Ordering N°: Year: 2014

Thesis
Cotutelle-de-thèse

Usage-Driven Unified Model for User Profile and Data Source Profile Extraction

Submitted to

University of Passau
Department of Informatics and Mathematics
Passau, Germany

National Institute Of Applied Sciences
Doctoral School of Computer Science and Mathematics
Lyon, France

In fulfillment of the requirements for

Doctoral Degree

Prepared by

Lyes Limam

Defended on the 24th of June 2014

Examining Committee

Prof. Christine Verdier	Université Joseph Fourier, France	Rapporteur
Dr. HdR. Markus Zanker	Université de Klagenfurt, Autriche	Rapporteur
Prof. Michael Granitzer	Université de Passau, Allemagne	Examinateur
Dr. Matthieu Exbrayat	Université d'Orléans, France	Examinateur
Prof. Ernesto Damiani	Université de Milan, Italie	Examinateur
Dr. Elöd Egyed-Zsigmond	INSA de Lyon, France	Examinateur
Dr. David Coquil	Université de Passau, Allemagne	Examinateur
Prof. Lionel Brunie	INSA de Lyon, France	Co-Directeur de thèse
Prof. Harald Kosch	Université de Passau, Allemagne	Co-Directeur de thèse

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

This thesis addresses a problem related to usage analysis in information retrieval systems. Indeed, we exploit the history of search queries as support of analysis to extract a profile model. The objective is to characterize the user and the data source that interact in a system to allow different types of comparison (user-to-user, source-to-source, user-to-source). According to the study we conducted on the work done on profile model, we concluded that the large majority of the contributions are strongly related to the applications within they are proposed. As a result, the proposed profile models are not reusable and suffer from several weaknesses. For instance, these models do not consider the data source, they lack of semantic mechanisms and they do not deal with scalability (in terms of complexity). Therefore, we propose a generic model of user and data source profiles. The characteristics of this model are the following. First, it is generic, being able to represent both the user and the data source. Second, it enables to construct the profiles in an implicit way based on histories of search queries. Third, it defines the profile as a set of topics of interest, each topic corresponding to a semantic cluster of keywords extracted by a specific clustering algorithm. Finally, the profile is represented according to the vector space model. The model is composed of several components organized in the form of a framework, in which we assessed the complexity of each component.

The main components of the framework are:

- a method for keyword queries disambiguation
- a method for semantically representing search query logs in the form of a taxonomy;
- a clustering algorithm that allows fast and efficient identification of topics of interest as semantic clusters of keywords;
- a method to identify user and data source profiles according to the generic model.

This framework enables in particular to perform various tasks related to usage-based structuration of a distributed environment. As an example of application, the framework is used to the discovery of user communities, and the categorization of data sources. To validate the proposed framework, we conduct a series of experiments on real logs from the search engine AOL search, which demonstrate the efficiency of the disambiguation method in short queries, and show the relation between the quality based clustering and the structure based clustering.

Résumé

La problématique traitée dans la thèse s'inscrit dans le cadre de l'analyse d'usage dans les systèmes de recherche d'information. En effet, nous nous intéressons à l'utilisateur à travers l'historique de ses requêtes, utilisées comme support d'analyse pour l'extraction d'un profil d'usage. L'objectif est de caractériser l'utilisateur et les sources de données qui interagissent dans un réseau afin de permettre des comparaisons utilisateur-utilisateur, source-source et source-utilisateur. Selon une étude que nous avons menée sur les travaux existants sur les modèles de profilage, nous avons conclu que la grande majorité des contributions sont fortement liés aux applications dans lesquelles ils étaient proposés. En conséquence, les modèles de profils proposés ne sont pas réutilisables et présentent plusieurs faiblesses. Par exemple, ces modèles ne tiennent pas compte de la source de données, ils ne sont pas dotés de mécanismes de traitement sémantique et ils ne tiennent pas compte du passage à l'échelle (en termes de complexité). C'est pourquoi, nous proposons dans cette thèse un modèle d'utilisateur et de source de données basé sur l'analyse d'usage. Les caractéristiques de ce modèle sont les suivantes. Premièrement, il est générique, permettant de représenter à la fois un utilisateur et une source de données. Deuxièmement, il permet de construire le profil de manière implicite à partir de l'historique de requêtes de recherche. Troisièmement, il définit le profil comme un ensemble de centres d'intérêts, chaque intérêt correspondant à un cluster sémantique de mots-clés déterminé par un algorithme de clustering spécifique. Et enfin, dans ce modèle le profil est représenté dans un espace vectoriel. Les différents composants du modèle sont organisés sous la forme d'un framework, la complexité de chaque composant y est evaluée. Le framework propose :

- une methode pour la désambiguisation de requêtes ;
- une méthode pour la représentation sémantique des logs sous la forme d'une taxonomie ;
- un algorithme de clustering qui permet l'identification rapide et efficace des

centres d'intérêt représentés par des clusters sémantiques de mots clés ;

- une méthode pour le calcul du profil de l'utilisateur et du profil de la source de données à partir du modèle générique.

Le framework proposé permet d'effectuer différentes tâches liées à la structuration d'un environnement distribué d'un point de vue usage. Comme exemples d'application, le framework est utilisé pour la découverte de communautés d'utilisateurs et la catégorisation de sources de données. Pour la validation du framework, une série d'expérimentations est menée en utilisant des logs du moteur de recherche AOL-search, qui ont démontrées l'efficacité de la désambiguisation sur des requêtes courtes, et qui ont permis d'identification de la relation entre le clustering basé sur une fonction de qualité et le clustering basé sur la structure.

Zusammenfassung

Die Arbeit befasst sich mit der Nutzungsanalyse von Informationssuchsystemen. Auf Basis vergangener Anfragen sollen Nutzungsprofile ermittelt werden. Diese Profile charakterisieren die im Netz interagierenden Anwender und Datenquellen und ermöglichen somit Vergleiche von Anwendern, Anwendern und Datenquellen wie auch Vergleiche von Datenquellen. Die Arbeit am Profil-Modell und die damit verbundenen Studien zeigten, dass praktisch alle Beiträge stark auf die entsprechende Anwendung angepasst sind. Als Ergebnis sind die vorgeschlagenen Profil-Modelle nicht wiederverwendbar; darüber hinaus weisen sie mehrere Schwächen auf. Die Modelle sind zum Beispiel nicht für Datenquellen einsetzbar, Mechanismen für semantische Analysen sind nicht vorhanden oder sie verfügen übe keine adequate Skalierbarkeit (Komplexität). Um das Ziel von Nutzerprofilen zu erreichen wurde ein einheitliches Modell entwickelt. Dies ermöglicht die Modellierung von beiden Elementen: Nutzerprofilen und Datenquellen. Ein solches Nutzerprofil wird als Menge von Themenbereichen definiert, welche das Verhalten des Anwenders (Suchanfragen) beziehungsweise die Inhalte der Datenquelle charakterisieren. Das Modell ermöglicht die automatische Profilerstellung auf Basis der vergangenen Suchanfragen, welches unmittelbar zur Verfügung steht. Jeder Themenbereich korrespondiert einem Cluster von Schlüsselwörtern, die durch einen semantischen Clustering-Algorithmus extrahiert werden. Das Modell umfasst mehrere Komponenten, welche als Framework strukturiert sind. Die Komplexität jeder einzelner Komponente ist dabei festgehalten worden. Die wichtigsten Komponenten sind die Folgenden:

- eine Methode zur Anfragen Begriffsklärung
- eine Methode zur semantischen Darstellung der Logs als Taxonomie
- einen Cluster-Algorithmus, der Themenbereiche (Anwender-Interessen, Datenquellen-Inhalte) über semantische Cluster der Schlüsselbegriffe identifiziert

- eine Methode zur Berechnung des Nutzerprofils und des Profils der Datenquellen ausgehend von einem einheitlichen Modell

Als Beispiel der vielfältigen Einsatzmöglichkeiten hinsichtlich Nutzerprofilen wurde das Framework abschließend auf zwei Beispiel-Szenarien angewendet: die Ermittlung von Anwender-Communities und die Kategorisierung von Datenquellen. Das Framework wurde durch Experimente validiert, welche auf Suchanfrage-Logs von AOL Search basieren. Die Effizienz der Verfahren wurde für kleine Anfragen demonstriert und zeigt die Beziehung zwischen dem Qualität-basiertem Clustering und dem Struktur-basiertem Clustering.

Part I.

Introduction

1. Introduction and Motivations

1.1. General Context

Nowadays, the amount of information available in the digital world grows up exponentially. According to a study conducted by the International Data Corporation (IDC) [GR11], the amount of information created and replicated in the world is doubling every two years and has exceeded 1,8 Zettabyte in 2011. In addition to that, the rapid growth of information volume affects both structures of smaller and larger scale. Exploiting such a huge amount of information in structures of small scale like enterprise or large scale like the Internet, is usually confronted with the challenge of relevant information seeking.

Naturally, the structure that controls the information takes the form of a network. Moreover, a network is a set of nodes exchanging data where a node is either a user or a data-source. In an information system, the user is *the entity that requests and consumes information in the form of text, multimedia objects or services that correspond to its interest,* while the data-source is *the entity which is able to deliver information as a response for a user request.* In order to facilitate access to relevant information, organizing the network is necessary from both the user side and the data-source side.

In the user side, such an organization consists in grouping together the users with similar interests. Therefore, a user who needs information from others has to ask the group who shares similar interests with him. In fact, dealing with groups of users, rather than individuals reduces the search space, which has the advantage of saving time and efforts and improving the quality of information. As an example, in social networks, the process of users grouping is known as community discovery.

Unlike the user, the data-source plays the role of the provider. Its function is to answer, when it is possible, to the user query. Organizing the data-sources consists

in grouping together the ones whose contents cover the same topics. This results in two advantages: first, in a search session, the user query is submitted to a group of data sources whose contents match the user interests, which reduces the search space. Second, grouping data-sources improves the global efficiency of the system because it facilitates data transfers within the same group. For instance, it enables workload balancing by taking decisions about data placement within the same group. We call the process of grouping data sources, data source categorization.

In summary, both user grouping and data source grouping should be based on the interest. Moreover, we note three types of relationships (see Fig. 1.1) :

1. User to a Group of users: a relationship in which a user exchange information with users with similar interests.

2. User to Group of data-sources: a relationship in which a user requests information from data sources covering his interest.

3. Data-source to Group of data-sources: a relationship in which a data source exchanges information with other data sources covering similar topics.

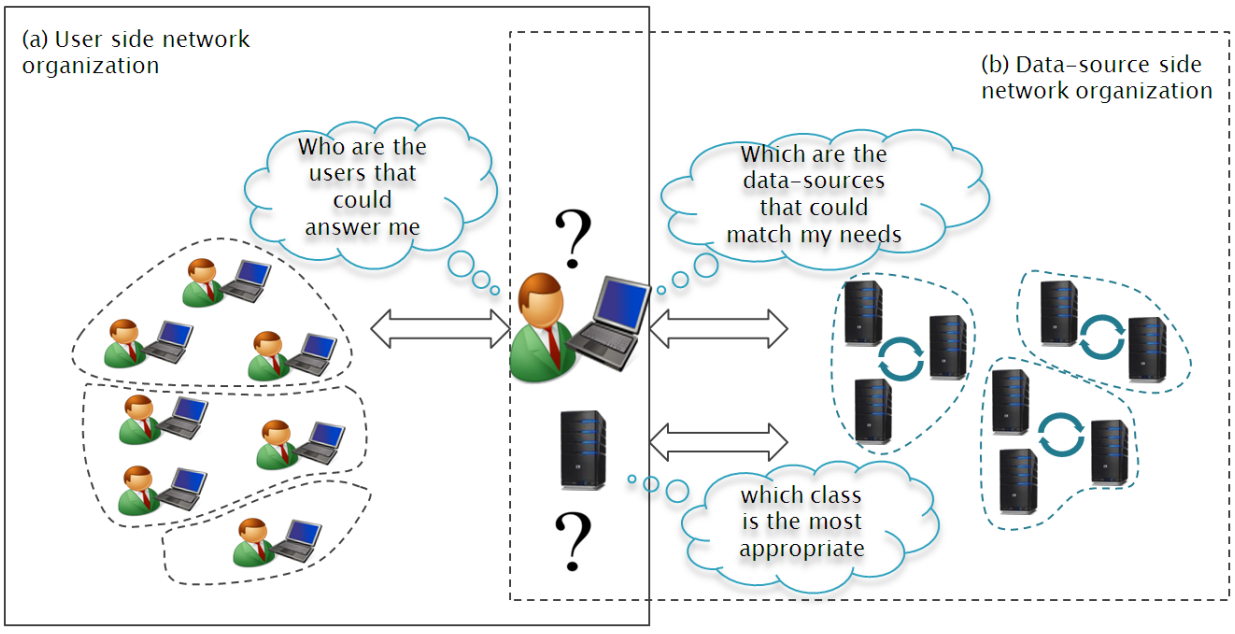


Figure 1.1.: Interest based Network Organization

1.2. Research Questions

As we have shown previously, network organization consists in splitting the network into groups of users having similar interests and into groups of data-sources covering similar topics. Such an organization facilitates the access to relevant information by reducing the search space and improving the efficiency of the system. In fact, the core problem in either user grouping or data-source grouping is a modeling problem. Indeed, the grouping process consists basically in comparing the entities (users and/or data-sources) of the system. However, comparing entities should be done on the basis of a model that represents the characteristics of those entities. Actually, if we consider the heterogeneous nature of the user and the data-source, there is a need of two different models, each of which performing a set of tasks:

The User Model:

- Representing the user interests
- Enabling comparisons:
  - User to User: Comparison enables to construct groups of users with similar interest and to simplify interactions between users.
  - User to Data-source: Comparison enables to identify the data-sources which are likely to match the users interests.

The Data Source Model

- Representing the topics provided by the data-source
- Enabling comparisons:
  - Data-source to Data-source: Comparison enables to construct groups of data-sources with similar contents and to simplify data transfers between data-sources.

The main objective of this thesis is to propose a profile model for both user and data source. Thus, through our proposal, we attempt to answer a number of specific research questions, which are:

1. *What is the best technique to represent the user interests and the data source topics, to be interpreted by the system, and understandable by the user?*

2. *Is there a way to characterize both the user and the data source in a unique model?*
3. *What is the data that should be analyzed to construct the model? Is the data collected implicitly or explicitly?*
4. *Should the model characterize the user short term interests and/or long term interests?*
5. *How can semantic mechanisms contribute to enrich the model?*
6. *Can the model perform the three types of comparison (i.e., user-user, user-data source, and data source-data source)?*
7. *Can the model construction method be scalable to deal with large amount of data?*

1.3. Contribution

In this thesis, we design a model and a framework for user and data-source profiling based on the analysis of usage. The main idea of the framework is to exploit the history of queries in which the users interact with data-sources. In our study we use information retrieval system application example, but the principles of the framework are still valid in other interaction-based applications, such as social network applications, microblogging platforms (e.g., Twitter), etc. The main components of the framework comprise:

- Data gathering and query log preprocessing (cf. sec. 4.2)
  This consists in collecting the user queries submitted to a web search engine, and in preprocessing the resulting log by filtering out meaningless queries and keywords. In this component, we propose two types of analysis: lexical and semantical. The former consists in using text analysis techniques (tokenization, stemming and, lemmatization). The latter consists in integrating semantic resources to check the validity of the keywords.
- Semantic representation of the log by using keywords taxonomy (cf. chapter 5)
  This consists in preparing the logs to the effective process of user and data source profiling. The main objective is to enrich the textual query log with semantics. In this component, we represent the query log in the form of keywords

taxonomy. Thus, the main contributions are: first, a method for keyword disambiguation, which consists in replacing each keyword by its corresponding sense, and second, a semantic distance function, which measures the distance between keywords in the taxonomy.

- Usage-based user profile and data-source profile modeling (chapter 6)
  This consists in extracting the user profile and the data source profile from the keywords taxonomy. In this component, we first, transform the taxonomy into a model representing the general user interests (i.e., the interests that concern all the users), then we instantiate the previous model for each individual user and individual data-source. The main contributions in this component are: first, a pruning clustering algorithm, which splits the taxonomy into a set of general interests, second, a method to represent the user profile and the data source profile in the vector space model.

The framework we propose answers the previous research questions (cf. sec. 1.2) as follows:

1. *What is the best technique to represent the user interests and the data source topics, to be interpreted by the system, and understandable by the user?*
   In our proposal, we use basic elements (keywords/synsets) and structures with positive added-value. In fact, as basic elements we use sense of the words also called synsets instead of text of words, extracted from the WordNet database. In addition to that, we consider clusters of concepts (synsets) as composite elements. The objective is to extract the user interests by exploiting the semantic relatedness between the concepts. Moreover, we introduce two structures: a hierarchy structure and a vector structure. The hierarchy structure relates the concepts and is used as a base on which user interests are extracted in the form of clusters. The vector structure is used to represent numerically the user interests.

2. *Is there a way to characterize both the user and the data source in a unique model?*
   In our proposal, the profile model enables to represent both of the data source and the user in the same model.

3. *What is the data that should be analyzed to construct the model? Is the data collected implicitly or explicitly?*
   In our proposal, we use the user queries as usage data to analyze. This type

of data is collected implicitly from the search engine. Of course explicit usage data (e.g., age, gender, etc.) can improve the precision of the constructed profile, but collecting explicit data requires effort from the user and is time consuming.

4. *Should the model characterize the user short term interests and/or long term interests?*
   In our proposal, we make the choice of long term interest as it is the more general, but it would be possible to extend to short term in the future.

5. *How can semantic mechanisms contribute to enrich the model?*
   In our model, we use WordNet, a linguistic ontology, as a source of semantics. The advantage is that WordNet is domain independent which enables to analyze any type of usage data, especially that coming from a regular user. Thus, we use WordNet to enrich the profile model (which is based on keywords) with semantics by providing senses to words and inferring the hierarchical relations between them.

6. *Can the model perform the three types of comparison (i.e., user-user, user-data source, and data source-data source)?*
   In our model the comparison between user profiles is done by comparing their corresponding vector of interests.

7. *Can the model construction method be scalable to deal with large amount of data?*
   In our proposal, the scalability of the model is demonstrated by calculating the complexity of the steps composing the modeling process.

1.4. Thesis Organization

The remaining of the dissertation comprises three parts:

The first part is the state of the art; it is divided into two chapters:

Chapter two reviews existing approaches in user modeling and discusses the possibility of using query logs as usage data representative of the user interests. Chapter three reviews the main existing distance/similarity functions and the most well-known clustering algorithms. Their performance and characteristics are discussed.

The second part is devoted to the contributions of the thesis:

Chapter four is an overview of the framework that we propose. Chapter five presents a method for semantically representing the query log. Chapter six discusses the design of a model of user interests, which is transformed to a profile model. In addition to that, it shows how to use the profile model in the process of user group identification and data-source categorization.

The dissertation ends with chapters discussion and conclusion.

Part II.

State of the Art

2. User Modeling

2.1. Introduction

User modeling usually means characterizing and representing the user in a way that enables to exploit information about his behavior and his preferences; it is also called "**user profiling**". The model aims to inform the system about the user's needs so that it enables the system to adapt to him. This is usually made by inferring the user interests based on his interactions with the system. Thus, constructing the user profile is essential for inferring user interests.

A user profile is constructed from the explicit and implicit interactions of the user with the system. In other words, usage data for user profiling can be collected implicitly or explicitly. Explicit collection usually requires the user's active participation, which allows the user to control the information in his profile. Explicit profiling can take different forms. The user may fill out a form, take part in a survey, submit personal information to register, etc. One advantage of this method is that it lets the users directly inform the system what they need and how they need it. One major inconvenience of this method is that it needs time and effort from the user, especially when the user interests may change frequently. Conversely, implicit profiling does not require the users' input and is performed as a background task. It usually means tracking and monitoring users' behavior in order to identify patterns [Bra04]. Amazon.com, for example, keeps track of each user's purchasing history and recommends specific purchases.

In general, two fundamental characteristics must be found in a user profile. First, the profile must be able to represent the user's multiple interests in different topics. Second, the profile extraction must be fast enough to adapt to the users' change in interests and possibly to deal with big amounts of user data. However, there are other characteristics (e.g., representation, usage data, semantics, etc.) that should be taken into account as it will be shown in the next section. Thus, different profiling

approaches have been proposed for different applications. These approaches rely on a variety of techniques like: knowledge management, classification, clustering, etc.

The rest of the chapter is composed of two parts. In sec. 2.2, we analyze chronologically the most important proposals done in the last twenty years in the field of user modeling. The review is done according to several dimensions. Then, in sec. 2.3 we show how it is possible to exploit query logs as usage data.

2.2. Profiling Model Approaches

In this section, we propose to analyze the existing profiling models. Thus, we show chronologically the major improvements that have been made according to a number of dimensions, each of which defines a profiling model parameter[1]. In fact, this results in a set of classes of approaches more or less independent. The parameters in question are illustrated in (Fig. 2.1):

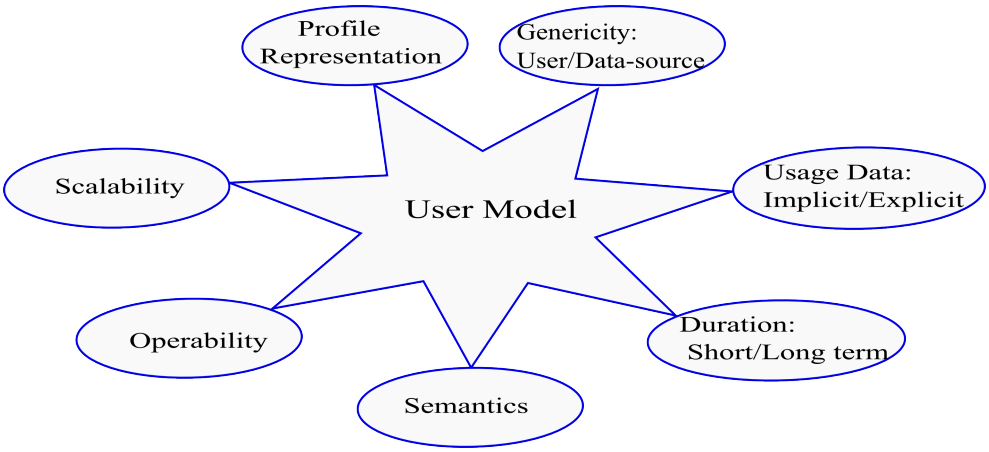


Figure 2.1.: User Model Parameters

1. **Profile Representation:** The profile representation technique used in the construction method. We focus on the most common techniques: keywords based, concepts based and structures based (semantic networks, hierarchical representations, etc.) [GSCM07].
2. **Genericity**: Capability to model both the user and the data source.

[1]In this chapter, we use the words “dimension” and “parameter” equivalently to mean one parameter of analysis

3. **Usage data**: Type of usage data to analyze, i.e., implicit and/or explicit.
4. **Duration**: Capability to characterize both short and/or long term user interests.
5. **Semantics**: Integration of semantic mechanisms and/or statistical models.
6. **Operability**: Capability to make operations on the profiles, specifically binary comparison and statistical analysis in general.
7. **Scalability**: Capability to deal with large amounts of data.

The user modeling is one of the early problems that has interested the computer scientists [PAC78, Ric79b, Ric79a, PA80]. At that time, the concept of user modeling was integrated in the application system and no separation was done between the user model components and the components of the other tasks [Kob01]. The big problem with this trend of *user-adaptive* applications is that the user model is application specific and cannot be reused. The separation between the user model and the user adaptive applications has started in the mid-eighties (e.g. [Sle85]). Since then an important number of profile models have been proposed. In our bibliographic study, we focused on works which match fully or partially the parameters that we cited above.

2.2.1. Kobsa et al.

One of the first generic and application independent user modeling systems was proposed by A. Kobsa and W. Pohl [KP94]. The authors proposed BGP-MS which is a user modeling shell system that assists the user application in adapting to the user by taking into account: knowledge, beliefs, and goals. In order to infer the user model, BGP-MS and the user application dialog permanently to exchange information about the user behavior. This is done by means of a set of messages, including:

- The user application informs BGP-MS about observed user beliefs and goals: Beliefs and goals are described by a specific description language (BGDL). Moreover, the content of the message is described following a first order logic syntax. When BGP-MS receives this message from the application, it stores it as primary information.

- BGP-MS sends an interview question to the user: The objective is to infer assumptions directly from the user answers, which will be integrated in the user model.
- The user application sends the answer to BGP-MS.
- The user application informs BGP-MS about observed user actions: This information allows BGP-MS to infer all the possible assumptions about the user beliefs and goals.
- The user application asks BGP-MS for its current assumptions about the user: The objective is to allow the application to adapt to the assumptions about the user model.
- BGP-MS gives the current assumptions to the application.
- BGP-MS informs the user application about important events in the user model. When important assumptions are inferred in the user model, BGP-MS can inform the application without waiting for a question.

The communication between BGP-MS and the user application is done via KN-IPCMS which is a platform-independent message-oriented communication protocol which allows asynchronous communication (e.g., observation about the user beliefs, goals and actions) and synchronous communication (e.g. BGP-MS current assumptions about the user or interview questions) when the user-application waits for an answer from BGP-MS.

In order to represent the user model and the domain knowledge, the authors use a partition hierarchy mechanism based on a partition mechanism called KN-PART [FH93]. KN-PART enables to represent knowledge and assumptions about the user (beliefs and goals) in a conceptual representation scheme as well as in first-order logic. It has two principal functions: First, it represents assumptions about the user model and the domain knowledge in the form of several separated partitions where each partition is a set of assumptions of the same type. The objective is to enable inference of new assumptions from each partition. Second, it relates these partitions to form a hierarchy. Finally, the common content of subordinate partitions is propagated to their superordinate partition and the content of this partition is inherited by the subordinate partitions. In order to collect assumptions about the user, BGP-MS uses inference mechanisms carried out on the basis of first order logic. The inference process is based essentially on the observed beliefs and goals, interviews, user's actions and the current user model.

The BGP-MS user modeling system matches the parameters we discussed above, as follows:

1. *Profile representation:* The model derived by BGP-MS consists of a set of assumptions organized into a hierarchical partition. These assumptions are represented both in conceptual scheme and in first order logic.
2. *Genericity:* The proposed model is exclusively a user model.
3. *Usage data:* The modeling system uses both implicit (e.g., the user actions) and explicit (e.g., interview questions) usage data.
4. *Duration:* The proposed model is based principally on current assumptions collected and inferred from the user interview answers, beliefs, goals and actions. Thus, the model is constantly re-adapted according to new events. In fact, the model is more adapted to capture short term user assumptions.
5. *Semantics:* The construction of the user model is based only on the interactions between the user and the application and between the user and the modeling system. Moreover, there is no way to insure that collected assumptions are correct (e.g. the user can misunderstand an interview question because of ambiguity). In case of an error in an assumption, BGP-MS continues until it reaches an inconsistency state. The use of semantics could significantly improve the capacity of BGP-MS in detecting inconsistency situations.
6. *Operability:* The authors do not show how operations could be done on individual user models.
7. *Scalability:* The scalability issue is not discussed by the authors. Nevertheless, in our opinion, the efficiency of the construction process depends mainly on two factors, which are: the communication protocol and the inference mechanism. Depending on the user application, functions the construction process could be notably slowed down because of the high number of generated messages at each application event and the inconsistencies that the inference mechanism has to manage.

Other work

There are many proposals that can be compared to the work of Kobsa et al. We can cite [FK02] which applies BGP-MS mechanisms (assumptions, user modeling

server, communication protocol, etc.) in the framework of a personalized city tours application.

2.2.2. Armstrong et al.

At almost the same period of time, Armstrong et al. proposed WebWatcher [AFJM95], an agent-based system for user modeling in the web. The goal of the system is to assist the user in searching information on the web by identifying hyperlinks which are the most likely to be relevant for the user. To do this, the system relies on the results of previous searches by analyzing both desired and undesired retrieved information by a machine learning method.

Technically, WebWatcher calculates a function $LinkUtility$ that measures the probability that a hyperlink leads to the user goal:

$$LinkUtility : Page \times Goal \times User \times Link \rightarrow [0, 1]$$

Where $Page$ is the current page, $Goal$ is the user goal, $User$ is the current user and $Link$ is a hyperlink on the current page.

In order to calculate the values returned by the function, the agent should pass a learning phase. The Learning process is composed of two parts: the training data collection and the learning method. The training data collection consists of recording for each user link selection an entry in a log file. The content of each entry is represented in a fixed length vector of Boolean elements corresponding to the occurrence or not of a word in the user selection. The vector is composed of four sub-vectors that represent underlined words, words of the sentence containing the link and the words used to describe the goal.

The objective of collecting training data is to calculate for each link the probability that it will be selected for a specific goal (i.e., a user query). The learned model is defined by the function $UserChoice$:

$$UserChoice : Page \times Goal \times Link \rightarrow [0, 1]$$

The second part of the learning process is the learning method. The authors apply three different learning methods: Winnow, Wordstat, TFiDF (see [AFJM95] for

more details), in addition to a reference random method (i.e., without learning). The objective is to measure the effectiveness of each one with respect to the proposed user model.

The authors show two interesting results. First, the precision of WebWatcher when using the three learning methods is significantly higher compared to a random recommendation. This result shows the impact of the proposed model. Second, the precision of the learning methods can be improved by reducing the coverage. This means that WebWatcher proposes recommendation only when its confidence is higher than a certain threshold which consequently reduces the number of link recommendations.

According to our parameters we deduce the following:

1. *Profile representation:* The data of the user model is textual and is first, organized into a set of vectors of words and then, transformed to vectors of features, where each feature corresponds to a word occurrence or nonoccurrence in the user selection of link.
2. *Genericity:* The proposed model concerns only the user and does not consider the data source as it focus only on what the user wants (goal) and never what the website proposes. Nevertheless, the model could be extended by taking into account words of the web page.
3. *Usage data:* The learning process is based only on searches done by the user (goal), which means that the model is fully implicit.
4. *Duration:* The user model is constructed thanks to a learning method. Consequently, the result user model is more adapted to capture the long term user goal. The reason is that whatever the goal is, it is always affected by all of the previously learned goals.
5. *Semantics:* The training data considered by the authors is textual and organized into vectors of words. One of the major problems that could affect the user model is the fact that words could be interpreted differently by the users because of ambiguity. Indeed, the authors do not deal with this issue.
6. *Operability:* One of the advantages of the proposed model is its presentation. The fact that the training data has numerical form, enables operations such as comparison between users.
7. *Scalability:* The proposed user model is intended to assist users on the web to select hyperlinks that are the most likely to lead to their goal. Considering

the size of the web, the model should be adapted to deal with big amounts of data (i.e., the training data). However, the author did not deal with the scalability issue even if they are convinced that the bigger the set of training data the better for precision. Thus, the experiments were done on 30 user sessions of depth of 6 searches for each session. Nevertheless, the authors showed further in [JFM97] that WebWatcher is still efficient even with a more significant number of users and data.

Other Work

WebWatcher can be compared with other similar work. For instance, the authors in [PBMW97] proposed an agent based system that assists the user to find interesting websites. The proposed approach is based on a naive Bayesian classifier that enables to learn and revise the user profile by using the user feedback and features (keywords) extracted from the visited webpages. In the context of the web, the authors in [HDSB03] proposed another approach to model the user profile by exploiting the history of navigation. The objective is to predict the user navigation behavior by using probabilistic Markov models. Almost in the same field of application, the authors in [DLSA03] proposed an approach to extract the user interests in e-commerce. The objective is to recommend new objects by exploiting the history of purchase. The approach is based on a naive Bayes machine learning method applied on text annotations provided with objects (books).

2.2.3. Bollackar et Al.

In [BLG99] the authors introduced a tracking system which they integrate to *CiteSeer,* a system for Autonomous Citation Indexing (ACI) of scientific publications on the Web, that they proposed previously in [GBL98]. The objective of the tracking system is to keep up to date scientific researchers with publications that correspond to their research topics in web digital libraries. To do this, the tracking system constructs the profile of the user that is used to track new interesting publications. Contrary to the previously proposed approaches, the user profile is not based only on keywords. The authors introduced the concept of *Heterogeneous profile.* It consists of making several and different representations of the user interest's topics. Indeed, the profile consists in a heterogeneous set of features (representative key-

words, related papers, citation links, metadata). The authors claim that, by using *Heterogeneous profile* a wider variety of users may be accommodated as each user has his/her own set of techniques of searching papers.

The profile building process is launched with a CiteSeer's searching session to find papers of interest. Thus, one needs three types of data: search keywords, citing paper and related paper of a paper that interests the user. The search keywords are words that separately match: the title, the header, the abstract or the main text body. The citing papers are papers that cite the paper of interest. The related papers are papers found thanks to relatedness measures and not by a citing relationship.

In fact, by using one of the above information, the tracking system can start building the profile. For instance, in the *CiteSeer* interface, if the user submits a keyword query containing the title of a paper, the tracking system retrieves the documents that corresponds to that query. If the user wants that the tracking system tracks new documents corresponding to those keywords, they will be saved in his/her profile. Then, if the user selects one of the retrieved items (i.e., papers) and wants to keep up to date with related papers, the paper in question will be saved in his/her profile and future related papers will be recommended to him/her. In the end, if the user wants to look into the list of citing papers and track one of them, he/she can reflexively save the paper in his/her profile and/or look for other related papers.

In addition to the possibility of adding keywords and documents to the user profile, the system allows explicit tuning of the profile by the user. He/she can do this, for example, by deleting component (keywords, documents) that do not belong anymore to his/her interests.

In summary, the heterogeneous profile proposed by the authors is composed of keywords and documents that interest the user. The advantage of such a profile model is the complementarity that exists between keywords and documents. For example, in a searching session, a relevant document that is not recommended to the user because of no or partial matching of the profile keywords could be proposed to the user thanks to a related document in his/her profile.

Let's show how [BLG99] fits our parameters:

1. *Profile representation:* The profile model is represented by keywords, in addition to their related documents.
2. *Genericity:* Only the user is considered in the profile model. Although, it

would be very useful for the user to know, for example, interesting data sources (libraries) which need to be modeled.

3. *Usage data:* The process of constructing the user profile is based on data (keywords and documents) collected during a search session or recommended by the tracking system. The process is done implicitly. Nevertheless, it comes to the user to explicitly decide if the collected/recommended data can be used to tun his profile.
4. *Duration:* The fact that it is possible to tune the profile by the user himself makes possible for the profile to keep up to date with both long and short term user interests.
5. *Semantics:* The authors do not deal with semantics, likely because they are limited to the domain of scientific research papers. Consequently, the problem of ambiguity is not considered.
6. *Operability:* Since the authors have used the vector space model to realize keywords matching and TFiDF (Term Frequency X Inverse Document Frequency) with CCiDF (Common Citation X Inverse Document Frequency) to measure the documents relatedness, operations between user profiles seems to be possible.
7. *Scalability:* We can consider two levels of scalability. Local data base scalability and search engine combined with crawling agents scalability. Although there are no performance measures shown by the authors, the system seems to be scalable on the web as it depends on the performance of a web search engine[2].

Other Work

Bollacker et al. work [BLG99] can be compared to Amato et al. work [AS99]. In fact, both of the two proposals are interested in modeling the user in the context of digital libraries. The most important resemblance in the two proposals is that both of them consider the fact that the presentation of interests could be very heterogeneous (keywords, documents, etc.) and hence has to be taken into account in the user profile model. However, we note a small difference which comes from the fact that the former focuses on scientific libraries whereas the latter deals with

[2]http://citeseerx.ist.psu.edu/index, 2013

general libraries. Consequently, in term of precision the Bollacker et al. model can be better than the Amato et al. model. The reason is that the topics of interest are well defined in Bollacker et al. (i.e., research topics) compared to Amato et al. (i.e., general libraries).

2.2.4. Kim et al.

The authors in [KC03] proposed a hierarchical approach to extract user's interest topics by implicitly analyzing previously visited web pages. The objective is to provide the search system with a robust context for personalization.

The proposed profile model consists in a set of classes of interests organized from general to specific. General classes represent long term interests and specific classes represent short term interests. The process starts by extracting the document (web page) features. It consists of representing each document by a set of words that represent the most accurately its content. Then, the set of all the words is split recursively into a cluster hierarchy by using a top down clustering algorithm called DHC. The algorithm works as follows:

First of all, the similarity between each couple of words is calculated using a similarity function. Thus, the set of words is transformed into a similarity graph. Second, the algorithm calculates a clustering threshold in the current cluster. Third, the graph is split into a set of connected components (sub-graphs) by eliminating edges whose weights are less than the threshold value. The resulting sub-graphs are the child clusters. Fourth, the two first steps are recursively applied on each child cluster. The clustering stops when there are no components to split or when the number of words is less than a predefined threshold (different from the splitting threshold).

The similarity function and threshold-finding function are the two major parts of the algorithm. Moreover, both of the threshold and the similarity matrix are calculated dynamically at every new child cluster. This property has an advantage since it enables to adapt to the size of the current child cluster.

The similarity function measures the relatedness of words occurring in the same documents. The authors used three different functions:

- $AEMI$ (Augmented Expected Mutual Information): It measures the similarity of two words by considering co-occurrence:

$$AEMI\,(A,\,B) = p(a,\,b)\,log\,\frac{p(a,b)}{p(a)p(b)} - \sum_{(A=a,\,B=\overline{b})(A=\overline{a},\,B=b)} p(A,\,B)\,log\,\frac{p(A,\,B)}{p(A)p(B)}$$

where:

- $p(a)$ is the probability of a document containing a ; $p(b)$ is the probability of a document containing b ;
- $p(a,b)$ is the probability of a document containing a and b.
- $p(\overline{a})$ is the probability of a document not containing a ; $p(\overline{b})$ is the probability of a document not containing b

- $AEMI\_SP$: Enhances AEMI by considering the IDF (inverse document frequency) of a word. It is expressed by the function SP:

$$AEMI\_SP = AEMI - \frac{SP}{2}$$

$$SP(m) = \frac{1}{1 + \exp(0.6 * (m * 10.5 - 5))}$$

$$m = MAX(P(a), P(b))$$

The clustering threshold is determined dynamically by different threshold-finding methods. The first method consists of finding the threshold value that optimizes the number of eliminated edges on each step. To do this, the authors use the concept of *Valley* on the graph involving the number of edges by regions of weights. It consists first of dividing the difference between the smallest edge weight and the highest one into ten equal regions. Then, the value of the threshold corresponds to the deepest point (the lowest weight) in the widest and steepest valley in the graph (# of edges / region).

The second threshold-finding method consists of finding the threshold that generates the maximum number of child clusters. After having specified the regions, the threshold will correspond to the boundary between two regions that generates the maximum child clusters.

In the end, we notice that the evaluation of the constructed hierarchy is done by analyzing the shape and the number of words within a cluster.

The hierarchical approach of [KC03] fits our parameters as follows:

1. *Profile representation:* The profile is represented by a hierarchy of interests where each interest is a cluster of keywords.
2. *Genericity:* The profile model considers only the user and does not model the data source (the web site).
3. *Usage data:* The usage data is implicitly extracted from previously visited web pages
4. *Duration:* The profile model represents both long term interests and short term interests. The authors consider general interests in the hierarchy to be long term interests and specific interests to be short term interests.
5. *Semantics:* The authors do not use any external source of semantics. However, using the concept of relatedness by word co-occurrence can fix some problems of ambiguity.
6. *Operability:* The extracted profile model depends on the usage data of an individual user. Moreover, the hierarchy and the interest clusters can be very heterogeneous from a user to another one, since both of the hierarchy generation and the clustering are based on dynamic threshold. Consequently, profile comparison is not possible.
7. *Scalability:* The scalability issue was not studied by the authors. The experiments were performed on a small number of users.

In the work of Kim et al., there are two aspects that we use in our approach (cf., chapter 5 and chapter 6): hierarchical classification and clustering algorithm. However, the techniques and the results are different. First, the Kim et al. word hierarchy puts into clusters words that are related by the co-occurrence relationship in documents. However, in our proposal, the clusters are constructed by using the semantic relationship between word meanings. Second, Kim et al. propose a top-down clustering algorithm. It produces large clusters in the top of the hierarchy as long term interests and short clusters in the bottom of the hierarchy as short term interests. However, our proposal is a bottom-up clustering, that favors clusters in the bottom of the hierarchy as they identify more accurately the user interests. Third, the user interests extracted by Kim et al. algorithm do not represent the degree of interest of a user regarding an interest. In contrary, our clustering algorithm extracts and affects interest weight for each cluster. Indeed, the interests hierarchy proposed by Kim et al. consists of an input for user profiling while our proposal is a profiling model.

Other Work

The authors in [SDW06] proposed an approach to construct the user profile in the context of news filtering items. The user profile in this approach is defined as a hierarchy of user interests. This work differs from Kim et al. [KC03] in three aspects: First, the construction process needs explicit feedback from the user while [KC03] use implicit feedback. (In [SDW06], the user is presented with a set of news articles and is required to indicate if a news article is of interest or not of interest). Second, the interest hierarchy is evaluated based on news articles re-ranking while [KC03] analyze the resulting hierarchy in terms of the shape of the hierarchy tree and clusters of terms. Third, in [SDW06], once the interest hierarchy is constructed, its leaf categories are updated after each user session, based on explicit feedback while [KC03] does not update the hierarchy.

2.2.5. Middleton et al.

Middleton et al. [MSDR04], proposed one of the first ontological user profiling approach destined to assist academic researchers by recommending on-line academic research papers.

The user profile is created and maintained by two recommender systems: Quickstep and Foxtrot.

Quickstep first classifies papers that have been discovered by the users. To do this it uses classes of research topics of an ontology (extracted from the dmoz open directory project). The documents classification is done by a k-Nearest Neighbor classifier using training documents, which are represented in term-vector space to enable closeness measurement by the TFiDF metric. After the classification phase, Quickstep creates the user profile by measuring the interest of the user with respect to the different topics. The topic interest is calculated considering the user's feedback when papers are recommended. The feedback consists in a set of actions which are: paper interest rating (positive or negative), browsing a paper, following a recommended paper and paper topic's correction (helps only the classifier). Moreover the *topic interest* function is time dependent, which means the less the papers of a topic receive feedback over time the less the topic interests the user. Hence the authors propose the following formula to measure the topic interest:

$$Topic\_interest = \sum_{i=1}^{n} Interest\_value(i) \,/\, days\_old(i)$$

Where n is the number of documents within a topic, *day_old(i)* is the number of days since the last feedback on the document *i*, *interest_value* is the rate given to the feedback of the document *i*. It is equal to "1" for paper brows, "2" for following recommendation, "10" for positive interest, "-10" for negative interest (not interested).

One particular characteristic of Quickstep is that it enables the inference of new user interest topics. In fact, a new topic is a super class whose *interest_value* is rated as 50% of the *interest_value of its subclass*.

Foxtrot is the second recommendation system that constructs the user profile based on the user interest feedback. Foxtrot and Quickstep are very similar, since both of them use similar ontologies and user feedback. Thus, the author used them together to enhance the recommendation precision and to resist to the famous problem of cold start. However, even if the two systems are similar, Foxtrot has the advantage to be more accurate in acquiring the user feedback. In fact, Foxtrot uses a graphic visualization of the user profile which enables the user to indicate the exact level and duration of interest for a topic by drawing interest bars onto the time/interest graph.

In summary, the profile model proposed by the authors shows two novel ideas compared to the previous work. First, the profile is represented in the form of an ontology of interest topics and second, it is visualized in the form of a graph. These two original ideas have several advantages. First, the use of ontology enables to provide a base on which the users can be compared. This can be done by comparing documents of the same topic (class). Second, the profile model can be easily mapped with other models by merging their ontologies. Third, the ontology provides a more understandable vocabulary to the users, which facilitate feedback acquiring. Forth, the model is flexible since it can infer new topics. Finally, the graphic profile visualization increases the user profile accuracy.

In contrary, one inconvenient in using ontological profile could be the fact that the profile is strongly related to the domain ontology. For instance, if one wants to construct the user profile in different domains the user might have several sub-

profiles instead of one, which for example complicates comparisons.

[MSDR04] fits our parameters as follows:

1. *Profile representation:* The profile is represented by a research topic ontology which classifies documents that interests the user.
2. *Genericity:* The data-source is not modeled, only the user is considered.
3. *Usage data:* The user browsing behavior is monitored by a web proxy. Thus, papers that interest the users are unobtrusively saved in a central database which makes the data collection implicit. However, during the profile construction process the profile is explicitly tuned by the user in the profile graph (called profile feedback by the authors).
4. *Duration:* The profile model can represent both short and long term interests. This is done thank to the time/interest graph visualization of the profile which allows the user to teach the system which interests are short or long term by specifying both the interest level and the duration.
5. *Semantics:* The ontology provides a unified vocabulary which facilitates the profile interpretation and avoid ambiguity, in particular, when the users return their profile feedback (they use the terms of the ontology).
6. *Operability:* The profile model proposed by the authors is intended to be used in the context of recommendation by collaborative filtering. Thus, the profile model is represented in a way that it enables to compare individual users. The ontological representation facilitates comparison since it separates interests documents into a set of topics and hence the comparison is made between documents of the same topics. The author used Pearson-r metric [Sti89] to measure correlation between profiles.
7. *Scalability:* The experiments presented by the authors were done on small and large scale.

In the work of Middleton et al. the semantics aspect is strongly highlighted. In fact, by using an ontology to represent the profile, the authors intend to address the problems related to ambiguity. In this sense, the semantics aspect is also one of the issues that we addressed in this thesis. However, there are two main differences compared to the work of Middleton et al. First, in this thesis we use keywords to represent the profile while Middleton et al. use documents. Thus, in our profiling model, the interests are clusters of keywords while in Middleton et al. the interests

are class of documents. Second, Middleton et al. use a domain ontology (research topic ontology) as a source of semantics while in this thesis we use WordNet. Thus, in Middleton et al. the modeling process is intended for specific public of users (i.e., research community) while in this thesis the modeling process is intended to any type of users.

Other Work

In addition to Middleton et al. work, the use of ontologies to model the user profile was the subject of many other proposals. The common point of these proposals is that they focus on the issue of semantics. [SMB07] for instance, proposed an approach to produce a semantic representation of the user interests by analyzing search history and using a predefined ontology. In [GA06] the authors introduced the notion of conceptual clustering. Their approach consists of representing the user interests as clusters of visited documents, where each cluster is attached to the most specific concept of a conceptual hierarchy extracted from the whole set of documents. The authors in [MS04] proposed to model the user interests as a semantic network. Their approach consists of transforming the word based document visited by the user into word sense based document by using the WordNet database. Thus, words (senses) co-occurring in documents are related to form a semantic network.

2.2.6. Shen et al.

The authors in [STZ05] proposed an approach to improve the accuracy of traditional web search engine by employing *eager* user profile. The authors claim that using immediate user feedback in the construction of the user profile can significantly improve the quality of search as it adapts in real time to any change of the user needs. Moreover, the proposed approach does not need any explicit feedback from the user to construct the profile. This means, that the user model is based only on implicit and immediate user feedback.

Since the user model has to dynamically adapt to every user needs change, it is necessary to take into account the time factor. The authors define the user model to be composed of four components:

$$m_t = (S,\ \vec{x},\ A_t,\ R_{t-1})$$

Where:

- S is the set of documents which have been seen by the user (i.e., the navigation history),
- $\vec{x} = (x_1, ..., x_n)$ describes the system understanding about what the user is interested in. It consists in a vector whose components are weights attributed to a set of keywords $v = \{w_1, ..., w_n\}$. The keywords are extracted from the history of search queries and documents. Indeed, the $\vec{x}$ values change instantly regarding the user's actions like: submitting a keyword query, viewing a document, etc.
- A_t represents the set of actions made by the user up to a time t,
- R_{t-1}represents all the system responses just before the last user action $a_t \in A_t$. The model $m_t$is used later to generate the system response $r_t \in R_t$

When a query is submitted by a user, the system responds with a set of document summaries. The accuracy of the response depends on the current user model m_t, which depends on $\vec{x}$:

$$\vec{x} = \alpha\vec{q} + (1-\alpha)\frac{1}{k}\sum_{i=1}^{k}\overrightarrow{s_i}$$

where $\vec{q}$ is query term vector, $\overrightarrow{s_i}$ the summary term vector (i.e., terms weighted by a TF-IDF function) of a document the user clicked on following the current query. k is the number of summaries, α $(= 0,5)$ is a parameter that controls the influence of the clicked summaries on the inferred $\vec{x}$.

The proposed user model fits our parameters as follows:

1. *Profile representation:* The user model m_t is composed of four components: keywords $(\vec{x})$, viewed documents (S) and the user-system interaction history (actions A_t, responses R_{t-1}). The two fist components form the model (profile) representation while the others enable to update the model over time.

2. *Genericity:* The proposed model concerns only the user and is not extended to the data source.
3. *Usage data:* The data used to construct the model is based on implicit user feedback. The model is constructed on the base of the user search queries and clickthrough information (viewed documents).
4. *Duration:* The proposed profile model is based on immediate user feedback; it is an eager user profile. Consequently, only current user interests are considered.
5. *Semantics:* The user model is represented by the term vector $\vec{x}$ which is calculated and updated based on the history of viewed documents $\overrightarrow{s_i}$ and the current query $\vec{q}$. However, the model does not integrate any semantic mechanism, in spite the textual nature of analyzed data.
6. *Operability:* Operations between user profiles are possible, since the profile is represented in the vector space model ($\vec{x}$, $\overrightarrow{s_i}$)
7. *Scalability:* The construction of the user profile is scalable for two reasons. First, the user profiling component is deployed in the client side and not on the server which reduce significantly the server load. Second, only current user interests are considered which reduce the amount of analyzed data.

2.2.7. Vallet et al.

Recently, the authors in [VCJ10] have proposed two approaches for web search personalization by exploiting the new concept of *folksonomy*. The main objective is to make ranking algorithms more effective compared to that of classical web search engines by using personalized ranking.

The authors therefore propose a profile model based on the user's annotations of interesting web pages. The user profile is defined by the set of tags that the user gives to the documents (web pages) that she/he has retrieved. Formally, the profile of a user u_m is defined as a vector $\overrightarrow{u_m} = (u_{m,1}, ..., u_{m,l})$ of annotations, where $u_{m,i}$ is the number of times the user u_m used the term "t_i" for annotating documents. In addition to the user profile, the authors define the document profile by the tags that the users have given to it. By the same way as for the user, the document profile is represented by a vector $\overrightarrow{d_n} = (d_{n,1}, ..., d_{n,l})$ of annotations, where $d_{n,i}$ is the number of times the document d_n has been annotated using the term t_i.

The above definitions of the user profile and the document profile are used to re-rank the documents retrieved by the search engine such that the document re-ranking corresponds to the user preferences. For a given retrieved set of documents, the personalization system measures the similarity between the documents and the user profile; then, according to the obtained scores, the documents are re-ranked. To do this, the authors propose two different approaches. The first approach uses a similarity function based on the vector space model and the second approach uses a similarity function based on a probabilistic model.

In the end, the experiments conducted by the authors show a significant improvement (23.7%) of the ranking quality of the Yahoo search engine.

The Vallet et al. profile model matches our parameters as follows:

1. *Profile representation:* The profile model enables to represent the user profile and the document profile. It consists in a vector of tags used by the users for annotating documents. A component of a user profile vector represents the number of times the corresponding tag has been used for annotating documents, while a component of a document profile vector represents the number of times the corresponding tag has been used by users for annotating that document.

2. *Genericity:* The profile model enables to represent both the user and the document. Nevertheless, even if one considers a document to be a data source, the model mostly covers one topic of interest. It would be more interesting and efficient if the model was capable to represent data sources covering multiple topics. Indeed, extending the profile model in such a way would enable to perform a high level comparison between the user profile and the data sources, e.g. web sites, digital libraries, etc.

3. *Usage data:* The profile model is extracted from the user folksonomy which is made of users' annotations. Moreover, since the user is not asked to annotate systematically the interesting documents, we consider the usage data collection to be hybrid (implicit and explicit).

4. *Duration:* The profile model does not take time into consideration. Hence, all the user's tags are used to construct both the user profile and the document profile whatever if there are old or recent. Consequently, the profile model represents long term user interests.

5. *Semantics:* The profile construction process relies only on the user tags that form the folksonomy and does not use any semantic resource. In fact, the model considers only the word form of the tags, since neither the word sense nor the semantic relationship between words are extracted.

6. *Operability:* The model is well adapted to make profile comparison. Indeed, the ranking personalization approaches proposed by the authors are based on the comparison of the user profile and the document profile.

7. *Scalability:* The experiments presented by the authors are made on 2000 users, 161,542 documents and 69,930 distinct tags extracted from the social bookmarking website[3] "Delicious" (formerly del.icio.us). Even if the data set is relatively large, it is still not large enough to prove the scalability of the profile construction process. Nevertheless, since the user and the document profiles are based on the counting of the tag usage, the construction method is scalable.

2.2.8. Abel et al.

More recently, the authors in [AGHT11] have discussed an approach to construct a semantic user profile by exploiting user posts in the microblogging platform Twitter. The authors deal with two issues:

The first issue is to enrich the user tweets with semantics as it is difficult to use them in their crude form. The idea proposed by the authors consists of finding possible links between the news articles and the user posts (tweets) as there is usually an influence of news on the users' discussions. The objective is to provide a contextual base to enrich the user messages with semantics. Therefore, different strategies are proposed to analyze the link between tweets and news articles. The first strategy consists of detecting whether there is an URL within the message that leads to a news article or whether the message is a reply to another one citing a news URL. The second strategy consists of analyzing the content of the message and the content of recent (two days) articles by extracting and comparing their features. The features could be: bag of words, hash tags or entities. Thus, the relationship between the message and the article is measured by the sum of $TF \times IDF$ of the features of the message.

[3] http://www.delicious.com

$$sim(t, n) = \sum_{i=1}^{m} TF_i \times IDF_i$$

Where: "t" is a vector of features extracted from the message, "n" is the corresponding vector of features extracted from a news article, "TF_i" is the frequency of a feature "i" in the article "n", "IDF_i" is the inverse document frequency of a feature in the article "n".

According to the above function, the first ranked tweet-news pairs (t, n) are considered to be related and thus, linked. Moreover, the content of the related articles is analyzed in order to extract elements that enrich the user message. To this end, the authors use web services provided by OpenCalais[4] (a platform for semantic analysis of web pages) to extract entities (people, organizations, etc.) and topics embedded within the article. Then, these elements, in addition to the user tweets and related articles, are used to construct a RDF graph representing the enriched message. The RDF graphs are then used to construct the user profile, which is the second issue.

The second issue that the authors deal with is how to use the enriched messages (i.e., the RDF graphs) to construct the user profile. The authors distinguish between two parts of the profile: entity based profile and topic based profile. Thus, the user profile is defined as entities and topics the user is interested in. The user interest representation in terms of entities and topics is based on the RDF graph representation of the messages. In fact, after having constructed the RDF graph representing all the user messages, the entities and topics are weighted. To do this, the authors use two different strategies. The first weighting strategy is based on the number of tweets. Each entity or topic receives a weight corresponding to the number of related tweets. The second weighting strategy is based on the number of news articles. Each entity or topic receives a weight corresponding to the number of related news. The experimental results presented by the authors show that the news weighting strategy is more precise that the tweet-base strategy.

The approach proposed by [AGHT11] matches our parameters as follows:

1. *Profile representation:* The user profile is represented by an RDF graph which connects entities and topics that interest the user, in addition to the user's tweets and related news articles.

[4]http://www.opencalais.com

2. *Genericity:* Only user interests are represented in the profile model.
3. *Usage data:* The usage data is implicit, since the profile model bases only on the user tweets and related news article.
4. *Duration:* The user profile is constructed using only recent user tweets and recent news articles. Consequently, only short term interests are modeled.
5. *Semantics:* The construction of the profile model is based on enriched user posts. The use of RDF graph to represent the user posts with their related news enables to infer valuable semantic information.
6. *Operability:* Individual user profiles are compared by analyzing their corresponding RDF graphs. Comparing RDF graphs consists of verifying whether the two graphs are isomorphic or to find part of the graphs that are isomorphic [w3c01]. However, isomorphism does not enable to calculate the similarity value between two graphs, which limits the operability of the model.
7. *Scalability:* The evaluation has been done on large data-set (more than 3 million tweets posted by more than 45,000 users).

2.2.9. Synthesis

Since user profiling has become application independent, it has been possible to plug in a profiling component in different adaptive applications. However, no model is compatible with all applications. In fact, it is necessary that the model respects certain requirement to be compatible with the application's functions. For this reason, in order to generalize the use of profile models to a higher number of compatible applications we have enumerated a set of *generic parameters*.

Tab. 2.1 summarizes chronologically the bibliographic study we have done on the most important work of each period of time. According to the review of the previous work regarding our generic parameters, proposed user profiling models match only a part of these parameters. In fact, the choice made by the authors depends on their objectives and hence on the target application's functions. In this thesis, we aim to match all these parameters. The objective is twofold: on one hand, to make the profile model as generic as possible in order to be compatible with higher number of applications; and on the other hand, to attempt to overcome shortcomings of previous work.

In the following we summarize the different contributions according to our parameters:

The first parameter we studied is the profile representation. It is the way in which the user profile is modeled in the system. Thus, it should first, represent the most accurately the user interests and second, be as simple as possible to be interpreted by the system and understandable by the user. In the previous work we have reviewed, the profile is composed of basic elements (keywords, concepts or tags), composite elements (assumption, clusters or documents), in addition to a structure (hierarchy, vector, ontology or RDF graph) on which these elements are organized. In terms of basic elements, keywords are the mostly used (Armstrong et al., Bollackar et al., Kim et al., Shen et al., Abel et al.). The advantage of using keywords is the simplicity of the profile representation and hence its interpretation. However, using keywords in their textual form gives rise to problems of ambiguity because of their contextual usage. Consequently, the use of an external source of semantics is mandatory in this case; this was not taken into account in the above work. Exceptionally, Kobsa et al. and Abel et al. used receptively concepts and RDF entities and topics instead of simple keywords. However, their profile models still suffer from semantics and operability. In terms of composite element, Kobsa et al. used assumptions, Bollackar et al., Middleton et al., and Shen et al. used documents while Kim et al. used clusters. In fact, while using assumptions and documents aims only to complete the information contained in basic elements, clustering aims to organize them into topics of interests to facilitate the interpretation. In terms of structure, Kobsa et al. and Kim et al. used a hierarchy structure to represent the topics of interests and their relationships. Armstrong et al. and Vallet et al. used a vector structure to represent a certain feature of the basic elements (keywords): In the former, it is the occurrence or nonoccurrence of a keyword (0 or 1) and in the latter, it is the number of times the tag is used for annotation. Middleton et al. and Abel et al. used respectively ontology and RDF graph as structure to represent the profile elements. The former classifies user interest documents into ontology classes while the latter connects user interest entities and topics in an RDF graph. The advantage of using such structures is the ability to fix semantic issues.

The second parameter we have studied is the genericity of the profile models. The objective was to study the extendability of the existing profile models to the data source. In fact, in this thesis we consider the data source and the user to be symmetric. Thus, if the profile model should improve the system behavior regarding

the user it should do the same regarding the data source (see sec. 1.3). In all the work we have reviewed, the data source side is not considered, except by Vellet et al. who define the document profile jointly with the user profile. However, even if we consider a document to be a data source, the document profile defined by the authors covers only one topic while the profile by definition should cover multiple topics (cf. sec. 2.1).

The third parameter is the usage data. It consists of describing the data used to construct the profile. As we said previously (sec. 2.1), the data used for user profile analysis can be implicitly collected from the user interaction with the system (usage) or explicitly given by the user. The previous proposals can be divided into two groups: those using both explicit and implicit usage data (Kobsa et al., Bollackar et al., Middleton et al.) and those using exclusively implicit data (Armstrong et al., Kim et al., Shen et al., Vallet et al., Abel et al.).

The fourth parameter is the duration. It defines the lifetime of the user interests as it is taken into account in the user profile. In the literature, the profile models consider short or long term user interests or both of them. The choice between short or long term user profiling depends on the user change in interests. For instance, research topics of a scientist do not change as frequently as the interests of a regular user surfing on the Internet. In fact, it all depends on the domain application.

The fifth parameter is the semantics. The objective of this parameter is to check the existence of mechanisms used in the profiling process to deal with semantic issues. In the reviewed work, only Middleton et al. and Abel et al. effectively deal with this issue. The former work uses predefined ontology to classify interest documents while the latter work, uses RDF graphs to connect entities and topics that interest the user. The advantage of using domain ontology (or RDF graphs) is the unification of the vocabulary and hence the avoidance of ambiguity while the advantage of using RDF graphs is the ability to put profile elements (entities and topics) in their right context by means of links. However, there are problems related to each of the two. Using predefined domain ontology is restrictive since the ontology is limited to a certain domain, which prevents the emergence of real user interests. Moreover, using RDF graphs does not favor the operability of the model. The reason is that the comparison between RDF graphs is limited to finding equality by means of isomorphism [Car02], which does not quantify the similarity.

The sixth parameter is the operability of the profile model. The objective of this

parameter is to check the ability of the model to allow operations, namely, comparison of user profiles. In fact, this parameter is related to the profile representation. More specifically, it depends on the structure used to represent the profile. For instance, the vector structure (Armstrong et al. , Middleton et al. and Vallet et al.) enables comparison between user profiles by calculating the Cosine similarity or the Euclidean distance.

The seventh parameter is the scalability of the profile model. It checks the ability of the profile construction method to deal with big amounts of usage data. The study that we have conducted on this parameter is based on our analysis of the profile construction method presented by the authors. In fact, in most of the papers, this parameter is not explicitly tested; except Armstrong et al., which shows the scalability of their profile model.

2.3. Search Query Log Analysis and Positioning of the Thesis

In this thesis, we use search query logs as usage data to analyze. The reason is that the search query log is without doubt one of the richest sources of information about the user interests and opinions. In addition to that the queries are collected implicitly, which does not need any effort from the user. Nevertheless, analyzing search query logs comprises some specificities that should be taken into account. The following sections outline some of these specificities discussed in early work.

2.3.1. Search Query Log Analysis and its Challenges

Since the emergence of the first search engines, many researchers have proposed methods for extracting information from search logs [BC01, CFS05, JST08, SMHM99]. This topic is frequently referred to as Search Query Log Analysis, or SLA. In the SLA context, a search log is defined as an electronic record of interactions between a search engine and users searching for information on that search engine [Jan06]. Indeed, the set of queries issued by a user in a specific period of time contains information about his/her topics of interest; one goal of SLA is the extraction of this information. However, several authors [BBD+98, Kur93, PSF04] have criticized analysis processes that rely uniquely on search logs because they record neither the

users' perceptions of search nor their satisfaction with the results. However, this definitely complicates the task of accurately identifying user search behavior. On the other hand, these valid critics do not imply that search logs are useless. First, a number of applications of SLA can provide useful results without needing the missing information identified by the critics (e.g., [LOPS07, ZN08, MCL+10, VLB+12]). Moreover, the quality of the knowledge produced by SLA can be enhanced by using external sources of information in addition to query logs. Indeed, we argue in this thesis (cf. chapter 5) that it is possible to extract large amounts of information from search logs by including the implicit semantic relations between query terms in the process [LBCK08, LCKB10].

2.3.2. Search Query Log and Semantics

Jansen [Jan06] proposed a unified methodology to conduct the analysis of search query logs. He defined three levels of analysis: term level (terms as the basis for analysis), query level (queries as the basis for analysis) and session level (in-session interactions as a basis for analysis). A closer examination of the first two levels reveals that the analysis is mostly statistical and does not consider semantics. Indeed, for term level analysis, all proposed metrics (term occurrence, total terms, unique terms, high usage terms and term co-occurrence, etc.) are statistical measures that consider neither the problem of the polysemy of query terms nor the semantic relations between terms. The query analysis level is also based on statistical metrics: initial query, modified query, identical query, unique query, query complexity and failure rate. These metrics are actually based on the classic definition of a keyword search query, which is a list of strings of zero or more terms submitted to a search engine [Jan06]. Besides, the session level characterizes the time aspects (session duration, query frequency, etc.) in a user interaction approach which is out of the scope of this thesis. Another limitation of existing metrics is that they are exclusively based on string comparison. For instance, the queries "Hotel Booking" and "Booking Hotel" are considered to be different though they are in fact semantically equivalent. In our work, we strive to address these shortcomings by enhancing the SLA methodology with semantics. We argue that a larger amount of more meaningful information may be extracted from the logs if external information about semantic relations between terms is coupled with the query logs before the start of the analysis phase. We propose a method to achieve this, which results in a novel

way of representing the logs in the form of a taxonomy of query terms. Such a global representation lends itself very well to the definition of a metric that measures the distance between query terms. Subsequently, the taxonomy equipped with the distance function enables us to define a semantic query clustering method, which can extract user interests from search data.

2.3.3. Usage Analysis in Search Query Logs

The query log represents the history of usage that a user or a group of users have operated on the system. In the literature, we observed lots of and various works that attempt to use query logs as a base for analyzing the user behavior. The applications are essentially personalization, recommendation, data mining, community discovery, etc. For example, in [CCHB10], the authors used the lick-through data (e.g., hypertext links), in the form of query logs. Hence, they investigate the utility of topic models for the task of personalizing search results based on the information present in the query log. They define generative models that take both the user and the clicked document into account when estimating the probability of query terms. These models can be used to rank documents by their likelihood given a particular query and user pair. In the same domain, i.e., personalization, the authors of [AGM+11] proposed a proactive approach that couples a MultiDimensional eXpressions (MDX) based language for expressing OLAP preferences to a mining technique for automatically deriving user preferences. The main idea is to use the queries issued by a user in a mining process to extract set of association rules that relate sets of frequent query fragments; then, for each future query, a subset of rules is selected and translated into a preference that is used to annotate the user's query. A set of experimental results proves the effectiveness and efficiency of the approach. In [SN09] the authors propose a method that exploits the click history of each user to build a topical ontology, i.e., a model of the different topics that interest the user. The topical ontology is used to guide the search engine when it receives a query. In addition, the authors define a ranking function to sort the search result according to the user preferences. Their experiments show that user preferences can be learned accurately thanks to the topical ontology. In the content management domain, the analysis of queries is used for instance to determine the most frequently asked questions. [WNZ02] deals with this issue by treating it as a query clustering problem. They propose to combine the queries with the documents selected by the

user. Query clustering is conducted based on the idea that queries are similar if the selected documents are the same. [CC02] constructs a hierarchical classification of query terms, which is called taxonomy. The terms are grouped together in hierarchical clusters based on a vector space model. Each term is represented by a vector V of characteristics and each component V_{ij}(i.e. the j^{th} unique term corresponding to the i^{th} query term) is calculated by a $TF * IDF$ weighting function on the top ranked documents. The clusters are then constructed by comparing the terms using the Euclidean distance.

In this thesis, we propose an approach that uses query logs as usage data to analyze the user profile. Details and deeper explanations of our proposal and contributions are discussed in the next chapters.

	Profile representation (Basic element)	Genericity	Usage data	Duration	Semantics	Operability	Scalability	**Application Domain**
Kobsa et al. (1994)	Concepts, Assumptions, Hierarchy	No	Implicit, Explicit	Short	No	No	No	Personalization, Adaptation
Armstrong et al. (1995)	Keywords, Vector of occurrence	No	Implicit	Long	No	Yes	Yes	Recommendation
Bollackar et al. (1999)	Keywords, Documents	No	Implicit, Explicit	Long, Short	No	Yes	Yes	Recommendation
Kim et al. (2003)	Keywords, Clusters, Hierarchy	No	Implicit	Long, Short	No	No	No	Personalization
Middleton et al. (2004)	Documents, Ontology	No	Implicit, Explicit	Long, Short	Yes	Yes	Yes	Recommendation
Shen et al. (2005)	Keywords, Documents	No	Implicit	Short	No	No	No	Personalization
Vallet et al. (2010)	Tags, Vector of annotations	Yes	Implicit, Explicit	Long,	No	Yes	Yes	Personalization
Abel et al. (2011)	Entities and topics RDF graph	No	Implicit	Short	Yes	No	Yes	Personalization, Social Networks
Our Model (2013)	Concepts (Synsets), Clusters, Hierarchy, Vector of interests	Yes	Implicit	Long,	Yes	Yes	Yes	Recommendation, Personalization, community discovery

Table 2.1.: User profiling models

3. Overview on Similarity Functions and Clustering Algorithms

3.1. Introduction

In order to measure the relatedness between objects, we can process the objects in two different but comparable ways. The first consists in calculating how the objects are far according to their representations: this concept is called *Distance* or *dissimilarity*. The second way consists in measuring how an object resembles to another according to their representations: this concept is called *Similarity*.

Clustering is a process whereby a set of objects is split into groups of similar objects. The most of researchers describe a cluster by considering the internal homogeneity and the external separation, which means that patterns in the same cluster should be similar to each other, at the opposite of patterns in different clusters [JD88, HJ97]. Thus, the clustering relies on a distance/similarity function to measure the object relatedness. Indeed, the choice of a distance/similarity function is essential since the clustering quality depends on it. From a practical point of view, clustering plays an important role in many domains such as databases, information retrieval, Web analysis, and many others.

3.2. Distance and Similarity Functions

In general, both of the distance and the similarity functions are used to measure the relatedness between two objects, an object and a cluster, or a pair of clusters. In the last two cases, only one object of the cluster (a member of the cluster or a virtual member, e.g., centroid) is used to calculate the distance or the similarity. Moreover, the definition of a distance or a similarity function has to respect certain conditions [DD09, Lux04].

A distance function defined on a data set X should satisfy the following conditions:

1. Symmetry. $\forall x,\ y \in X,\ D(x,y) = D(y,x)$
2. Positivity. $\forall x,\ y \in X,\ D(x,y) \geq 0$
3. Triangle inequality. $\forall x,\ y,\ z \in X,\ D(x,z) \leq D(x,y) + D(y,z)$
4. Reflexivity. $D(x,y) = 0$ Iff $x = y$.

A similarity function should satisfy the following conditions:

1. Symmetry. $\forall x,\ y \in X,\ S(x,y) = S(y,x)$
2. Positivity. $\forall x,\ y \in X,\ 0 \leq S(x,y) \leq 1$ (the similarity function is positive and normalized)
3. Reflexivity.$S(x,y) = 1$ Iff $x = y$ (the similarity function is reflexive and normalized)

In case the similarity function satisfies condition 2 (i.e., positive and normalized), one can transform the similarity function into a distance function, such as: $\forall x,\ y \in X,\ D(x,y) = 1 - S(x,y)$.

Since the problem of objects clustering or classification has been investigated, many distances and similarity functions have been proposed [XW05, Cha07, DD09]. The choice of a distance or similarity function determines the quality of the clustering. This means that the best choice is the one that insures the highest modularity, (i.e., strong intra-cluster similarity and a week inter-cluster similarity). Backer and Jain [BJ81] formulate it as, "*in cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity, such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups*". In this thesis, we distinguish two classes of similarity/distance functions: similarity functions between objects described by a vector of multiple features[1], and semantic similarity functions in which only semantic features[2] are taken into account. Thus, the comparison between objects is done by measuring the distance/similarity between them on the basis of the same representation.

[1]Quantitative features

[2]The features are usually extracted from a semantic resource: ontology, dictionary, annotated corpus, etc.

3.2.1. Similarity and Distance Functions Based on Vector of Features

In several domains, such as statistics, objects are naturally represented by their vectors of features. Thus, in the aim of comparing these objects one can use a number of functions ([Cha07, DD09]).

The most used distance functions are derived from the Minkowski distance. This class of functions enables to measure the distance between two vectors by calculating the norm of their difference. The Minkowski distance is a *n-norm* distance where $n \geq 1$ is the order of the distance. Indeed, in the special cases where $n = 1$, $n = 2$ and, $n = +\infty$ we obtain respectively, the Manhattan distance, the Enclidean distance, and the Chebyshev distance. In general, the Minkowski class is well adapted to non-sparse vectors of low dimensionality (i.e., vectors with a small number of features)

Another class of distances is based on the correlation parameter. The Mahalanobis distance measures the distance between two vectors by taking into account the correlation of all the other vectors. Thus, the covariance matrix is calculated in advance. The Pearson distance between two vectors, is based on their linear correlation. Thus, the Person distance is based on the Person correlation coefficient.

Another class of distances is based on the inner product. This class is represented by the Cosine similarity[3]. The Cosine similarity measures the similarity between two vectors by calculating the cosine of the angle between them. Moreover, the cosine similarity is known to be well adapted to the high dimensionality (i.e., vectors with high number of features) and space data.

Tab. 3.1 gives the formulas and summarizes the three families of distances [Cha07, DD09].

3.2.2. Semantic Similarity Functions

In the domain of natural language processing (NLP) there are many applications where the semantic similarity/distance is the key concept, such as word sense disambiguation, document classification, semantic search, etc. Oppositely to the similarity functions based only on vectors of features (cf. sec. 3.2.1), the semantic similarity functions are usually based on semantic resources, such as dictionaries, thesauri,

[3]It can be transformed to a distance since it is normalized

Measure	Formula	Comment
Minkowski distance	$D(x,y) = \left(\sum_{i=1}^{d} \lvert x_i - y_i \rvert^n\right)^{1/n}$	It measure the distance between two vectors by calculating the norm of their difference. The Minkowski distance is a *n-norm* distance.
Euclidean distance	$D(x,y) = \left(\sum_{i=1}^{d} \lvert x_i - y_i \rvert^2\right)^{1/2}$	Special case of Minkowski distance, with $n = 2$.
Manhattan distance	$D(x,y) = \sum_{i=1}^{d} \lvert x_i - y_i \rvert$	Special case of Minkowski distance, with $n = 1$.
Chebyshev distance	$D(x,y) = \max_{1<i<d} \lvert x_i - y_i \rvert$	Special case of Minkowski distance. The limit of the Minkowski distance, with $n \to \infty$.
Mahalanobis distance	$D(x,y) = (x-y)^T S^{-1} (x-y)$, where S is the covariance matrix,	It measure the distance between two vectors by taking into account the correlation of all the data set. S is calculated based on all of the vectors
Pearson correlation distance	$D(x,y) = \frac{(1 - r_{xy})}{2}$, where $r_{xy} = \frac{\sum_{i=1}^{d} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d} (x_i - \bar{x})^2 \sum_{i=1}^{d} (y_i - \bar{y})^2}}$	It measures the distance between two variables based on their linear correlation. $r_{xy} \in [-1, 1]$ is the Pearson correlation coefficient.
Cosine similarity	$S(x,y) = \cos\alpha = \frac{x^T y}{\lVert x \rVert \lVert y \rVert}$	It measures the distance between two vectors by calculating the cosine of the angle (α) between them.

Table 3.1.: Similarity and distance based on vector of features

and ontologies. Indeed, the semantic similarity functions can be organized into four classes [SAR10].

The similarity functions of the first class consist in calculating the path separating two concepts in the ontology where they are represented. The length of the path is calculated in term of the number of edges or nodes it contains. This class of functions includes: Wu-Palmer similarity, Zargayouna similarity, Leacock-Chorodow similarity, Resnik (Edge) similarity, Hirst-St.Onge similarity.

The functions of the second class are based on the concept of Information Content (IC). In information theory, the IC is considered as a measure that quantifies the amount of information a concept contains. The IC of a concept c is: $IC(c) = -log(P(c))$. With $P(c)$ the probability that the concept c is found in a corpus. This class of functions includes: Resnik (IC) similatiry, and Lin similarity.

The functions of the third class are based on the matching of features (attributes and relations) of the concepts. The similarity between two concepts c_1 and c_2 is measured by considering the common features of c_1 and c_2, the features of c_1 that are not in c_2, and the features of c_2 that are not in c_1. As an example of common features, both of concepts *car* and *bus* are used for transport. In this class of functions one supposes that each concept is defined by a set of features. This class of functions includes Tversky similarity and Pirrò similarity.

The last class of functions is the class of hybrid functions. It includes functions that combine techniques from different classes. For example, the Jiang-Conrath similarity is a combination of a path based (edge counting) approach and an information content (IC) approach. The OSS (*Ontology Structure based Similarity*) is another hybrid semantic similarity function. The OSS similarity is based on the a-priori scoring of concepts (a weighing function of concepts) and the length of the path separating them in the ontology. Thus, the similarity between two concepts c_1 and c_2 consists of measuring the transfer of score (denoted T(c_1, c_2)), which is the propagation of the a-priori score of c_1 to c_2 along the path relating them. The transfer (i.e., T(c_1, c_2)) is then transformed to a similarity value.

More recently, [BMI11] proposed a semantic similarity function, which is based on the page counts and text snippets retrieved from a search engine for two words. The idea consists of measuring the co-occurrence of two words by using page counts and extracting patterns of semantic relations (i.e., the different types of semantic relations) existing between two words to identify the features of word pairs. Those

features measure the relation between the two words. Thought the proposed approach outperforms some existing approaches, the problem is that it is solely based on the search engine, which can limit its use.

Tab. 3.2 summarizes the semantic similarity functions discussed above.

3.2.3. Thesis Positioning

3.2.3.1. A Semantic Distance Function

In this thesis, we propose a semantic distance function to measure the distance between search query keywords sec. 5.2.3. Indeed, we are interested in the class of path based functions for several reasons. First, this class of functions is independent from the corpus (contrary to IC based class). Second, it is based on the structure of the ontology (or taxonomy) on which the concepts are represented. This property fits well into our scheme since we are using a hierarchical structure of concepts. Third, in practice it is easier to calculate the semantic similarity between two concepts by measuring the length of the path separating them than measuring their IC or extracting their features. Indeed, in this thesis we propose a semantic distance which is a path-based function. One characteristic of this function is that it is based on a weighting strategy that favors concepts of low level of abstraction. Hence, two adjacent concepts of low level of abstraction are considered to be more similar that two others of higher level of abstraction. The objective of this function is not to calculate the semantic similarity in absolute terms, but to be used to identify clusters in a taxonomy structure.

3.2.3.2. Euclidean Distance for Profiles Comparison

In this thesis, we use the Euclidean distance as a metric to measure the distance between vectors of features (cf. sec. 6.4). The vectors represent both the user profile and data source profile, while the features represent the user interests. The objective is to cluster similar user and data sources. The use of the Euclidean distance is motivated by its efficiency in measuring the distance between vectors of features. In addition to that the usage of the other distances is out of scope of this thesis (e.g., Cosine similarity is adapted for sparse and high dimensional data and Pearson correlation distance is based on the correlation parameter).

Measure	Formula	Comment
Wu-Palmer similarity [WP94]	$S_{Wu-Pa}(c_1, c_2) = \frac{2 * depth(LCS)}{depth(c_1) + depth(c_2)}$ LCS is the least common subsumer of c_1 and c_2	Path based similarity (Edge counting)
Zargayouna similarity [ZS04]	$S_{Zarga}(c_1, c_2) = \frac{2 * depth(LCS)}{depth(c_1) + depth(c_2) + spec(c_1, c_2)}$ $spec(c_1, c_2) = depth(LC) * length(LC,\ c_1) * length(LC,\ c_2)$ LCS is the least common subsumer of c_1 and c_2, LC is the most specific concept in the ontology	Path based similarity (Edge counting) $spec(c_1, c_2)$ calculates the specificity of the two concepts compared to the most specific concept
Leacock-Chodorow similarity [LC98]	$S_{Le-Ch}(c_1, c_2) = -\log\left(\frac{\min\ length(c_1, c_2)}{2 * D}\right)$ D is the max depth of the taxonomy	Path based similarity (node counting)
Resnik (Edge) similarity [Res95]	$S_{ResEdge}(c_1, c_2) = 2D - length(c_1, c_2)$ D is the max depth of the taxonomy	Path based similarity (Edge counting)
Hirst-St.Onge similarity [HSO98]	$S_{Hirst}(c_1, c_2) = T - length(c_1, c_2) - k * R$ T and k are experimental constants R is the number of changes of direction in the path c_1c_2	Path based similarity (Edge counting)
Resnik (IC) similatiry [Res95]	$S_{ResIC}(c_1, c_2) = IC(LCS)$ where,$IC(c) = -log(P(c))$	Information content based similarity
Lin similarity [Lin98]	$S_{Lin}(c_1, c_2) = \frac{2 * IC(LCS)}{IC(c_1) + IC(c_2)}$	Information content based similarity
Tversky similarity [Tve77]	$S_{Trv} = \alpha F(\psi(c_1) \cap \psi(c_2)) - \beta F(\psi(c_1)/\psi(c_2)) - \gamma F(\psi(c_2)/\psi(c_2))$ F reflects the salience of a set of features $\psi(c)$ represents the features of the concept c α, β, γ express the focus on each component	Features matching based similarity
Pirro similarity [Pir09]	$S_{pirro}(c_1, c_2) = \begin{cases} S_{Trv'}(c_1, c_2) & c \neq c_2 \\ 1 & c = c_2 \end{cases}$ $S_{Trv'}(c_1, c_2) = 3 * IC(LCS) - IC(c_1) - IC(c_2)$	Features matching based similarity an adaptation of the Tversky similarity to IC
Jiang-Conrath similarity [JC97]	$S_{Ji-Co}(c_1, c_2) = \frac{1}{IC(c_1) + IC(c_2) - 2 * IC(LCS)}$	Hybrid similarity measure It combines a node counting approach and an IC approach
OSS similarity [SZF07]	$S_{OSS}(c_1, c_2) = 1 - \frac{log(T(c_1, c_2))}{maxD}$ $T(c_1, c_2)$ is the transfer of the a-priori score from c_1 to c_2, $maxD$ is the max distance between any concepts in the ontology	Hybrid similarity Combines the a-priori scores of concepts in specific contexts and the distance between them in the ontology

Table 3.2.: Semantic similarity functions

3.3. Baseline Clustering Algorithms

The literature on clustering algorithms is generally divided into several classes. The most known classes include, hierarchical, partitioning and density based algorithms [XW05, Ber06, Cha07, DD09]. The next subsections detail some baseline algorithms of each class. The algorithms are selected according to some parameters in which we are interested. These parameters are: complexity, granularity, and outliers processing. The complexity defines the scalability of the algorithm. The granularity discusses the capability of the algorithm to deal with data in different level of granularity. The outliers processing parameter discusses the resistance of the clustering method against outliers. In this thesis, the granularity parameter is highlighted in chapter 6 (sec. 6.2).

Tab. 3.3 summarizes the algorithms we have selected.

3.3.1. Hierarchical Algorithms

The hierarchical clustering methods consist in constructing a hierarchy of clusters in the form of a tree, called *dendrogram* [Ber06]. Each node of the dendrogram contains a child cluster, which is a part of a parent cluster. Thus, the hierarchical partition enables clustering on different levels of granularity. Moreover, hierarchical clustering methods are divided into agglomerative (bottom-up) and divisive (top-down) [JD88]. An agglomerative clustering starts with clusters of one object and recursively merges the two most appropriate clusters until the cluster of all of the objects. In the opposite, a divisive clustering starts with one cluster containing all the objects and recursively splits the most appropriate cluster until clusters of one objects. The result of the clustering in both of the two approaches corresponds to a cluster partition that satisfies a given criterion.

Obviously, the advantage of hierarchical clustering is the flexibility regarding the level of granularity. However, the major drawback is that there is no way to backward once a cluster is constructed, which prevent any possible improvement.

3.3.1.1. Linkage Methods

Merging or splitting (agglomerative/divisive) clusters in hierarchical clustering relies basically on a similarity or a dissimilarity (distance) function (d). However,

a similarity/distance function is defined to compare individual objects. Since, the hierarchical clustering involves comparison between clusters (partition), the similarity/distance function has to be generalized to clusters (D). Indeed, there are three different methods to calculate the distance between two clusters C_1 and C_2:

First, the distance between C_1 and C_2 is the distance between their two closest objects, this method is called single linkage:

$$D(C_1, C_2) = min\,\{d(x, y) \,|\, x \in C_1,\ y \in C_2\}$$

Second, the distance between C_1 and C_2 is the distance between their two farthest objects, this method is called complete linkage:

$$D(C_1, C_2) = max\,\{d(x, y) \,|\, x \in C_1,\ y \in C_2\}$$

Third, the distance between C_1 and C_2 is the average distance between their all of objects, this method is called average linkage:

$$d(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{x \in C_1} \sum_{y \in C_2} d(x, y)$$

These methods has been respectively implemented by: [Sib73], [Sei89], [Def77]. In general, the time complexity of linkage based hierarchical clustering is $O(n^2)$.

3.3.1.2. Shape and Outliers Sensitive Methods

The linkage based methods suffer from two problems. The first problem is that these methods are adapted to clusters of convex shape, while in real life data, the clusters shape is more complex. The second problem is that these methods are not adapted to deal with noise and outliers, which can distort the clusters. Some attempts to deal with these issues are CURE [GRS98] and CHAMELEON [KHK99]. CURE represents the clusters by a fixed number of representative objects (a sample). Thus, the distance between two clusters is the minimum distance between two objects of

their corresponding representatives. The Complexity of CURE is $O(n^2_{sample})$, with n_{sample} the size of the sample (i.e., the representatives). CHAMELEON is composed of two steps. In the first step, it represents the data set (the objects) as a k-nearest neighbor graph, in which each object is connected only to its k-nearest neighbors (the rest of the edges are removed). The result of this step is a set of sub-clusters of k-connected objects (initialization step). In the second step, the sub-clusters are merged using a hierarchical merging approach to find the clusters. The merging process depends on the inter-connectivity within the clusters and their closeness. Thus, two clusters are merged if their inter-connectivity is high and they are close together. Moreover, the size of a cluster has to be higher than a threshold fixed by the user. The complexity of CHAMELEON is $O(n\,m + N\,log\,N + m^2\,log\,m)$, with m the number of the initial sub-clusters and n the size of the data set.

3.3.2. Partition-Based Algorithms

The principle of the partition based algorithms is to split a given set of objects into a fix number of clusters. Unlike hierarchical methods, in which it is not possible to backward during the clustering process, partition based algorithms construct the clusters by iterative improvements. Thus, these algorithms use some heuristics to optimize the number of iterations when choosing the most appropriate objects to cluster. The advantage of the partition based algorithms is that they are more scalable compared to hierarchical algorithms, while the drawbacks are the difficulty to define the best number of clusters, the fact that the result of the clustering is a local optimum, and the sensibility to outliers. In terms of granularity these algorithms are not adapted since the clustering is iterative and depends on a fix number of clusters.

The most well-known methods in this class of algorithms are K-Means and K-Medoids.

3.3.2.1. K-Means Methods

The K-Means algorithm [HW79] is based on the parameter k, which represents the number of clusters to find. The K-Means algorithm consists mainly of four steps. In the first step the algorithm selects randomly k objects, which initialize k clusters. Those, objects are initially the means of these clusters. In the second

step, each of the remaining objects is assigned to the closest cluster (i.e., the cluster mean). This is done by calculating the distance (the Euclidean distance) between the object and the mean of the cluster. In the third step, the means (the mean of a cluster is not necessarily a real object) of each cluster is recalculated according to the new configuration (i.e., the clusters). The steps two and three are repeated until a stopping criterion is achieved, which is the square-error criterion: $E(m) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\|^2$, with x an object of cluster C_i and m_i the mean of the cluster C_i. It is the sum of the square error between the objects of each cluster the mean of that cluster. The clustering process stops when $E(m)$ cannot be reduced. The complexity of the general K-Means algorithm is $O(nkl)$, where n the number of objects, k the number of clusters, and l the number of iterations until the stopping criterion.

3.3.2.2. K-Medoids Methods

The K-medoids algorithm [KR87] processes as K-Means. However, contrary to K-Means, which represents a cluster by its means, in K-medoids a cluster is represented by one of its points, which is the medoid. In the first step, the algorithm selects randomly k objects as medoids. In the second step, each of the remaining objects is assigned to the closest medoid. In contrary to k-means, the distance measure is not necessarily the Euclidean distance; it could be any dissimilarity measure. In the third step, the medoids are replaced by new ones, such that they minimize the square-error: $E(o_m) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - o_m\|^2$, with x an object of cluster C_i and o_m the medoid of the cluster C_i. The steps two and three are repeated until $E(o_m)$ cannot be reduced. The complexity of the general K-Medoids algorithm is $O(n^2)$, where n the number of objects.

3.3.3. Density-Based Algorithms

Another class of clustering algorithms is based on the concept of density. In this class, there are mainly two methods. In the first method the density is defined by a set of training data points. This method is implemented in DBSCAN. In the second method the density is defined by a point associated with a density function. This method is implemented in the algorithm DENCLUE. The advantages of the

density-based algorithms are a good scalability and a good resistance against outliers. However, there are two main drawbacks. First, the clustering depends on the user input parameters, and second, the algorithm often fails to distinguish between two clusters when their densities are very similar.

In terms of granularity, these algorithms are not adapted for two reasons. First the clustering depends usually on a user threshold (the min size of a cluster), and second, the clustering process consists in merging sub-clusters with high density.

3.3.3.1. DBSCAN

DBSCAN (Density Based Spatial Clustering of Applications with Noise) [EKSX96] is one of the very known density-based clustering algorithms. The algorithm has two input parameters, which are:

1. "ϵ": A threshold distance used to delimit theϵ-*neighborhood* of the point x representing the points whose the distance from x is less than ϵ. Theϵ-*neighborhood* denoted as $N_\epsilon(x)$ is defined as: $N_\epsilon(x) = \{y \in X \,|\, dist(x, y) \leq \epsilon\}$, where X is the set of all the points.
2. *"MinPts"*: A minimum number of points required in the ϵ-*neighborhood* of x to form a cluster. If the condition$|\ N_\epsilon(x)\ | \geq MinPts$ is satisfied, then x is called a *core point*.

The authors define a cluster by using the key concepts of *density-reachable* and *density-connectivity* relationships wrt. ϵ and *MinPts*.

The density-reachable relationship is defined between a core point and another arbitrary point. A point y is *density-reachable* from a core point x if y is in ϵ-*neighborhood* of x or there is a finite sequence of core points x, $x_1 ... \ x_n$, y such as each point in this sequence belongs to the ϵ-*neighborhood* of its predecessor.

The density-connectivity extends the definition of density-reachable, since it enables to define a density based relationship between two points when they cannot be connected by a simple density-reachable relationship. Thus, two points x and y are *density-connected* if there is a core point o from which both x and y are density-reachable.

The authors define a density-based cluster as a subset of D (a database of points) satisfying the following conditions:

- $\forall\, x,\, y$: if $x \in C$ (cluster) and y is density-reachable from x, then $y \in C$
- $\forall\, x,\, y \in C : x$ is density-connected to y

To discover the clusters, the DBSCAN algorithm follows the steps bellow:

1. For each unvisited p in dataset D
   a) Mark p as visited
   b) Discover the ϵ-*neighborhood* of p,
   c) If p satisfies the *MinPts* condition (p is a core point) then
      i. Create a new cluster C
      ii. Add p to C,
      iii. ExpandCluster (p, C)
   d) Else
      i. p is a noise (p could be part of another cluster if it is density-reachable)

ExpandCluster function explores the ϵ-*neighborhood* of the core point p and try to find new density-reachable points. The function follows the steps bellow:

1. For each q in ϵ-*neighborhood* of p,
   a) If q is not visited
      i. Mark q as visited
         A. Discover the ϵ-*neighborhood* of q,
         B. If q satisfies the *MinPts* condition,
            Add ϵ-*neighborhood* of q to ϵ-*neighborhood* of p,
   b) If q do not belong to any other cluster then
      i. Add q to C (q could be either a core-point, a non-core point or considered previously as noise of another cluster)

The complexity of the DBSCAN algorithm is mainly related to the process of discovering theϵ-neighborhoods of a point. In order to discover the ϵ-neighborhoods of each point the algorithm relies on R*-tree indexation whose complexity is of $O(log(n))$ where n represents the whole number of points. Therefore, the overall complexity of DBSCAN is $O(n\, log(n))$ where n is the number of points.

In the end, the DBSCAN algorithm was generalized to consider more complex objects rather than simple points (e.g., polygons). This gave rise to the GDBSCAN algorithm [SEKX98].

3.3.3.2. DENCLUE

The algorithm DENCLUE (DENsity-based CLUstEring) [HHK98] is another density-based algorithm whose the approach is different from the DBSCAN. In fact, instead of density-connectivity and density-reachable concepts, DENCLUE uses the concept of density function. To define the density function, the authors introduced first, the influence function which models the influence of a data point within its neighborhood. Thus, the density function at a point x is defined as the sum of the influence functions of the overall data points at point x. Given a basic influence function f_B, the general definition of the density function f_B^D at point x is defined as:

$$f_B^D(x) = \sum_{i=1}^{n} f_B(x, x_i) = \sum_{i=1}^{n} f_B^{x_i}(x)$$

Where $D = x_1, ..., x_N$ is the set of feature vectors of the overall N points and $f_B^{x_i}(x)$ is the influence function of point x_i at point x. In real data set, the influence function of each point can take different forms, such as square wave influence function f_{Square}, or Gaussian influence function f_{Gauss} (details are included in [HHK98]).

The definition of a cluster in DENCLUE relies on the concept of *density-Attractor* and *density-attracted* points.

A point x^* is a *density-attractor* for a given influence function f_B, iff $x^*$is a local maximum of the density-function f_B^D.

This leads to the definition of *density-attracted* point. A point x is density-attracted by x^* iff :

$\exists\, k \in N\ : d(x^k,\, x^*) \leq \epsilon$ with: $x^0 = x,\ x^i = x^{i-1} + \delta . \dfrac{\nabla f_B^D(x^{i-1})}{\parallel \nabla f_B^D(x^{i-1}) \parallel}$, (i.e., find i that corresponds to k)

$\nabla f_B^D(x)$ is the gradient of $f_B^D(x)$.

The DENCLUE algorithm finds the clusters following the next two steps:

The first step is the pre-clustering. It consists in constructing a map of highly populated areas by dividing the data space into a set of neighboring hypercubes. The number of the hypercubes depends on the value of σ, which determines the length of a hypercube (2σ). Thus, the hypercubes are numbered according to their position from a given origin in the map. The highly populated neighboring hypercubes are merged. More formally, if two clusters c_1 and c_2 are highly populated (according to a predefined threshold ξ) and $d(mean(c_1), mean(c_2)) \leq 4\sigma$ then c_1 and c_2 are merged. Obviously, the objective of this step is to produce a map of highly populated areas of connected hypercubes (the non-connected hypercubes represents the outliers).

The second step is the effective clustering step. In fact, the algorithm calculates the *local* density function at each point x (i.e., $\hat{f}^D(x)$), which is the density function at x that takes into account the influence of the points in its neighborhood (i.e., a distance of 4σ). Furthermore, the algorithm uses the local density function to identify the density attractor (x^*) (i.e., the point that maximizes the local density function). Indeed, the clusters are identified by finding the density attracted points of each density attractor.

The complexity of the DENCLUE algorithm in the worst case is $O(n\,log\,n)$, with n the number of points.

Cluster algorithm		**Time Complexity**	**Granularity**	**Outliers**
Hierarchical clustering	Linkage metrics	$O(n^2)$	Yes	No
	CURE	$O(n^2_{sample})$	Yes	Yes
	CHAMELEON	$O(n\,m + N\,log\,n + m^2\,log\,m)$	Yes	Yes
Partitioning clustering	K-Means	$O(nkl)$	No	No
	K-Medoids	$O(n^2)$	No	Yes
Density-based clustering	DBSCAN	$O(n\,log\,n)$	No	Yes
	DENCLUE	$O(n\,log\,n)$	No	Yes

Table 3.3.: Baseline clustering algorithms comparison

3.3.4. Thesis Positioning

In this thesis, we propose a clustering algorithm based on a taxonomy structure of keywords sec. 6.2.1.1. The objective of the algorithm is to cluster the keywords by pruning the taxonomy. Indeed, the algorithm can be classified with the hierarchical algorithms by assimilating the taxonomy to a simplified dendrogram. In fact, the difference between a real dendrogram and the keyword taxonomy is that the in the dendrogram the nodes are child clusters and parent clusters related by the partition relationship, while in the taxonomy the nodes are single keywords (concepts) related by the IS-A relation. The algorithm is characterized by a low complexity and defines a certain clustering granularity. Moreover, the algorithm measures the distance between keywords by using a semantic distance (cf. sec. 5.2.3).

3.4. Summary

The state of the art on distance functions distinguishes two types of distance functions (cf. Tab. 3.1 and Tab. 3.2); those based on vectors of features and those based on semantic features. This chapter reviews the main existing functions for both of the two types of functions. Thus, functions based on vector of features are split into three classes, each of which has a specific usage. The choice made for a function depends on the target application and the usage of these functions. Indeed, the Minkowski class is well adapted to non-sparse vectors with low dimensionality. At the opposite of the inner product (i.e., the cosine similarity) class which is well adapted to high dimensionality and sparse vectors. Otherwise, the class of functions based on the correlation parameter is mostly used in statistics when the objects are supposed to be highly correlated. The functions based on semantic features are split into four classes. The choice made for a function depends on the semantic resource being used. Indeed, the class of path based functions is well adapted in case the there is a semantic structure that represents the concepts (e.g., ontology, taxonomy, etc.). The class of functions base on the information content (IC) is well adapted in case the concepts are extracted from an existing annotated corpus (BNC, SimCor, etc.). The class of features matching based functions is well adapted in case the concepts are clearly defined by their corresponding attributes and features. The class of hybrid functions is well adapted in case there are multiple semantic resources.

The state of the art on clustering algorithms is divided into several classes (cf.

Tab. 3.3). This chapter reviews the main three classes, which are the class of hierarchical algorithms, the class of partition-based algorithm, and the class of density based algorithms. The review is done following four parameters: the complexity, the granularity and the outliers processing. Indeed, the algorithms of this class are the best in term of granularity, since they enable to show several clustering levels with different size of clusters (partitions), which corresponds to the granularity levels. However, the class of hierarchical clustering is characterized by relatively high time complexity compared to the other classes. In addition to that, they are sensible against outliers, except some algorithms (CURE, CHAMELEON), which are relatively resistant against outliers. The partition based algorithms are characterized by a less complexity compared to hierarchical clustering and better resistance against outliers (K-Medoids). However, this class of algorithms does not define any kind of granularity since the cluster are defined iteratively from the initial set of clusters. The class of density based algorithms is characterized by a lowest complexity over the reviewed classes; in addition to that it has a good resistance against outliers. However, as for partition based algorithms, this class do not define any kind of granularity, since the clustering depends usually on a user threshold (e.g., the min size of a cluster), and the fact that the clusters are obtained by merging sub-clusters (e.g., ϵ-neighborhood and hypercubes).

Part III.

A Framework for Usage Analysis in Information Retrieval Systems

4. Overview of a Framework for Usage Analysis in IR systems

4.1. Introduction

Nowadays, the world of computer science is overwhelmed by applications intended to be used by always connected end-users. In this kind of applications, the quality of service is directly related to the user satisfaction. This implies that the design process should integrate the user model. Depending on the role given to the model, we can distinguish two different levels of application: individual and group levels. At the individual level, the user model, which is also the user profile, is mostly used for adaptation and personalization, i.e., basically, to improve the human-machine interaction (the interface) [Fis01] or to adapt the information content with respect to the user interests and characteristics [JCECWtC00]. As examples of individual user profile we can cite: web page personalization and recommender system. At the group level, the user profile is extended to groups of users sharing similar characteristics. In addition to the advantages related to a single user profile, the profile of a group allows the users to exchange easily their content within the group, especially in some specific network applications, like: sharing data over P2P networks, data propagation on mobile networks or extracting documents in a distributed information retrieval system.

Along with the need to identify the user profile in a distributed environment, the need to model the data-source emerges for the same reasons as for the user. In fact, a data-source can play symmetrically the role of a user Fig. 1.1. If we consider, that the user's interactions are requesting data from data-sources and exchanging data with other users, the data-source interactions are responding to the user request and exchanging data with other data-sources. Consequently, the applications deployed on a distributed environment composed of users and data-sources should integrate

both the user model and the data-source model.

There is an important amount of work, that has attempted to resolve the problem of user modeling. However, existing techniques (sec. 2.2) depend strongly on the domain application and the usage data. We mean by usage data all the information about the user, resulting **implicitly** from an interaction with the system or specified **explicitly** by the user (e.g. user preferences). For instance, in the domain of interface adaptation the usage data are essentially composed of user behavioral actions (click-through, focus areas, etc.) and user preferences (color, screen size, etc.). In this perspective, the technique used to model the user definitely depends on the usage data to analyze.

In this thesis, we focus on the problem of modeling the user in a distributed environment. The objective is to propose a model which offers the possibility of organizing the network of users and data sources in the form of groups which share similar interests, so that, the model should in the end, improve the communication within the network. In our approach we consider that:

- The domain application is a network of users and data sources, which means that it is necessary to model the two entities.
- The usage data are textual, which means that the users interactions with the system have a textual form.

To resolve this problem we proposed a generic framework (Fig. 4.1) that integrates a unified model to extract both the user profile and the data source profile from textual usage data. The main strong points of the proposed model are :

- Ability to model both of the user and the data source in one unified model
- Semantic analysis of usage data
- The efficiency of the model in the profile extraction which enables high scale computations and easy updates.

In our study, we consider as usage data the user query logs extracted from a web search engine (Fig. 4.1). The log has a form of a table with three entries which are: the user_id (the user), the textual query (the usage data) and the selected_url (the data source). However, the model is actually applicable for any textual usage data. In the next sections, we give a short overview of the different steps of the profiling model. Step 3 is detailed in chapter V while steps 4, 5 are detailed in chapter VI.

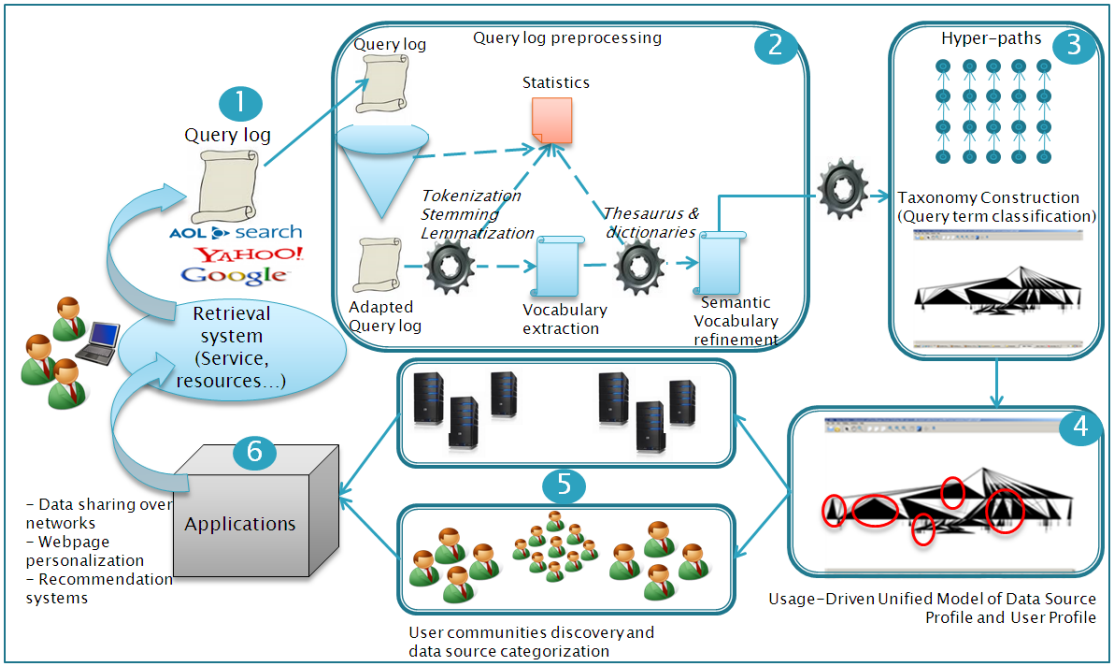


Figure 4.1.: A Framework for Usage Analysis

4.2. Data Gathering and Query Log Preprocessing

Data gathering and query log preprocessing (Fig. 4.1, components 1 and 2) are the two first steps in the usage analysis framework. It consists first, in collecting the user queries submitted to a web search engine[1] (Fig. 4.2(a)), and second, in preprocessing the resulting log by filtering out meaningless queries and keywords.

The collecting process consists in recording the queries submitted by the user through the search engine and for which he/she has selected at least one result item (i.e., a URL). The objective behind keeping only queries for which the user selected result items is twofold. First, it enables to avoid wrong queries (i.e., queries that are re-formulated by the user after reading the summaries). Second, it enables to join the user and the URL, which represents the data source. As we will show further in this thesis, the fact that the user and the data-source are joined by the query represents the basis on which the profile model will be constructed. In the end, the result of this step is an adapted query log containing a set of entries ordered by timestamp, where each entry is composed of the user ID, the text query and the URL.

The preprocessing step aims to obtain a refined vocabulary of query terms that will

[1]In our experiments, the AOL search engine

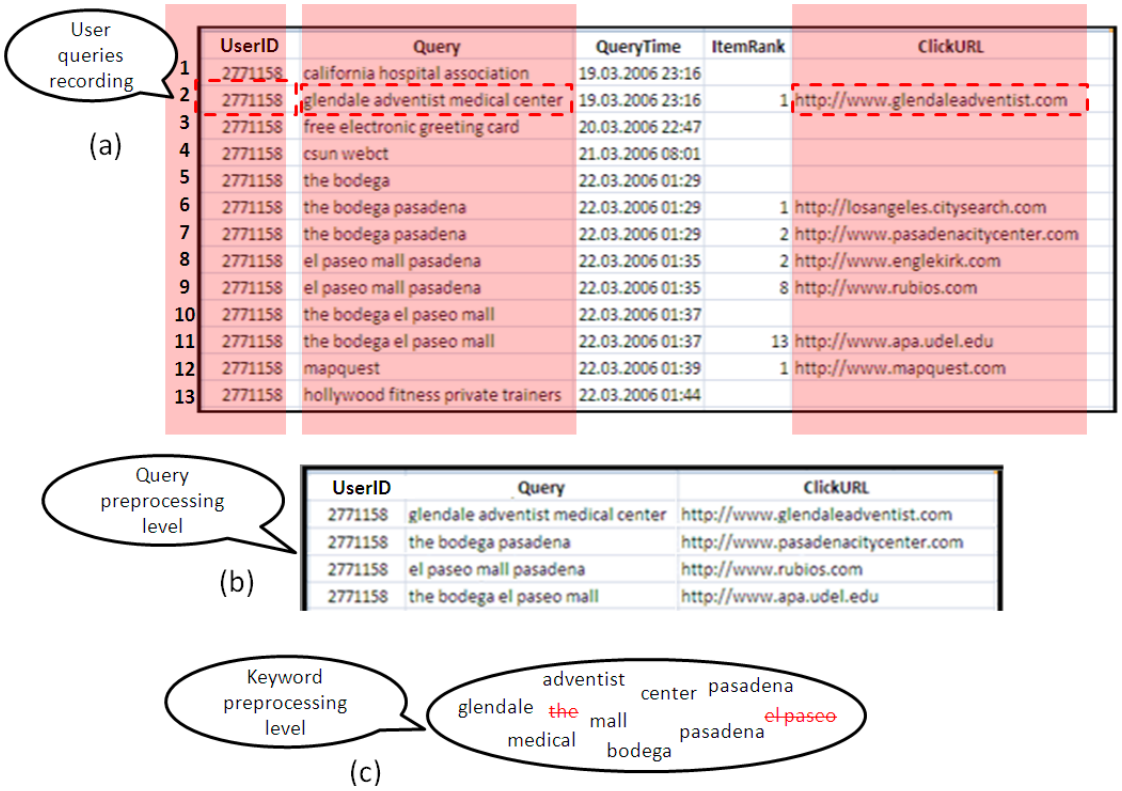


Figure 4.2.: AOL Search Query Log

form the set of elements of a semantic representation of the log. There are two levels of preprocessing: the query level and the term level. Query preprocessing is mostly a filtering process. First, identical queries[2] resulting from the structure of the logs are deleted. Second, the "bad queries" are filtered out. We mean by a bad query a non interpretable query, e.g., a wrong URL, an email address, etc. This kind of query may for example occur when the user is trying to check a text spelling without expecting relevant results. At the term level, we apply a classical lexical analysis on the queries. First, each query is split into a set of terms by means of a tokenizer. The goal is to extract the set of meaningful keywords. The tokenizer deletes all stop words, such as articles and prepositions. Second, each term is processed separately using stemming and lemmatization methods. These tools group together the different inflected forms of a word into a single item, e.g., the plural and singular form of a noun are identified as a single item. Third, query terms that appear more than once in the log are deleted. Fourth, we match each term with a WordNet synset; terms that are not in WordNet are considered as

[2]The query is duplicated if the user clicks on several result items

non-interpretable.

Fig. 4.2 shows a part of a log that concerns the user "2771158" (cf. step (a) in Fig. 4.2). Applying the preprocessing steps (i.e., at query level and at keyword level) to these queries produces first, a new (adapted) log (cf. step (b) in Fig. 4.2) and then a set of refined keywords (cf. step (c) in Fig. 4.2). In more details:

At query level preprocessing :

- The queries without click (data source) are filtered out. In the example, these queries are: 1, 3, 4, 5, 10, 13
- The wrong queries, which are reformulated (i.e., by adding keywords) are filtered out. In the example, query 5 is reformulated to query 6. Hence, query 5 could be filtered out.
- The bad queries (i.e., non interpretable queries) are filtered out. In the example, it is query: 12
- The duplicated queries are filtered out. In the example, these queries are: 6, 8, 10

At keyword level preprocessing :

- The stop words are deleted. In the example, the stop word is: "the"
- The words that are not found in the dictionary (WordNet) are deleted. In the example, the non-defined word is: "el paseo"

4.2.1. Preprocessing Statistics

The preprocessing step is mostly a filtering process in which same queries and keywords are deleted. In order to measure the usability of the post-processed query log, we specify two kinds of thresholds : local usability threshold T_l and global usability threshold T_g. The local usability threshold enables to determine weather a query is kept in the log or is deleted. This threshold is the result of the keyword preprocessing level where several keywords are deleted. This threshold depends on the ratio r between the number of remaining keywords and the number of keywords in the input query. If $r \geq T_l$ then the query is kept else it is deleted. The global usability threshold enables to determine whether the log is ready for further analysis. It depends on the ratio R between the number of post-processed (non-deleted) user

queries and the number of input user queries. If $R \geq T_g$ than the result of the step 2 passes to the step 3, else the process waits for additional queries.

4.2.2. Lexical Analysis

The lexical analysis is based on three main techniques : Tokenization, Stemming, Lemmatization.

4.2.2.1. Tokenization

The tokenization is the process whereby a text (phrase, query, etc.) is transformed into a set of keywords; empty words (articles, piece of URL, numbers, etc.) are deleted from the output set of keywords. The process consists in parsing an input query and extracting words separated by predefined separators. In the case of a query, keywords are mostly separated by white spaces.

4.2.2.2. Stemming

The stemming aims to analyze morphologically the keywords obtained by tokenization to reduce them to their canonical form, i.e, the *stem* (e.g. are->ar, saw,see->s, cars,car's->car). The stemmer we use in this thesis is based on the Porter stemmer [Por80] which determines the stem of a word by eliminating or replacing the suffix. The Porter stemmer is based on a list of rules organized by groups into five steps. Each rule deals with one suffix and is composed of a condition and an action. The condition checks the part of the word before the suffix, and the action eliminates or replaces the suffix. Thus, depending on the part before the suffix a decision is made about eliminating or replacing the suffix. For a given word "*Stem+suffix*" the stemmer starts parsing by the end to identify the suffix that matches one of the rules. If the condition of the rule is satisfied the action is applied and the stemmer swishes to the next step, until the end of the rules.

4.2.2.3. Lemmatization

The lemmatization aims to reduce the input word to its base form by considering the part of speech (word class: noun, verb, preposition, etc.) of the word; the result

of the process is a *Lemma* (e.g. am, is, are->be). Lemmatization has the same objective as stemming. However, lemmatization is more complex, since in addition to deal morphologically with the word, the retrieved lemma should be a real word. In contrast, a stem is sometimes a truncated word. One consequence of this difference is that a lemmatizer is often slower compared to stemmer. In fact, a lemmatizer can be defined as composed of two phases. In the first phase, the lemmatizer has to identify the part of speech of the word which represents the category of the word (noun, verb, etc) within its context (phrase, query, etc.). A word whose the POS is identified is said to be annotated. The POS of the word is used to determine correctly the action to perform in the next phase. This first phase is the one that slows the lemmatization since its based on a training model. This latter is a set of previously annotated words (a corpus) used to predict the POS of future words by using a probabilistic model. In the second phase, the lemmatizer applies the appropriate rule to eliminate or to replace the inflectional part of the input word. These rules are different from those used in stemming since the objective is not necessarily to deal with suffixes and produce very often real words. The function "LuceneLemmatizer" is based on the Stanford POS tagger[3] [TKMS03, Rat96]. It deals with the input words all at once; it analyzes them by using a pre-trained model (trained on the *Wall Street Journal*) to identify the POS of each word. The words and their corresponding POS are used by the reduction rules to identify the lemmas.

4.2.3. Semantic Analysis

In this thesis, we use mainly WordNet as source of semantics. Nevertheless, there exist other sources of semantics such as DBpedia that is possible to combine with WordNet:

- WordNet : It is a lexical database for the English language [FM98]. In Wordnet, words (nouns, verbs, adjectives) are grouped into sets of synonyms called synsets. As a result, a polysemous word can be included in several synsets. In addition to synsets, wordnet provides definitions, and presents several semantic relations between synsets, for instance : Is-A, Part-of, etc. One of the most important objective of Wordnet is to support automatic text analysis. Moreover, it is widely used by text disambiguation technics. The Wordnet dataset describes about 200000 words which represent about 115000 synsets.

[3]http://nlp.stanford.edu/software/corenlp.shtml

- DBpedia : It is a structured representation of the information available in the Wikipedia resources[4]. It provides users with an interface to query relationships and properties associated with Wikipedia resources, including links to other related datasets. The DBpedia dataset describes more than 3.64 million entities, including, persons, places, organizations, etc. In order to represent the relationships between the objects, DBpedia uses the Resource Description Framework (RDF).
- Combined dictionaries, thesauruses and ontologies: In order to complete the lack of information in a dictionary or in an ontology, we need to combine them with others. This is the case with Wordnet. Although it covers a large space of words, Wordnet suffers from the lack of named entities. The solution consists in combining it with another ontology. For example, DBpedia and Wordnet are already linked by 318,000 RDF links[5].

4.3. Taxonomy Construction : a Support for Analysis

The result of the first two steps is a refined vocabulary of query terms which are the inputs of the next step. The third step in the framework (Fig. 4.1, component 3) attempts to produce a basis on which all the future calculations of the model will be done. More specifically, it consists in producing a global semantic reference for the whole log before starting any process of query log mining. Query log mining is frequently done by extracting meaningful patterns from the whole set of queries. In this thesis, we consider the problem of patterns extraction to be a clustering problem. This means that more attention is given to groups of queries independently from the whole set. However, we argue that it is also important to understand the relationships between the different discovered patterns, and to compare patterns between users. In the case of keyword-based queries, the extracted global reference should take into account semantic relations that exist between natural language words.

To this end, we outline a method to construct a taxonomy over the terms used in keyword search logs. This method uses the WordNet lexical database of English terms. The construction of the taxonomy is based on two principal aspects: the first

[4]http://wiki.dbpedia.org/About
[5]http://wiki.dbpedia.org/UseCases, 2012

is the hierarchical relationship between the terms established by the hypernymy and generalization/specialization relations ("is a"). This kind of relationship enables to sort the terms into different levels of abstractions and then gives rise to a tree structure. The second aspect is the semantic distance between each pair of neighboring terms; it represents the weight of the edges of the tree. Here, we introduce a weighting function that takes into account the abstraction level of each element. In fact, two terms at the bottom of the hierarchy are considered to be closer than two terms located at the top of the hierarchy.

4.4. Usage-Based User Profile and Data Source Profile Modeling

The step four (Fig. 4.1, component 4) is the step when the profile model is established. It is composed of two sub-steps. In the first sub-step, we apply a pruning algorithm on the taxonomy (of step 3) which is actually a clustering algorithm on a tree structure. The goal is to extract the general users' interests by considering together all the queries, independently from the users. Hence, the result of this sub-step is a model of user interests. In the second sub-step, the model of interests evolves to a profile model. It consists in instantiating the model to take into account the characteristics of each individual user, and each individual data source.

4.4.1. Extracting User Interests : a Pruning Algorithm

The goal of the clustering algorithm is to transform the tree structure obtained in step 3 into a set of sub-trees. To achieve this purpose, the algorithm is based on three parameters, which are: the size of a cluster and the abstraction level in one side and the semantic distance between clusters in the other side. The key idea is to maximize a certain quality function F which depends on these parameters. The specification and the role of each parameter are defined as follows :

- Size (s) : It measures the number of terms within a cluster by taking into account the frequency of each term. Thus, $s = \sum_{i=1}^{k} fr(t_i)$, with t_i a term of the cluster, and k the number of terms within the cluster.

  - Frequency (fr) : It measures the number of occurrence of a term in the queries.

- Abstraction level (l_i): It measures the abstraction level of a cluster i based on the depth of the its elements in the taxonomy.
- Distance (d) : It measures the distance between terms and between clusters. The distance between two terms is directly related to there abstraction level l. Thus, the lowest the abstraction levels of (adjacent) terms is, the smallest the distance is.

The size s_i of a cluster i and the abstraction level l_i of its elements enable to determine the local cluster quality value "$v = f(s_i, l_i)$" while the distance enables to determine the value of the threshold for clustering.

4.4.2. A Two-Face Model of User Profile and Data Source Profile

After having built the model the users of interests (cf.sec. 4.4.1), the next step consists in adapting it to characterize individual users and individual data sources. Basically, the task is to recalculate the value v_i of each cluster by taking into account only the queries submitted by each individual user.

The process consists in three sub-steps. First, we calculate the contribution of each individual user in the query log by deleting the terms that do not appear in his/her own queries. It consists in recalculating the value of v_i of each cluster according to the remaining terms. Second, in order to include the $influence$ of the child clusters on the their corresponding parent, the value of $v_i$of each child cluster is propagated to the parent cluster. The propagated value is equal to v_i/d where d is the distance between the child cluster and his parent cluster. The third sub-step consists in normalizing the vectors v_i obtained from the last sub-step. It consists to divide the components of the each vector v_i by the component with the max value. The objective of this step is to remove the gap between active users (with frequent query) and less active users.

Symmetrically, in order to calculate the vector v_j of a data source, the same process is applied from the data-source side. The only difference is in the subtracting step. The terms that do not appear in the queries submitted to the data source are subtracted from the initial clusters.

4.5. Usage of the Model

The dual nature of the profile model introduces two equivalent but independent applications (cf. sec. 6.4): user communities discovery and data source categorization (cf. Fig. 4.1, component 5). The main objective of these applications consists in grouping together entities that share similar objects. Therefore, we define a community as a set of users, who share similar interests while a data source category as a class of data sources, which share similar contents. In addition to these two applications, the profile model enables to directly compare the user profile to the data source profile which enables to create mappings between the users and the data sources. In fact, the mappings process is typically a recommendation process.

User Community Discovery In order to group the users according to their interests, their respective vector of profile are clustered together by using a clustering algorithm.

Data Source Categorization Symmetrically, In order to group the data sources according to their similar contents, their respective vector of profile are clustered together by using a clustering algorithm

Mappings discovery between the users and the data sources The mappings are identified by calculating the distance between each user profile vector and data source profile vector. Thus, if the distance is greater that a predefined threshold a mapping is created between the user and the corresponding data source.

4.6. Summary

In this chapter, we have briefly presented the framework we propose for usage analysis. The goal was to give a global description of the contribution and to show relationships between the different components of the framework.

The proposed framework is composed of six steps. The step one consists in collecting the textual usage data to be analyzed. The step two (per-processing step), consists in preparing the data for further analysis; the objective of this step is to obtain a refined vocabulary that will be used in the next step. In the step three, we

construct the basis on which the profile model will be calculated; it consists in an enhanced taxonomy of terms obtained from the previous step. In the step four, the model is calculated in two sub-steps. First, we apply a clustering algorithm that extracts a model of user interests in the form of clusters of keywords. Then, the model of interests is enhanced to a user profile model that takes into account the characteristics of each user. The step five, presents direct usages of the model, which are the communities discovery and the data sources categorization. Finally, the step six represents the applications that can be deployed on the network that can make use of communities and data source categories. Details of steps three, four and five will be given in the next two chapters.

5. Keyword-Based Query Log Analysis

5.1. Introduction

Search query logs have always been considered to be a rich source of information about the user behavior [JST08]. Mining processes are usually applied to search query logs in order to extract knowledge about the usage [Jan06]. This is in particular a necessary step for the design of true user-centric applications in which user search interests are identified and taken into account [ZN08, LOPS07, WNZ02]. In recent years, much research has been done in the domain of search query logs analysis. To date, researchers have mostly focused on analytical and statistical methods for extracting knowledge from log [WGB12, JCG10, HGM+10, BLL+10, Jan07, PCT06, SMHM99]. However, most of these proposals lack a mechanism to extract semantic features and include them in the analyzing process. This makes them unable to deal with problems related to the semantics of the data such as the identification of users search interests. In this chapter, we discuss possibilities of resolving this issue using semantic mechanisms (query terms relations, dictionaries, disambiguation, semantic distance) in order to enrich traditional mining processes. The ultimate objectives are to produce a semantically enhanced global reference for the whole log that integrates these semantic mechanisms and to construct a base on which the user interests and the data source content can be identified.

5.2. Keyword Taxonomy Construction: A Global Semantic Representation

The objective of this section is to detail the process of constructing a semantics-based global reference of the query logs. To start, we define this global reference as a data structure organizing the query terms in a taxonomy equipped with a semantic distance function. Thus, we describe an approach to construct this taxonomy over the terms used in keyword search logs by means of the WordNet lexical database of English terms. More specifically, the construction of the taxonomy is based on two main aspects. The first is the hierarchical relation between the terms established by the hypernymy (generalization/specialization relations) "IS-A". This kind of relation enables sorting the terms into different levels of abstraction and organizing them in a hierarchical structure. The second aspect is the semantic distance between two terms connected by a IS-A relation. Here, we introduce a weighting function that takes into account the abstraction level of the terms. In fact, two terms in the bottom of the hierarchy are considered to be semantically closer than two terms situated at the top of the hierarchy. The taxonomy construction step follows a preprocessing step (cf. sec. 4.2 for more details). The two steps are summarized in Fig. 5.1.

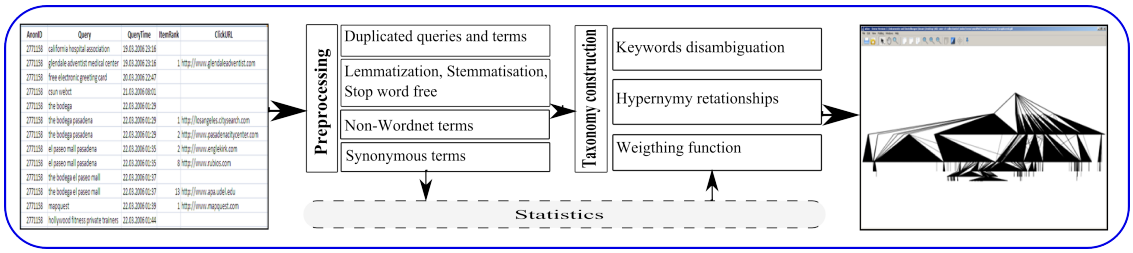


Figure 5.1.: Taxonomy Extraction from Keyword Search Log

5.2.1. Keywords Disambiguation by Using External Source of Semantics:

One of the objectives targeted in this chapter is to enhance search query log analysis with semantics. More specifically, this consists in identifying the meanings of query log terms and the semantic relations between them. These aspects are key elements in the taxonomy construction process.

5.2.1.1. Overview on Existing Disambiguation Methods

In the literature, text disambiguation refers to *Word sense disambiguation (WSD)*, which is a field of research on text mining that deals with the problem of disambiguating polysemous words within a text. Navigli, in [Nav09], classifies the WSD methods based on two dimensions, supervision and knowledge.

The supervision dimension divides the methods into three classes: supervised methods, semi-supervised methods, and unsupervised methods. The supervised methods use machine-learning techniques to learn a classifier from manually annotated text (mainly word senses). This class of methods includes: decision mechanisms, naive Bayes classifiers, neural networks, example-based learning, support vector machines, and combined classifiers. The semi-supervised methods use different techniques to learn sense classifiers from minimally annotated text. This class of methods include: self-learning classifiers by bootstrapping, and self-learning classifiers by bootstrapping and monosemous words. The unsupervised methods are based on unlabeled text, and do not use any manually annotated text. This class of methods includes: word context clustering, synonymous words clustering, and co-occurrence graphs of grammatical relations.

The knowledge dimension characterizes the disambiguation methods that use external source of knowledge/semantics such as dictionaries, thesauri, and ontologies. Knowledge-based methods are divided into three classes: methods based the overlap of sense definitions, methods based on selectional preferences, and structural methods. The overlap of sense definitions methods consist in calculating the overlap of the textual definitions of the word in the dictionary with its word context. The definition with the highest overlap then corresponds to the sense of the target word. The methods based on selectional preferences consists in using selectional preferences (rules) to reduce the number of possible meanings of a word occurring in its context. The selectional preferences are identified from the syntactical relations between the words of the context. The structural methods use semantic structures to calculate the words relatedness; most of the existing approaches use the WordNet structure. The first part of these methods are based on semantic similarity in WordNet. Thus, disambiguating a word consists in finding the sense that maximizes the sum of the binary similarities with the word senses of the same context. The second part of these methods are based on the graph representation. The senses of the words of the same context are represented in the form of a graph which nodes are senses and

which edges are the semantic relations between them (is-kind-of, has-part, etc.). The disambiguation then consists in finding the set of senses that maximizes a score function (which is not a semantic similarity function) that takes as input one sense of a word and the set of senses of its context.

5.2.1.2. A Graph-Based Disambiguation Method for Search Query Keywords

As in the natural language, in search queries, keywords (query terms) are very often polysemous and therefore have several interpretations. Consequently, there is a need to attribute a sense to each keyword. To achieve this objective, we make use of the concept of *synset* defined in WordNet. *A synset is a set of words that are synonymous in a specific context where a specific meaning is attached to each synset* [FM98]. A keyword supporting multiple senses belongs to as many synsets as it has senses. Indeed, finding the meaning of a keyword comes to find the best mapping to the corresponding synset. We call the process of mapping keywords to synsets: **keywords disambiguation**.

In this thesis, we present a disambiguation method that tackles the problem of polysemous keywords in search queries. This method is knowledge-based (structural), as it is based on the dictionary and the structure of WordNet, and supervised, as the WordNet inventory is manually annotated (cf. sec. 5.2.1.1)[Nav09]. Moreover, the method uses the *keyword context,* which is a set of keywords appearing together within the same user query. The goal behind the use of the context is to extract the sense of each keyword by considering the semantics shared by the keyword and the keywords of its context (i.e., the subject of the query). The way of dealing with keywords in this process differs from that used in classical text mining: indeed, syntactical and grammatical relations are not taken into account.

The example in (Fig. 5.2) illustrates the application of the method to real AOL search queries:

The disambiguation method includes two steps:

First, for each keyword in the query, a ranked list of synsets is extracted form the WordNet database. This list of synsets corresponds to all senses that a keyword could have. Formally, given a query $Q = \{w_1..w_n\}$, for each w in Q we attribute a set of synsets S_w, such as:

$\forall w \in Q, \exists S_w = \{s_1..s_k\}$: $w \in \bigcap_{i=1}^{k} s_i$ *(means that the word "w" belongs to "k" synsets)*

Use query: (*taken from the AOL search log*)
"paper line border"
"scientific journal paper"

Synsets of the keyword "paper": (*from WordNet*)

1. paper – (a material made of cellulose pulp derived mainly from wood or rags or certain grasses)
2. composition, paper, report, theme – (an essay (especially one written as an assignment); "he got an A on his composition")
3. newspaper, paper – (a daily or weekly publication on folded sheets; contains news and articles and advertisements; "he read his newspaper at breakfast")
4. paper – (a scholarly article describing the results of observations or stating hypotheses; "he has written many scientific papers")
5. paper – (medium for written communication; "the notion of an office running without paper is absurd")
6. newspaper, paper, newspaper publisher – (a business firm that publishes newspapers; "Murdoch owns many newspapers")
7. newspaper, paper – (a newspaper as a physical object; "when it began to rain he covered his head with a newspaper")

Result of the disambiguation: (*the integers at the front of the keywords corresponds to the synset number*)
"**paper** line border"≡"**paper.1** line.2 border.1"
"scientific journal **paper**"≡"scientific.1 journal.2 **paper.4**"

Figure 5.2.: Keyword Disambiguation

$\forall s_i,\ s_{i+1} \in S_w : s_i > s_{i+1}$ *(">" is read like "precedes")*

In WordNet, senses (i.e., synsets) of the same word are ranked based on the frequency of occurrence of each sense in the British National Corpus [Lee93, sev09, pri12]. Thus, the ranking of the synsets corresponds to an ordered set of synsets from the most common sense to the less common for a given word.

The second step consists in finding the best combination of synsets $\hat{S}_Q = \{\hat{s}_1..\hat{s}_n\}$ that is *semantically equivalent* to the query $Q = \{w_1..w_n\}$ (noted: $Q \equiv \hat{S}_Q$). From the list of synsets $(S_w, >)$ attributed to each keyword "w", the method selects the synset with the closest sense to "w" in the context of Q:

$Q \equiv \hat{S}_Q \Longleftrightarrow \forall w_i \in Q,\ \exists \hat{s}_i \in \hat{S}_Q :\ w_i \in \hat{s}_i$ (*with* $\hat{s}_i$ *the closest synset of* w_i *in the*

context of Q)

To find $\hat{S}_Q$, the method relies on the WordNet structure. Indeed, in addition to the concept of synset, WordNet defines different types of relations between synsets (cf. sec. 5.2.1.4). Thus, the WordNet structure can be assimilated to a graph which nodes are the synsets and which edges are the relations between them. Disambiguation then consists in finding in the WordNet graph the best combination of synsets $\hat{S}_Q$ corresponding to the keywords of Q. By considering the set of synsets S_w of every keyword $w \in Q$, we can generate several graphs, each of which represents a combination of synsets. We consider the problem of finding $\hat{S}_Q$ as a problem of calculating the best minimum spanning tree (MST) among the set of graphs generated by Q. More formally, we calculate the MST of each graph $G = (S_Q, E)$ where $S_Q = \{s_1..s_n\}$ is a combination of synsets corresponding to $Q = \{w_1..w_n\}$ and E a set of weighted edges whose weights represent the length (*i.e.,* number of edges) of the WordNet path linking s_i to s_j.

As each keyword in Q supports multiple synsets, there are different combinations to analyze and hence, several MST to calculate. Finding $\hat{S}_Q$ using an exact approach amounts to explore all the combinations, which is a combinatorial problem of exponential complexity. The complexity of such a method is related to the number of time the MST algorithm is applied. Let n be the number of keywords and $k_1..k_n$ be the number of synsets corresponding respectively to keywords $w_1..w_n$. The complexity of the general MST algorithm is $O(n+m)$, with n the number of keywords and m the number of edges [FW94]. The number of all possible combinations of synsets is equal to $\prod_{i=1}^{n} k_i$. Consequently, the overall complexity of an exhaustive search method is equal to $O(\left(\prod_{i=1}^{n} k_i\right)(n+m))$.

The search space in the proposed method is significantly reduced thanks to a heuristic approach. The main idea of this heuristics is to suppose that a keyword in a search query is used to express its most common sense. This assumption is supported by the fact that the queries in a public search engine are usually submitted by regular users. The Algorithm 5.1 describes how to approximate $\hat{S}_Q$ by exploiting the order of the synsets[1] of a word w defined in $(S_w, >)$. Here, the graph of synsets is incrementally constructed by following the order of the synsets in $(S_w, >)$. Thus,

[1] the synsets of a word are ranked from the most common the least common by using the SemCor corpus

the construction of the final graph goes forward step by step. In the first step, all the nodes are initialized to the first synset (i.e., the most common sense); the first MST is calculated. Thus, each node tries to reach the next step by replacing the current synset by the next one in $(S_w, >)$. At each replacement, the MST is recalculated. If the MST is not improved, the node is marked as final and is excluded from the next step; the previously considered synset is established. If the replacement improves the MST, the new synset is considered in the next step; the previous synset is temporarily established in the current step.

Algorithm 5.1 Keywords Disambiguation Algorithm

Input: $Q = \{w_0, w_1, \ldots, w_n\}$ /* *the keyword query Q*

Output: $\widehat{S}_Q = \{\widehat{s}_0, \widehat{s}_1, \ldots, \widehat{s}_n\}$ /* *the set of synsets corresponding to Q*

```
Create G(S_Q, E)   /* G is the graph of Q, S_Q is a combination of sysnets corresponding
                      to Q, and E the set of edges representing the length of the shortest
                      path between each couple of synsets
for all node N in S_Q do   /* Initialize the nodes
    Initialize N with the first synset of the corresponding keyword in Q
    Mark N as non-final and non-visited
end for
for all edge e in E do   /* Initialize the edges
    Initialize e with the length of the shortest path between the two first synsets
end for
Calculate the MST of G
while G contains non-final nodes do
    for all non-visited node N do
        Mark N as visited
        G_temp = G
        if N is non-final then   /* N represents the keyword w and s_w is the set of synsets of w
            Replace the current synset s by s+, the next synset in (s_w,<)
            Update G
            Calculate the MST of G
            if MST is not improved then
                Mark N as final
            end if
            Replace s+ by s   /* return to the previous synset
            G = G_temp   /* return to the previous graph
        end if
    end for
    for all N non-final do
        Replace s by s+
        Mark N as non-visited
    end for
    Update G
end while
Return the list of final nodes corresponding to Ŝ_Q
```

In the next step, only non-final nodes are considered. The process continues until all the node are marked as final, which means that no improvement is possible. The synsets of the final nodes represent the approximation of the best combination of the query synsets (Algorithm 5.1).

We notice that in the particular case of a query composed of unrelated keywords in WordNet, the query is split into several sub-queries (*i.e.,* sub-graphs) and the

disambiguation method is applied separately to each sub-query. In the case of a single keyword sub-query, the first synset is automatically selected.

The complexity of the disambiguation method depends on the number of keywords in the query and on the number of synsets of each keyword. Let n be the number of keywords and $k_1..k_n$ be the number of synsets corresponding respectively to keywords $w_1..w_n$. The complexity of the MST algorithm is $O(n+m)$ [FW94] (n the number of nodes and m the number of edges). According to Algorithm 5.1, the MST is recalculated as many times as one synset is replaced by the next synset. Thus, the MST algorithm is applied at most $\sum_{i=1}^{n} k_i$ times because the method follows a greedy approach and hence a synset is visited only once. Consequently, the complexity of the disambiguation heuristics is equal to $\left(\sum_{i=1}^{n} k_i\right) \times O(n+m)$. Suppose k_{max} be the maximum number of synsets that a keyword in Q could have. The disambiguation complexity is then equal to: $k \times n \times O(n+m) = O(n^2+nm)$, "$m$" being the number of edges among the graphs of Q. In the worst case of a complete graph (*i.e.,* $m = \frac{n(n-1)}{2}$), the complexity of the disambiguation is equal to: $O(n^3)$ which is polynomial.

The proposed method has three main advantages. First, it uses a reliable source of semantics, which is WordNet with possible mappings to other resources (cf. sec. 5.2.1.3). Second, it enables to find with a satisfactory precision (cf. sec. 5.5.1) the sense of a keyword by considering the context. Third, it is based on a greedy algorithm. However, in spite of these strong points, the method can relatively suffer from some problems. The first problem is related to the size of the query (cf. sec. 5.5.1). In fact, the method works with queries of multiple keywords, and supposes that the query contains only one subject (i.e. the context). However, if a query contains several sub-subjects the method can fail to find the right synset, since it cannot find the context of the keyword. Indeed, the method works well with relatively short queries. The second problem that can affect our method is related to the capabilities of WordNet. Indeed, even though the WordNet database covers most English words, it contains only a few named entities. Fortunately, thanks to the mapping possibility included in WordNet, it is possible to extend its capacity by considering other databases of named entity like DBpedia sec. 4.2.3. Such a mapping is for instance realized in the YAGO project which aims to unify WordNet and Wikipedia [SKW07]. The third problem is related to the heuristics. The principle of the heuristics is to exclude a node from the next step when it does not improve the

MST. This heuristics may fail (i.e., its precision can be affected) if the excluded node could improve the MST when it is considered with nodes of farther steps. However, we consider this case to be rare since it concerns synsets of less common sense.

5.2.1.3. Discussion: WordNet as a Main Source of Semantics

There are several reasons that motivated us to choose WordNet as a main source of semantics:

1. First, WordNet is one of the richest thesaurusi, as it comprises more than 200000 word-synset pairs;
2. Second, it provides many types of relations which enables to connect the synsets between them. This property is particularly used by the Disambiguation method;
3. Third, it contributes to the definition of the structure of the taxonomy thanks to the IS-A relation between synsets;
4. Finally, it is compatible with a large number of dictionaries and other semantic sources e.g. DBpedia, which provides mappings to the synsets and therefore extends its semantic scope (cf. sec. 4.2.3).

5.2.1.4. Keywords and Semantic Relations

The process of keyword disambiguation enables to specify the meaning of a keyword. However, the construction of the taxonomy needs also to determine the relations between keywords. In our context, what we mean by semantic relations between keywords is more precisely defined as relations between synsets. As we deal with textual queries, a linguistic ontology can provide us with information about synsets and the relations between them. To this end, we use the WordNet structure that includes several types of semantic relations (about twenty) e.g. hyponymy vs. hypernymy (IS-A), holonymy vs. melonymy (is part of), etc. In the framework of the disambiguation method, all the relations are considered to construct the graphs of the query. Indeed, using different relations contributes to the enrichment of the query graphs since it enables the emergence of all possible relations between synsets. Fig. 5.3 shows the different relations included in WordNet that are related to the synsets of the word “paper”. This figure was created using visuwords [Log13].

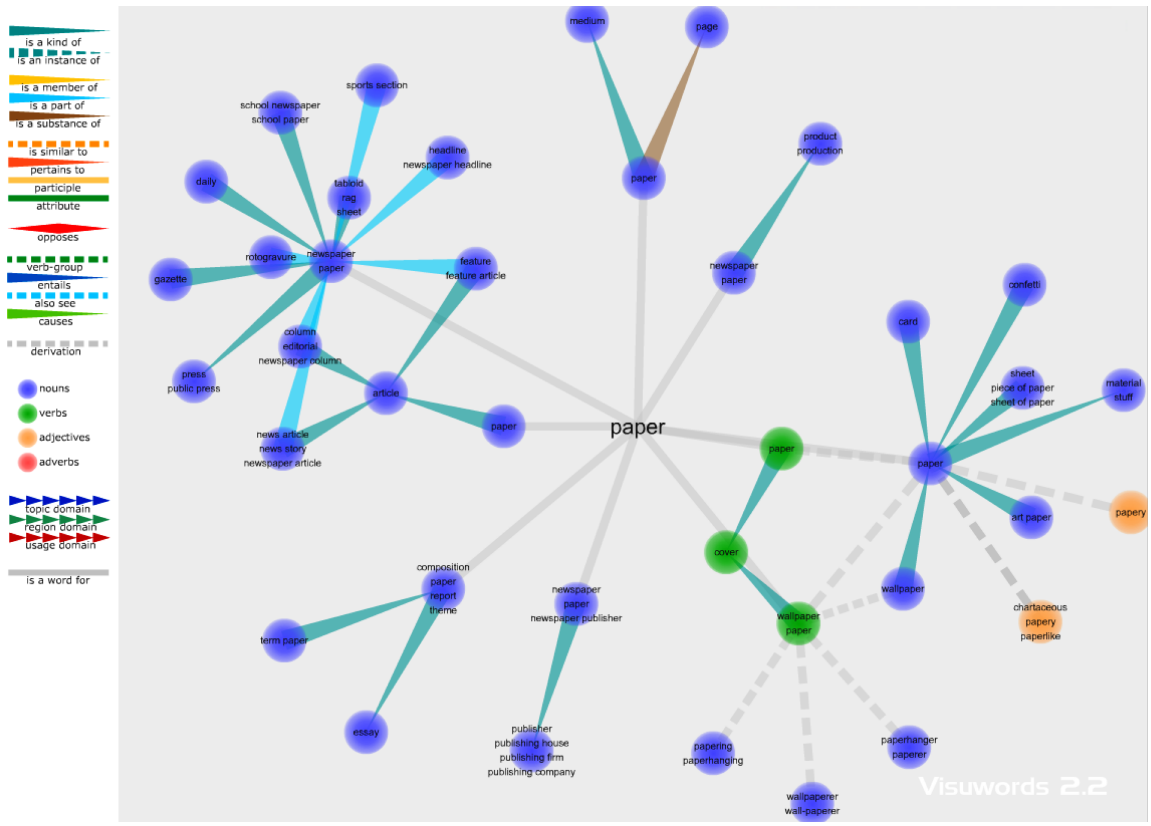


Figure 5.3.: The WordNet Structure

In the remainder of the process of constructing the taxonomy, we use the hypernymy relation (IS-A) as a basic structural characteristic to define a hierarchical order between the keywords (*i.e.*, sysnsets). Semantically, it cooresponds to the relation of being super-ordinate or belonging to a higher rank or class, e.g., "paper" is the hypernym of "wallpaper". Fig. 5.3 shows the synsets of the word "paper" and the relations with their direct subordinates.

5.2.2. Basic Hypernymy Structure

The aim of this step is to construct a hierarchical structure that semantically relates the synsets corresponding to the keywords extracted from search logs. To this end, we use the hypernymy relation "IS-A". We choose the IS-A relation for two main reasons: first, it enables to classify the keywords with a high degree of granularity; second, it provides a means to measure the distance between keywords in the taxonomy (cf. sec. 5.2.3).

Such a relation enables to identify for each synset the hierarchy of hypernyms and to relate them by merging those hierarchies. Merging different hierarchies of hypernyms gives rise to two possible structures. If we consider a simple hypernymy *i.e.,* a synset accepts only one hypernym, the merging process produces a tree structure. However, if we consider that a synset accepts multiple hypernyms, then the merging process produces a semi-lattice. In our proposal, we call both structures *Taxonomy*. The hypernymy structure is generated as follows. In the first step, for each synset we identify the list of its direct hypernyms in WordNet. It is a set of hypernyms to which the sysnet is directly connected by an IS-A relation. The process is then applied recursively to each hypernym until it reaches the last hypernym which hypernym is the root[2]. For example, let us take the term: "football". Suppose that this term matches the synset: "football.1" (the integer "1" reflects the rank of the synset). According to WordNet, sport.1 and game.2 are the hypernyms of football.1, diversion.1 is the hypernym of sport.1, activity.1 is the hypernym of game.2 and diversion.1 (cf. Fig. 5.4).

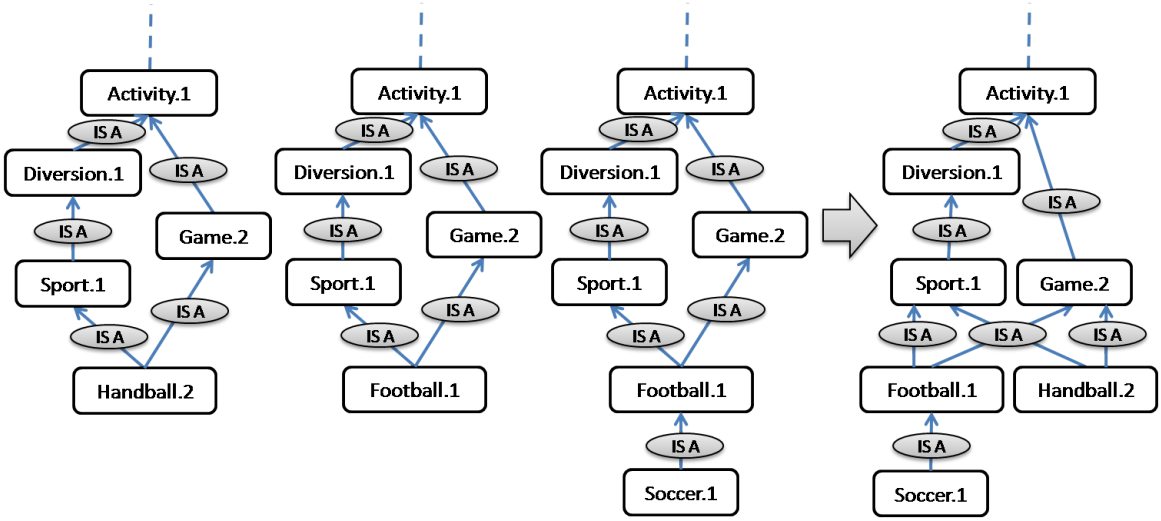


Figure 5.4.: Hypernymy Structure Construction by Hpaths Merging

Hence, we get the sub-paths "football.1" $is-a$ sport.1 $is-a$ "diversion.1 $is-a$ "activity.1" and "football.1" $is-a$ game.2 $is-a$ "activity.1" (cf. Fig. 5.4). We call the hierarchy of hypernyms "*Hpath*".

[2]Note that WordNet does not have a inique global root. For technical purposes, one can assume the existence of a virtual root. Moreover, in recent version of WordNet (from v2.1) there exists a root for nouns, which is "Entity".

Algorithm 5.2 Hypernymy Structure Construction Algorithm

Input: S /* *Set of synsets obtained after disambiguation*
Output: T /* *hypernymy structure*
1: $T = \{\emptyset\}$
2: **for all** $s \in S$ **do**
3: $Hpath = getHpath$(s)
4: $T =$ MERGE(Hpath,T)
5: **end for**
6: Return T

7: **function** $getHpath_{simple}(s)$ /* *This function is used if every hypernym is defined with one direct hypernym*
8: $Hpath = \emptyset$
9: **while** $s \neq R$ **do** /* R *represents the root*
10: $t =$ GETHYPERNYM(s) /* GETHYPERNYM *returns one direct hypernym*
11: $Hpath = Hpath \cup (s, t)$
12: $s = t$
13: **end while**
14: Return $Hpath$
15: **end function**

16: **function** $getHpath_{multiple}(s)$ /* *This function considers the general case of hypernyms with multiple direct hypernyms*
17: $Hpaths = \emptyset$
18: **if** $s \neq R$ **then** /* R *represents the root*
19: $H =$ GETHYPERNYMS(s) /* GETHYPERNYMS *returns the set of all the direct hypernyms*
20: **for all** $t \in H$ **do**
21: $Hpaths = getHpath_{multiple}(t) \cup (s, t)$
22: **end for**
23: **end if**
24: Return $Hpaths$
25: **end function**

26: **function** MERGE($Hpath_1, Hpath_2$)
27: return($Hpath_1 \cup Hpath_2$)
28: **end function**

Depending on the type of hypernymy (simple or multiple), the Hpath is either straight or contains derivations (sub-paths). In the second step, the produced Hpaths are merged based on their common edges to form a tree or semi-lattice structure. Fig. 5.4 shows the Hpaths of Handball.2, Football.1 and Soccer.1 connected in one semi-lattice. These synsets corresponds to keywords extracted from queries in Fig. 5.5.

The process of hypernymy structure construction is summarized in Algorithm 5.2.

AnonID	Query	QueryTime	ItemRank	ClickURL
31910	southern california **football** schedule	02.03.2006 17:39	1	http://usctrojans.collegesports.com
1079144	**handball** videos	18.03.2006 18:35	2	http://www.hand-ball.org
1042633	**soccer** coaching education course	19.04.2006 12:07	10	http://www.brucebrownlee.com

Figure 5.5.: Examples of Queries Containing Keywords: Handball, Football and Soccer

The complexity of the hypernymy structure construction is related to the complexity of the functions $getHpath_{multiple}$ (if we consider the general case) and MERGE.

The complexity of $getHpath_{multiple}$ is related to the number of hypernyms (including hypernyms of hypernyms) and to the number of edges linking these hypernyms in the Hpath. In Algorithm 5.2, the number of edges equals the number of times the $getHpath_{multiple}$ is called recursively. We consider the GETHYPERNYMS function complexity to be constant, since it is a selection query submitted to the WordNet data base.

Let k be the number of hypernyms (including the root R) of a query synset s (i.e., a keyword of a query after disambiguation), and A the number of edges. Thus, we can write:

$$A = \sum_{i=1}^{k-1} H_i \tag{5.1}$$

with H_i the number of hypernyms of hypernym "i" (H_1 represents the set of direct hypernyms of s and $H_k = 0$ represents the set of hypernyms of the root R).

Moreover, because a synset (or hypernym) can have at most k hypernyms (if it is linked to all of the hypernyms of the Hpath):

$$\forall i \in \{1..k\},\ 0 \leq H_i \leq k \tag{5.2}$$

From 5.1 and 5.2 we obtain:

$$0 \leq A \leq k(k-1)$$

Indeed, the complexity of the $getHpath_{multiple}$ algorithm in the worst case is then:

$O(k^2)$.

The MERGE function consists in unifying the Hpaths by merging their corresponding sets of hypernyms and edges while deleting duplicates. Since the Hpaths take the form of semi-lattices (i.e., tries), the merge function can be represented as a "union-find" problem [KT06, GF64]. Thus, the complexity of such a function is $O\left(log\left(k_1 + k_2\right)\right)$ with k_1 and k_2 being the number of hypernyms of $Hpath_1$ and $Hpath_2$. Assuming k_1 is comparable to k_2, the complexity of MERGE function is $O\left(log\left(k\right)\right)$. Consequently, the $getHpath_{multiple}$ complexity is higher than the MERGE complexity: $O\left(log\left(k\right)\right) \subseteq O(k^2)$.

The complexity of the whole process of hypernymy structure construction is therefore $O(X\,k^2)$ with k the number of hypernyms in a Hpath and X the number of query synsets in the taxonomy.

The hypernymy structure produced in this step defines the structure of the keyword taxonomy. The keywords are organized into different abstraction levels where each keyword is attached to one synset represented by an integer (the sense number). In the next step, the taxonomy is provided with a means to measure the semantic distance between keywords.

5.2.3. Semantic Distance Function

In order to provide the taxonomy with a means of quantitative evaluation of the semantic distance between two terms (keywords), we first introduce a weight associated to the individual "$is-a$" links. This weight is defined as a decreasing function of the semantic proximity between the parent and the child (*i.e.,* the higher the weight, the less related the terms). As all relations contained in the taxonomy are of type "$is-a$", the nodes go from the most general at the top to the most specific at the bottom. Therefore, two connected terms at the bottom of the taxonomy are more closely related than two connected terms at the top. The weighting function should thus be decreasing with respect to the level of the terms. The choice of a decreasing function is motivated by the fact that the relations between concrete and less abstract terms are more relevant and hence stronger than relations between general terms. In addition to that, a decreasing weighting function enables to characterize the dimension of the domain covered by the query terms. This means that depending on the terms used, the query varies from general (*i.e.,* large domain di-

mension) to specific (small domain dimension). We argue that the more general the terms, the larger is the distance between them and hence the larger is the domain covered by the query. For example, let us consider two queries Q1 and Q2 where Q1 is "Sport diversion", which is a general query about sport, and Q2 is "football player X", a more specific query about a football player. In this example, the domain covered by "Sport" and "diversion" is larger that the one covered by "football player" and a given person "X". It is why the weight of the relation "Sport $is-a$ diversion" is higher that the one of the relation: "X $is-a$ football player". Based on this, we define the weight function W as:

Definition 1. *Let x, y two terms (i.e., synsets) related by the direct relation y "is-a" x. The weight function W is defined on x and y as:*

$W(x,y) = \frac{1}{l(y)}$, *with "l" is the function that returns the level of "y".*

Fig. 5.6 illustrates the use of this function.

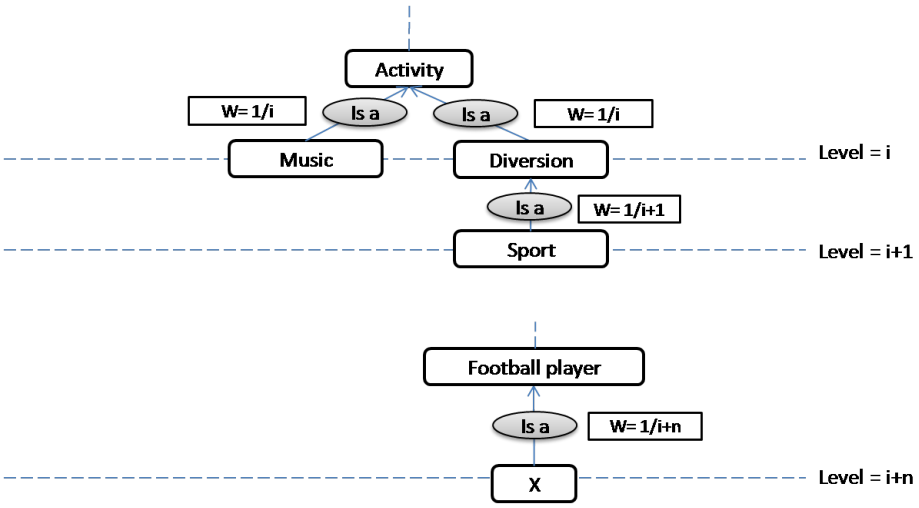


Figure 5.6.: Decreasing Weight Function

In order to be able to compare every couple of terms of the taxonomy, we introduce a function that depends on the weights of the edges composing the shortest path between the terms. Depending on the type of path, there are two ways to compute the distance.

- If the path is straight (cf. Fig. 5.7), the distance is the sum of the weights of its edges; in case of multiple paths, the shortest one is considered.

- If the path is deviated (cf. Fig. 5.7), the distance is the sum of distances of the two sub-paths (straight paths) that have a common hypernym as a upper bound and form together the shortest path.

Hence, the function is defined as:

Definition 2. *Let x, y two terms (i.e., synsets) of the taxonomy. The dissimilarity (or distance[3]) function D is defined on x and y as:*

$$D(x,y) = \begin{cases} 0, \text{ if } (x = y). \\ \sum_{i=l(y)+1}^{l(x)} (\frac{1}{i}), \text{ if } x \rightarrow y \text{ (the shortest straight path) exists.} \\ \sum_{i=l(x)+1}^{l(y)} (\frac{1}{i}), \text{ if } y \rightarrow x \text{ (the shortest straight path) exists.} \\ \sum_{i=l(c)+1}^{l(x)} (\frac{1}{i}) + \sum_{i=l(c)+1}^{l(y)} (\frac{1}{i}), \text{ with } x \rightarrow c \leftarrow y \text{ the shortest path deviated by c exists.} \end{cases}$$

with "l" being the level function (the level function returns the depth of a term, its value increases when going to the bottom of the taxonomy) and "c" the common hypernym of terms "x" and "y".

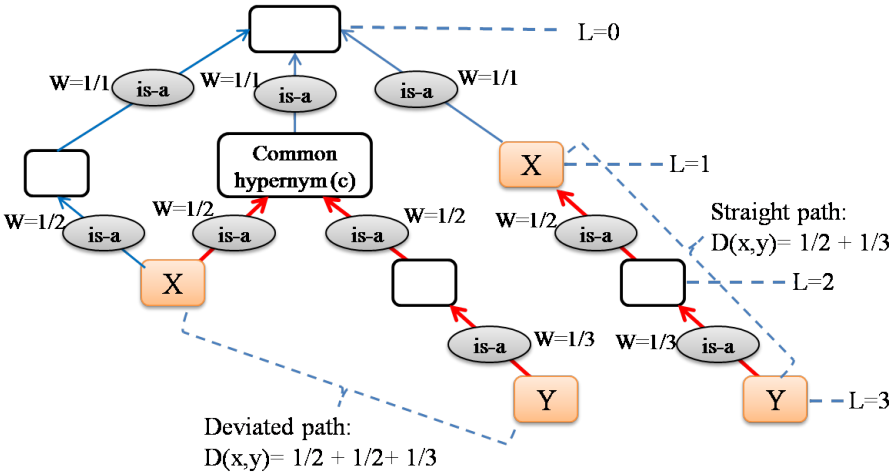


Figure 5.7.: Distance Measurement

With this definition, we aim to focus on the abstraction level of terms to evaluate their distance. The notion of semantic distance between words has been extensively

[3]In the rest of this chapter, we call the function D "distance" as it is more illustrative and therefore commonly used in the domain. In mathematical terms, a distance should satisfy the four conditions presented in sec. 3.2. However, the function D does not satisfy the triangle inequality as we will see in this section. Indeed, the function D is in this case a semimetric.

studied in the litterature, especially in the information retrieval domain (cf. sec. 3.2.2 for more details). However, the aim of the proposed semantic distance "D" is not to measure the absolute semantic relatedness of terms but to provide a means to measure the semantic proximity in the specific context of our taxonomy. Therefore, we aimed to model the semantic variation (the weight function) of the "$is-a$" relation depending on the abstraction level of the related terms in the taxonomy (cf. Fig. 5.6). This property had never been explicitly studied in earlier works. However, we could verify its effectiveness with respect to the other functions (cf. sec. 5.5.2).

The function presented in definition 2 satisfies the following conditions:

1. Positivity. $\forall x, y \in V$, $D(x,y) \geq 0$ (*with* V *the set of terms*)

   - In case of a straight path $(y \rightarrow x)$ or $(x \rightarrow y)$, the distance is the sum of positive weights of the edges composing the path.
   - In case of a deviated path $(x \rightarrow c \leftarrow y)$: the distance is the sum of the two straight paths $x \rightarrow c$ and $y \rightarrow c$

Consequently, D satisfies the positivity condition.

2. Symmetry.$\forall x, y \in V$, $D(x,y) = D(y,x)$

   - In case of a straight path $(x \rightarrow y)$: $D(x,y)$ is the sum of positive weights of edges composing the path. Consequently:
     $D(x,y) = \frac{1}{l(y)+1} + ... + \frac{1}{l(x)} = \frac{1}{l(x)} + ... + \frac{1}{l(y)+1} = D(y,x)$
   - In case of a straight path $(y \rightarrow x)$:
     $D(x,y) = \frac{1}{l(x)+1} + ... + \frac{1}{l(y)} = \frac{1}{l(y)} + ... + \frac{1}{l(x)+1} = D(y,x)$
   - In case of a deviated path $(x \rightarrow c \leftarrow y)$: $D(x,y)$ is the sum of two straight paths (cf. the previous case).

Consequently, D is symmetric.

3. Reflexivity. $\forall x, y \in V$ $D(x,y) = 0$ Iff $x = y$.
There are two cases :

   - if $x = y \Rightarrow D(x,y) = 0$ (directly from the definition 2)
   - if $D(x,y) \neq 0 \Rightarrow x \neq y$
     In fact, if $D(x,y) \neq 0$ means that at least:
     - $D(x,y) = \frac{1}{l(y)}$ (i.e., y "$is-a$" x) $\Rightarrow x \neq y$

- $D(x,y) = \frac{1}{l(x)}$ (i.e., x "$is-a$" y) $\Rightarrow x \neq y$
- Otherwise the path between x and y is the sum of positive weights (positivity)$\Rightarrow x \neq y$

Consequently, D is reflexive.

The distance D does not satisfy the triangle inequality (Fig. 5.8 shows a counter example).

By definition, the triangle inequality is formalized as: $\forall x, y, z \in V$, $D(x,z) \leq D(x,y)+D(y,z)$. Thus, let us suppose the following concepts represented in Fig. 5.8:

- X, Y, Z: three different players.
- C_1: a player in a football club
- C_2: a player in a national football team
- C: the concept football player

In the example (cf. Fig. 5.8), the shortest path between player X and Z passes through the common concept "football player", while the shortest path between players X and Y passes through the concept "player in a football club" and the shortest path between players Y and Z passes through the concept "player in a national football team". According to definition 2, $D(X,Z) = 3$, $D(X,Y) = 1$, and $D(Y,Z) = 1$. Consequently, $D(X,Z) > D(X,Y) + D(Y,Z)$, which means that D do not satisfy the triangle inequality.

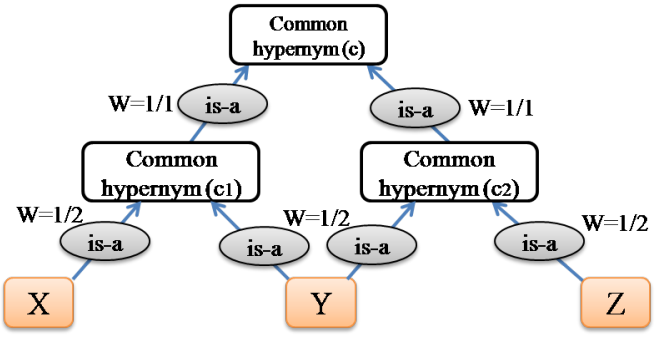


Figure 5.8.: Counter example for triangle inequality

5.3. The Computational Complexity of the Taxonomy Construction

The complexity of the taxonomy construction depends on the complexity of its different steps including the preprocessing and the effective process of construction (Algorithm 5.3). The algorithm includes five main functions:

5.3.1. Tokenization:

The complexity of the function (LUCENETOKENIZER(q)) is related to the cost of parsing the query, which is itself related to the size of the input text; it is equal to $O(l)$ with l the number of words (including stop words) within the query q (cf. sec. 4.2.2.1). Indeed, this function transforms q to R_q being the number of the keywords in q without stop words.

5.3.2. Stemming

The complexity of stemming a word is related to the cost of parsing the suffix to identify the rule, to the cost of parsing the stem to check the condition, and to the number of rules (cf. sec. 4.2.2.2). The complexity of stemming one word is $O(kp)$ with p the length of the word and k the number of rules. Since k is fixed $O(kp) = O(p)$. The complexity of the stemming function (LUCENESTEMMER) takes also into account the number of input keywords. Indeed, the complexity of LUCENESTEMMER is $O(np)$ with n ($n =\mid R_q \mid$) the number of keywords to stem in q.

5.3.3. Lemmatization

Lemmatization (LUCENELEMMATIZER) consists of two phases (cf. sec. 4.2.2.3). The complexity of the first phase applied on a set of words of size n ($n =\mid R_q \mid$) is $O(nTAB)$, where: T is the number of available tags for a word (verb, preposition, noun, etc.), A is the number of available lexical features for a tag given its context (the query), and B represents the number of the B-best annotations of R_q used to estimate the best annotation. For more details, see [Rat96]. The complexity of the

second phase is related to the number of rules and to the length of the word, as for stemming, which is $O(np)$ with n the number of words to analyze and p the length of the word. Indeed, as the complexity of the function LuceneLemmatizer depends on the parameters of the first and the second phases, it is $O(n(TAB + p))$.

Algorithm 5.3 Keyword-based Taxonomy Construction

Input: $Q = \{q_0, q_1, \ldots, q_z\}$ /* *set of queries extracted from a search log*
Output: T /* *taxonomy of query terms*
1: $R_q = \emptyset$ /* *sequence of keywords*
2: $R_{stem} = \emptyset$ /* *sequence of stems*
3: $R_{lemma} = \emptyset$ /* *sequence of lemmas*
4: $S = \emptyset$ /* *set of synsets*
5: **for all** $q \in Q$ **do**
6: $R_q =$ LUCENETOKENIZER(q)
7: $stemsuccess = false$
8: **for all** $w \in R_q$ **do**
9: $stem =$ LUCENESTEMMER(w) /* *retrieve the stem of w*
/* *check stem in WordNet*
10: **if not** CHECK_WORDNET $(stem)$ **then**
11: $stemsuccess = false$
12: **break**
13: **else**
14: $stemsuccess = true$
15: $R_{stem} = R_{stem} \cup stem$
16: **end if**
17: **end for**
18: **if** $stemsuccess = true$ **then** /* *stemsuccess is true iif all stems are in wordnet*
19: $R_q = R_{stem}$
20: **else** /* *retrieve the lemmas of* R_q
21: $R_{lemma} =$ LUCENELEMMATIZER(R_q)
22: **for all** $lemma \in R_{lemma}$ **do**
/* *check lemma in WordNet*
23: **if not** CHECK_WORDNET $(lemma)$ **then**
24: $R_{lemma} = R_{lemma} \backslash lemma$
25: **end if**
26: **end for**
27: $R_q = R_{lemma}$
28: **end if**
29: **if** $R_q \neq \emptyset$ **then**
30: $S = S \cup$ GETSYN(R_q)
31: **end if**
32: **end for**
33: **if** $S \neq \emptyset$ **then**
34: GETSTRUCTURE(S)
35: **end if**

36: **function** LUCENETOKENIZER(q)
37: Implements the Lucene tokenizer
38: Return R_q
39: **end function**

40: **function** LUCENESTEMMER(e)
41: Implements the Porter algorithm for stemming
42: Return *stem*
43: **end function**

44: **function** LUCENELEMMATIZER(e)
45: Implements the Lucene Stanford Lemmatizer
46: Return R_{lemma}
47: **end function**

48: **function** GETSYN(R_q) /* *affect a synset number to each query term*
49: Implements the keywords disambiguation alg.
/* *Algorithm 5.1*
50: **end function**

51: **function** GETSTRUCTURE(S) /* *return a hierarchical structure of S*
52: Implements the hypernymy construction alg.
/* *Algorithm 5.2*
53: **end function**

Note: In the algorithm (Algorithm 5.3) both stemming and lemmatization are used though their objectives are similar (which is word normalization). In fact, the objective behind combining these two techniques is, on the one hand, to exploit the simplicity of stemming and on the other hand to benefit from the efficiency of lemmatization. Indeed, the algorithm is designed in a way that avoids the senseless words produced by stemming and the slowness caused by lemmatization.

5.3.4. Disambiguation

The complexity of the disambiguation function (GETSYN(R_q)) is $O(n^3)$ (the case of a complete graph) with $n =\mid R_q \mid$ the number of keywords in the query (cf. sec. 5.2.1.2).

5.3.5. Hypernymy Structure

The complexity of the hypernymy construction function (getStructure(S)) is $O(X\,k^2)$ with X the whole number of query synsets (i.e., the number of all the keywords in taxonomy) and k the max number of hypernyms of a synset (cf. sec. 5.2.2).

5.3.6. Overall Complexity

The algorithm deals with a set of queries as input and produces a taxonomy of keywords as output.

According to Algorithm 5.3, the complexity of the construction process is the sum of the complexities of the above functions applied to the whole set of queries. Let "z" be the number of queries, n the maximum number of keywords within one query (without stop words), $X = z * n$ the maximum number of keywords[4], t the number of stop words, $l = n+t$ the number of keywords including stop words, p the length of keyword, and k the max number of hypernyms (including hypernyms of hypernyms) of a synset:

- The complexity of tokenization is $O(z\,l) = O(z(n+t)) = O(z\,n) = O(X)$ (since n and t are in the same order)
- The complexity of stemming is $O(z\,n\,p) = O(X\,p)$
- The complexity of lemmatization is $O(z\,n(T\,A\,B + p)) = O(X(T\,A\,B + p))$
- The complexity of disambiguation is $O(z\,n^3) = O(X\,n^2)$
- The complexity of hypernymy structure construction is $O(X\,k^2)$

In addition to the above functions, the function "check_wordnet" is used to check the availability of a word in WordNet. The complexity of such a function is considered to be the complexity of a binary search on a sorted list (*i.e.,* WordNet words). Thus the complexity of check_wordnet is $O(log(G))$ with "G" the number of WordNet words [LBB81]. In the main algorithm:

- The complexity of checking all the words in WordNet is $O(X\,log(G))$ (Notice that the function is called in lines *10* and *23* of Algorithm 5.3)

Consequently, the complexity of the taxonomy construction algorithm is:

[4] X is the number of all the keywords of the taxonomy

$O(X(1+p+TAB+n^2+k^2+log(G)))$, with X the number of keywords, p the average size of a word, T the number of available tags for a word (verb, preposition, noun, etc.), A the average number of available lexical features for a tag, B the number of B-best annotations for a sequence of keywords (*i.e.,* query) used to estimate the best annotation, n the number of keywords in a query, k the average number of hypernyms (*i.e.,* the size of Hpath) of a synset and G the number of words in WordNet.

Indeed, except the number of input keywords X which is not bounded, the other parameters are bounded and can be estimated in advance. Therefore, the complexity of the taxonomy construction can be written: $O(\lambda X)$ with $\lambda = 1 + p + TAB + n^2 + k^2 + log(G)$. As λ is bounded, the algorithm complexity is $O(X)$.

5.4. Using the Taxonomy for Analysis

The keywords taxonomy can be used to analyze the query log. The analysis can be either general and based on the structure of the taxonomy or empirical and based on the content of the taxonomy. In the following we detail the two ways:

5.4.1. Structure Analysis

Before actually using the keywords taxonomy for its semantric value, its structure can be studied in order to estimate its utility. The parameters that characterize this structure are: its size, its width, and its average depth.

5.4.1.1. Size

This parameter represents the number of nodes independently from their position in the taxonomy. The size of a taxonomy T is:

$$S_T = \mid \{x \in V : \ x \ has \ a \ node \ in \ T\} \mid,$$

with V the set of keywords.

The size of the taxonomy can be used as a precondition for analysis since the analysis cannot provide meaningful results if the query log that is not large enough. We can therefore threshold the number of keywords (i.e., the size of the taxonomy) to decide when the analysis can be started.

5.4.1.2. Average Depth

This parameter enables to measure the granularity of the taxonomy. In other words, it measures the level of detail between the root and the keywords of the taxonomy. Technically, it is the average length of a path from the root to a keyword of T:

$$L_T = \frac{\sum_{j=1}^{|S_T|} length(P_j)}{\mid S_T \mid},$$

with $length(P_j)$ the length (the number of edges) of the path P_j from a keyword (represented as a synset) w_j to the root, and m the number of keywords.

The average depth measures the level of detail of the users topics of interests. A deep taxonomy is focused on a number of topics of interests.

5.4.1.3. Width

This parameter enables to show how thick or thin is the taxonomy. Technically, it is the average number of nodes in one level. Formally, the width of a taxonomy T is:

$$W_T = \frac{S_T}{L_T},$$

with S_T the size of T, and L_T the average depth of T.

The width of the taxonomy is inversely related to its depth. It enables to study the diversity of the users topics of interest. A wide taxonomy means that it involves a large number of different users topics of interest.

5.4.2. User Interests Analysis

The main objective of the keywords taxonomy is to provide traditional query logs with a semantically rich representation. This representation is the basis for further analysis in the next steps. In this thesis, we use the keywords taxonomy to build the user profile by extracting its interests. The method for extracting the interests is based on a clustering algorithm applied on the taxonomy. Therefore, the hierarchical representation in addition to the semantic distance are the main aspects used by the algorithm. Details for user interests extracting and user modeling are given in chapter 6.

5.5. Experimental Results

In order to verify the consistency of the taxonomy we have conducted three series of experiments, each one including validation tests. In the first experiment, we check the consistency of the disambiguation method. In the second experiment, we analyze the effect of the taxonomy level on our semantic distance function in order to compare it with other functions of state of the art. In the third experiment, we analyze the taxonomy that we obtained from real-world search logs. These logs originate from the AOL search engine [PCT06]; they consist of almost 20M Web queries collected from almost 650k users over three months in 2006.

5.5.1. Consistency of the Disambiguation Method

To test the validity of our method, we measure its precision and its recall. To this end, we randomly select from the AOL search log 100 queries of different sizes in terms of number of keywords. Using the WordNet dictionary, we apply our disambiguation method to determine the most appropriate synset to each keyword in the context of the query. The obtained results are then compared to a manual disambiguation of the keywords.

The precision of the method is the number of correctly attributed synsets divided by the number of attributed synsets (i.e., the number of keywords that match a synset regardless of whether the matching is correct or not):

$$P = \frac{\#\ correctly\ attributed\ synsets}{\#\ attibuted\ synsets}$$

The recall is the number of correctly attributed synsets divided by the number of keywords:

$$R = \frac{\#\ correctly\ attributed\ synsets}{\#\ keywords}$$

The recall in our method is sensitive to the WordNet coverage (i.e., the capacity to attribute a synset to each keyword). In fact, if the queries contain mostly keywords defined in WordNet, the recall will be high. However, if the queries are mostly composed of non-WordNet words the recall will be affected. This is in particular the case with named entities.

The results of our experimentation are summarized in Tab. 5.1:

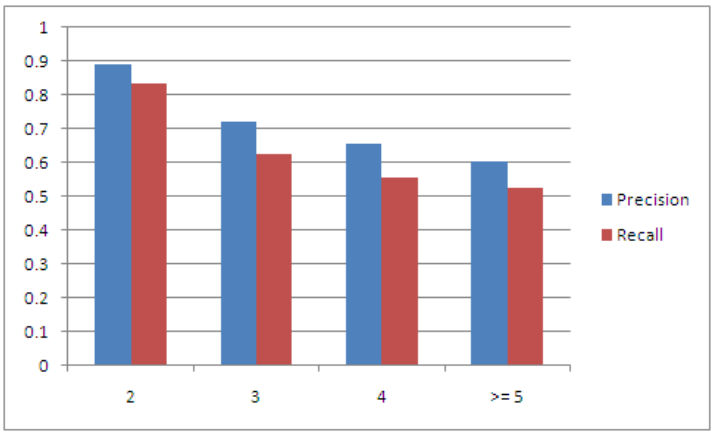


Figure 5.9.: The Disambiguation Precision and Recall

The graph in Fig. 5.9 shows the precision and the recall of the method with respect to different sizes of the queries in terms of number of keywords[5].

The highest measure of the precision (0.89) is obtained for queries with 2 keywords. It decreases when the size increases, and stabilizes (0.60) when the number of keywords is more than five. Otherwise, the overall average value of the precision is 0.70

[5]In the experimentation, we consider only queries whose the size is more that 2 keywords. Queries with one keyword are directly matched to the first synset of the keyword.

(cf. Tab. 5.2). The precision of the method can be affected by three factors. The first factor is related to the size of the query. The more keywords the query contains, the lesser the precision of the method. This is due to the fact that the method is based on the minimum spanning tree to approximate the most appropriate synsets. Thus, it deals with a query as composed of one subject, whereas a query of multiple keywords seems to contain several sub-subjects. Since the MST tries to find the synsets that are close each to other, the method can fail to find the right synset since the synset can belong to several subjects. The fact that the method is well adapted to relatively short queries makes it pertinent for web search query term disambiguation since the average size of a search queries is known to be small (2,4 according to[SWJS01]). The second factor is related to the heuristics used by the method. Since this heuristics supposes that a synset of a keyword can be found in the first top synsets, the method can fail in some specific cases in which the synsets are in the end of the list (cf. the last paragraph in sec. 5.2.1.2). The third factor is related to the specific case of synsets which are not defined in WordNet. In fact, the method in this case attributes a default synset.

The highest measure of the recall (0.83) is obtained at 2 keywords. It decreases and stabilizes (0.52) after 5 keywords. The overal average value of the recall is 0.63 (cf. Tab. 5.2). In fact, the recall depends on the WordNet coverage. The more keywords exists in WordNet, the higher the recall of the method. Thus, the recall can be affected in two cases. First, if the keyword is not defined in WordNet at all, and second, if the synset corresponding to the keyword is not in the list of synsets of the keyword.

We notice that in the SemEval competition [Nav09], which is dedicated to the evaluation of systems based on semantic analysis, some methods for word sense disambiguation reach a high level of accuracy (e.g., NUS-ML with 88.7% and UBC-ALM with 86.9%). However, these methods were only tested on regular text, and none of them was tested for keywords disambiguation.

5.5.2. Characterization of the Semantic Distance

The main objective of our semantic distance is to measure the relationship between keywords in the taxonomy. In addition to that, this distance distinguishes between keywords (i.e., synsets) related by the “is-a” relation (e.g., a couple of synset-hypernym) situated at the bottom of the taxonomy (i.e., specific keywords)

The Number of keywords (n)	Precision	Recall
$n = 2$	0.89	0.83
$n = 3$	0.72	0.62
$n = 4$	0.65	0.55
$n \geq 5$	0.60	0.52
Average measure	0.70	0.63

Table 5.1.: Validation tests of the disambiguation method

and those situated at the top of the taxonomy (i.e., general keywords). Indeed, specific keywords are more closely than general keywords. In this section, we attempt to check this property with respect to other semantic similarity metrics of the state of the art. The metrics are selected from different classes (edge counting, information content, hybrid). To do this, we first randomly select keywords from each level of the taxonomy (i.e., levels of abstraction), and thenmeasure for each keyword its similarity/distance to its parent (i.e., hypernym) Tab. 5.2.

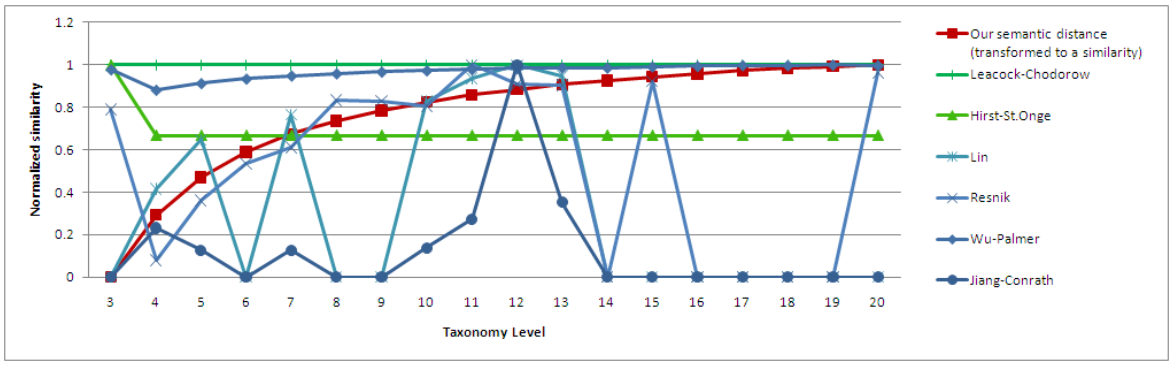


Figure 5.10.: Semantic distance with respect to the taxonomy level

The graph in Fig. 5.10 shows the variation of the similarity between consecutive synsets with respect to the level of the taxonomy. In order to enable comparison, the values of the similarity shown on the figure are normalized with respect to the maximum value of each metric. The real values are presented in Tab. 5.2. In this graph, we can distinguish three types of curves: those that are constant (Leacock-Chodorow and Hirst-St.onge), those that are variating randomly (Lin, Resnik, Wu-Palmer and Jiang-Conrath), and one that is variating monotonously (our semantic distance[6]). The first type of curves (Leacock-Chodorow and Hirst-St.onge) shows that the metrics are independent from the level of the taxonomy, and do not make

[6]The semantic distance is normalized and transformed to a similarity by $s = 1 - d$

difference between specific keywords and between general keywords. The second type of curves (Lin, Resnik, Wu-Palmer and Jiang-Conrath) shows that the metrics are affected by the level of synsets in the taxonomy but in a random way. The curve of our distance (transformed to a similarity) shows that the distance variates in a monotonous way with respect to the level of the taxonomy. In fact, this property is very important since it enables to cluster similar synsets and to separate specific keywords from general keywords. This property does not exist neither in the first type nor in the second type of metrics.

	Wu-Palmer	Leac.-Chod.	Hir.-St.O.	Resnik	Lin	Jian.-Conr.	Ours
level 3	0.952	2.995	6	7.758	0	0	0.333
level 4	0.857	2.995	4	0.779	0.404	0.436	0.250
level 5	0.888	2.995	4	3.558	0.629	0.239	0.200
level 6	0.909	2.995	4	5.254	0	0	0.166
level 7	0.923	2.995	4	6.000	0.742	0.240	0.142
level 8	0.933	2.995	4	8.182	0	0	0.125
level 9	0.941	2.995	4	8.154	0	0	0.111
level 10	0.947	2.995	4	7.915	0.804	0.259	0.100
level 11	0.952	2.995	4	9.819	0.909	0.513	0.090
level 12	0.956	2.995	4	8.932	0.971	1.884	0.083
level 13	0.960	2.995	4	8.875	0.921	0.664	0.076
level 14	0.960	2.995	4	0	0	0	0.071
level 15	0.965	2.995	4	9.057	0	0	0.066
level 16	0.967	2.995	4	0	0	0	0.062
level 17	0.971	2.995	4	0	0	0	0.058
level 18	0.971	2.995	4	0	0	0	0.055
level 19	0.973	2.995	4	0	0	0	0.052
level 20	0.971	2.995	4	9.463	0	0	0.050

Table 5.2.: Semantic distance with respect to the taxonomy level

5.5.3. Results From the AOL Keywords Taxonomy

Fig. 5.11 shows the shape of the taxonomy obtained from the AOL search logs. The taxonomy is composed of 36177 synsets, which are distributed over 20 levels. The most populated levels are situated between the levels 6 and 14, which means that the taxonomy is mostly composed of synsets of moderate abstraction level. Indeed, the average depth of the taxonomy is equal to 10. In addition to that, the width

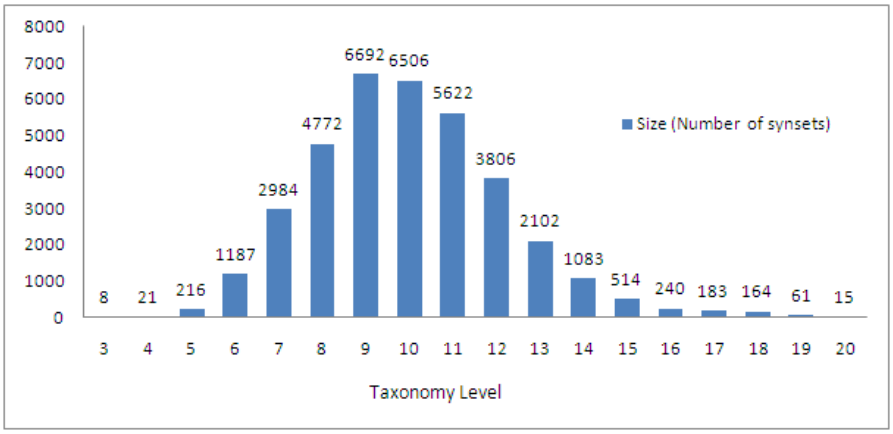


Figure 5.11.: Synsets Distribution over the taxonomy

of the taxonomy, which represents the distribution of the keywords over its levels is equal to 3617. In fact, the shape of the taxonomy and its parameters can be used for further analysis (e.g., keywords clustering).

Size	Average Depth	Width
36177	10	3617

Table 5.3.: Results from the AOL keywords taxonomy

5.6. Summary

The aim of this chapter is to enhance query log analysis with semantics. To achieve this objective we described a process to extract a global semantics reference in form of a taxonomy. The construction process comprises, in addition to the preprocessing steps, the proposition of a method for keyword disambiguation, the definition of semantic distance function and the effective process of the taxonomy construction. The proposed disambiguation method is a graph-based approach that uses a heuristics combined with the MST algorithm. We conducted experiments on real search query logs, which show that the method reaches very high performance, especially with very short queries. This performance makes it well adapted for web search queries. In addition, we proposed a semantic distance function which objective is to calculate the distance between each two keywords in the context of the taxonomy. In the experimentation, we showed that this function has a specific

property of distinguishing specific keywords and general keywords, since it consider that specific keywords are closer compared to general keywords. This property does not exist in the metrics of the state of the art, which makes this function the most appropriate function in our case, as this property is a requirement of the further analysis process described in the next chapter. The taxonomy construction process uses the relation of hypernymy between the synsets identified after matching the keywords to WordNet synsets and disambiguating them. The taxonomy structure is obtained by merging the Hyperpaths of each synset. The taxonomy construction process is efficient, as it is of linear complexity. In the next chapter, we will aim to demonstrate the usefulness of the extracted taxonomy for extracting user profiles.

6. Usage-Based Profile Modeling

6.1. Introduction

The object of this chapter is to describe a method to construct a profile model from search query logs. The method distinguishes between two different types of profiles: user profile and data source profile. The former represents the user interests, which depict what he/she is looking for. The latter represents the data source topics of interest, which depicts the documents provided by the data source. As a first step, the method extracts a model representing the interests of all the users of the log. This model is then instantiated for each user and data source in form of individual profiles. This is done based on a clustering algorithm applied on the keyword taxonomy (see chapter 5); thus, the interests are extracted in the form of clusters of keywords.

6.2. Extracting User Interests

The goal behind the search log analysis is the extraction of the user interests. Technically, the problem is defined as a query terms clustering problem. This problem has been discussed earlier in different ways [WNZ02, CC02] but without considering the semantic features involved in query terms. On the contrary, we propose to take advantage of the keyword taxonomy (see chapter 5) that extracts the semantic features of the query terms by replacing terms by synsets related by the hierarchical relation $is - a$.

6.2.1. Query Terms Clustering

6.2.1.1. Pruning Algorithm

The main objective of the algorithm is to split the taxonomy structure into a set of smaller structures corresponding to clusters (see Fig. 6.1). The proposed algorithm is designed to take advantage of the decreasing property of our distance function D (see sec. 5.2.3) i.e., the fact that the distance between two adjacent elements in the bottom of a taxonomy T is smaller than between two adjacent elements in the top. This property directly affects the clustering algorithm since it favors clustering elements at the bottom rather than elements at the top of T. This implies that the size of the clusters increases in the bottom of T. The advantage of such a property is to form clusters with the maximum of specific terms [LCKB10].

To identify the clusters, the algorithm uses a threshold value δ on the semantic distance, such that $D(s_d, s_1) \leq \delta \leq D(s_d, s_n)$ with $P_T = s_d, s_1 ... s_n$ the longest path in the taxonomy T. In addition to that, the algorithm defines a quality function F_δ that attributes a global quality value to the clustering. It is the sum of the local quality values $v(C)$ of each cluster C , which depends on the size, the frequency and the abstraction level of the cluster elements.

We define a cluster in a keyword taxonomy as follows:

Definition 3. *Let T be a keyword taxonomy, $C = \{s_1 ... s_k\}$ a set of synsets in T, D the semantic distance and F_δ a clustering quality function.*

C is a cluster iff: $\forall s_i,\ s_j \in C$: $D(s_i, s_j) \leq \delta$ and F_δ is maximal.

The quality function $F_\delta(\mathscr{C})$ defined on a set of clusters $\mathscr{C}$ is:

$$F_\delta(\mathscr{C}) = \sum_{C \in \mathscr{C}} v(C) = \sum_{C \in \mathscr{C}} f(z_C,\ l_C) = \sum_{C \in \mathscr{C}} z_C / l_C$$

With:

- $z_C = \sum_{term \in C} fr_{term}$ denotes the size of a cluster C and fr_{term} the frequency of a term in C (i.e., the number of times the term occurs in the queries)

- $l_C = 1/(1 + \sum_{term \in C} depth_{term})$ denotes the abstraction level of a cluster C and $depth_{term}$ the depth of a term in the taxonomy. The abstraction level of a cluster is inversely related to the sum of the depths. Thus, the deeper the terms, the lower the abstraction level of the cluster.

The threshold δ and the quality function F_δ are actually directly related since the modification of δ implies the modification of the set of clusters $\mathscr{C}$, and hence the modification of $F_\delta(\mathscr{C})$. Indeed, the value of δ to find is the one which maximizes F_δ.

Algorithm 6.1 General User Interests Extraction

Input:
1: T /* *taxonomy with weighted links*
2: S_0 /* *the set of all the synsets of* T
3: D /* *distance function*
4: F_δ /* *clustering quality function*
Output: $\mathscr{C}$ /* *the set of clusters*
5: $P_T = getLongestPath(T)$ /* *list the weights of edges of the longest path*
6: $\delta = 0$, $i = 0$, $q = 0$
7: **for all** $edgeWeight \in P_T$ **do**
8: $\delta = \delta + edgeWeight$ /* *update the threshold value*
9: $S = S_0$
10: $\mathscr{C}_{temp} = \{\emptyset\}$ /* *set of clusters*
11: $C = null$ /* $C \in \mathscr{C}_{temp}$
12: **while not** $isEmpty$(S) **do**
13: $s_d =$ getDeepest(S) /* *find the deepest term ; randomly select one in case of several terms*
14: **for all** $\theta = parentOf(s_d)$ **do**
15: $C = \{s_d\} \cup$ CLUSTER_UP(s_d, s_d, θ)
16: **end for**
17: $\mathscr{C}_{temp} = \mathscr{C}_{temp} \cup \{C\}$
18: $S = S - C$
19: **end while**
20: i = i++
21: **if** $F_\delta(\mathscr{C}_{temp}) >$ q **then**
22: q $= F_\delta(\mathscr{C}_{temp})$
23: $\mathscr{C} = \mathscr{C}_{temp}$
24: **end if**
25: **end for**

26: **function** CLUSTER_UP(deepest, predecessor, s)
27: **if** $D(deepest, s) \leq \delta$ **then**
28: **for all** $\sigma = childOf(s)$ **such as** $\sigma \neq predecessor$ **do**
29: $C' = C' \cup$ CLUSTER_DOWN($deepest$,s, σ)
30: **end for**
31: $C' = C' \cup \{s\}$
32: **end if**
33: **for all** $\pi =$ parentOf(s) **do**
34: CLUSTER_UP(deepest, s, π)
35: **end for**
36: **return** C'
37: **end function**

38: **function** CLUSTER_DOWN(deepest, predecessor, s)
39: **if** $D(deepest, s) \leq \delta$ **then**
40: **for all** $\pi = parentOf(s)$ **such as** $\pi \neq predecessor$ **do**
41: $C'' = C'' \cup$ CLUSTER_UP($deepest$,s, π)
42: **end for**
43: $C'' = C'' \cup \{s\}$
44: **end if**
45: **for all** $\sigma =$ childOf(s) **do**
46: CLUSTER_DOWN(deepest, s, σ)
47: **end for**
48: **return** C''
49: **end function**

The clustering algorithm (see Algorithm 6.1) acts as follows:

1. It initializes the value of the threshold δ with the weight of the first edge of P_T (*i.e.*, $D(s_d, s_1)$), (cf. lines 6 and 8).

2. It initializes the cluster with the deepest term (s_d) in the taxonomy T (cf. lines 13 to 16); the goal is to get the most specific terms in the cluster currently being built .

3. It tries to find the closest terms to the deepest term s_d that respect the threshold by switching in two dimensions, height (parents) and width (children) of the taxonomy (cf. lines 15, 29, 34, 41, 46). Thus, first it checks the distance between s_d and its parents with respect to the threshold using the function "cluster_up" then, if the threshold condition is satisfied for a parent, it uses cluster_down to similarly check the children terms related to that parent. The process is recursively applied on the next parents (parents of parents) until there is no term that respects the threshold condition with the initial term s_d. The result of this iteration is one cluster; all terms of the cluster are excluded from the next iterations (cf. lines 17, 18).

4. It seeks the deepest term again and repeats 2 and 3 on the rest of the terms until they are all clustered (cf. loop while, lines 12-21).

5. It calculates $F_\delta(\mathscr{C})$ with $\mathscr{C}$ the current set of clusters and increases the value of the threshold by the next edge of P_T (cf. lines 21-24 and line 8).

6. It repeats steps 1,...,5 until δ is equal to the length of P_T. The set of clusters $\mathscr{C}$ corresponds to the one which maximizes $F_\delta(\mathscr{C})$.

Starting with the deepest term and using our distance D favors the construction of large clusters of specialized term at the expense of general terms. In fact, since the distance between a term and its child is smaller than the distance between the term and its parent (the property of the distance function, cf. sec. 5.2.3) the clustering grows in width rather than in height. For example (cf. Fig. 6.1), the algorithm produces the cluster[1]: "Contact-sport, rugby, boxing, fight, football, hokey" but puts the terms "sport" and "diversion" into two different clusters as they are situated at the top of the taxonomy and hence less related.

[1]In the example, we consider that each term has a frequency equal to 1

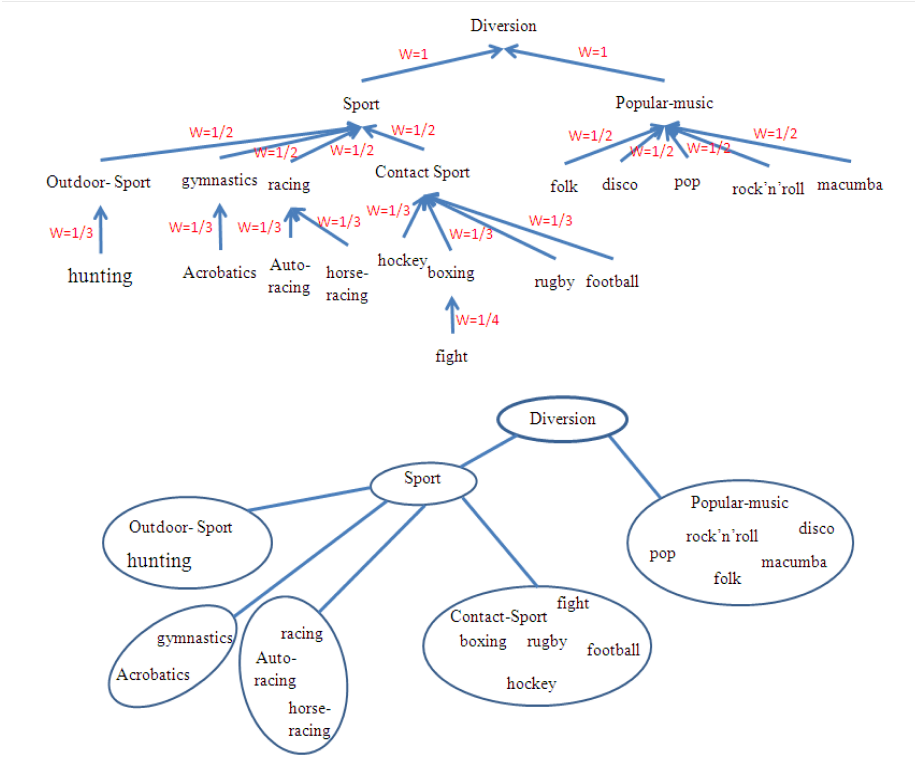


Figure 6.1.: Example of Taxonomy Before and After Clustering

6.2.1.2. The Computational Complexity

One of the strong points of the algorithm is its computational complexity. As the algorithm works on already sorted elements (*i.e.* by the $is-a$ relation), the elements are visited only once. As a result, the number of clustering operations (i.e., cluster_up and cluster_down functions) is limited to "$(\mid P_T \mid) \times n$", with n the number of terms in the taxonomy and P_T the longest path. Thus, the complexity of the algorithm is $O(kn)$, k being the length of P_T. Note that the algorithm is based on a self optimization approach, since it tries several values of δ to calculate $F_\delta(\mathscr{C})$ (i.e., as many values as the length of P_T). In fact, if δ is known in advance (i.e., experimentally defined) the algorithm is linear and has as complexity $O(n)$.

6.2.2. A Model of General Interests

The query terms clustering algorithm applied on the keyword taxonomy produces a set of clusters which elements are semantically related by the $is-a$ relation. In addition to that the clusters are sorted into different abstraction levels. Since the

keywords in the queries are specified by the user, they express implicitly the user's interests. In fact, the clusters of keywords computed by the previous algorithm represent the set of general interests expressed by the whole set of users registered in the log. The example in Fig. 6.1 shows the keyword taxonomy transformed into a set of seven clusters of user's interests in music and sport. Therefore, we define the model of general user interests as follows:

Definition 4. *Let T be a keyword taxonomy, $\mathscr{C} = \{c_1...c_n\}$ the set of clusters of T. The model of general interests $M_{\mathscr{C}}$ is defined on $\mathscr{C}$ as a vector such as:*

$M_{\mathscr{C}} = (z_{c_1}...z_{c_n})$, *with* $z_{c_i} = \sum_{term \in c_i} fr_{term}$ *denote the size of the cluster* c_i *and* fr_{term} *the frequency of "term" in* c_i

6.3. Implicit User Profile and Data Source Profile Modeling

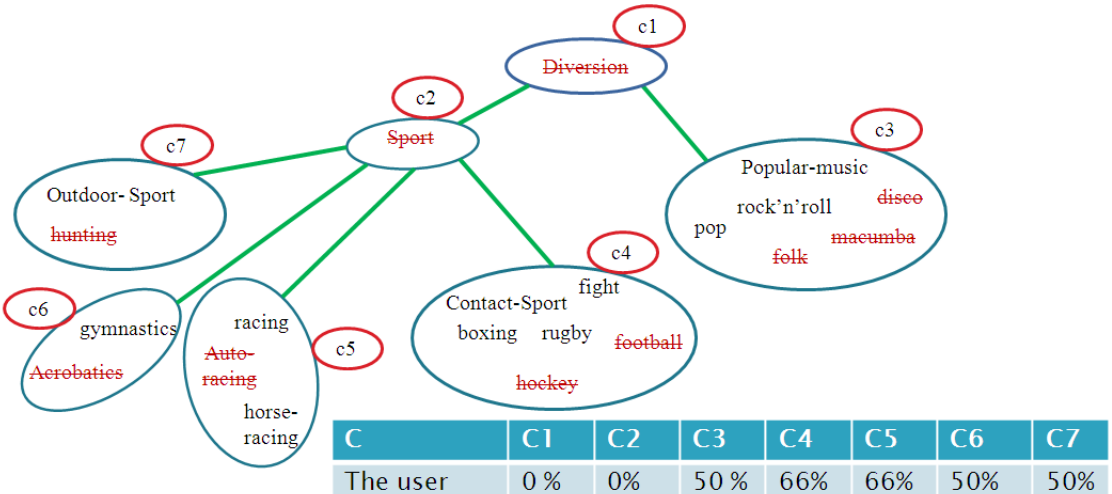
The model of interests $M_{\mathscr{C}}$ described in definition 4 is a global representation of the interests of the whole set of users. The user profile is defined by the set of the user's personal interests. Therefore, the user profile is defined on $M_{\mathscr{C}}$ as the contribution of the user in each cluster c_i in $M_{\mathscr{C}}$. The user contribution in a cluster represents the ratio between the keywords submitted by the user and the whole set of the keywords in the cluster. In fact, the ratio represents the level of interest of the user in one interest identified in the model. The user profile is formally defined below:

Definition 5. *Let T be a keyword taxonomy, $\mathscr{C} = \{c_1...c_n\}$ the set of clusters of T, $M_{\mathscr{C}}$ the model of general interests defined on $\mathscr{C}$, and u a user contributing in $M_{\mathscr{C}}$. The user profile $M^u_{\mathscr{C}}$ is defined on $M_{\mathscr{C}}$ such as:*

$M^u_{\mathscr{C}} = (\frac{z^u_{c_1}}{z_{c_1}}...\frac{z^u_{c_n}}{z_{c_n}})$,

with $z^u_{c_i}$ the number of keywords of the user u in the cluster c_i, and z_i the number of all the keywords of the cluster c_i.

Fig. 6.2 is an example of a user profile extracted from the model of interests in Fig. 6.1. The struck out keywords are those that the user did not include in his/her queries u (e.g., $z^u_{c_1} = 0$, $z^u_{c_2} = 0$, $z^u_{c_3} = 3$, etc.).



C	C1	C2	C3	C4	C5	C6	C7
The user contribution	0 %	0%	50 %	66%	66%	50%	50%

Figure 6.2.: Basic Profile Extraction

The user profile definition given in definition 5 represents the user's contribution in each cluster of the model of interests. This definition is valid when the clusters are completely independent. In contrast, since the clustering algorithm is operated on the taxonomy structure, the clusters are hierarchically related by the $is-a$ relation. This relation between clusters is derived from the relations between their corresponding elements. For example, in Fig. 6.2 cluster c_4 and c_2 are hierarchically related since the element "contact-sport" of the cluster c_4 is related to the element "sport" of the cluster c_2 by the $is-a$ relation. In fact, if we consider the clusters hierarchy one has to consider the influence of the child cluster on the parent cluster. For instance, in the example of the Fig. 6.2, the fact that the user is interested in "contact-sport" means implicitly that he/she is interested in "sport" but partially. To this end, we introduce the concept of **hierarchical normalization**.

The hierarchical normalization consists in adding the influence of the child clusters to their corresponding parents (e.g. clusters $c_1$and c_2 in Fig. 6.3). Moreover, the influence of a child cluster c_{child} on the parent cluster c_{parent} depends on the distance between the two clusters. We define the distance $D(c_{child}, c_{parent})$ to be the distance between their corresponding closest elements. The distance between two elements has been defined previously (cf. sec. 5.2.3). Let $\mathscr{A}$ be the set of child clusters of c_{parent}. We denote $f_{c_{parent}}$ the influence on the parent cluster c_{parent} coming from child clusters. It is defined recursively as:

$$f_{c_{parent}} = \sum_{c_{child} \in \mathscr{A}} \frac{z_{c_{child}} + f_{c_{child}}}{D(c_{parent}, c_{child})}$$

With: $f_{c_{child}} = 0$, if c_{child} is a leaf cluster.

Consequently, the influence of a child cluster on its parent is higher in the bottom of the clusters hierarchy as the distance is smaller. Indeed, the next definition of the user profile takes into account this hierarchical normalization:

Definition 6. *Let T be a keyword taxonomy, $\mathscr{C} = \{c_1 ... c_n\}$ the set of clusters of T, $M_{\mathscr{C}}$ the model of general interests defined on $\mathscr{C}$, and u a user contributing in $M_{\mathscr{C}}$. The user profile $M^u_{\mathscr{C}}$ is defined on $M_{\mathscr{C}}$ such as:*

$M^u_{\mathscr{C}} = (I^u_{c_1}, ..., I^u_{c_n})$,

with:

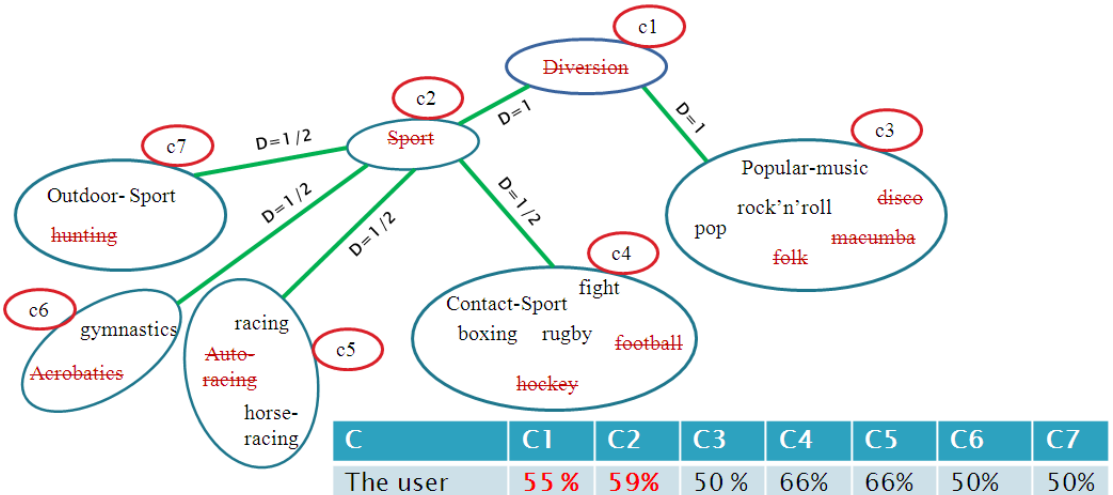
- $I^u_{c_i} = \frac{z^u_{c_i} + f^u_{c_i}}{z_{c_i} + f_{c_i}}$ *the contribution of the user u in the cluster c_i after hierarchical normalization*
  - $f^u_{c_i} = \sum_{j=1}^{k} (\frac{z^u_{c_j} + f^u_{c_j}}{D(c_i,\ c_j)})$ *the influence of the k child clusters on the parent cluster c_i, which considers only the keywords of the user u*
  - $f_{c_i} = \sum_{j=1}^{k} (\frac{z_{c_j} + f_{c_j}}{D(c_i,\ c_j)})$ *the influence of the k child clusters on the parent cluster c_i*
  - *If a child cluster "c_j" is a leaf cluster: $f^u_{c_j} = 0$ and $f_j = 0$, where c_i is the parent of c_j*

In the example in Fig. 6.3, the contributions of the user in clusters c_1 and c_2 are updated to include the influence of the child clusters. In this way, the cluster c_2 is influenced by the clusters c_4, c_5, c_6, and c_7, and the cluster c_1 is influenced by the clusters c_2 and c_3. Indeed, the contribution of the user u in the clusters c_1 and c_2 are respectively updated to 50% and 59% (cf. Fig. 6.3):

- $I^u_{c_2} = \frac{z^u_{c_2} + f^u_{c_2}}{z_{c_2} + f_{c_2}} = \frac{0 + 16}{1 + 26} \approx 0.59 \equiv 59\%$
  - $f^u_{c_2} = \frac{z^u_{c_4}}{D(c_2, c_4)} + \frac{z^u_{c_5}}{D(c_2, c_5)} + \frac{z^u_{c_6}}{D(c_2, c_7)} + \frac{z^u_{c_7}}{D(c_2, c_7)} = \frac{4}{1/2} + \frac{2}{1/2} + \frac{1}{1/2} + \frac{1}{1/2} = 16$

- $f_{c_2} = \frac{z_{c_4}}{D(c_2,c_4)} + \frac{z_{c_5}}{D(c_2,c_5)} + \frac{z_{c_6}}{D(c_2,c_7)} + \frac{z_{c_7}}{D(c_2,c_7)} = \frac{6}{1/2} + \frac{3}{1/2} + \frac{2}{1/2} + \frac{2}{1/2} = 26$

- $I^u_{c_1} = \frac{z^u_{c_1} + f^u_{c_1}}{z_{c_1} + f_{c_1}} = \frac{0+19}{1+33} \approx 0.55 \equiv 55\%$
  - $f^u_{c_1} = \frac{z^u_{c_2} + f^u_{c_2}}{D(c_1,c_2)} + \frac{z^u_{c_3}}{D(c_1,c_3)} = \frac{0+16}{1} + \frac{3}{1} = 19$
  - $f_{c_1} = \frac{z_{c_2} + f_{c_2}}{D(c_1,c_2)} + \frac{z_{c_3}}{D(c_1,c_3)} = \frac{1+26}{1} + \frac{6}{1} = 33$



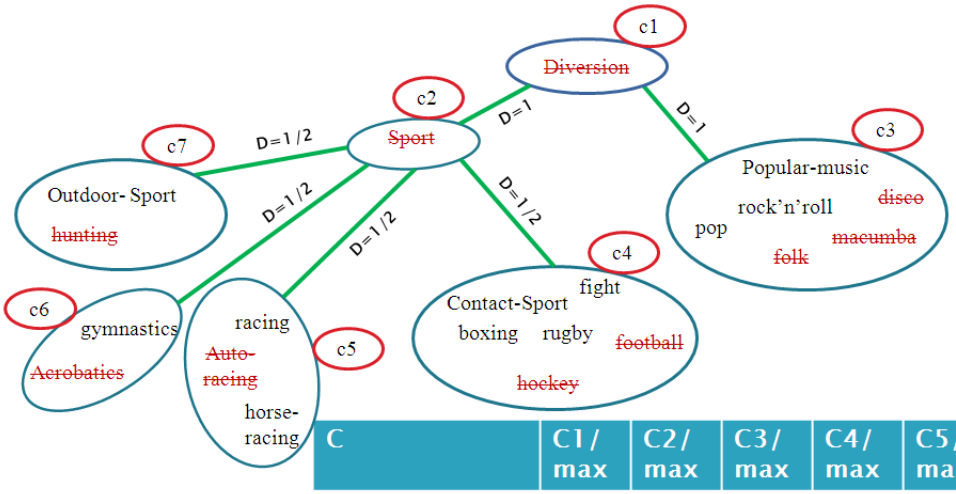
C	C1	C2	C3	C4	C5	C6	C7
The user contribution	55 %	59%	50 %	66%	66%	50%	50%

Figure 6.3.: Hierarchical Normalization

The hierarchical normalization introduced in definition 6 enables to update the user profile by taking into account the hierarchical relations between the clusters. Nevertheless, the definition 6 is still not adapted to allow comparison between the users profiles. The problem is that the users do not have the same number of search queries in the same search session. As a result, the profile vectors calculated from the model of general interests are affected by the number of the search queries, which may artificially create differences between otherwise similar profiles. For example, two users u_1 and u_2 who have respectively the profiles $M^{u_1}_{\mathscr{C}}(50\%, 57\%, 50\%, 66\%, 33\%, 100\%, 50\%)$ and $M^{u_2}_{\mathscr{C}}(5\%, 5, 7\%, 5\%, 6, 6\%, 3, 3\%, 10\%, 5\%)$ are similar in term of interests distribution but different if we calculate the distance between them. In fact, the only difference is that user u_1 is ten times more active than user u_2. In order to remove the search activity effect we add a **linear normalization** to the process. This normalization consists in recalculating the profile vector in relation to the maximum contribution of the user within the vector (see Fig. 6.4). To this end, we propose the following definition:

Definition 7. *Let T be a keyword taxonomy, $\mathscr{C} = \{c_1 ... c_n\}$ the set of clusters of T, $M_{\mathscr{C}}$ the model of general interests defined on $\mathscr{C}$, and u a user contributing in $M_{\mathscr{C}}$. The user profile $M^u_{\mathscr{C}}$ is defined as:*

$$M^u_{\mathscr{C}} = (\frac{I^u_{c_1}}{\max\{I^u_{c_1} ... I^u_{c_n}\}} ... \frac{I^u_{c_n}}{\max\{I^u_{c_1} ... I^u_{c_n}\}}),$$



C	C1/ max	C2/ max	C3/ max	C4/ max	C5/ max	C6/ max	C7/ max
Normalized user contribution	83%	89%	75%	100%	100%	75%	75%

Figure 6.4.: Inter-Cluster Normalization

The user profile in definitions 5, 6, 7 is based on the model of general interests in definition 4. This model is a hierarchy of clusters of keywords. The keywords are extracted from search queries of the users registered in the search log. In fact, using search queries to analyze the user interests is very advantageous since a query enables to relate the user (*i.e.*, the requester) to the data source (*i.e.*, the provider). Indeed, we can exploit the relation induced by the query to similarly model the data source profile in order to describe the content of the data-source. This profile is calculated in the form of a vector which components represent the contributions of the data source to the model of general interests. The contribution of a data source in a cluster is the ratio between the number of keywords for which the data-source has delivered content and the number of the keywords composing the cluster. The definition of the data source profile is similar to definition 7 , the definition of the user profile:

Definition 8. *Let T be a keyword taxonomy, $\mathscr{C} = \{c_1 ... c_n\}$ the set of clusters of T, $M_{\mathscr{C}}$ the model of general interests defined on $\mathscr{C}$, and d a data source contributing in $M_{\mathscr{C}}$. The data source profile $M^d_{\mathscr{C}}$ defined on $M_{\mathscr{C}}$ is:*

$$M^d_{\mathscr{C}} = (\frac{I^d_{c_1}}{\max\{I^d_{c_1}...I^d_{c_n}\}}...\frac{I^d_{c_n}}{\max\{I^d_{c_1}...I^d_{c_n}\}}),$$

with:

$I^d_{c_i} = \frac{z^d_{c_i} + f^d_{c_i}}{z_{c_i} + f_{c_i}}$ *the contribution of the data source d in the cluster* c_i *after hierarchical normalization.*

6.4. Community Discovery and Data Source Categorization

The two direct applications of the user profile and data source profile models respectively are community discovery and data source categorization. The main objective of these applications is to group together entities that have similar profiles. Therefore, we define a community as a set of users, who share similar interests while a data source category is defined as a set of data sources, which share similar contents. In fact, the profile model we presented above has specific advantages in these applications related to its characteristics.

First, the user/data sources profiles are calculated from a set of semantic clusters of synsets, which enables more precise profile comparison as opposed to comparison based only on the text form of keywords.

Second, interests are represented in the form of a hierarchy that separates general interests from specific ones, which enables a fine grained comparison.

Third, the user interests are transformed into vectors of numerical values, which enables the use analytical tools (such as clustering, PCA, etc.).

Fourth, it provides a unified representation of the user profile and the data source profile as they are derived from the same model (*i.e.,* the model of general interests). The advantage of such a unified representation of the profiles is the ability to make all kinds of comparisons: user to user, data source to data source, and user to data source. The first two comparisons enable respectively to identify user communities and to categorize the data sources while the third enables to identify the mapping between the user interests and the data source contents, which is typically used for query routing. These three applications are formalized in the next sections.

6.4.1. User Community Discovery

The definition of the concept of community is based on definition 7 , which defines the user profile. A community is then a set of users whose profile vectors are grouped together in one cluster:

Definition 9. *Let $M_{\mathscr{C}}$ be the model of general interests, $\mathscr{U}_{\mathscr{C}} = \{U^1_{\mathscr{C}}, ..., U^n_{\mathscr{C}}\}$ the set of user profiles defined over $M_{\mathscr{C}}$, and ψ a clustering function. We define a community of users as a subset of user profiles $\mathscr{R}_{\mathscr{C}} \subseteq \mathscr{U}_{\mathscr{C}}$, such that $\mathscr{R}_{\mathscr{C}}$ is a cluster generated by ψ.*

6.4.2. Data Source Categorization

The definition of a data source category is based on definition 8 , which defines the data source profile. A data source category is then a set of data sources which profile vectors are grouped together in one cluster:

Definition 10. *Let $M_{\mathscr{C}}$ be the model of general interests, $\mathscr{D}_{\mathscr{C}} = \{D^1_{\mathscr{C}}, ..., D^m_{\mathscr{C}}\}$ the set of data source profiles defined over $M_{\mathscr{C}}$, and ψ a clustering function. We define a category of data sources as a subset of data sources $\mathscr{S}_{\mathscr{C}} \subseteq \mathscr{D}_{\mathscr{C}}$, such that $\mathscr{S}_{\mathscr{C}}$ is a cluster generated by ψ.*

6.4.3. Mapping the Users to the Data Sources

The mapping between the users and the data sources is the set of couples *user* × *data source* such that the distance between the user profile and the data source is less than a predefined threshold. The value of the threshold represents the degree of similarity between the user needs and the contents of the data source in question. For instance, in an information retrieval system, it can be used for keyword query routing [TZ14], which is the process whereby search queries are only routed to relevant data sources in order to reduce the processing cost.

Definition 11. *Let $M_{\mathscr{C}}$ be the model of general interests, $\mathscr{U}_{\mathscr{C}} = \{U^1_{\mathscr{C}}, ..., U^n_{\mathscr{C}}\}$ the set of the user profiles, and $\mathscr{D}_{\mathscr{C}} = \{D^1_{\mathscr{C}}, ..., D^m_{\mathscr{C}}\}$ the set of the data source profiles defined over $M_{\mathscr{C}}$. We define a mapping $P_{\mathscr{C}} \subseteq \mathscr{U}_{\mathscr{C}} \times \mathscr{D}_{\mathscr{C}}$ as a set of couples user × data source such that :*

$P_{\mathscr{C}} = \{(U^i_{\mathscr{C}}, D^j_{\mathscr{C}}) \in \mathscr{U}_{\mathscr{C}} \times \mathscr{D}_{\mathscr{C}} : \ d(U^i_{\mathscr{C}}, D^j_{\mathscr{C}}) \leq t\}$

with:

- $U^i_{\mathscr{C}}$ *and* $D^j_{\mathscr{C}}$ *are respectively the profiles of the user "i" and the data source "j".*
- d*: a distance function (e.g., the Euclidean distance).*
- t *: the mapping threshold.*

Note that each user profile could be associated with several data source profiles, and conversely, each data source could be associated with several user profiles.

6.5. Experimental Results

6.5.1. Clustering Evaluation

From a structural point of view, the quality of the clustering depends on a separation criterion that measures the intra-cluster relatedness and the inter-cluster unrelatedness. Thus, a clustering of good quality should maximize the intra-cluster similarity and minimize the inter-cluster similarity. In our experimentation we use the Davies–Bouldin index (DBI) [DB79] to evaluate our clustering algorithm. The DBI is given by the following formula:

$$DBI = \frac{1}{n}\sum_{i=1}^{n}\left(\max_{i \neq j}\frac{s_i + s_j}{d(c_i, c_j)}\right)$$

With,

- s_i corresponding to the average distance of all elements in cluster "i" to centroid c_i. It is a measure of the scattering within the cluster "i". (s_j is defined symmetrically on the cluster "j").
- $d(c_i, c_j)$ the distance between the centroids of cluster i and cluster j.

The DBI calculates the separation between the clusters by considering the distance between the centroids and the scattering within each cluster. The DBI determines first for each cluster "i" the cluster "j" that maximizes the ratio between $s_i + s_j$ and $d\,(c_i, c_j)$. The value of the DBI is then the average value of these ratios. Indeed, the best clustering corresponds to the one which has the smallest value of the DBI.

Cluster5: bren={SynID: 02897097} kalashnikov={SynID: 03607659} luger={SynID: 03695857} spyglass={SynID: 03333129} spandau={SynID: 04267091} glass={SynID: 03333129} peacemaker={SynID: 02907656} colt={SynID: 03073296} lance={SynID: 03637618} garand={SynID: 03416775} m-1={SynID: 03416775} lancet={SynID: 03637618} dragunov={SynID: 03231819},...

Cluster11: freezer={SynID: 03170635} slicker={SynID: 03844815} oilskin={SynID: 03844815} mac={SynID: 03702719} blazer={SynID: 02850358} mack={SynID: 03702719} burberry={SynID: 02921406} fridge={SynID: 03273913} lumberjack={SynID: 03696909} mink={SynID: 03770954} swallowtail={SynID: 04368496} banyan={SynID: 02788462} undies={SynID: 04509171} rotisserie={SynID: 04111531},...

Cluster23: stash={SynID: 13366912} hoard={SynID: 13366912} cache={SynID: 13366912} fund={SynID: 13367070} reserve={SynID: 13368052} rent={SynID: 13296270} dividend={SynID: 13408023} taxation={SynID: 13261916} revenue={SynID: 13261916} killing={SynID: 13259797} cleanup={SynID: 13259797} margin={SynID: 13260762} net={SynID: 13258362} profit={SynID: 13258362} payoff={SynID: 13260190} earnings={SynID: 13258362-},...

Cluster28: entomophobia={SynID: 14385160} arachnophobia={SynID: 14382075} acrophobia={SynID: 14382766} photophobia={SynID: 14384351} triskaidekaphobia={SynID: 14384684} androphobia={SynID: 14381997} aaa={SynID: 14106456} pvc={SynID: 14362841} frenzy={SynID: 14391876} tachycardia={SynID: 14363027} agoraphobia={SynID: 14381840} hysteria={SynID: 14391876} bradycardia={SynID: 14362510},...

Cluster30: christmas={SynID: 15196186} ascension={SynID: 15193052} xmas={SynID: 15196186} easter={SynID: 15188154} passover={SynID: 15195928} purim={SynID: 15196870} pentecost={SynID: 15197042} pesach={SynID: 15195928} shavuot={SynID: 15197042} hanukkah={SynID: 15199033} hanukah={SynID: 15199033} chanukah={SynID: 15199033} hannukah={SynID: 15199033} chanukkah={SynID: 15199033} stp={SynID: 13780339} thanksgiving={SynID: 15201116} dormition={SynID: 15194194},...

Cluster73: revisionism={SynID: 08368516} international={SynID: 08366071} communism={SynID: 08365855} socialism={SynID: 08366202} capitalism={SynID: 08364143} purdah={SynID: 08379882} segregation={SynID: 08380340} feudalism={SynID: 07972425} pluralism={SynID: 08367683} industrialism={SynID: 08364757} patriarchy={SynID: 07972674} patriarchate={SynID: 07972674} separatism={SynID: 08380340},...

Figure 6.5.: Examples of Semantic Clusters

6.5.2. Semantic clustering of the AOL search keywords

The graph in Fig. 6.7 shows the variation of the clustering quality function with respect to the threshold values. The threshold values are represented with intervals where there is no variation of the clustering quality[2]. As shown in definition 3, the clusters should correspond to the maximum value of the quality function. The value of the threshold that maximizes the quality function in Fig. 6.7 corresponds to the interval $[1,314, 1,514[$. This means that the sum of the ratios between the size of a cluster and its abstraction level is maximal in the set of the obtained clusters. Oppositely, the values of the threshold that are out of this interval decreases the clustering quality, since for the values that are less than this interval the clusters are smaller, and for the values that are higher than this interval the abstraction level is higher within the clusters. Fig. 6.5 shows some examples of clusters corresponding to

[2]The reason is that, the result of the clustering do not change as well

the threshold in $[1,314, 1,514[$. Indeed, cluster5 seems to correspond to a collection of arms, cluster11 seems to corresponds to house appliances, cluster23 is related to financial concerns, cluster28 corresponds to human psychological status including phobias, cluster30 corresponds to religious holidays and finally cluster73 corresponds to political trends.

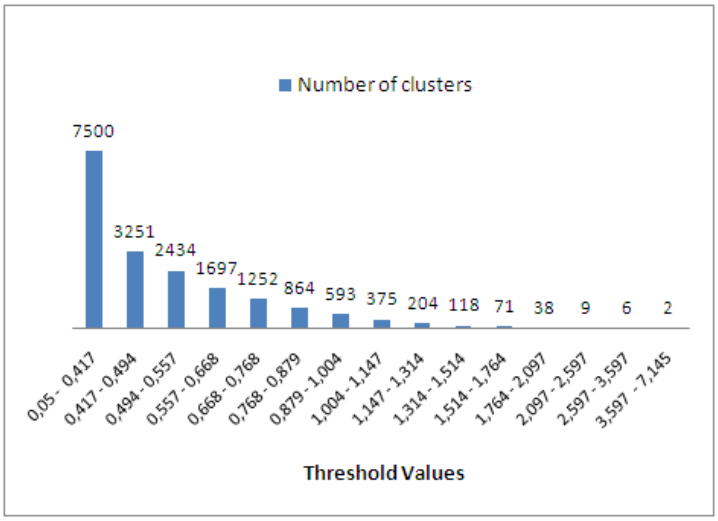


Figure 6.6.: Number of Clusters Corresponding to the Threshold Values

The number of clusters corresponding to the different values of the threshold[3] are shown in Fig. 6.6. The figure shows a monotonous decrease of the number of clusters between the two extreme values of the threshold. Indeed, since the clustering is based on a pruning approach, the number of clusters is directly affected by the value of the threshold. Thus, the more the threshold increases the more the size of the clusters increases and, hence, the number of cluster decreases. Furthermore, the speed of the decrease of the number of clusters depends on the shape of the taxonomy (cf. sec. 5.4). For instance, a wide taxonomy favors large clusters because a child element is closer than a parent element (cf. the distance function) and hence, reduces the number of clusters; to the contrary, a deep taxonomy favors small clusters and hence, increases the number of clusters.

[3]The interval $[0, 05, 0, 417[$ shows the average number of clusters corresponding to the first five smallest values of threshold

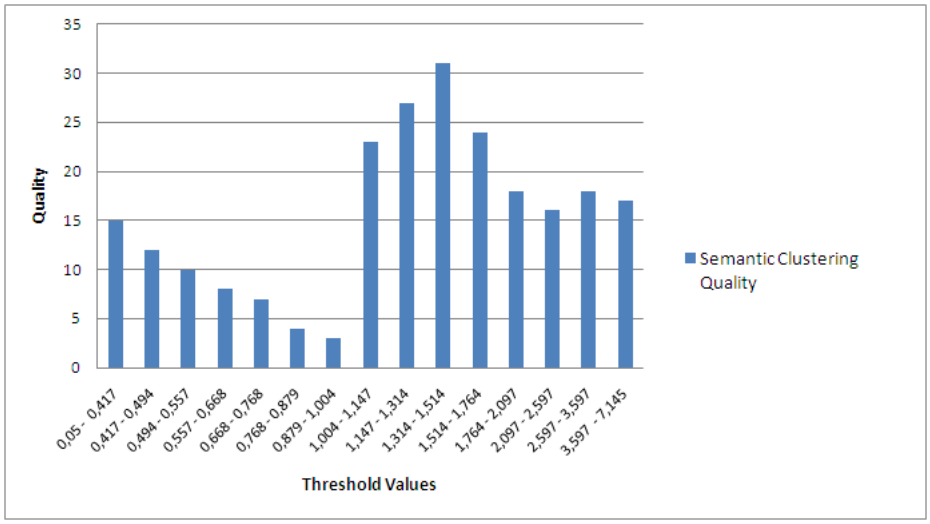


Figure 6.7.: Semantic Clustering by Threshold Tuning

6.5.3. Structure-Based Clustering Evaluation

In order to evaluate the clusters from a structural point of view, we use the DBI measure. Contrarily to the clustering quality function, the best value of the DBI is the smallest one. Fig. 6.8, shows the DBI for the same values of threshold as defined previously. The DBI corresponding to the highest value of the quality function is not necessarily the lowest one. In fact, the graph shows the DBI of the best value of the quality function at the 11th position from the 15 values represented in the graph. Nevertheless, the values of the DBI corresponding to thresholds $[1,764, 2,097[$, $[2,097, 2,597[$, $[2,597, 3,597[$ and $[3,597, 7,145[$ are better than the the value of the DBI corresponding to the best value of the quality function. These results are related to the definition of the DBI. By definition, the DBI determines for each cluster "i" the cluster "j" that maximizes the ratio between $s_i + s_j$ and $d(c_i, c_j)$. Thus, its value can be directly affected by s_i and s_j. In the case of the previous thresholds, the clusters contain more specific keywords (i.e., close each to other) than general keywords, which makes the average distance smaller compared to other clusters. Indeed, Fig. 6.6 shows that the number of clusters corresponding to these thresholds is very small, meaning that these clusters are large and contain a large amount of specific keywords (cf. the shape of the taxonomy in sec. 5.5.3). Consequently, semantic clustering cannot be performed on the taxonomy by considering only the structure of the clusters, but should also make use of a clustering quality function.

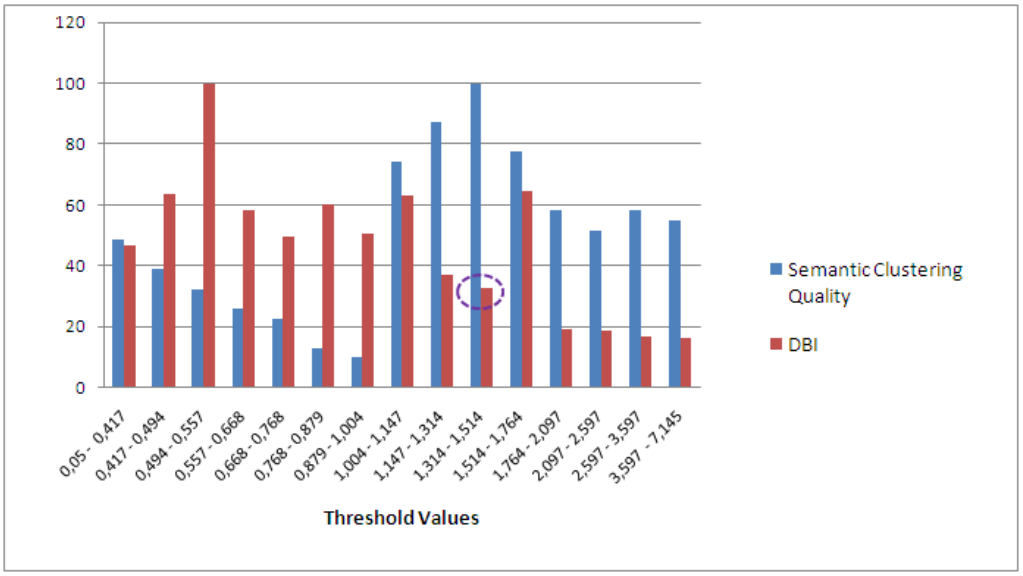


Figure 6.8.: Structure-based Clustering Evaluation

6.6. Summary

In this chapter we present a methodology to extract the user profile and the data source profile, by using the keyword taxonomy. Thus, we first apply a clustering algorithm on the taxonomy to extract the general user interests. The algorithm makes use of a custom clustering quality function F_δ, which depends on the size of the clusters and on their abstraction level. Second, the model of interests is instantiated to each user and data source to calculate their corresponding profiles. The instantiation is done by calculating the contribution of each user/data source in the model. The experimentation conducted on the clustering algorithm enabled us first to evaluate the clustering with respect to our clustering quality functionF_δ. We then evaluated our clustering algorithm using the clustering structural quality measure DBI and studied the relationship between clustering based on the quality function F_δ and clustering based on the structure. Indeed, we conclude from the experimentation that semantic clustering can be performed on the taxonomy by considering both the structure and the clustering quality function.

Part IV.

Conclusion

7. Conclusion and Futures Perspectives

The user profile is the cornerstone in user-centric systems such as adaptive information retrieval systems. Therefore, much research has been done to find the best profile model. In this thesis, we assessed the most important work on user modeling since the first proposals. Our study was based on a set of parameters, which we deemed necessary to consider in the development of any user model. These parameters are: the representation, the genericity, the type of data, the duration, the semantics, the operability and the scalability (i.e., in terms of complexity). In our study, we found that most of the previous models consider only a part of these parameters, which makes them dependent on the application in which they have been proposed.

In this regard, we developed in this thesis a profile model which considers all these parameters so that it can be integrated in any user-centric application. In addition to that, it gives a special attention to the genericity, the semantics and the scalability parameters. The reasons are that, first, the data source is as important as the user, thus it should be integrated in the model (genericity), second, it should be able to integrate semantic mechanisms to enhance its interpretabilty (semantics), third, it should be able to deal with big amount of data (scalability). Furthermore, the construction of the model is based on the analysis of the user web search logs, which represents one of the challenges of this thesis because of their textual nature.

The model we proposed is composed of several components which are organized in the form of a framework. The main contributions can be summarized as follows:

- A preprocessing methodology to extract the query keywords.
- An algorithm for keywords disambiguation, which enhances the interpretabilty of the model and avoids the problems related to polysemous keywords. The algorithm shows satisfactory performance in short queries, which makes it

well adapted to deal with search queries. The complexity of the algorithm is polynomial.

- An algorithm to construct a keyword taxonomy based on the sense of each keyword. The taxonomy represents the semantic basis on which the profile is calculated. The complexity of the algorithm is linear.
- A semantic dissimilarity function which enables to distinguish between specific keywords and general keywords.
- A clustering algorithm to extract semantically the user interests from the keyword taxonomy. The algorithm is based on a pruning approach and on a quality function. Thanks to the dissimilarity function, the algorithm distinguishes specific clusters and general clusters. The experimentation shows that the algorithm is consistent with the DBI measure in terms of structure.
- A methodology to calculate the user and the data source profiles from the profile model. The strong points of this method are that, first, it takes advantage of the semantic representation of the user interests, and second, it represents the profile in the form of a vector which favors comparison between profiles.
- A unified profile model which can represent both of the user profile and the data source profile

The first objective of the model is to consider all of our parameters, so that it can be integrated in any user-centric application. In this way, the model opens several new perspectives. In fact, the objective in the near future is to assess its contribution to some of potential applications in the domains of information retrieval.

- Extraction of Documents in a Distributed Information Retrieval System
  In distributed information retrieval systems, the user requests documents from multiple data-sources that correspond to a specific information need. In this kind of system, the users are separated from the data-sources. This means that the users are considered to be the requesters while data-sources are considered to be the providers. Therefore, the network organization should consider both of the user side and the data-source side. User grouping favors information flows between users [Tor12, YCB13] while the data-source categorization enables in one hand, to improve the retrieval process and on the other hand, to balance the workload. This is illustrated in the following interaction scenario:
  - User to Group of users: The user submits a query which returns the

documents which have been extracted by users with similar interests. This is typically a collaborative filtering process.

- User to a Group of data-sources: The user submits a query through the retrieval system which distributes the query over the appropriate data-sources and merges the returned results.
- Data-source to a Group of data-sources: The data-source replicates some documents within a group of data-sources of similar topics. The objective is to avoid bottlenecks related to an increasing demand by balancing the workload between data-sources with similar topics [SGD+02, CC04].

- Data Propagation on Mobile Networks
  In mobile networks (e.g., MANET), the users exchange information by means of their devices. Oppositely with the previous application, there is only one type of entity that composes the network (i.e., the users) since a user can play both roles of the requester and the provider of information. Usually, a mobile network is divided into a set of small and disconnected subnets where each user has the possibility to share information with the neighbors of its subnet [RT99]. Moreover, the user has the possibility to move among the subnets, which enables to discover new neighbors. In practice, the user saves a list of physical neighbors and updates it when he/she moves from a subnet to another one. The network organization consists in discovering and grouping the users who have similar interests. In the following scenario we distinguish two kinds of users: requester and provider:
  - Requester user to a Group of users: The user requests information from its neighborhood. The user first, discovers the physical neighbors in its subnet and then, he/she identifies the group of users whose interest is similar to his own. In this way, the requests are sent only to similar users, which optimizes the number of interactions within the subnet.
  - Provider user to Group of users: The provider user proposes to replicate its resources to its neighbors of similar interests. The objectives can be to recommend resources, or to insure the data availability over the network.

Bibliography

[AFJM95] Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: a learning apprentice for the world wide web. In *1995 AAAI Spring Symposium on Information Gathering from Heterogeneous Distributed Environments*, pages 6–12, 1995.

[AGHT11] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Semantic enrichment of twitter posts for user profile construction on the social web. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part II*, ESWC'11, pages 375–389, Berlin, Heidelberg, 2011. Springer-Verlag.

[AGM+11] Julien Aligon, Matteo Golfarelli, Patrick Marcel, Stefano Rizzi, and Elisa Turricchia. Mining preferences from OLAP query logs for proactive personalization. In *Proceedings of the 15th international conference on Advances in databases and information systems*, ADBIS'11, pages 84–97, Berlin, Heidelberg, 2011. Springer-Verlag.

[AS99] Giuseppe Amato and Umberto Straccia. User profile modeling and applications to digital libraries. In *Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries*, ECDL '99, pages 184–197, London, UK, 1999. Springer-Verlag.

[BBD+98] Deborah D. Blecic, Nirmala S. Bangalore, Josephine L. Dorsch, Cynthia L. Henderson, Melissa H. Koenig, and Ann C. Weller. Using transaction log analysis to improve OPAC retrieval results. *College & Research Libraries*, 59(1):39–50, 1998.

[BC01] R. Baeza-Yates and C. Castillo. Relating web characteristics with link based web page ranking. In *International Symposium on String Processing and Information Retrieval*, pages 21–32, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[Ber06] P. Berkhin. A survey of clustering data mining techniques. In Jacob Kogan, Charles Nicolas, and Marc Teboulle, editors, *Grouping Multidimensional Data*. 2006.

[BJ81] Eric Backer and Anil K. Jain. A clustering performance measure based on fuzzy set decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 3(1):66–75, January 1981.

[BLG99] Kurt D. Bollacker, Steve Lawrence, and C. Lee Giles. A system for automatic personalized tracking of scientific literature on the web. In *Digital Libraries 99 - The Fourth ACM Conference on Digital Libraries*, pages 105–113. ACM Press, 1999.

[BLL+10] Jiang Bian, Xin Li, Fan Li, Zhaohui Zheng, and Hongyuan Zha. Ranking specialization for web search: a divide-and-conquer approach by using topical RankSVM. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 131–140, New York, NY, USA, 2010. ACM.

[BMI11] D. Bollegala, Y. Matsuo, and M. Ishizuka. A web search engine-based approach to measure semantic similarity between words. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):977–990, 2011.

[Bra04] Sviatoslav Braynov. Personalization and customization technologies. In *The Internet Encyclopedia*. John Wiley & Sons, Inc., 2004.

[Car02] Jeremy J. Carroll. Matching RDF graphs. In Ian Horrocks and James Hendler, editors, *The Semantic Web – ISWC 2002*, number 2342 in Lecture Notes in Computer Science, pages 5–15. Springer Berlin Heidelberg, January 2002.

[CC02] Shui-Lung Chuang and Lee-Feng Chien. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 75—82, Washington, DC, USA, 2002. IEEE Computer Society.

[CC04] Tzung-Shi Chen and Kuo-Lian Chen. Balancing workload based on content types for scalable web server clusters. In *18th International*

Conference on Advanced Information Networking and Applications, volume 2, pages 321–325, Fukuoka, Japan, March 2004.

[CCHB10] Mark J. Carman, Fabio Crestani, Morgan Harvey, and Mark Baillie. Towards query log based personalization using topic models. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1849–1852, New York, NY, USA, 2010. ACM.

[CFS05] Michael Chau, Xiao Fang, and Olivia R. Liu Sheng. Analysis of the query logs of a web site search engine. *Journal of the American Society for Information Science and Technology*, 56(13):1363–1376, November 2005.

[Cha07] Sung-Hyuk Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4), 2007.

[DB79] David L. Davies and Donald W. Bouldin. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1(2):224–227, February 1979.

[DD09] Michel Marie Deza and Elena Deza. Encyclopedia of distances. In *Encyclopedia of Distances*, pages 1–583. Springer Berlin Heidelberg, 2009.

[Def77] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[DLSA03] Marco Degemmis, Pasquale Lops, Giovanni Semeraro, and Fabio Abbattista. Extraction of user profiles by discovering preferences through machine learning. In *Intelligent Information Processing and Web Mining, IIS'03*, volume 22 of *Advances in Soft Computing*, pages 69–78. Springer Berlin Heidelberg, 2003.

[EKSX96] Martin Ester, Hans-peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[FH93] J. Fink and M. Herrmann. KN-PART: ein verwaltungssystem zur benutzermodellierung mit prädikatenlogischer wissensrepräsentation:

Konzeptionelle grundlagen. Technical report, Departement of information science, University of Konstanz, Germany, 1993.

[Fis01] Gerhard Fischer. User modeling in Human-Computer interaction. *User Modeling and User-Adapted Interaction*, 11(1-2):65–86, March 2001.

[FK02] Josef Fink and Alfred Kobsa. User modeling for personalized city tours. *Artif. Intell. Rev.*, 18(1):33–74, September 2002.

[FM98] Christine Fellbaum and George Miller. *WordNet: An Electronic Lexical Database*. The MIT Press, May 1998.

[FW94] Michael L. Fredman and Dan E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *J. Comput. Syst. Sci.*, 48(3):533–551, June 1994.

[GA06] D. Godoy and A. Amandi. Modeling user interests by conceptual clustering. *Information Systems*, 31(4):247–265, 2006.

[GBL98] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *International Conference on Digital Libraries*, pages 89–98. ACM Press, 1998.

[GF64] Bernard A. Galler and Michael J. Fisher. An improved equivalence algorithm. *Commun. ACM*, 7(5):301–303, May 1964.

[GR11] John Gantz and David Reinsel. Extracting value from chaos. Technical report, June 2011.

[GRS98] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 73–84, New York, NY, USA, 1998. ACM.

[GSCM07] Susan Gauch, Mirco Speretta, Aravind Chandramouli, and Alessandro Micarelli. User profiles for personalized information access. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The adaptive web*, volume 4321, pages 54–89. Springer-Verlag, Berlin, Heidelberg, 2007.

[HDSB03] Younes Hafri, Chabane Djeraba, Peter Stanchev, and Bruno Bachimont. A markovian approach for web user profiling and clustering.

In *Proceedings of the 7th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'03, pages 191–202, Berlin, Heidelberg, 2003. Springer-Verlag.

[HGM+10] Jian Huang, Jianfeng Gao, Jiangbo Miao, Xiaolong Li, Kuansan Wang, Fritz Behr, and C. Lee Giles. Exploring web scale language models for search query processing. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 451–460, New York, NY, USA, 2010. ACM.

[HHK98] Alexander Hinneburg, Er Hinneburg, and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. pages 58–65. AAAI Press, 1998.

[HJ97] Pierre Hansen and Brigitte Jaumard. Cluster analysis and mathematical programming. *Mathematical Programming*, 79(1-3):191–215, October 1997.

[HSO98] Graeme Hirst and David St-Onge. Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database.*, chapter 13, pages 305–332. MIT Press, 1998.

[HW79] JA Hartigan and MA Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, 28(1):100–108, 1979.

[Jan06] Bernard J. Jansen. Search log analysis: What it is, what's been done, how to do it. *Library & Information Science Research*, 28(3):407–432, 2006.

[Jan07] Bernard J. Jansen. Investigating the relevance of sponsored results for web ecommerce queries. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 857–858, New York, NY, USA, 2007. ACM.

[JC97] Jay J. Jiang and David W. Conrath. An information-theoretic definition of similarity. In *Proceedings of International Conference Research on Computational Linguistics*, Taiwan, 1997.

[JCECWtC00] Wu Jon C.S., Hsi Eric C.N., Kate Warner ten, and Pe-

ter M.C. Chen. A framework for web content adaptation. http://www.w3.org/2000/09/Papers/Philips-v2.html, 2000.

[JCG10] Bernard J. Jansen, Gerry Campbell, and Matthew Gregg. Real time search user behavior. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 3961–3966, New York, NY, USA, 2010. ACM.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[JFM97] Thorsten Joachims, Dayne Freitag, and Tom Mitchell. WebWatcher: a tour guide for the world wide web. In *Proceedings of the fifteenth International joint Conference on Artificial Intelligence*, pages 770–775. Morgan Kaufmann, 1997.

[JST08] Bernard J. Jansen, Amanda Spink, and Isak Taksa, editors. *Handbook of Research on Web Log Analysis*. IGI Global, October 2008.

[KC03] Hyoung R. Kim and Philip K. Chan. Learning implicit user interest hierarchy for context in personalization. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 101–108, New York, NY, USA, 2003. ACM.

[KHK99] G. Karypis, Eui-Hong Han, and V. Kumar. CHAMELEON: a hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[Kob01] Alfred Kobsa. Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11(1-2):49–63, March 2001.

[KP94] Alfred Kobsa and Wolfgang Pohl. The user modeling shell system BGP-MS. *User Modeling and User-Adapted Interaction*, 4(2):59–106, 1994.

[KR87] Leonard Kaufman and Peter Rousseeuw. Clustering by means of medoids. In *Statistical Data Analysis Based on the L1-Norm and Related Methods*, pages 405–416. Y. Dodge, North-Holland, 1987.

[KT06] Jon Kleinberg and Éva Tardos. *Algorithm design*. Pearson/Addison-Wesley, Boston, 2006.

[Kur93] Martin Kurth. The limits and limitations of transaction log analysis. *Library Hi Tech*, 11(2):98–104, 1993.

[LBB81] Gilbert N. Lewis, Nancy J. Boynton, and F. Warren Burton. Expected complexity of fast search with uniformly distributed data. *Information Processing Letters*, pages 4–7, 1981.

[LBCK08] Lyes Limam, Lionel Brunie, David Coquil, and Harald Kosch. Query Log Analysis for User-Centric Multimedia Databases. In *the 8th International Conference on New Media Technology (I-Media'08)*, pages 441–444, Graz, Austria, September 2008.

[LC98] Claudia Leacock and Martin Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database.*, chapter 13, pages 265–283. MIT Press, 1998.

[LCKB10] Lyes Limam, David Coquil, Harald Kosch, and Lionel Brunie. Extracting user interests from search query logs: A clustering approach. In IEEE, editor, *the 7th International Workshop on Text-based Information Retrieval (TIR '10) in conjunction with the 21st International Conference on Database and Expert Systems Applications (DEXA '10)*, Bilbao, Spain, September 2010.

[Lee93] Geoffrey Leech. 100 million words of english. *English Today*, 9(01):9–15, 1993.

[Lin98] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[Log13] Visuwords tm online. visual dictionary, visual thesaurus. http://www.visuwords.com/?word=paper, 2013.

[LOPS07] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Fabrizio Silvestri. Mining query logs to optimize index partitioning in parallel web search engines. In *Proceedings of the 2nd international conference on Scalable information systems*, InfoScale'07, pages 43:1–43:9, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Lux04] Ulrike von Luxburg. *Statistical learning with similarity and dissim-*

ilarity functions. PhD thesis, Technische Universität Berlin, Berlin, 2004.

[MCL+10] Tobias Mayer, David Coquil, Lyes Limam, Florian Stegmaier, Mario Döller, and Harald Kosch. Live-Ticker Supported Sports-Video Annotation Enabling Tactic Analysis. In *11th International Workshop of the Multimedia Metadata Community*, pages 53–56, Barcelona, Spain, May 2010.

[MS04] Bernardo Magnini and Carlo Strapparava. User modelling for news web sites with word sense based techniques. *User Modeling and User-Adapted Interaction*, 14(2-3):239–257, June 2004.

[MSDR04] Stuart E Middleton, N. R Shadbolt, and D. C De Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88, January 2004.

[Nav09] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):10:1–10:69, February 2009.

[PA80] C. Raymond Perrault and James F. Allen. A plan-based analysis of indirect speech acts. *Comput. Linguist.*, 6(3-4):167–182, July 1980.

[PAC78] C. Raymond Perrault, James F. Allen, and Philip R. Cohen. Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, TINLAP '78, pages 125–132, Stroudsburg, PA, USA, 1978. Association for Computational Linguistics.

[PBMW97] Michael Pazzani, Daniel Billsus, S. Michalski, and Janusz Wnek. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[PCT06] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*, InfoScale '06, New York, NY, USA, 2006. ACM.

[Pir09] Giuseppe Pirró. A semantic similarity metric combining features and intrinsic information content. *Data Knowl. Eng.*, 68(11):1289–1308, November 2009.

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, December 1980.

[pri12] Synsets ranking using the british national corpus. http://wordnet.princeton.edu/wordnet/download/standoff/, August 2012.

[PSF04] A. Phippen, L. Sheppard, and S. Furnell. A practical evaluation of web analytics. *Internet Research*, 14(4):284–293, 2004.

[Rat96] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Empirical Methods in Natural Language Processing*, pages 133–142, 1996.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[Ric79a] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3(4):329–354, October 1979.

[Ric79b] Elaine Alice Rich. *Building and exploiting user models.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1979.

[RT99] E.M. Royer and Chai-Keong Toh. A review of current routing protocols for ad hoc mobile wireless networks. *Personal Communications, IEEE*, 6(2):46–55, Apr 1999.

[SAR10] K. Saruladha, G. Aghila, and S. Raj. A survey of semantic similarity methods for ontology based information retrieval. In *2010 Second International Conference on Machine Learning and Computing (ICMLC)*, pages 297–301, 2010.

[SDW06] Michael Shepherd Singh, Sarabdeep and, Jack Duffy, and Carolyn Watters. An adaptive user profile for filtering news based on a user interest hierarchy. *Proceedings of the American Society for Information Science and Technology*, 43(1):1–21, 2006.

[Sei89] Hamid K. Seifoddini. Single linkage versus average linkage clustering in machine cells formation applications. *Computers & Industrial Engineering*, 16(3):419–426, 1989.

[SEKX98] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, June 1998.

[sev09] British national corpus. http://www.natcorp.ox.ac.uk/, January 2009.

[SGD+02] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy. An analysis of internet content delivery systems. *SIGOPS Oper. Syst. Rev.*, 36(SI):315–327, December 2002.

[Sib73] R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, January 1973.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[Sle85] D. Sleeman. UMFE: a user modelling front-end subsystem. *International Journal of Man-Machine Studies*, 23(1):71–88, July 1985.

[SMB07] Ahu Sieg, Bamshad Mobasher, and Robin D. Burke. Learning ontology-based user profiles: A semantic approach to personalized web search. *IEEE Intelligent Informatics Bulletin*, pages 7–18, 2007.

[SMHM99] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *ACM SIGIR Forum*, 33(1):6–12, September 1999.

[SN09] Sofia Stamou and Alexandros Ntoulas. Search personalization through query and page topical analysis. *User Modeling and User-Adapted Interaction*, 19(1-2):5–33, February 2009.

[Sti89] Stephen M. Stigler. Francis galton's account of the invention of correlation. *Statistical Science*, 4(2):73–79, 1989.

[STZ05] Xuehua Shen, Bin Tan, and Chengxiang Zhai. Implicit user modeling for personalized search. In *Proceedings of CIKM*, 2005.

[SWJS01] Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, and Tefko Saracevic. Searching the web: The public and their queries. *J. Am. Soc. Inf. Sci. Technol.*, 52(3):226–234, February 2001.

[SZF07] Vincent Schickel-Zuber and Boi Faltings. Oss: A semantic similarity function based on hierarchical ontologies. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 551–556, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[TKMS03] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[Tor12] Zeina Torbey. *Increasing Data Availability in Mobile Ad-hoc Networks: A Community-Centric and Resource-Aware Replication Approach.* Phd thesis, INSA DE LYON, September 2012.

[Tve77] Amos Tversky. Features of similarity. *Psychological Reviews*, 84(4):327–352, 1977.

[TZ14] Thanh Tran and Lei Zhang. Keyword query routing. *IEEE Transactions on Knowledge and Data Engineering*, 26(2):363–375, Feb 2014.

[VCJ10] David Vallet, Iván Cantador, and Joemon M. Jose. Personalizing web search with folksonomy-based user and document profiles. In *Proceedings of the 32nd European conference on Advances in Information Retrieval*, ECIR'2010, pages 420–431, Berlin, Heidelberg, 2010. Springer-Verlag.

[VLB+12] M. Viviani, L. Limam, N. Bennani, E. Egyed-Zsigmond, and D. Coquil. Multi-application personalization: Data propagation evaluation on real-life query log. In *6th IEEE International Conference on Digital Ecosystems Technologies (DEST)*, pages 1–6, Campione, Italy, 2012.

[w3c01] How to diff RDF - semantic web standards. http://www.w3.org/2001/sw/wiki/How_to_diff_RDF, 2001.

[WGB12] Ingmar Weber, Venkata Rama Kiran Garimella, and Erik Borra. Mining web query logs to analyze political issues. In *Proceedings of the 3rd Annual ACM Web Science Conference*, WebSci '12, pages 330–334, New York, NY, USA, 2012. ACM.

[WNZ02] Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1):59–81, January 2002.

[WP94] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[XW05] Rui Xu and D. Wunsch,II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[YCB13] Yulian Yang, Sylvie Calabretto, and Lionel Brunie. Centrality-based peer rewiring in semantic overlay networks (S). In *IEEE International Conference on Research Challenges in Information Science*, pages 1–6, May 2013.

[ZN08] Zhiyong Zhang and Olfa Nasraoui. Mining search engine query logs for social filtering-based query recommendation. *Applied Soft Computing*, 8(4):1326–1334, September 2008.

[ZS04] H. Zargayouna and S. Salotti. Mesure de similarité sémantique pour l'indexation de documents semi-structurés. In *12ème Atelier de Raisonnement à Partir de Cas*, Villetaneuse, France, 2004.