purposes of the present thesis as the process would require a rather large amount of

time. As a consequence, I will now on assume that a *relevance engine* provides a set of weights to each incoming piece of information. The overall matching score between the user and the item we want to recommend him will therefore be a weighted sum of all the different sources of contextual information we consider.

5.1 Design of the recommendation algorithm

I will now go into the details of the design of the algorithm that has been realized to implement a recommendation service for the FABULA application.

In section 3.3 I have overviewed the existing types of RS available and listed their major advantages and drawbacks: the solutions we adopted for the algorithm developed here are meant to gather and exploit the positive aspects of both approaches, by combining ideas that are typical of both content-based and collaborative filtering RS.

As introduced in the previous chapter, the dimensions of context that we identified are the users, the places, the activities and the services: they are also the different categories of items that it is possible to recommend to a user. The algorithm developed follows different strategies for each category, mainly due to the semantic structure underlying the representation of a given item of that category.

5.1.1 Recommendation of a friend

In the context model developed, a user has been described by means of an explicit and an implicit profile. As a consequence, the matching value m between two users u_a and u_b is:

$$m(u_a, u_b) = \\ implicit\_profile\_matching(u_a, u_b) + explicit\_profile\_matching(u_a, u_b)$$

The explicit profile The attributes that we can retrieve from the explicit profile in order to compute a matching with the user the system is making the recommendation for are: the age, the gender, the education level, the country of origin, the languages spoken, the friends on FABULA and the interests listed. It is rather random to decide which values of the first four attributes are to be considered as a good match: the user can state beforehand what values he accepts, but the preferences may vary depending on the situation. However, the matching procedures between u_a and u_b adopted for the other types of attributes are:

- Friends: the result of the matching is:

$$friends\_matching(u_a, u_b) = \frac{common\_friends(u_a,u_b)}{friends(u_a)+friends(u_b)}$$

The same approach is used for the languages spoken;

- Interests: the result of the matching of the interests listed by the two users is the sum of two values:
  - the ratio of the interests in common out of the total number:
    $$interests\_matching(u_a, u_b) = \frac{common\_interests(u_a,u_b)}{interests(u_a)+interests(u_b)}$$
  - since two users might use different keywords to express very close interests (e.g., *reading* and *books*), the second value is the ratio of the semantically similar interests out of the total number of items listed by the two user. I argue that the semantic analysis of terms can enhance the traditional matching approaches of RS algorithms.

The lexical database WordNet [30] has been used to define the semantic similarity between two concepts. Several semantic distance measures based on WordNet are available in literature [37]. Mainstream approaches obtain the semantic similarity either by exploiting the relationships connecting the concepts lexicalized in WordNet or by finding their shared information content. The description of Wordnet and the measures for semantic similarity is available in appendix A.

For this work, the semantic distance between two words w_1 and w_2 is computed with the *path* measure [37]:

$$sim(w_1, w_2) = \frac{1}{shortest\_path(w_1,w_2)}$$

where the *shortest_path* is the shortest path connecting the two words considering the is-a hierarchy of WordNet. We opted for this measure because it does not rely on corpora analysis and thus avoids the problem dealing with the sparsity of data, since it is not possible to make any preliminary assumption about the meaning of the terms analyzed.

Figure 5.1 shows an example[1] of a `is-a` hierarchy in WordNet.

Once the similarity values between all the pairs of words are ready, we consider as similar for our purposes the pairs having a similarity value above the following threshold:

$$T = avg\_sim\_value + k * \sigma$$

where *avg_sim_value* is the average similarity value found and σ is the standard deviation. k is a parameter to be set experimentally. In the beginning, we assume k = 1.

[1] http://www.cs.princeton.edu

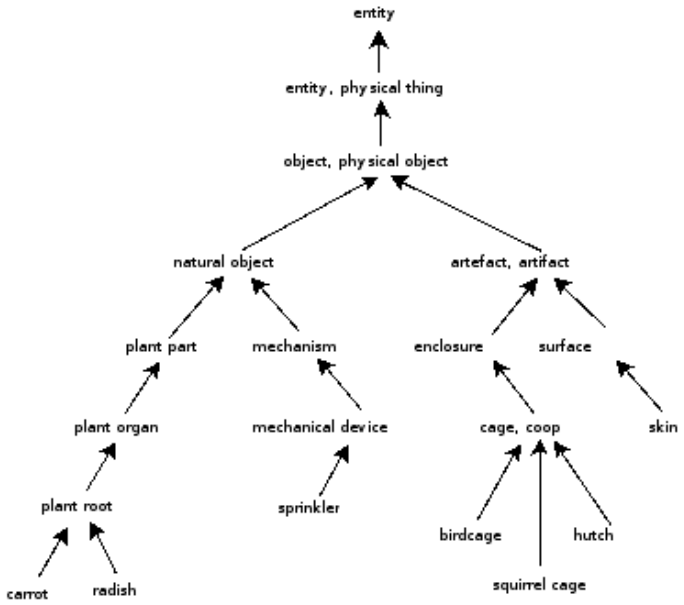

Figure 5.1: An is-a relationship example

The implicit profile The attributes that compose the implicit profile of a user are the history of the activities the user has been engaged in and of the places he has visited, the chat logs with other users and the information from Facebook, if the user is a member.

As far as the history of activities done and places visited is concerned, in both cases a ratio is computed dividing the common items by the sum of the two user's items.

To match the chat history of each pair of users, the two preliminary step on each log file are:

- stopwords removal;
- stemming process.

The output is a list of basic terms that, as before, undergo two matching phases, a keyword-based one and a semantic one based again on WordNet.

The Facebook profile of the user, if existing, contains many pieces of information: it is a social network that the user has been building since some time. It represents a trust network whose weight has to be taken into account. The trustworthiness of the relationships exisisting in that network is already granted. When computing the matching values between the current user's Facebook profile and any other user's one, a way to take this network of relationships into account is to match the current user's Facebook profile with the other users' Facebook profiles, from the point of view and the access privileges of the current user on Facebook. The distance between the users u_a and u_b inside the Facebook network and the mutual visibility (what they can see of each other's profiles) is already a trust value. As a

result, the matching value based on the Facebook information is calculated for each user and involves the following kinds of items:

- the list of friends. In particular, if u_a and u_b are already friends on Facebook, it is likely that they want to be recommended to each other also on FABULA;
- the list of interests;
- the items the user labeled with an *I like it* tag (from now on I will refer to these items as the user's *likes*);

The matching values for the interests and the likes is the sum of two matching phases, one keyword-based and the other based on the semantic distance between the two terms in WordNet, as for the interests listed in the explicit profile.

However, the matching procedure is slightly different from the one applied to the explicit profile. The general formula is:

$$FB\_matching(u_a, u_b) = \sum_{i \in FBItems} matching_i(u_a, u_b)$$

where *FBItems* are the types of items retrievable on Facebook.

In particular, when recommending u_a to u_b, each matching i is computed with the formula:

$$matching_i(u_a, u_b) = \frac{common\_items(u_a, u_b)}{items(u_b \cup items(u_a)} * penalty\_factor$$

The penalty value The *penalty_factor* in the previous formula represents a penalization for a user (u_a) having too many or too few items inside his profile if compared to the general behavior of the users in the system. In other words, in the recommendation process, we want to punish a user whose behavior is too far from that of the others. This penalization has five different levels and, from the opposite perspective, it represents the different levels of trustworthiness of a user inside a social network.

The overall behavior of the users of the system is represented by the mean value of items they have. We applied different statistical approaches to obtain the mean value, each of them encompassing an aspect that we wanted to describe. In particular, given a sorted list containing the number of items each user in the system has, we calculated:

1. the arithmetic mean (or average), used to understand the central tendency;
2. the trimmed mean, meant to smooth the effect of an equal amount of the high and the low ends and avoid the penalization being affected by too high or too small values. We decided to follow a common rule of discarding 25% of the ends;

3. the weighted means, where the weight of a value is the frequency of that value inside the list: for instance, if inside a list of twelve values six users have only one item, the numbers of the first three users are cut by the computation of the trimmed mean, but their value is instead of great importance for the weighted mean since it is common to half of the users.

Based on the number of items of u_a and u_b (from now on *items(a)* and *items(b)*), two different situations can occur:

- *items(a) - items(b)* $\leq$ *0*: in this case, no penalization for u_a is applied because u_b to whom we want to recommend u_a has already more items than him.
- *items(a) - items(b)* $>$ *0*: given the distribution of the values around the mean (e.g., the arithmetic mean), the different levels of penalization are defined by using the standard deviation and u_a is penalized based on the interval in which *items(a)* falls. The penalization is directly proportional to the distance from the mean value.

The standard deviation is a good tool to measure how much a user is far from the average behavior: the intervals L_1, L_2,..., L_5 are then defined as:

$$L_i = |avg - i * st\_dev|$$

where st_{dev} is the standard deviation, avg is the mean obtained in one of the three ways listed and $i \in \{1, 2, ..., 5\}$. See figure 5.2 for an example normal distribution, where each colored band has a width of one standard deviation. As a result, the correlation between u_a and u_b becomes:

$$matching_i(u_a, u_b) * \alpha$$

where $\alpha \in \{1.0, 00.8, 0.6, 0.4, 0.2\}$. In the case of $\alpha = 1.0$ (i.e., *items(a)* $\in L_1$), no penalization is applied.

5.1.2 Recommendation of places and activities

In order to foster the re-usability and independence of the algorithm developed, activities and places are represented with a very similar structure in the developed ontology. This is why, for this work, the same algorithm can be used to handle the recommendation of a place or of an activity to the user. I will now on refer to either places and activities as items. In both cases, the profile consists of:

- a description in free text about the item;
- a rating value ranging from 1 to 10 for the historical, social and cultural attractactiveness of an item: at first, this values are predefined, then they are updated when a user provides a new rating;

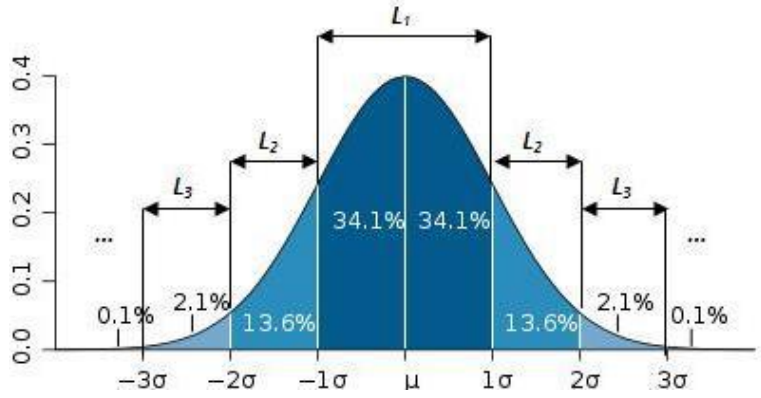

Figure 5.2: An example plot of a normal distribution

– a set of tags, either system or user-defined.

Due to the structure of the profiles, it was necessary to turn the attention to Information Retrieval (IR) models. In general, an IR model is a quadruple:

$$[D,\ Q,\ F,\ R(q_i,d_j)]$$

where:

– D is a set composed of logical views for the documents in the collection considered;
– Q is a set composed of logical views for the user information (queries);
– F is a framework for modeling document representations, queries, and their relationships;
– R(q_i,d_j) is a ranking function which associates a real number with a query $q_i \in Q$ and a document representation $d_j \in D$. Such ranking defines an ordering among the documents with regard to the query q_i.

In the present work, we want to recommend an item to a user: as a result, the profiles of items represent the documents, and the profiles of the users are used as queries.

The procedure to obtain the documents out of the profiles of the items is the following. The description in free text has to be transformed into a list of index words t, by removing the stopwords and stemming the terms left. The output words will be added to the system-defined tags st to construct the document describing each item:

$$d_i = [w_1, w_2, ...w_i, ..., st_1, st_2, ...st_i, ...]$$

The queries are instead constructed out of the profiles of the users by collecting the interests from his implicit ($impl\_interest_i$) and explicit profile ($expl\_interest_i$). Documents and queries are therefore vectors of terms.

$$q_i = [impl\_interest_1, ...impl\_interest_i, ..., expl\_interest_1, ...expl\_interest_i, ...]$$

Such vectors of terms need to be turned into vectors of numerical values. For each term, its *tf-score* is computed. The *tf-score* of term i inside document j is the frequency of i in j.

However, the *tf-score* alone is not enough: it is necessary to add a weight to a term based on how many times it appears out of the total number of terms in the same document. This weight is represented by the *idf-score*:

$$idf_i = \log \frac{N}{n_i}$$

where N is the number of documents and n_i the number of documents in which the term i appears.

The resulting value for each term i in document j is then:

$$TFIDF_{i,j} = tf_{i,j} * idf_i$$

At this point, the recommendation is computed in two different ways, by combining solutions typical of the CF RS and trying to overcome some of their drawbacks:

1. *Cluster-based recommendation*: clusters of similar items are build and the user is recommended those items that belong to the same cluster of the items he already has, weighted by a specific function;
2. *Similarity-based recommendation*: the similarity between the document representing the item and the query representing the user is computed with the *vector model* and the user is recommended the items he is more correlated to, based on a mathematical procedure.

Cluster-based recommendation With this solution, we want to take into consideration the history of the user inside the system. This can be represented as a list of documents (the activated he did or the places he visited) $D = d_1, d_2, ..., d_i, ...$ In this way, as for model-based CF algorithms, a model of the user behavior is constructed.

Let us now consider all the N documents available. The inter-document similarity is found through the TFIDF cosine measure, that allows us to look at the similarity between two documents as the cosine of the angle θ the two n-dimensional vectors d_1 and d_2 used to represent the documents form in an n-dimensional space:

$$sim(d_1, d_2) = cos\theta = \frac{\sum_{i=1}^{n} d_{1i} * d_{2i}}{\sqrt{\sum_{i=1}^{n} {d_{1i}}^2} * \sqrt{\sum_{i=1}^{n} {d_{2i}}^2}}$$

The result of this procedure over every pair of documents is a *NxN* matrix s where each cell *s(i,j)* contains the similarity between the documents i and j.

To build clusters of similar items (vectors) we decided to use a *hierarchical agglomerative algorithm* with *single linkage*. In the beginning, each vector, that is a point in the n-dimensional space, is a single cluster. At each step, every cluster is merged with the closest one. The result of this procedure in two dimensions is a nested cluster diagram, as shown in figure 5.3.

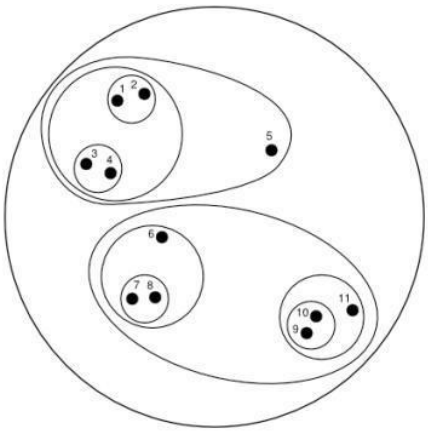

Figure 5.3: An example of nested cluster diagram

The distance *dist* between two clusters C_i and C_j is obtained with the single linkage method, it is to say that *dist* is the minimum of all the similarities between the points (i.e., documents) $x \in C_i$ and $y \in C_j$:

$$dist(C_i, C_j) = \min_{x \in C_i, y \in C_j} dist(x, y) = \min_{x \in C_i, y \in C_j} sim(x, y)$$

In other words, given an initial cluster C_i, the merging criteria is to find the cluster C_j such that the distance $dist(C_i, C_j)$ above is maximized.

A *stop condition* for the algorithm was required in order to find the number of clusters that best fits the purposes of this work. The steps to find it are the following:

- first of all, the number of different types T of items is computed by combining the system-defined tags available;
- thus the heterogeneity H of the environment of the items is given by:

  $$H = \frac{total\_items}{T}$$

  where *total_items* is the total number of items available inside the system (e.g., the number of the places having a profile);
- the dimension that at least one cluster must reach in order to stop the algorithm is then given by the inverse of H.

Once the clusters are available, the user is recommended the items that belong to the same clusters of those in the history of the user.

However, a weight function *w(x)* can to be added, where x is the number of items belonging to the history of the user inside that cluster. The idea is to give a low weight ($\simeq 0$) to the items belonging to a cluster where the user has very few items, and a high weight ($\simeq 1$) to those inside clusters where the user has a lot of items. This can be achieved by using the hyperbolic tangent function taken inside the interval [0,1]. The formula is the following:

$$w(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Similarity-based recommendation This solution is oriented at applying approaches typical of the memory-based CF algorithms, where the recommendation is statistically made out of the behavior of a set of neighbors having a history of agreeing with the user we are making the recommendation for. In this work, this idea is re-elaborated as follows.

In the first phase, we need to find the correlation between the user and the items. As a result, the vector model is applied. Given the vectors of values d_1, d_2, ..., d_n as the documents (i.e., the profiles of the items), and q as the query (i.e., the profile of the user u_a) obtained as described above, this second approach computes the similarity between the user with each document d_k:

$$sim(d_k, q) = cos\theta = \frac{\sum_{i=1}^{n} d_{ki} * q_i}{\sqrt{\sum_{i=1}^{n} {d_{ki}}^2} * \sqrt{\sum_{i=1}^{n} {q_i}^2}}$$

The result of this matching is the cosine of the angle between the two vectors d_i and q. Documents are ranked according to their degree of similarity to the query.

So far, this solution does not take into account the behavior of the other users. In addition, the entire dataset of items has been considered in order to generate a recommendation. As I recalled when describing content-based RS in section 3.3.1, this leads us to stick to what the user explicitly listed, and to lose what might be interesting for him.

As a consequence, in the second phase, $sim(d_k, q)$ has to be weighted according to the vote, or ranking, the neighbors gave to d_k, and to their correlation with the user u_a we are computing the recommendation for.

The first problem deals with finding a neighbor for user u_a, made of the t users that are most similar to the user u_a. It is found by exploiting the procedure described in section 5.1.1 used to recommend a friend to the current user: the neighborhood $N = (u_1, u_2, ..., u_t)$ is therefore formed by the first t users ranked according to their correlation with u_a. For instance, the degree of correlation w between users u_a and u_b is the value $m(u_a, u_b)$.

Thus, we want to introduce the opinion of the neighbors. Let us now recall that, in FABULA, a user can express three ratings ranging from 1 to 10

for an item. A rating can be express for the cultural, social and historical attractiveness for an item.

Finally, for each level of attractiveness, we can define the recommendation value of item d_k for the user u_a as:

$$rec(d_k, u_a) = sim(d_k, q) * \sum_{u \in neighbor(u_a)} \frac{rank(u,d_k) * m(u_a,u)}{n * \sum_{u \in neighbor(u_a)} m(u_a,u)}$$

where n is the total number of users belonging to the neighbor of u_a that expressed a vote for d_k and $\sum_{u \in neighbor(u_a)} m(u_a, u)$ is used to weight the relative correlation between u and u_a.

Figure 5.4 shows a graphical schema of the procedures adopted.

5.1.3 Recommendation of services

The recommendation of a service is quite straightforward. A service for our purposes is required by one or more tasks of an application, and is made available in one or more specific geographical areas. As a result, when a user is trying to accomplish a given task in a location of the city, the system is able to recommend him the services available there to solve the task.

5.2 Implementation of the algorithm

So far, in this chapter I have described the design process of the recommendation algorithm. I will now proceed with the details of the implementation.

As mentioned, the algorithm was realized as independent of the underlying architecture. Java[2] was therefore chosen as a programming language.

Interface with the model of context The first thing to do is to provide an interface between the algorithm and the ontological model of context, in order to allow bidirectional communication and the possibility to capture the dynamicity of the changes in the context. The Jena Framework[3] was used for this purpose. See appendix A for detailed description of the software. The ontology developed with Protégé in OWL format is then loaded inside the `Ontology Model` created with Jena.

Afterwards, individuals of every dimension of context have to be added to this model.

A user or an item in a Jena `Ontology Model` is represented as an instance of the `Individual` class. Once an individual is created, properties can be added to it as instances of the classes `DatatypeProperty` and `ObjectProperty`.

[2]http://www.java.com
[3]http://jena.sourceforge.net/

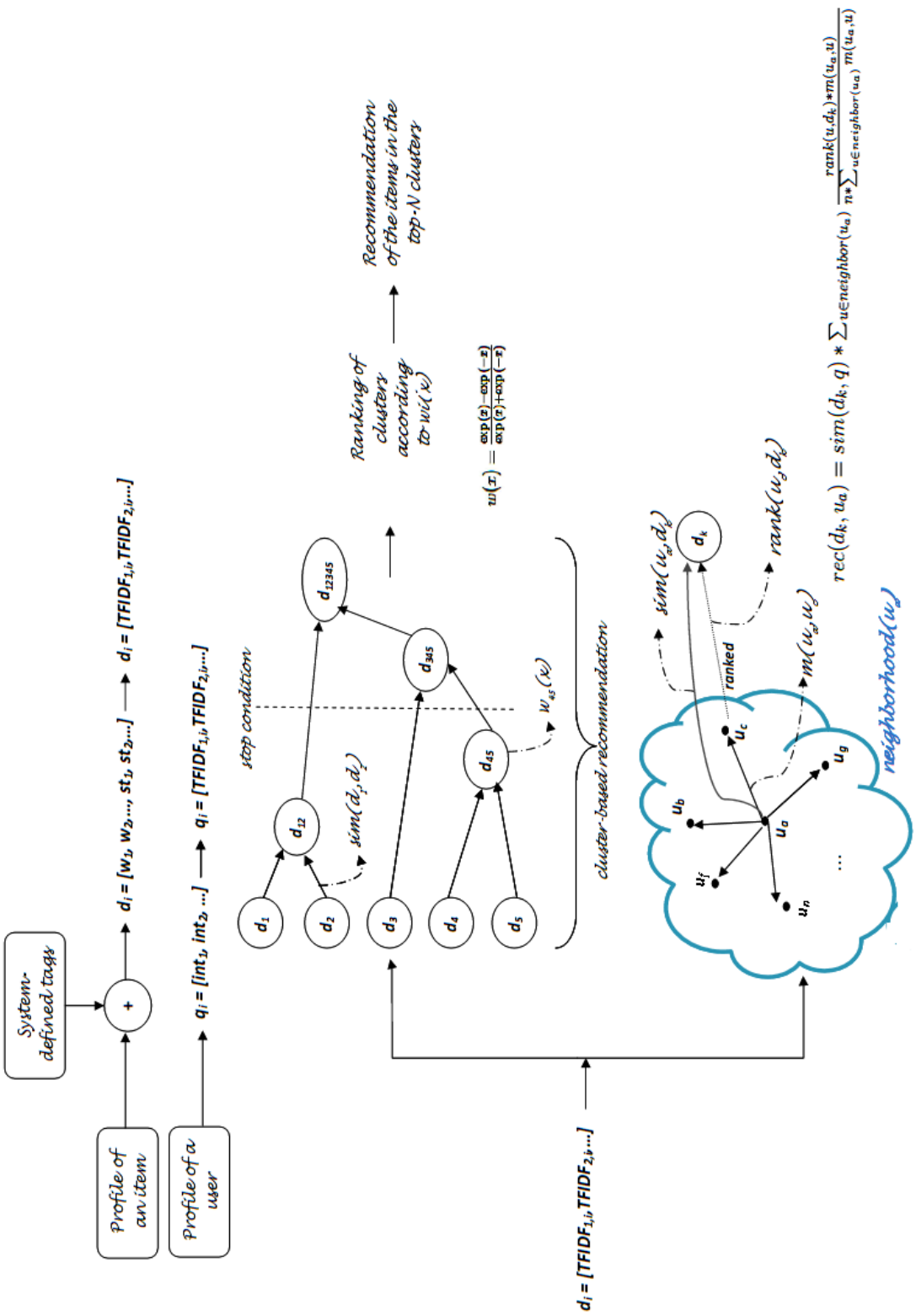

Figure 5.4: Schema of the similarity-based and cluster-based recommendation of items

Data about the users and the items are stored in different formats, based on their nature and their source: the basic information about each user (see the *Explicit Profile* class in section 4.1.1) is put in a database and later exported in an XML file, whereas the history of the activities done and the places visited and the chat logs are in text files.

Such files are parsed according to their format. It was necessary to develop a set of functions to parse the XML and text files. The latter format required also an additional preprocessing so to obtain a list of strings out of the free text. Functions to remove the punctuation and the stopwords and to stem the remaining terms were used for this purpose.

The information found on Facebook is instead handed in a separate way that I will describe later.

The pieces of information collected are thus inserted as either datatype or object properties of the instances of the Jena `Ontology Model`. Figure 5.5 shows the schema of how data are handled.

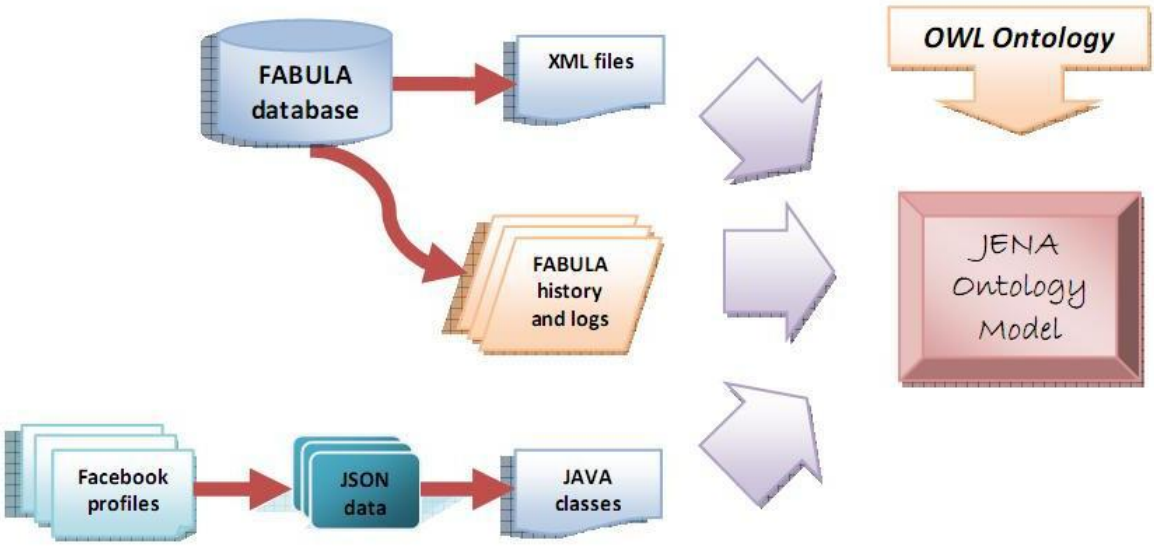

Figure 5.5: Schema of the data handling in the algorithm

Matching the users At this point, users are inserted in a hash table, a solution aimed to improve the speed of execution of the algorithm.

Every user is matched with every other user: the output is a user-user matrix *r*, where each cell *r(i,j)* is the sum of the matching values computed as I described in the previous section. At an implementational level, a different module for each type of data was realized.

Programmatically, data belonging to the FABULA profiles and data coming from Facebook are analyzed separately. This means that *r(i,j)* is:

$$r(i,j) = FABULA_{Score} + FB_{Score}$$

where $FABULA_{Score}$ is the output of the function used to match the information about two users that deals with the activity of the user inside FABULA, and FB_{Score} is the result returned by the function used to match the Facebook data.

Such division of the matching computation was chosen to enhance the modularity of the algorithm: since Facebook is just one of the many popular social networks available, if another one is used, there are no consequences over the general procedure but only the method returning the FB_{Score} has to be substituted.

Facebook data management The computation of the score resulting from the comparison between the Facebook profiles required an additional set of methods necessary to connect to the Facebook site and to handle the data found there.

The *Graph API*[4] is the tool provided by the Facebook Platform enabling an application to read and write data to the social graph. It provides a uniform view of the objects (e.g., users, pages, sets of interests, photos, etc...) and the connections between them (e.g, likes, tags, friendships, etc...): they can be fetched through their unique ID. The results are returned as JSON[5] objects, a lightweight data-interchange format, and are then wrapped into specific Java classes so to later handle them in the algorithm.

The *Graph API* uses the *OAuth 2.0*[6] protocol for the authentication and authorization of a user. It is out of the targets of this thesis to investigate the management of the user authentication. At a high level, it means that ofter the user logs in to Facebook, he is returned an *access token*, that can be used by the application to make authorized requests on behalf of that user.

Afterwards, given the ID of a user and the access token, an entry point must be created for the application as a `FacebookGraph` object. Since this process turned out to be computationally heavy, the instances of each user's `FacebookGraph` were stored in a hash table, so to quickly pick them when required by the algorithm.

Semantic matching As stated in the previous section, both the $FABULA_{Score}$ and the FB_{Score} are obtained through a keyword-based and a semantic matching of the profiles. As a result, a further class was therefore developed to manage the semantic analysis and comparison of the data. To foster reusability and modularity, items that have to be semantically matched are

[4]http://developers.facebook.com/docs/api
[5]http://www.json.org/
[6]http://developers.facebook.com/docs/authentication/

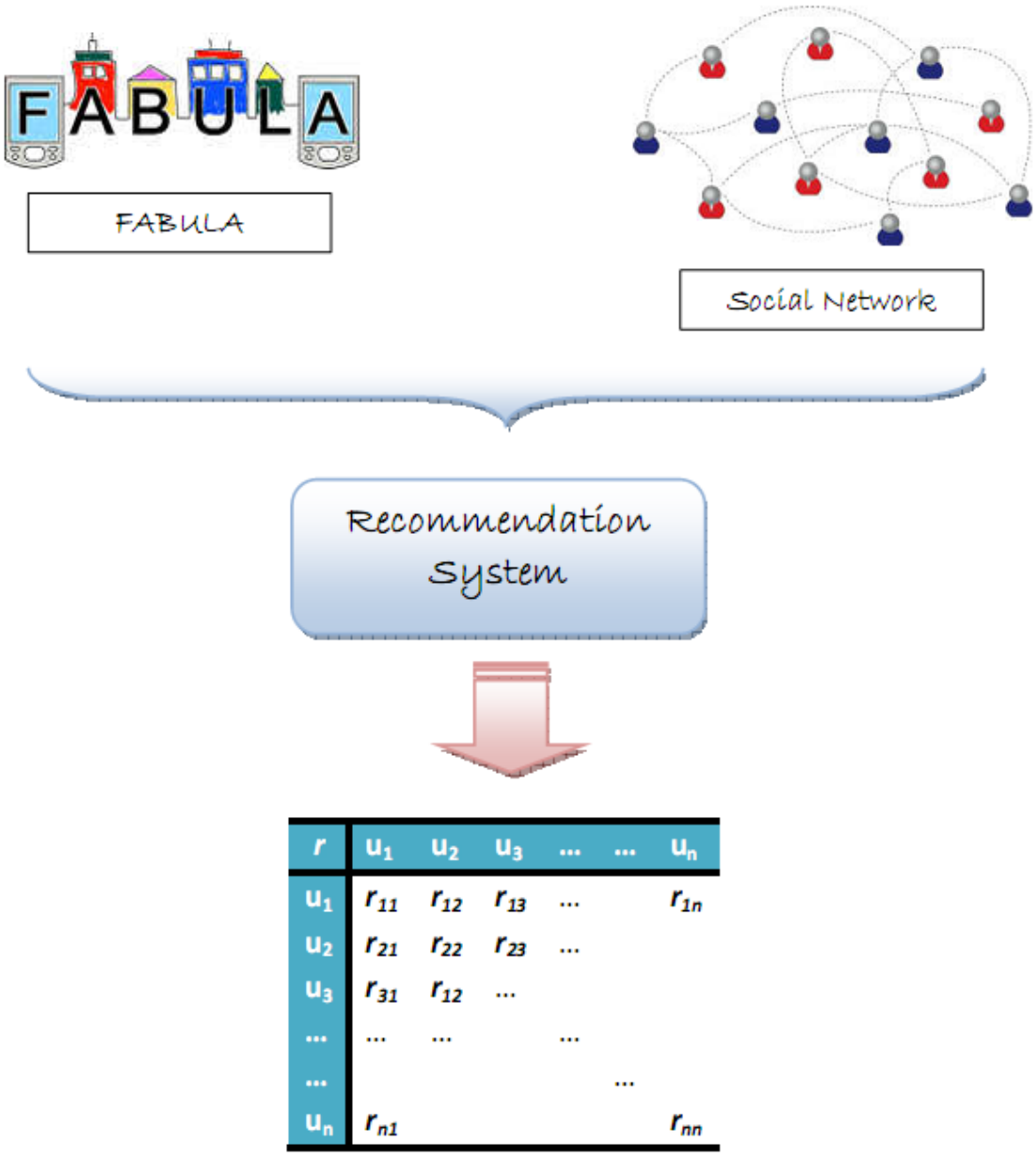

Figure 5.6: The integration of FABULA with a social network for the recommendation

transformed into strings and the method for the semantic matching computation takes two generic `ArrayLists` of these strings as input and returns the matching score.

As introduced, the semantic analysis was carried on by means of WordNet. Among the several APIs available online to connect to its graph and browse the relationships between words, I chose to rely on the *JavaWordNet Library*[7] (from here on referred to as JWNL) for the following reasons:

- it is written in pure Java code and is completely portable and independent of the platform of use;
- it is a project directly related to WordNet, which guarantees a good interaction with the lexical database;
- it makes the access to WordNet very easy: it is sufficient to create a link to the local installation of WordNet by instantiating the `Dictionary` object;
- it contains a simple and intuitive set of methods to navigate WordNet.

By means of JWNL, an instance of the WordNet dictionary is loaded and, if the given pair of strings are found, the shortest distance among them is found inside the graph. A description of how words are organized and the relationships among them is available in appendix A.

[7] http://sourceforge.net/apps/mediawiki/jwordnet/index.php?title=Main_Page

CHAPTER 6

Evaluation of the results

6.1 Preliminary information

In chapter 2, I have described the working scenario and the dataset available, which the algorithm developed was tested upon. I argue that a group of seventeen users, eleven of which with a profile on a social network like Facebook, active online on FABULA, is a rather likely scenario for a mobile learning application designed to be used inside the central area of the city of Trondheim.

However, the majority of my contribution lays in the design phase: the testing and the evaluation I will hereby explain are mainly meant to assess the impact of the choices made, by measuring the different matching values, or correlations, between users that the system can provide.

In fact, the recommendation will be carried on by means of the correlation m between two users. In other words, the problem of deciding whether to recommend u_b to u_a is solved by computing how the profiles of u_a and u_b are correlated, or similar, with the procedure described in the previous chapter.

My purpose here is to evaluate the performance of the recommendation system and the quality of the solutions adopted. My idea is to focus on one category of context so to weight up all of the relevant aspects: as a result, I will only analyze the case of the recommendation of friends to a user. Let us recall that in section 3.3.2, given a list of items $I = i_1, i_2, ..., i_n$, a recommendation to a user u_a is defined as the list of the top-N items $i \notin I_{ua}$ that u_a is assumed to like the most.

Specifically, I aim to show the effects of the following techniques:

- the benefits of the exploitation of the information coming from a Social Network like Facebook;
- the penalization procedure for a user on Facebook: firstly, its contribution to the overall matching values is tested;

- the impact of the point of view of the user inside Facebook: when computing a list of friends to recommend to a user, there is no central view over the Facebook social network, so the matching values vary based on the connections the user has;
- the semantic matching procedure based on WordNet: its impact to the final result is studied;

I did not fix a threshold to state if a recommendation of a friend to a user has to be made or not, because I wanted to investigate the trend of the results by applying the formulas introduced in the previous section and their variation due to the combination of the different techniques presented. The recommendation threshold, or the number N of friends to recommend, can be set after having a clear idea of the behavior of the system over the whole group of users.

6.2 Illustrative scenario

Nestore is a new exchange student at NTNU, in Trondheim, Norway. He arrived few days before the start of the semester, so he has some free time to visit the city and learn about it. He just installed the FABULA application and created his user profile. He does not know what to expect, so he just inserts the pieces of information about himself that occur to him: his age, gender, city of birth, languages spoken, education level and a short list of interests. Being new in town, Nestore has no friends yet on FABULA. On the server side, a new instance of the `Individual` class inside the FABULA ontology is created, and new properties are added to it. Data about users are stored separately from the OWL files containing the structure of the ontology.

Nestore then takes a stroll from his house towards the center and sets his status to *visible*. The user agent running on Nestore's mobile phone is identified by a unique ID and periodically sends to the server the updated position inside the city.

He has previously read online about the interesting places to visit in Trondheim: being interested in history, he starts visiting the things inside the city that he thinks are most relevant for him (*mobile learning*). For the time being, three learning places have been identified inside the center of Trondhiem and represented as boxes, where the longitude and latitude of the vertices is given. Through the geographical position, the user agent senses that Nestore has crossed the boundaries of one of the predefined learning places. In particular, he is approaching a learning opportunity, the Nidaros Cathedral. There, activities are available and the application advertises the possibility for Nestore to get engaged in some of them. He first chooses to play a quiz asking general questions about the history and the architecture of the cathedral, that requires the user to be located outside. After successfully completing the game, Nestore requests to play a treasure-hunt game:

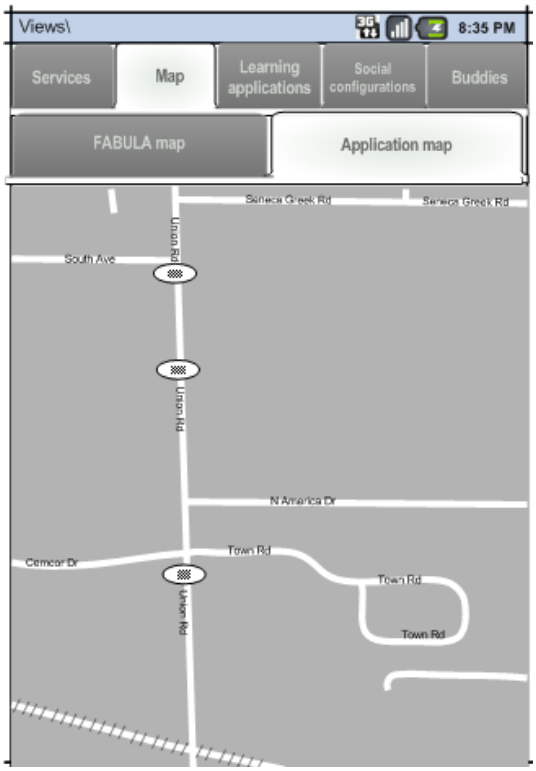

Figure 6.1: A screenshot of the map of the area around the user

when enough FABULA users are available to form at least two groups of players, the application tells him that the game is starting.

Now Nestore has a list of places he has visited and a list of activities he has done, but he wants to look for new friends. He therefore requests the *Friend suggestion* service to find some good matches for him. The service asks if Nestore wants to look for people inside the same learning zone, or in the whole city. He choses the second option.

When the *Friend suggestion* service is started, the recommendation algorithm computes the correlation between Nestore's data and other users' data.

But Nestore has also a profile on Facebook: the service then asks Nestore the authorization to access it, and Nestore grants the permission. Data about him are browsed: in particular, he has 37 friends, 15 items tagged as *I like it*, but no interests listed.

A good match for him is found, for instance, in the user Agrippina (*serendipitous interaction*): she did the same activities as Nestore, played the treasure-hunt game with him and she also visited two monuments that he really liked, too. They both speak English and Finnish. In addition, they are already friends on Facebook and have some items in common there. No penalty value is assigned to Agrippina since the number of items she has inside Facebook is close to the average of the other users from Nestore's point

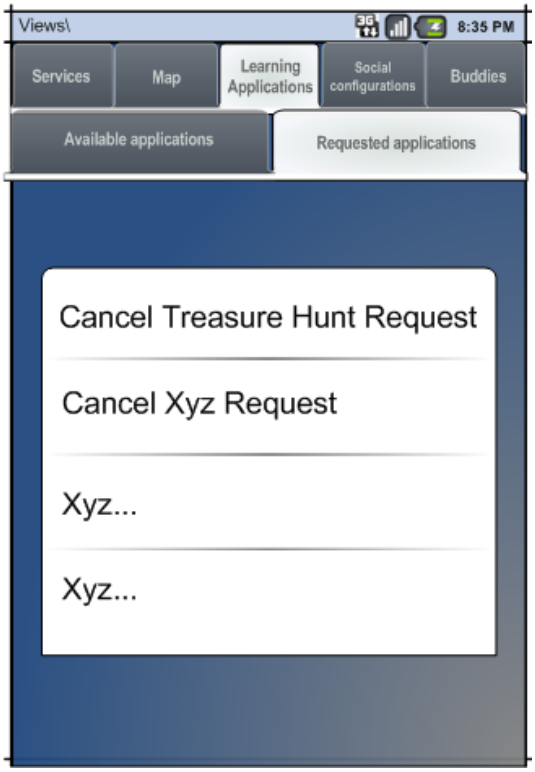

Figure 6.2: A screenshot of the waiting list of the services requested by the user

of view. As a result, their mutual matching score is of 1,435 points out of a average of 1,083 points of the matching values computed for Nestore with all other users; in particular, the Facebook correlation accounts for the 10% of this number. It is also intersting to notice how the overall correlation between Nestore and Agrippina is improved of the 20% thanks to the semantic-based similarity computation, when the average weight of this procedure is of 4% out of all the users and 5% out of only the Facebook members.

In the end, all users are then ranked according to their similarity with his profile, as shown in figure 6.3. Nestore can thus choose to have a look at what he has in common with any of them, and then decide to ask someone for friendship.

6.3 Impact of the Social Network

When presenting the choices made in order to represent the user class in the model of context in chapter 4, I have argued that looking for information about the user on some Social Network would help avoiding a redundancy in the information building. But is this actually useful in order to find an interesting match (or set of matches) to recommend to the user?

	Nestore
Agrippina	1,4355322
Pompeo	1,2873017
Romeo	1,284844
Licinio	1,269192
Miciu	1,2556261
Augusto	1,2203021
Edoardo	1,2162699
Calpurnia	1,2020202
Elena	1,1663283
Enrico	1,1432059
Munazio	1,0801587
Ottaviano	1,0178571
Noa	0,9787157
CaioGiulio	0,9777778
Silla	0,9615531
Caio	0,9143218

Figure 6.3: Users ranked according to their correlation with user Nestore

The first step to justify this choice is to analyze the behavior of the general algorithm. The output is a recommendation matrix r, where each cell *r(i,j)* is the sum of the matching of the explicit and implicit information about a user.

At a first glance, we see that the matrix is not symmetrical, as shown in figure 6.4: we have $r(i,j) \neq r(j,i)$. In general, recommending user u_a to user u_b is not the same as recommending u_b to u_a. This is due to the way the information from Facebook is used, and in particular to two reasons:

- the penalization is thought to be an asymmetric tool: if the algorithm is matching u_a and u_b in order to recommend u_b to u_a, then u_b can be penalized according to certain conditions. At opposite, when recommending u_a to u_b, it is not said that u_a will receive the same penalty value;
- what a user sees of another user depends on the other user's visibility settings: if u_a and u_b are not already friends on Facebook, it may happen that u_a sees everything of u_b, but u_b only sees little information about u_a.

Each user is matched with all of the other users inside the system, and the average of the output correlations is split in what is the result of the matching of the user information retrieved from FABULA and what is the outcome of the Facebook data comparison. If a user is not on Facebook, the matching relies only on the data on FABULA.

Figure 6.5 shows the percentages of contribution provided by the keyword and semantic-based comparison of the data coming from Facebook out of the total correlation of one user with all others.

We observe that this solution increases the correlations of the 11,17% on average for the users having a profile inside this Social Network. For instance, the Facebook analysis improves the correlation with the other users up to the 21% on average for user u_{10}, reaching a peak of the 41% when matching him with u_2. This means that, throught Facebook, u_{10} has found in u_2 a rather interesting match, that he perhaps would not have been aware of if only the FABULA information was considered.

6.4 Impact of the penalization procedure

In general, data coming from a Social Network (be it Facebook or any other one) where the user has been active since some time are affected by three main problems:

1. they can become a great amount without the user being aware of it: think, for instance, of all the items labeled with an *I like it* tag;
2. they are prone to be very sparse, i.e. belonging to many different uncorrelated domains;

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}	U_{11}	U_{12}	U_{13}	U_{14}	U_{15}	U_{16}	U_{17}
U_1	0	1,5046452	0,941342	0,8650793	0,9540044	1,0436509	1,3468254	1,5793798	1,2333333	0,92584	1,0678571	1,2203021	1,4544765	0,7490843	1,1166667	1,0182900	1,0992424
U_2	1,3078386	0	0,7583334	0,6607143	1,3964287	1,1051338	1,1666666	1,0184344	1,0734128	1,2714286	0,8472223	1,4355322	0,8777168	0,8388278	0,9207603	0,9493350	0,9263492
U_3	0,941342	0,7583334	0	0,7845238	1,0503968	0,4222222	0,8666667	0,5333333	0,7429293	0,769481	0,7992424	0,9143218	0,5467033	0,5205128	0,5861110	1,0333334	0,3650793
U_4	0,8650793	0,6607143	0,7845238	0	0,6345238	0,6984127	0,8095238	0,8289682	1,0107143	0,670635	0,702381	0,9777778	0,667735	1,0903541	1,0277778	0,7928571	0,6949495
U_5	0,9540044	1,3964287	0,6345238	1,0503968	0	0,8777778	1,2440476	1,0528139	1,1047981	0,951623	0,86829	1,2020202	0,7947802	1,1185898	1,2718254	0,7361111	0,6944443
U_6	1,0079366	1,0352037	0,4222222	0,6984127	0,8777778	0	0,8747159	1,0995215	0,6944445	0,711667	0,8722222	1,1432059	0,7095239	0,7095238	0,9294643	0,8450717	0,6868226
U_7	1,3468254	1,1666666	0,8666667	0,8095238	1,2440476	0,8727809	0	1,3141414	1,2555556	1,1071429	1,2361112	1,2162699	0,9752747	0,8062271	1,0694444	0,7972222	0,8095238
U_8	1,3714287	0,9472222	0,5333334	0,8289682	1,0528139	1,0723982	1,3262626	0	1,0511906	0,849206	1,1333333	1,1663283	0,7285715	0,7285714	0,8409092	0,6900793	0,9797981
U_9	1,2333333	1,0734128	0,7429293	1,0107143	1,1047981	0,6944445	1,2555556	1,0511906	0	1,1761905	1,0111111	1,2691920	0,9991453	0,7991453	1,2699494	0,7325758	1,0164142
U_{10}	1,0140756	1,4047620	0,7694805	0,6706349	0,9516233	0,7916666	1,2182540	0,9901155	1,1761905	0	1,0214286	1,2556261	1,2964286	0,6761905	1,0063283	1,2615577	0,8592857
U_{11}	1,0678571	0,8472223	0,7992424	0,702381	0,86829	0,8722222	1,2361112	1,1333333	1,0111111	1,0214286	0	1,0801587	0,6126374	0,8959707	1,1666667	0,6520202	0,5873016
U_{12}	1,2203021	1,3797183	0,9143218	0,777778	1,2020202	1,1432059	1,2162699	1,2615114	1,2691920	1,1998122	1,0801587	0	0,9787157	1,0178571	1,1761905	0,9766955	0,9615531
U_{13}	1,3556530	0,9777168	0,5467033	0,6677735	0,7947802	0,7539683	0,9752747	0,8941558	0,9991453	1,4107143	0,6126374	0,9787157	0	0,8845238	1,1200919	0,9571494	1,1457143
U_{14}	0,7490843	0,8388279	0,5205128	1,0903541	1,1185898	0,7095238	0,8062271	0,7285714	0,7991453	0,676191	0,8959707	1,0178571	0,8845238	0	1,0750000	0,6538462	0,8186508
U_{15}	1,0333333	0,9549708	0,5861111	1,0277778	1,2718254	0,9294643	1,0694444	1,1128954	1,2699494	1,1367377	1,1666667	1,2873017	1,0885130	1,0750000	0	0,9690948	1,1506377
U_{16}	0,9668615	0,9292578	1,0333334	0,7928571	0,7361111	0,8700717	0,9972222	0,8764224	0,7325758	1,3170040	0,6520202	1,284844	0,9085008	0,6538462	0,8344794	0	0,8755051
U_{17}	1,0628788	1,0920635	0,3650794	0,6949495	0,6944443	0,6868226	0,9345238	1,1183064	1,0164142	0,953571	0,5873016	0,9615531	1,1571429	0,8186508	1,1668539	0,9255050	0
AVG :	1,0292844	0,9980686	0,6599211	0,7725177	0,9383809	0,7966886	1,0202112	0,9760644	0,9670648	0,949322	0,856115	1,0830004	0,8635523	0,7872279	0,9752070	0,8229849	0,8041924

Figure 6.4: The overall recommendation matrix

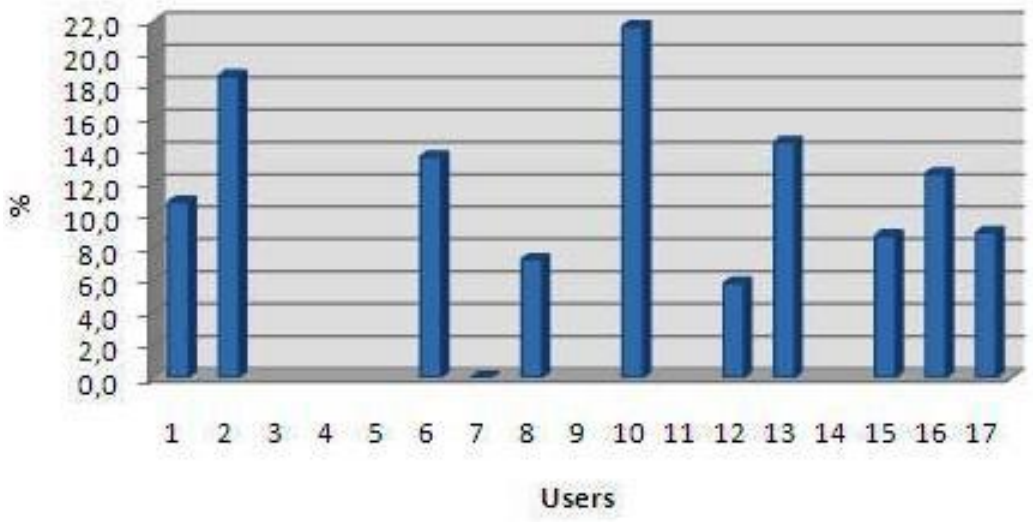

Figure 6.5: The % improvement on the matching values owed to the Facebook data matching

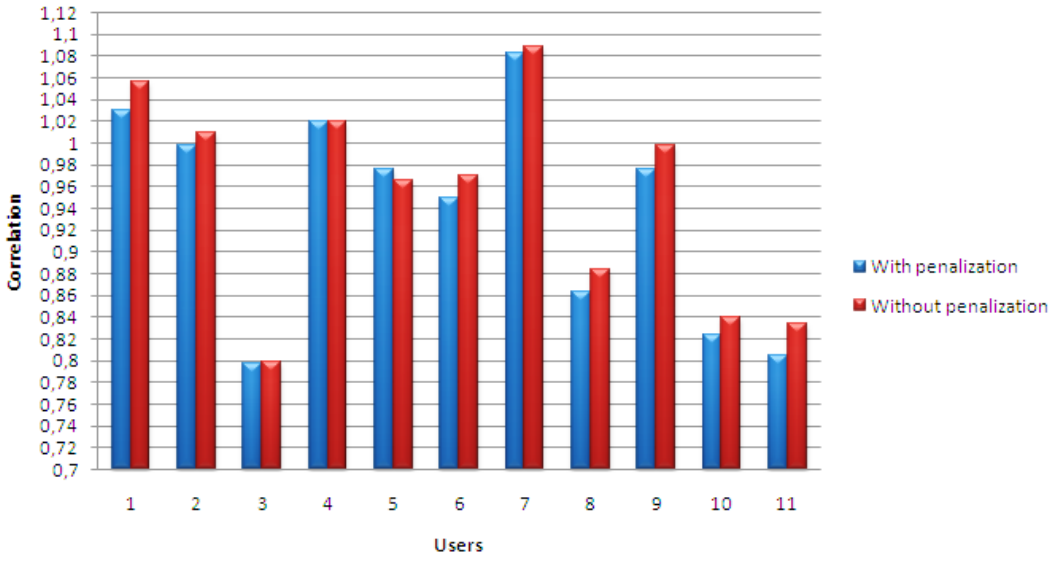

Figure 6.6: Average correlations for each user with and without the penalization procedure

3. at opposite, they can be very few, meaning that the level of activity of the user has been too low and his data become less interesting.

Points 1 and 3 are the main reasons why the penalty factor was introduced. To test its effectiveness, I compared the behavior of the recommendation algorithm with and without applying it. Figure 6.6 shows the different average correlations obtained for the Facebook users only, since this procedure is meant to act on their activity inside Facebook.

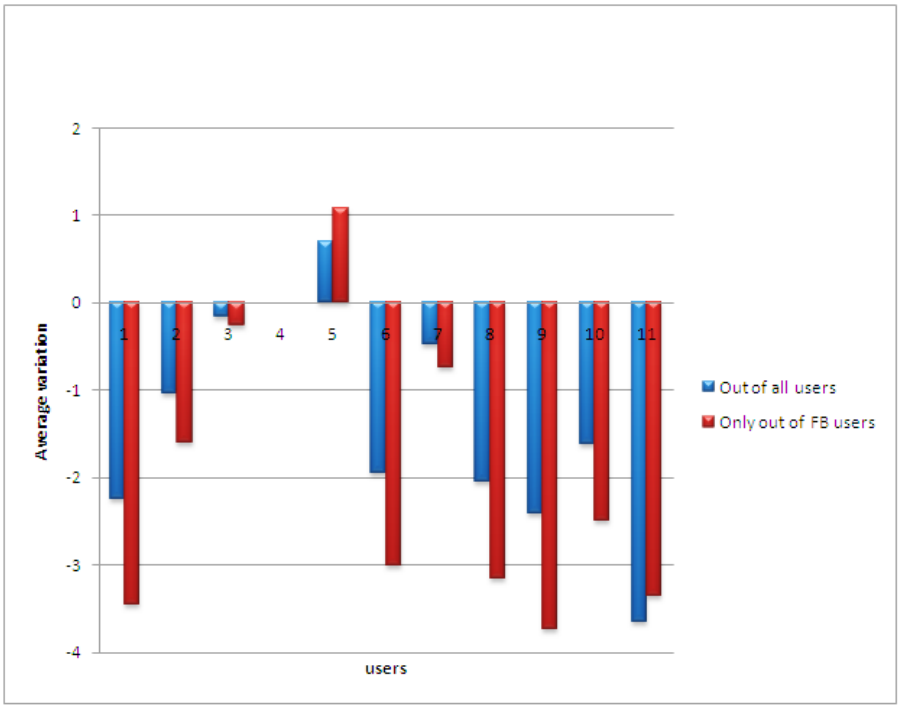

Figure 6.7: Average % variations of the correlations out of all the users and out of the Facebook members without applying the penalty factor

Figure 6.7 shows the percentages of variation of the average correlations of one user with the others, obtained without applying the penalty faction. First, all of the users in the FABULA dataset are considered, like before, then only the Facebook members. In the first case, we clearly see that adding the penalization lowers the correlations between users. Only in one case we obtain a positive variation. On average, adding the penalty value diminishes the correlations of 1.90% for the Facebook members and of 1.30% for all users. Only the first value is interesting for us anyway, because the penalty factor is applied to a user beased on his activity inside Facebook, and not inside FABULA.

By looking at the single averages, we see that the results are, in general, centered around this mean value in both cases.

This observation means that users with a number of items that is much higher or much lower if compared to any other user's are, on average, not

effectively affected by this procedure: if they get the same loss in their average correlation with other users, it might mean that the penalization procedure is not able to solve the problems it was thought for.

Nonetheless, to better understand the reasons of this phenomenon, let us analyze the percentages of variation of every single matching value between Facebook users, as of figure 6.8: we discover significant percentages of loss in the correlation between each user and other users we want to recommend to him due to the penalization. Such values are finally smoothed if considered on average, because they are affected by many null variations. This can be explained because each user has a view over the situation of the other users that is limited by their privacy setting.

	Augusto	Agrippina	Enrico	Edoardo	Elena	Miciu	Nestore	Nor	Pompeo	Romeo	Silla
Augusto	0	0	0	0	0	0	0	0	0	0	0
Agrippina	-12,8	0	0	0	-4,4	-10,5	0	-11,4	-5,4	-3,0	-17,9
Enrico	0	-6,8	0	0	0	-11,2	0	-6,3	0	-3,0	0
Edoardo	0	0	-0,2	0	16,3	0	0	0	0	0	0
Elena	0	0	-2,5	0	0	0	-8,2	0	0	0	0
Miciu	-1,2	0	0	0	0	0	0	-8,8	-17,7	-7,0	-11,0
Nestore	0	-4,1	0	0	0	-4,7	0	0	0	12,9	0
Nor	-7,3	0	0	0	0	0	0	0	0	0	-1,0
Pompeo	-8,0	-1,7	0	0	0	-4,2	0	-2,9	0	-1,6	-1,4
Romeo	-5,3	-5,2	0	0	0	-2,5	0	-5,4	-18,0	0	-5,7
Silla	-3,4	0	0	0	0	0	0	0	0	0	0
AVG VARIATION(%):	**-2,2**	**-1,0**	**-0,2**	**0**	**0,7**	**-1,9**	**-0,5**	**-2,0**	**-2,4**	**-1,6**	**-3,7**
AVG VARIATION FB USERS (%):	**3,5**	**1,6**	**-0,2**	**0**	**1,1**	**-3,0**	**-0,7**	**-3,2**	**-3,7**	**-2,5**	**-3,4**

Figure 6.8: The percentages of variation in the correlation values due to the penalization procedure

To now investigate how this penalization works, let us consider the example of our illustrative scenario. By looking at the users' profiles, a good match for user Nestore would also be user Elena. They have 32 friends and 7 items tagged with *I like it*. However, from the point of view of Nestore, the average number of friends of all the users is 3.5, with a standard deviation of 56, and the average number of liked items is 8.5, with a standard deviation of 38. Such high values for the standard deviation are due to the high distribution of item numbers of the users. Elena has been very active on Facebook, and she collected so far 332 friends and 205 liked items: as a result, the algorithm applies the highest penalization in both cases, i.e. 0.2, and the correlation between her and Nestore is lowered of around 8.2%, as we can see in figure 6.8.

6.5 Effects of the semantic matching

Another approach adopted in the recommendation algorithm whose usefulness I want to test is the matching procedure based on the semantic distance between concepts in WordNet.

Let us now get back to the illustrative scenario described in section 6.2. As we said, thanks to the semantic matching procedure the correlation between Nestore (u_{12}) and Agrippina (u_2) was increased of 4% out of all the users and 5% out of only the Facebook members. We see that the keywords have an average distance in the is-a hierarchy of WordNet of 9.34 if coming from the FABULA profile, and of 12.89 if coming from the Facebook profile. With the path method we decided to adopt for the semantic similarity computation, we register an average similarity of 0.083 between the Facebook concepts and of 0.139 between the FABULA concepts.

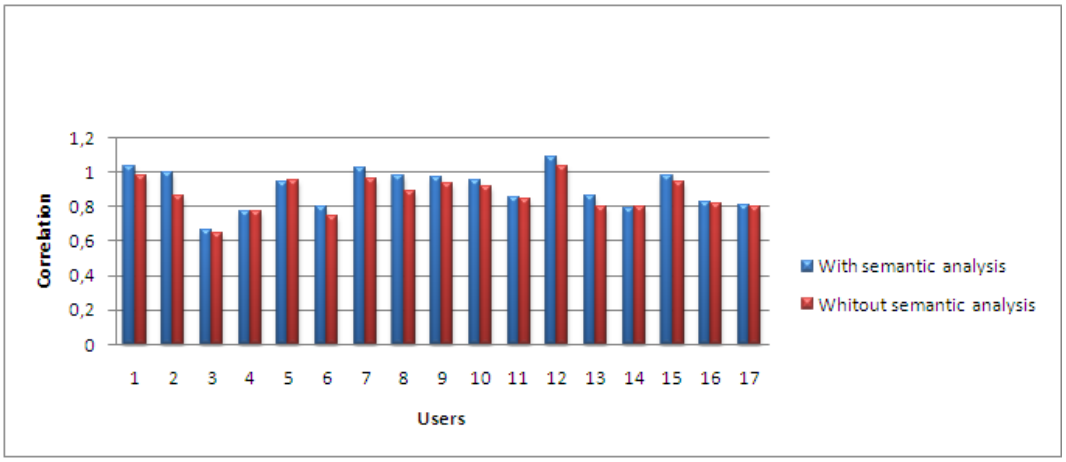

Figure 6.9: Average correlation values with and without semantic matching procedure

Figure 6.9 sheds a light on how the average correlation of one user with all the others is diminished if no semantic matching is performed on the concepts representing the items retrieved from FABULA and Facebook. If we now go into the details of these results and analyze the contribution of this procedure first on the FABULA data and then on the Facebook data, we see that it becomes particularly useful in the second case. In fact, the average improvement in the correlation values out of all the users owed to the semantic matching phase is of almost the 4%. For the users with a Facebook profile, it raises up to the 6.60%. In figure 6.10 the variations in the algorithm behavior if no semantic matching is performed are shown, out of all the users and only out of the Facebook members.

The reason for this increase in the semantic matching values on the Facebook data lays in the fact that inside Facebook the users have generally many more items than on FABULA, which a smaller network.

This type of analysis is however affect by one problem: not all of the keywords used for example for the liked items are retrievable in WordNet, mainly because:

- they are written in languages other than English;
- they are proper nouns or addresses of websites;

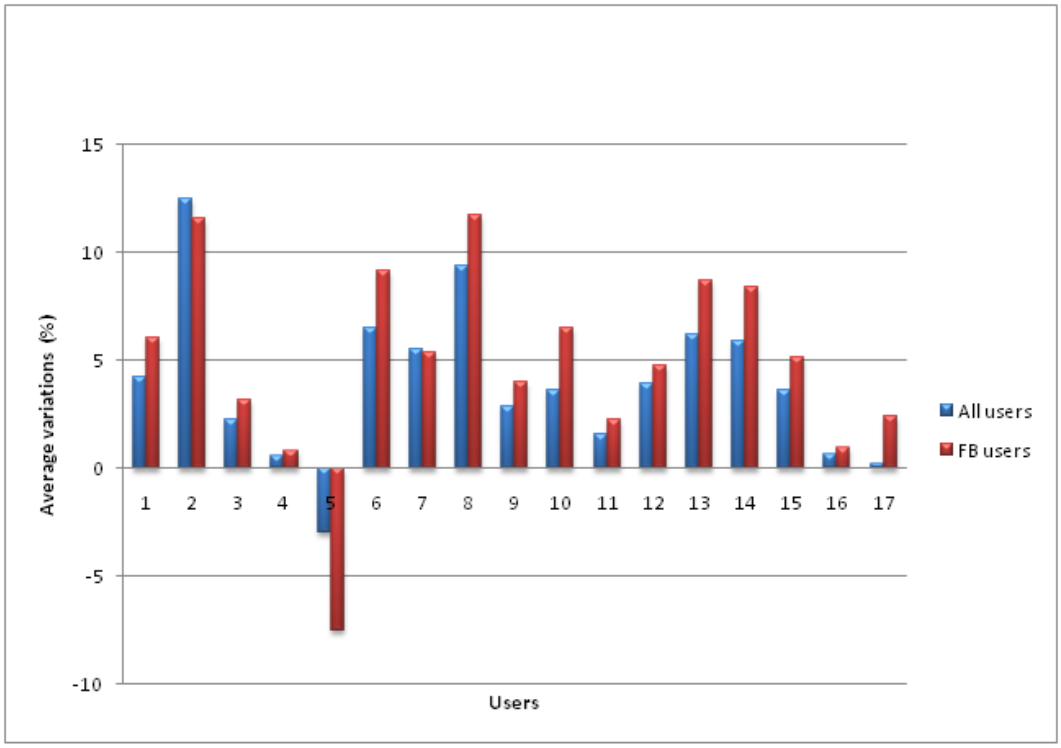

Figure 6.10: Average variation without semantic analysis out of all the users and out of the FB members

– they are written in strange, unpredictable formats and can contain special characters.

As a consequence, in these situations the correlation computation only relies on the keyword-based matching, which is a rather good tradeoff to handle this type of items.

CHAPTER 7

Conclusions and future work

This work has been developed as part of the FABULA project, aimed at finding novel solutions for supporting collaborative mobile learning activities in a city-wide context. It is mainly composed of two parts: the development of an ontology-based model of context and the design and implementation of an algorithm for a recommendation system grounded on the model of context.

As I said, the development of a context-aware framework involves four steps: raw information must be wrapped into a common format, and then they must be gathered and processed into a general representation of the context surrounding the application and the user. This information has then to be filtered according to a relevance assignment process, and finally presented, or suggested, to the user.

This thesis dealt with the second and fourth step of this procedure, even if, at a fist stage, a wrapping of the raw information had to be done.

First of all, a discussion about what context is according to the leading literature and what are the best ways to represented it was carried on. There is no common view in literature about these questions, nor is there a solution to the problem of assigning a relevance value to the incoming piece of information that suits our requirements.

Secondarily, I overviewed some relevant approaches to build a recommendation systems able to suggest content from this context to the user.

Context has by no means to be treated as a specific kind of knowledge. As a consequence, we decided to represent it with four main categories, or dimensions: the users, the places, the activities available in those places and the services provided for an activity inside a place. So, information about these categories had to be modeled: the best solution to do that seemed to be by means of an ontology, because it is a tool that enhances knowledge representation and reasoning techniques. It allows us to achieve a high level, formal representation (therefore re-usable and platform-independent) of the knowledge the application acquires and to address the dynamicity of the interaction with the physical world.

The advantage of such ontology-based model of context is that it is general enough to be exploited for different purposes. However, an open issue still deals with the phase of information filtering: I assumed that a *relevance engine* is in charge of adding a weight to the pieces of information retrieved by the application, since it was not possible to find the right solution to this problem that would probably require a separate, long analysis.

Among the dimensions of context, a detailed representation is provided for the users by merging the basic ontology with a specific ontology based on FOAF, the de-facto standard to represent information about one person and his connections inside other Social Networks. A suggestion for the future work is to find a specific ontology to also describe places, and in particular to achieve an efficient the realization of a representation of a place for one user out of a common physical space.

In the second part of this work, I carried on the design of a recommendation system, able to present and suggest contents to the user. A separate procedure has been realized for treating every dimension of context, and solutions adopted encompassed many techniques: free text processing, WordNet-based semantic matching of keywords, adding an interface to Facebook, clustering of documents and similarity computation based on the vector model. This algorithm was actually implemented in Java language and a set of tests have been done for the case of the recommendation of a friend to a user. A dataset of users' and items' profiles was realized for this purpose.

In particular, three methods used for the matching of the information about users where tested:

- the search for data about the user inside Facebook: thanks to the matching of the pieces of information found in the user's profile on Facebook, the average correlation between users is raised of 11% on average, reaching peaks of 41%;
- the introduction of a penalty factor for a user being too much or too little active inside the system: in this case, the level of activity of a user was defined in comparison to the average behavior of the others. Penalization occurs at five stages based on the standard deviation, directly proportional to the distance from the average value. It can be seen as a process to assign a trustworthiness to a user. Penalization values affect the correlation between user of up to 1.90% on average. By looking at the single matching values, we see that they are somethimes heavily hit by this process, but are smoothed if considered on average by the presence of many null values;
- the semantic matching of terms to find similarities between concepts in the users' profiles otherwise undisclosed: in this way, average correlation is increased to up to 6.60%, because new commonalities are discovered.

In the future, it might be interesting to investigate different ways to address this methods. For example, interfaces with other Social Networks can be created.

As far as the penalization procedure is concerned, the results I achieved are not completely satisfactory. The method presented here can be modified and adapted, and other ways to model the general behavior of the users can be investigated. Improvements can be made in two directions:

1. in order to have a clearer view over the general behavior of the other users, a way to see all the Facebook data of a user when matching him with another can be tested, so to overtake the obstacle of the privacy settings;
2. a different paradigm of penalization can be reached by adapting the Clarke Tax Algorithm, aimed at assigning a tax to a user depending on how the individual's utilities affect the rest of the group [16].

For the semantic matching, I used a a rather simple distance measure based on the is-a hierarchy of WordNet. The effects of different measures can be experimented: for example, similarity measures based on the information content of the concepts can be re-elaborated and applied.

References

[1] Badrul Sarwar, George Karypis, Joseph Konstan and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[2] Nicholas A. Bradley and Mark D. Dunlop. Toward a multidisciplinary model of context to support context-aware computing. In *Human-Computer Interaction*, volume 20(4), pages 403–446, 2005.

[3] Harry Bunt. Context and dialogue control. *THINK Quarterly*, 3, 1994.

[4] C. Leacock and M. Chodorow. Combining local context and WordNet similarity for word sense identification. In *WordNet: An electronic lexical database*, pages 265–283. MIT Press, 1998.

[5] Ilaria Canova Calori. Awareness for fostering serendipitous interaction in public intelligent environments. In *Intelligent Environments*. IOS Press, 2009.

[6] Matthew Chalmers. A historical view of context. In *Computer Supported Cooperative Work (CSCW)*, volume 13: 3-4, pages 223 – 247, 2004.

[7] Chiara Rossitto. *Managing Work at Several Places: Understanding Nomadic Practices in Student Groups, Doctoral Thesis in Human-Computer Interaction*. 2009.

[8] John H. Connolly. Context and the study of human languages and computer programming languages, 2001.

[9] John Dewey. *Context and thought*. University of California press in Berkeley, 1931.

[10] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany* , pages 304–307, 1999.

[11] Paul Dourish. What we talk about when we talk about context. In *Personal and Ubiquitous Computing*, volume 8(1), pages 19–30, 2004.

[12] Hamid R. Ekbia and Ana G. Maguitman. Context and Relevance: A Pragmatic Approach. In *Modeling and using context. International and interdisciplinary conference nr 3, Dundee, UK*, volume 2116, pages 159–169, 2003.

[13] Claudio Bettini et al. A survey of context modelling and reasiong techniques. In *Pervasive and Mobile Computing*, 2009.

[14] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider. *The description logic handbook: theory, implementation, and applications*. Cambridge University press, 2003.

[15] A. Frank. Tiers of ontology and consistency constraints in geographical information systems. *International Journal of Geographical Information Science*, 15(7):667–678, 2001.

[16] Michael A. Goodrich. The Clarke Tax Algorithm. 2005.

[17] Samuel Greenberg. Context as a dynamic structure. In *Human-Computer Interaction*, volume 16:2-4, pages 257–268, 2001.

[18] Guus Schreiber, Monica Crubézy and Mark Musen. A case study in using Protégé-2000 as a tool for CommonKADS. In *International Conference on Knowledge Engineering and Knowledge Management (EKAW2000)*, pages 33–48. Springer-Verlag Berlin Heidelberg, 2000.

[19] Harry Chen, Filip Perich, Tim Finin and Anupam Joshi. SOUPA: standard ontology for ubiquitous and pervasive applications. In *Proceedings of the first annual international conference on mobile and ubiquitous systems: networking and services*, 2004.

[20] Ilaria Canova Calori and Monica Divitini. Reflections on the role of technology in city-wide collaborative learning. In *International Journal of Interactive Mobile Technologies (iJIM)*, volume 3(2), 2009.

[21] Innar Liiv, Tuukka Ruotsalo, Tanel Tammet and Alar Kuusik. Personalized context-aware recommendations in SMARTMUSEUM: combining semantics with statistics. In *2009 International Conference on Advances in semantic proceedings*, 2009.

[22] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings on International Conference on Research in Computational Linguistics*, pages 19–33, 1997.

[23] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge university Press, 1991.

[24] Basit A. Khan and Mihhail Matskin. A Platform for Actively Supporting E-Learning in Mobile Networks. *International Journal of Mobile and Blended Learning*, pages 55–79, 2010.

[25] Basit A. Khan and Mihhail Matskin. AGORA Framework for Service Discovery and Resource Allocation. In IARIA, editor, *The Fifth International Conference on Internet and Web Application and Services*, 2010.

[26] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the International Conference on Machine Learning*, 1998.

[27] Chengzhi Liu. Interpersonal Relationship Recommendation Framework for Mobile Learning Community. 2010.

[28] Matthew Horridge. *A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools - Edition 1.2*. University of Manchester, 2009.

[29] Michal Laclavik, Marian Babik, Zoltan Balogh, Emil Gatial and Ladislav Hluchy. Semantic Knowledge Model and Architecture for Agents in Discrete Environments. *Frontiers in Artificial Intelligence and Applications*, 141:727–728, 2006.

[30] George A. Miller. WordNet: A Lexical Database for English. In *Communications of the ACM*, volume 38(11), pages 39–41, 1995.

[31] Moo Nam Ko, Gorrel P. Cheek and Mohamed Sheab. Social-networks connect services. In *Computer*, volume 43, pages 37–43, 2010.

[32] Natalya F. Noy and Deborah McGuiness. Ontology Development 101: A Guide to Creating Your First Ontology. 2001.

[33] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. In *Artificial Intelligence Review - Special issue on data mining on the Internet*, volume 13(5-6), pages 393–408, 1999.

[34] Pienaar Öztürk and Agnar Aamodt. A context model for knowledge-intensive case-based reasoning. In *International Journal of Human-Computer Studies*, volume 48 - Special issue: using context in applications, pages 331–355, 1998.

[35] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[36] S. M. Smith. *Environmental context-dependent memory*. 1988.

[37] Ted Pedersen, Siddharth Patwardhan and Jason Michelizzi. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (Intelligent Systems Demonstrations)*, 2004.

[38] Xiao Hang Wang, Da Qing Zhang, Tao Gu and Hung Keng Pung. Ontology based context modeling and reasoning using OWL. In *Proceedings of the second IEEE annual conference on pervasive computing and communication workshops*, 2004.

[39] Yiwen Wang, Natalia Stash, Lora Aroyo, Peter Gorgels, Lloyd Rutedge and Guus Schreiber. Reccomendations based on semantically enriched museum collections. *Web Semantics: science, services and agents on the world wide web*, 6:283–290, 2008.

[40] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[41] Elisabetta Zibetti and Charles Tijus. Understanding actions: contextual dimensions and heuristics, 2005.

APPENDIX A

Software tools

I will now provide a quick description of all the software tools involved in this work at different levels.

A.1 Protégé and ontologies

Protégé[1] is a free, open-source ontology editor and a knowledge representation system. It is being developed at Stanford University in collaboration with the University of Manchester and is based on Java. As of July 2010, version 4.1 has been released. However, this overview makes reference to the previous 4.0 version. All the software is available under the Mozilla Public License. At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats.

By ontology, we mean an explicit and formal description of a domain, including: concepts (called classes), more specific concepts (subclasses), their attributes and features (slots, roles or properties - describing the internal structure of concepts) and restriction on slots (facets or role restrictions). In this way, an ontology defines a common vocabulary providing a shared understanding of the given domain. The ontology and the set of individuals (instances) of classes constitutes a knowledge base.

The Protégé platform supports two main ways of modeling ontologies:

- *Protégé-Frames editor*: it enables the user to build and populate frame-based ontologies. In this model, an ontology consists of a set of classes organized in a subsumption hierarchy (i.e., a class A is subsumed by a class B if A is a subclass of B) to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties. This editor follows the standards provided by the OKBC[2] (Open

[1] http://protege.stanford.edu/
[2] http://www.ai.sri.com/

Knowledge Base Connectivity) protocol. Features of Protégé-Frames are:

1. A set of user interface elements that can be customized to enable user to model knowledge and enter data in domain-friendly forms;
2. A plug-in architecture that can be extended with custom-designed elements;
3. A Java-based Application Programming Interface (API) that makes it possible for plug-ins and other applications to access, use, and display ontologies created with Protégé-Frames.

– *Protégé-OWL editor*: it allows a developer to build or load an ontology in OWL[3] (Ontology Web Language), a W3C's standard language born to process information (classes and relations between them) on the web and based on XML, or in RDF. Protégé-OWL's flexible architecture makes it easy to configure and extend the tool. Protégé-OWL is integrated with Jena (to be described in the next section) and has an open-source Java API for the development of custom-tailored user interface components or arbitrary Semantic Web services.

In practical terms, developing an ontology includes:

– defining classes in the ontology;
– arranging the classes in a taxonomic (subclass-superclass) hierarchy;
– defining slots and describing allowed values for these slots;
– filling in the values for slots for instances.

We can then create a knowledge base by defining individual instances of these classes filling in specific slot value information and additional slot restrictions. For a detailed description on how to build an ontology step by step, see [32].

Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema.

Protégé is built on top of the OWL API, providing more flexible development and a straightforward migration path for OWL-based applications. In particular, Protégé can be used to build OWL-DL ontologies.

OWL-DL is a sublanguage of OWL including all OWL language constructs, but they can be used only under certain restrictions. The name is due to its correspondence with description logics.

After building an ontology, Protégé allows the user to check its consistency (i.e. check whether it is possible or not for a class to have instances) and automatically compute (i.e., infer) the ontology class hierarchy by using a Description Logic reasoner, also called classifier. Different reasoners are available for Protégé; for example, Protégé 4.0 is shipped with FaCT++,

[3] www.w3.org/TR/owl-ref/

that allows very fast classification times and reasoning with larger ontologies.

Besides reasoners, Protégé 4.0 supports many other useful plugins for developing ontologies, created by members of the CO-ODE team and other associates at the University of Manchester[4]. For example, OWLViz[5] enables the class hierarchies in an OWL ontology to be viewed and incrementally navigated, allowing comparison of the asserted (manually constructed) class hierarchy and the inferred (computed by the reasoner) class hierarchy.

A.2 Jena

As stated before, Protégé-OWL comes integrated with Jena. Jena[6] is a Java-based open-source framework for building semantic web applications developed at the HP Labs Semantic Web Program. Now the project has been dismissed by HP Labs and transferred under the liberal BSD-style license. Jena provides a programmatic environment for RDF, RDFS, OWL and SPARQL (a query language for RDF[7], whose implementation for Jena is called ARQ). It includes a rule-based inference engine and provides services for model representation, parsing, database persistence, querying and some visualization tools.

Protégé-OWL has always had a close relationship with Jena, in particular for parsing of OWL/RDF files and various other services such as species validation and datatype (i.e., type of data in XML Schema) handling have been reused from Jena. This integration allows programmers to user certain Jena functions at run-time, without having to go through the slow rebuild process each time.

OWL ontologies can import other ontologies and thus establish relationships between resources from multiple files. For example, it is common to keep instance data separate from classes and properties. In this case, the instances ontology would import the ontology that defines the classes. The Protégé user interface allows users to switch the "active" sub-ontology, so that users can select which file changes will be propagated to. Protégé stores OWL ontologies in triple tables, one table for each file. These tables are named after the Java interface `TripleStore`. Each `OWLModel` manages a list of triple stores by means of its `TripleStoreModel`. The key to the Protégé-Jena integration is the fact that both systems operate on a low-level "triple" representation of the model. Protégé has its native frame store mechanism, which has been wrapped in Protégé-OWL with the `TripleStore` classes. In the Jena world, the corresponding interfaces are called `Graph` and `Model`. The Protégé `TripleStore` becomes a Jena `Graph`, so that any read access

[4]http://www.co-ode.org/downloads/protege-x/plugins/
[5]www.co-ode.org/downloads/owlviz/
[6]http://www.openjena.org/
[7]http://www.w3.org/TR/rdf-sparql-query/

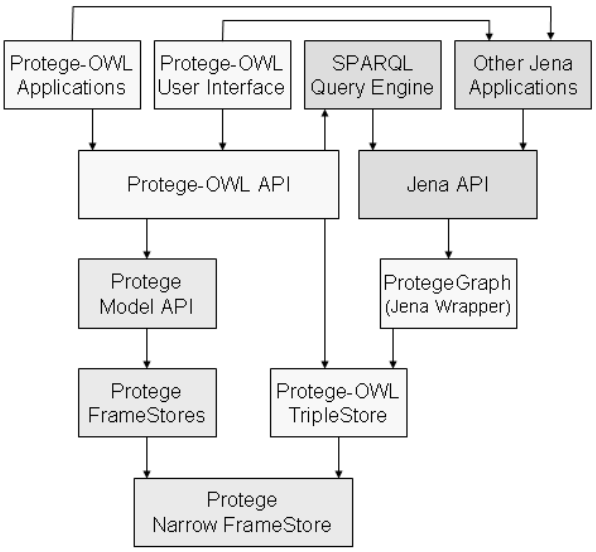

Figure A.1: The architecture of the Protégé-OWL/Jena integration

from the Jena API in fact operates on the Protégé triples. In order to modify these triples, the conventional Protégé-OWL API must be used. However, this mechanism allows the use of Jena methods for querying, while the ontology is edited inside Protégé. The `OWLModel` API has a new method, `getJenaModel()`, to access a Jena view of the Protégé model at run-time. This can be used by Protégé plug-in developers. Many other Jena services can be wrapped into Protégé plug-ins this way, by providing them a pointer to the `Model` created by Protégé.

Since Jena is fundamentally an RDF platform, Jena's ontology support is limited to ontology formalisms built on top of RDF: RDFS, the varieties of OWL, and the now-obsolete DAML+OIL. In particular, in addition to defining hierarchical classes that resources can belong to, OWL allows the characteristics of resources' properties to be expressed. For instance, OWL could be used to state that the `childOf` property is the inverse of the `parentOf` property. Another example is to state that the WordNet vocabulary's `hyponymOf` property is transitive.

Representing an ontology with Jena The Jena Ontology API is independent of the programming language: the Java class names do not mention the underlying language. For example, the `OntClass` Java class can represent an OWL class, RDFS class, or DAML class. To represent the differences between the various representations, each of the ontology languages has a profile, which lists the permitted constructs and the names of the classes and properties. Thus, for instance, in the OWL profile the URI for object property is owl:ObjectProperty (short for http://www.w3.org/2002/07/owl#ObjectProperty) and in the RDFS profile it is null since RDFS does not define object properties. The profile is

bound to an ontology model, which is an extended version of Jena's `Model` class. The base `Model` allows access to the statements in a collection of RDF data. `OntModel` extends this by adding support for the kinds of objects expected to be in an ontology: classes (in a class hierarchy), properties (in a property hierarchy) and individuals. When working with an ontology in Jena, all of the state information remains encoded as RDF triples (which Jena calls Statements) stored in the RDF model.

A.3 WordNet

The following review of WordNet is an adaptation of what I wrote for my Bachelor thesis, where I have extensively uesed this tool.

WordNet is a large lexical network for the English language developed at Princeton University under the supervision of George A. Miller. It is freely and publicly available at http://wordnet.princeton.edu/. The last versionon is 2.1 for Windows operating systems, and 3.0 for UNIX-based systems. The lexical categories represented here are:

- nouns;
- verbs;
- adjectives;
- adverbs.

Terms are grouped in sets of synonyms (called *synsets*) on a cognitive basis. Each synset expresses a distinct concept. One term can have different meanings, and based on the meaning it can belong to different synsets. Synsets are interliked by menas of semantic relations, while terms have lexical relations with other terms. Some relations exist only for a subset of sintactic categories.
For the purposes of this work, only the semantic relations are considered, so I will skip the details about the lexical relations. Here follows a description of the semantic relations available for WordNet.

A.3.1 Semantic relations

A semantic relation is a relation between meanings, that in WordNet are represented as synsets. It is therefore natural to think of this kind of relations as pointers between synsets. Semantic relations are symmetrical: if a relation R exisits between synset $x = \{x_1, x_2, ..., x_n\}$ and $y = \{y_1, y_2, ..., y_m\}$, then there exist also a relation R' between y and x.

Hyponymy and hypernymy A concept represented by synset $x = \{x_1, x_2, ..., x_n\}$ is a hyponym of the concept represented by synset $y = \{y_1, y_2, ..., y_m\}$ is an English native speaker accepts as true the statment that *x is a (kind of) y*. As far as the hyponymy beteween verbs is concerned, in this case the relation is called troponymy (manner of): verb x is a troponym of y if x expresses an action y done in some manner. The inverse relation of hyponymy is hypernymy: is is the link between a superordinate concept and another that is more specific and that inherits all the caracteristics, but has something different or additional. Hyponymy and hypernymy are the most used kinds of relations in WordNet and generate a `is-a` structure. For example, *house* `is-a` *dwelling*: here, *house* is a hyponym of *dwelling*, that is to say one of the possible type of *dwelling*; on the contrary, *dwelling* is hypernym of *house*, that is to say that it subsumes the concept of *house*).
This is the hierarchical relation most widely userd in disambiguation algorithm and to compute the similarity between words.

Meronymy and holonym A concept represented by synset $x = \{x_1, x_2, ..., x_n\}$ is a meronym of another concept represented by synset $y = \{y_1, y_2, ..., y_m\}$ if, for a native English speaker, the proposition *y has a x as part/member/substance* (or, simply, *x is a part of y*) is true. In this case, y is a holonym of x. Even the meronymy/holonymy relation is much used when realizing hierarchies out of the synsets of WordNet, in order to obtain an additional kind of linkage to the only *is-a* structure.

Entailment or troponymy We say that a verb x entails a verb y if x cannot happen unless y already happened or is happening. It is a relation connecting verbs, and is also known as troponymy. It expresses a relation close to meronymy, but has the caracteristic of being also a lexical relation (between the single verbs), and not only sematic (between the synsets to whom the two verbs belong).

Cause to The casual relation is meant to describe a type of entailment between verbs that does not require any temporal constrain.

Coordination Coordination is a derivate type of relation: two synsets are coordinate if they share a common hypernym, i.e. they are two different specialization of the same concept. For instance, *dog* and *wolf* are coordinate because they share *canine* as a common hypernym.

Similarity Similiarity is the kind of relation underlying the mutual organization of adjectives. Synsets made of adjectives can be connected by relation of antonymy (*hot* and *cold*, *heavy* and *light*): that is the reason why

they are called *head synset*. To these head synsets other synsets are connected by means of the similarity relation, sharing indirectly the antonymy relation with them.

A.3.2 Similarity measures

The similarity, or semantic distance, between concepts based on WordNet are mainly of two different classes: those that are based on the length of the path between the concepts inside a lexical graph (*path-based measures*), and those that are based on the information content (IC) subsuming the concepts (*IC-based measures*). Also available are a number of formulas to compute the semantic relatedness between concepts, that is a rather more general type of similarity based on additional information other than the length of the distance inside an `is-a` hierarchy. However, such kind of approach is out of the purposes of the present work.

Here follows a quick review of the measures most widely cited in literature [37].

Path-based similarity measures

a - The *path* measure is the simplest approach: the similarity sim between two words w_1 and w_2 is the inverse of the shortest path between the synsets (or concepts) c_1 and c_1 they belong to:

$$sim(w_1, w_2) = \frac{1}{shortest\_path(c_1, c_2)}$$

b - The *lch* measure [4] also relies on the shortest distance between two concepts, but scales this value by the total depth D of the taxonomy:

$$sim(c_1, c_2) = -\log \frac{shortest\_path(c_1, c_2)}{2D}$$

c - The *wup* measure [40] finds the lengh of the path to the root node of the lexical hierarchy, starting from the most specific concept that the two synsets share (i.e., their least common subsumer or *lso*).

IC-based similarity measures In brief, the definition of information content of a concept c is:

$$IC(c) = -\log p(c)$$

where *p(c)* is the probability of encountering an istance of the concept c. There are two ways to get the value of *p(c)*: one is to calculate it based on its frequency inside a corpora, and the other, which fits our case, is to get the IC(c) by means of the is-a hierarchy of WordNet. In such a hierarchy, the more specific a concept, the higher its IC: if (c_1 `is-a` c_2), then ($p(c_2) < p(c_1)$).

a - The *res* measure [35] computes the similarity as:

$$sim_R(c_1, c_2) = -\log p(lso(c_1, c, 2))$$

b - The approach followed by the *jcn* measure [22] is instead provided as a measure of semantic distance, the inverse of similarity. It is the conditional probability of encountering a child synset, given an instance of the parent synset.

$$sim_{JC}(c_1, c_2) = 2 * \log p(lso(c_1, c, 2)) - (\log(p(c_1)) + \log(p(c_2)))$$

c - The *lin* similarity measure [26] re-elaborates the same elements of the sim_{JC} measure:

$$sim_L(c_1, c_2) = \frac{2*\log p(lso(c_1,c,2))}{(\log(p(c_1))+\log(p(c_2)))}$$

Acknowledgements

I would like to thank my supervisors prof. Monica Divitini at NTNU and prof. Sonia Bergamaschi at Università di Modena e Reggio Emilia for their support and their assistance and for giving me the possibility to live this amazing experience abroad in a great university and a lovely city.

I am also grateful to prof. Mihhail Matskin, for his precious comments and suggestions.

A special thought to all of my friends in Trondheim, for making this experience unique, to my boyfriend Stefano, for his invaluable support, and to my dear friend Fabio, for his neverending and patient help and his priceless advice, without which this work would not have been possible.

In conclusion, last but not least, a great thank you to my family, because they have always believed in me during all the five years of my university studies.