Semantic Indexing and Retrieval based on Formal Concept Analysis

Victor Codocedo, Ioanna Lykourentzou, Amedeo Napoli

Semantic Indexing and Retrieval based on Formal Concept Analysis

Victor Codocedo, Ioanna Lykourentzou and Amedeo Napoli

June 29, 2012

Abstract

Semantic indexing and retrieval has become an important research area, as the available amount of information on the Web is growing more and more. In this paper, we introduce an original approach to semantic indexing and retrieval based on Formal Concept Analysis. The concept lattice is used as a semantic index and we propose an original algorithm for traversing the lattice and answering user queries. This framework has been used and evaluated on song datasets.

1 Knowledge discovery in databases (KDD)

1.1 Introduction

Knowledge discovery in databases can be likened to the process of searching for gold in the rivers: the gold nuggets that are researched are knowledge units, and the rivers are the databases under study. Huge volumes of data –and particularly documents– are available, without any intended usage. A fundamental question is to know if there may be something interesting in these data, and to find methods for extracting these "interesting things". The knowledge discovery in databases process –hereafter KDD– consists in processing a huge volume of data in order to extract knowledge units that are non trivial, potentially useful, significant, and reusable. From a global point of view, the KDD process may also be understood as a process turning data into information and then knowledge (see figure 1), considering the following equations [41, 51]:

- Data = signs + syntax.
- Information = data + meaning.
- Knowledge = information (syntax and semantics) + ability to use information.

In addition, the knowledge units extracted by the KDD system must be represented in an adequate representation formalism and then they may be integrated within the ontology to be reused for problem-solving needs in application domains such as agronomy, biology, chemistry, medicine...

1.2 Symbolic Methods in KDD

Knowledge discovery in databases (KDD) consists in processing a large volume of data in order to extract useful and reusable knowledge units from these data. An expert of

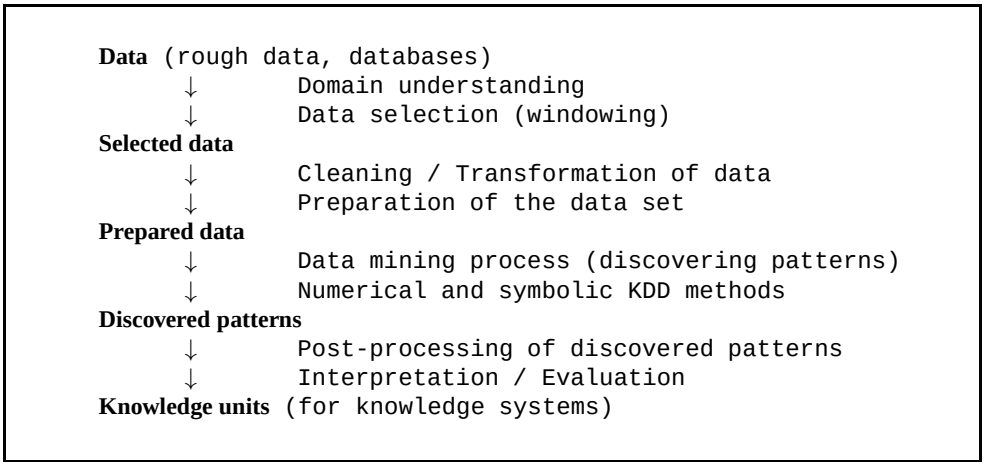

Figure 1: The KDD loop: from rough data to knowledge units. The overall objective process of the KDD process is to select, prepare and extract knowledge units from different sources, and then to represent the extracted knowledge units in adequate knowledge structures.

the data domain, the analyst, is in charge of guiding the extraction process, on the base of his/her objectives and domain knowledge. The extraction process is based on data mining methods returning information units from the data. The analyst selects and interprets a subset of the units for building "models" that may be further interpreted as knowledge units with a certain plausibility. The KDD process is performed with a KDD system based on components such as domain ontologies, data mining modules (either symbolic or numerical), and interfaces for interactions with the system, e.g. editing and visualization.

The KDD process can be considered along three main steps: data preparation, data mining, and interpretation of the extracted units. At each step, domain knowledge, possibly represented within ontologies, can play a substantial role for improving the KDD process [26]. Moreover, data mining methods can be either numeric or symbolic. In this talk, we will mainly focus on the second type and especially itemset search, association rule extraction, and Formal Concept Analysis (and extensions) [32].

The search for frequent itemsets consists in extracting from binary tables itemsets occurring with a support that must be greater than a given threshold [33, 3, 47, 53]. Given a set of objects and a set of properties, an item corresponds to an attribute or a property of an object, and an itemset (a pattern) to a set of items. The support of an itemset corresponds to the proportion of objects owning the itemset, with respect to the whole population of objects. An itemset is frequent if its support is greater than a given frequency threshold σ_{S}: a proportion at least equal to σ_{S} of objects own all items included in the itemset. The search for frequent itemsets is based on monotony constraints (base of the Apriori algorithm [1]). The search of frequent itemsets begins with the search of frequent itemsets of minimal length (or length 1). Then, the frequent itemsets are recorded and combined together to form the candidate itemsets of greater length. The non-frequent itemsets are discarded and all their super-itemsets. The can-

didate itemsets are tested and the process continues in the same way, until no more candidates can be formed.

From frequent itemsets it is possible to generate association rules of the form $\mathtt{A} \longrightarrow \mathtt{B}$ relating an itemset A with an itemset B, that can be interpreted as follows: the objects owning A also own B with a support and a confidence [1, 34]. More precisely, an association rule $\mathtt{A} \longrightarrow \mathtt{B}$ has a support defined as the support of the itemset $\mathtt{A} \cup \mathtt{B}$ and a confidence defined as the quotient $\mathtt{support}(\mathtt{A} \cup \mathtt{B})/\mathtt{support}(\mathtt{A})$ (that can be interpreted as a conditional probability). Then, a rule is valid if its confidence is greater than a confidence threshold $\sigma_{\mathtt{C}}$, and its support is greater than the frequency threshold for itemsets $\sigma_{\mathtt{S}}$ (a valid rule can only be extracted from a frequent itemset).

The numbers of extracted itemsets and rules may be very large, and thus there is a need for pruning the sets of extracted itemsets and rules for ensuring a subsequent interpretation of the extracted units. This is especially true when the interpretation has to be done –and this is usually the case– by the analyst who is in charge of interpreting the results of the KDD process [7].

Actually, the search for itemsets and association rules are related to concept lattices: they correspond to a breadth-first search in the concept lattice associated with the formal context under study.

1.3 Formal Concept Analysis and variations

1.3.1 The basic framework of FCA

The framework of FCA is fully detailed in [15]. FCA starts with a formal context (G, M, I) where G denotes a set of objects, M a set of attributes, or items, and $I \subseteq G \times M$ a binary relation between G and M. The statement $(g, m) \in I$ is interpreted as "the object g has attribute m". Two operators $(\cdot)'$ define a Galois connection between the powersets $(2^G, \subseteq)$ and $(2^M, \subseteq)$, with $A \subseteq G$ and $B \subseteq M$:

$A' = \{m \in M \mid \forall g \in A : gIm\}$ and $B' = \{g \in G \mid \forall m \in B : gIm\}$.

For $A \subseteq G$, $B \subseteq M$, a pair (A, B), such that $A' = B$ and $B' = A$, is called a formal concept. In (A, B), the set A is called the extent and the set B the intent of the concept (A, B). Concepts are partially ordered by $(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2$ $(\Leftrightarrow B_2 \subseteq B_1)$. With respect to this partial order, the set of all formal concepts forms a complete lattice called the concept lattice of (G, M, I). As already mentioned above, natural links exist between between concept lattices, itemsets, and association rules [3, 53, 46].

When one consider non binary contexts, e.g. numerical or interval data, conceptual scaling is often used for binarizing data and for obtaining a binary formal context [15]. Then, a numerical dataset is described by a many-valued context. (G, M, W, I) is a many-valued context where G is a set of objects, M a set of numerical attributes, W a set of values (e.g. numbers), and I a ternary relation defined on the Cartesian product $G \times M \times W$. The fact $(g, m, w) \in I$ or simply $m(g) = w$ means that the object g takes the value w for the attribute m.

Then, classical algorithms can be applied for designing concept lattices from scaled contexts [22]. However, adapted algorithms for designing a concept lattice may be directly applied on more complex data such as numbers, intervals, or graphs [23, 21, 18, 17].

1.3.2 Pattern Structures

Instead of applying discretization leading to space and time computational hardness, one may directly work on original data. A pattern structure is defined as a generalization of a formal context describing complex data [14, 21].

In classical FCA, object descriptions are sets of attributes, which are partially ordered by set inclusion, w.r.t. set intersection: let $P, Q \subseteq M$ two attributes sets, then $P \subseteq Q \Leftrightarrow P \cap Q = P$, and $(M, \subseteq)$, also written $(M, \cap)$, is a partially ordered set of object descriptions. Set intersection $\cap$ behaves as a meet operator and is idempotent, commutative, and associative. A Galois connection can then be defined between the powerset of objects $(2^G, \subseteq)$ and a meet-semi-lattice of descriptions denoted by $(D, \sqcap)$ (standing for $(M, \cap)$). This idea is used to define pattern structures in the framework of FCA as follows.

Formally, let G be a set of objects, let $(D, \sqcap)$ be a meet-semi-lattice of potential object descriptions and let $\delta : G \longrightarrow D$ be a mapping associating each object with its description. Then $(G, (D, \sqcap), \delta)$ is a pattern structure. Elements of D are patterns and are ordered by a subsumption relation $\sqsubseteq$: $\forall c, d \in D$, $c \sqsubseteq d \Longleftrightarrow c \sqcap d = c$. A pattern structure $(G, (D, \sqcap), \delta)$ gives rise to two derivation operators $(\cdot)^{\square}$:

$$A^{\square} = \sqcap_{g \in A} \delta(g), \quad for\ A \subseteq G \quad and \quad d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\}, \quad for\ d \in (D, \sqcap).$$

These operators form a Galois connection between $(2^G, \subseteq)$ and $(D, \sqcap)$. Pattern concepts of $(G, (D, \sqcap), \delta)$ are pairs of the form (A, d), $A \subseteq G$, $d \in (D, \sqcap)$, such that $A^{\square} = d$ and $A = d^{\square}$. For a pattern concept (A, d), d is a pattern intent and is the common description of all objects in A, the pattern extent. When partially ordered by $(A_1, d_1) \leq (A_2, d_2) \Leftrightarrow A_1 \subseteq A_2 \ (\Leftrightarrow d_2 \sqsubseteq d_1)$, the set of all concepts forms a complete lattice called pattern concept lattice. More importantly, the operator $(.)^{\square\square}$ is a closure operator and pattern intents are closed patterns. Existing FCA algorithms (detailed in [22]) can be used with slight modifications to compute pattern structures, in order to extract and classify concepts. Details can be found in [14, 18, 21].

Below, we analyze object descriptions as interval in numerical data. Pattern structures allows to directly extract concepts from data whose object descriptions are partially ordered. Considering a numerical dataset with objects in G and attributes in M, a meet operator $\sqcap$ on interval patterns can be defined as follows. Given two interval patterns $c = \langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$, and $d = \langle [e_i, f_i] \rangle_{i \in \{1,...,|M|\}}$, then: $c \sqcap d = \langle [minimum(a_i, e_i), maximum(b_i, f_i)] \rangle_{i \in \{1,...,|M|\}}$ meaning that a convexification of intervals on each vector dimension is operated. The meet operator induces the following subsumption relation $\sqsubseteq$ on interval patterns: $\langle [a_i, b_i] \rangle \sqsubseteq \langle [c_i, d_i] \rangle \Leftrightarrow [a_i, b_i] \supseteq [c_i, d_i],\ \forall i \in \{1, ..., |M|\}$ where larger intervals are subsumed by smaller intervals.

A numerical dataset with objects G and attributes M can be represented by an interval pattern structure. Let G be a set of objects, $(D, \sqcap)$ a meet-semi-lattice of interval patterns ($|M|$-dimensional interval vectors), and δ a mapping associating to any object $g \in G$ an interval pattern $\delta(g) \in (D, \sqcap)$. The triple $(G, (D, \sqcap), \delta)$ is an interval pattern structure (see examples and details in [21, 17]).

Pattern structures are very useful for building concept lattices where the extents of concepts are composed of "similar objects" with respect to a similarity measure associated to the subsumption relation $\sqsubseteq$ in $(D, \sqcap)$ [17].

1.3.3 Relational Concept Analysis

Relational datasets are composed of a binary tables ($\texttt{objects} \times \texttt{attributes}$) and inter-object relations
($\texttt{objects} \times \texttt{objects}$). Formally, these binary tables introduce a set of objects $\texttt{G}_\texttt{i}$ described by a set of attributes $\texttt{M}_\texttt{i}$, and, as well, a set of relations $\texttt{r}_\texttt{k} \subseteq \texttt{G}_\texttt{i} \times \texttt{G}_\texttt{j}$. Relational datasets arise in a wide range of situations, e.g. Semantic Web applications [43], relational learning and data mining [12], refactoring of UML class models and model-driven development [44].

Relational Concept Analysis (RCA) extends FCA to the processing of relational datasets in a way allowing inter-objects links to be materialized and incorporated into formal concept intents. Links are thus scaled to become relational attributes connecting first objects to concepts and then concepts to concepts as role restrictions do in Description Logics (DL) [2]. The new attributes are complex properties reflecting the relational aspects of a formal concept. They nevertheless abide to the same classical concept formation mechanisms from FCA which means that the relational concept intents can be produced by standard FCA methods. Due to the strong analogy between role restrictions and relational attributes in RCA, formal concepts can be readily translated into a DL-based formalism [40], e.g. for ontology engineering purposes as in [5, 4, 39].

RCA was introduced and detailed in [40]. The data structure is described by a relational context family, composed of a set of contexts $\{\mathcal{K}_\texttt{i}\}$ and a set of binary relations $\{\texttt{r}_\texttt{k}\}$. A relation $\texttt{r}_\texttt{k} \subseteq \texttt{G}_\texttt{j} \times \texttt{G}_\ell$ connects two object sets, a domain $\texttt{G}_\texttt{j}$ ($\texttt{dom}(\texttt{r}_\texttt{k}) = \texttt{G}_\texttt{j}$) and a range $\texttt{G}_\ell$ ($\texttt{ran}(\texttt{r}_\texttt{k}) = \texttt{G}_\ell$). RCA is based on a "relational scaling" mechanism that transforms a relation $\texttt{r}_\texttt{k}$ into a set of relational attributes that are added to the context describing the object set $\texttt{dom}(\texttt{r}_\texttt{k})$. To that end, relational scaling adapts the DL semantics of role restrictions.

For each relation $\texttt{r}_\texttt{k} \subseteq \texttt{O}_\texttt{j} \times \texttt{O}_\ell$, there is an initial lattice for each object set, i.e. $\mathcal{L}_\texttt{j}$ for $\texttt{O}_\texttt{j}$ and $\mathcal{L}_\ell$ for $\texttt{O}_\ell$. For a relation $\texttt{r}_\texttt{k} \subseteq \texttt{O}_\texttt{j} \times \texttt{O}_\ell$, a relational attribute, is associated to an object $\texttt{o} \in \texttt{O}_\texttt{j}$ whenever $\texttt{r}_\texttt{k}(\texttt{o})$ satisfies a given constraint, where $\texttt{r}_\texttt{k}(\texttt{o})$ denotes the set of objects in $\texttt{O}_\ell$ in relation with $\texttt{o}$ through $\texttt{r}_\texttt{k}$. The relational attribute is denoted by $\forall \texttt{r}_\texttt{k}.\texttt{C}$ (universal scaling) when $\texttt{r}_\texttt{k}(\texttt{o}) \subseteq \texttt{extent}(\texttt{C})$ with $\texttt{r}_\texttt{k}(\texttt{o})$ possibly empty. The relational attribute is denoted by $\exists \texttt{r}_\texttt{k}.\texttt{C}$ (existential scaling) when $\texttt{r}_\texttt{k}(\texttt{o}) \cap \texttt{extent}(\texttt{c}) \neq \emptyset$. Other relational scaling operators exist in RCA and follow the classical role restriction semantics in DL.

Actually, RCA is a powerful mechanism for managing relations in the framework of FCA. In CBR, it could be used for example for associating elements of problem statements with elements of problem solutions, an association that was not possible in [11].

1.4 Elements for Discussion

Usually, considering knowledge systems, and CBR systems as well, knowledge units may have two major different origins: explicit knowledge (and cases) can be given by domain experts and implicit knowledge can be extracted from databases of different kinds, e.g. domain data or textual documents. Moreover, a KDD system, as any other knowledge system, improves its performance when it is guided by domain knowledge [26]. Hereafter, some requirements for KDD systems, adapted from [7, 51], are listed for discussion:

- A KDD system is a knowledge system: it should present to the user the underlying domain in an appropriate fashion and rely on domain knowledge (e.g. an

ontology).

- Extending the system knowledge: domain representation should be extensible by addition of new concepts or classes resulting from mining or querying processes. Concepts and their instances must be reusable in queries. The question of extracting cases from data, which have to be made precise, remains open [10].
- Alternative classification and mining tools: it should be possible to define alternative classifications of data, e.g. alternative concept lattices. A set of different classification and mining tools should be available, possibly combining numerical and symbolic methods.
- Support to analysts: analysts should be supported by adequate visualization tools and in the interpretation of extracted units as well, in particular by domain knowledge.
- Monitoring and documenting the system evolution: tools managing versions can be used for monitoring changes in classes or concepts over time. The system should document the different steps of the knowledge discovery process.
- KDD is a flexible process and its results should reflect the plural nature of knowledge, i.e. extracting procedural or declarative knowledge units, and, as well, meta-knowledge units.
- KDD provides knowledge units for extending ontologies, and, reciprocally, knowledge systems and CBR systems can be used to guide and improve KDD.

Finally, the relations between knowledge representation, reasoning, and knowledge discovery with FCA, are explained as follows in [51]. Formal concepts and concept lattices provide a mathematization of real-world concept hierarchies. This yields a mathematical support to human reasoning, especially using the graphical representation of concept lattices. Then, conceptual knowledge discovery, considered as pattern discovery plus knowledge creation, can be guided by the design of concept lattices and a subsequent representation of the formal concepts within a knowledge representation formalism such as description logics. The process can be repeated until a satisfactory knowledge base is obtained.

2 Semantic indexing

Semantic indexing and retrieval is a growing research area [38]. Semantic indexing refers to organizing a set of available information items inside an index according to the semantic relations and concepts that they share, while semantic retrieval refers to searching within this index to identify the items, the context of which matches a given user query [19]. In a certain sense, semantic retrieval is based on flexible and partial matching techniques contrasting exact matching techniques, traditionally used in standard information retrieval.

The organization and retrieval of information based on its *context* can, if effectively carried out, significantly improve the automated understanding of the meaning of the information, as well as to provide users with richer and more meaningful search results. For these reasons, context-based methods of classification and retrieval are applied on multiple domains and types of information, ranging from text-based documents to multimedia content, such as image or video.

One of these domains, interesting for its originality and for the potential added value that it can create, is the domain of song indexing and retrieval. In particular, current song indexing and retrieval systems organize songs based on low or mid-level descriptors [49]. Examples of low-level descriptors include a song's bit-rate, length or pitch and examples of mid-level descriptors include a song's title, artist, genre, year and so on. Using these elements, songs are categorized and retrieved, either based on the user's past known preferences (content-based recommendation) [28], or on the user's interactions with other users (collaborative filtering) [45].

However, songs contain more and richer information than the one described through the aforementioned descriptor elements. Indeed, songs, and especially song lyrics, include information referring to a specific context, event or place. In addition, songs, being art pieces, may be linked to a broader context, such as a historical event, a certain period of time or a cultural idiosyncrasy. Based on this knowledge, a user may be interested in generating compilations of songs based on high-level descriptions, for instance "Songs about the Vietnam War", "Protest Songs", "Songs about peace" etc. As an indication of the growing interest that content-based song indexing receives by users, Wikipedia has so far indexed over 4000 songs under the category "Songs by Theme", which in turn includes a hierarchy of over 250 sub-topics created by Wikipedia contributors [1].

Despite user interest, only a few research works focus on subjects related to context-based song indexing and retrieval. Among the most representative, the work in [48] tries to extract meaningful annotations about songs, using text-mining of expert reviews, targeted user-feedback or social game-based models. The work in [27] performs automated lyrics analysis, using probabilistic latent semantic analysis, to determine artist similarity based on the songs' semantics. To the best of the authors' knowledge, no work so far explores the problem of semantic context-based song indexing and retrieval. An opportunity therefore emerges to use the context-related information of song datasets, revealed through lyrics or through external knowledge resources (such as Wikipedia or Wordnet), in order to provide users with more meaningful search results that will complement the results brought by current song retrieval systems.

Therefore, shifted to the domain of songs, the problem of semantic information indexing and retrieval can be considered as follows:

1. Create a semantic index for organizing songs according to their content (their lyrics).

2. Develop automated ways for searching within this index to retrieve songs semantically linked to a user query.

An example of an indicative question for this problem would be: "Given a set of high-level descriptors provided by the user, find the set of songs whose lyrics are semantically related to these descriptors".

To address this problem, we propose a semantic indexing and retrieval of songs based on Formal Concept Analysis (FCA [15]). In particular, FCA is used to construct a concept lattice, which classifies songs w.r.t. high-level descriptors that they share, i.e the semantic annotations of their lyrics. This concept lattice is then used as a semantic index for songs. In addition, a heuristic search algorithm is proposed, which traverses the lattice in a novel way, in order to match queries based on high-level descriptors to contextualized sets of songs.

[1] `http://en.wikipedia.org/wiki/Category:Songs_by_theme`

Regarding standard approaches of semantic indexing and search, the main contributions of this work lie in the following:

- The use of FCA and a concept lattice as a semantic index.
- The development of a novel algorithm that traverses a concept lattice to retrieve information based on the content of concepts.
- The incorporation of semantic indexing to current song retrieval needs.

An additional characteristic is addressing the problem of semantic indexing and retrieval as a 3-step knowledge discovery process, comprising the following main steps: data preparation, discovery and filtering of the results.

The rest of this document is organized according to the main steps of the process. Firstly, Section 3 proposes the state of the art for positioning the present work. Section 4 presents the data sources and the preparation of the dataset. Section 5 introduces FCA and analyzes its use for semantic indexing and querying. Next, Section 6 presents the evaluation results obtained for our approach. Finally, Section 7 discusses possible extensions of our work and research.

3 Related Work

Song information retrieval is a relatively young field, which started to draw attention in the late 1990s, both in terms of commercial systems and academic research [8].

On the commercial side, many well-known song retrieval systems may be found, including Internet applications (Last.fm[2], Grooveshark[3], Songsterr[4]), desktop applications (iTunes, Zune, Amarok, Songbird, Winamp) or even embedded applications in mobile devices (ipod, android, zone).

Academic approaches have also elaborated on different aspects of the song indexing and retrieval domain. First, a significant number of works focus on song annotation, i.e. describing the song in terms of symbolic, audio or textual metadata (indicatively [13, 20]). Symbolic and audio annotation belong to the so-called low-level descriptors and characterize the song in terms of musical notes and sound features such as bit-rate, length, pitch or melody. Text-based annotations belong to the category of mid-level descriptors and use annotation elements such as artist, genre and title. The annotation task is performed through hand-labeled expert feedback, automated text-mining of music reviews or, more recently, through social song annotation games [48].

Song annotations are then used by music search engines for a number of retrieval tasks. Low-level descriptors are used for example to enable fingerprinting, i.e. searching within song datasets to retrieve exact or similar matches to a particular recording [50] or to facilitate the resolution of copyright or plagiarism issues [49]. Mid-level descriptors are used mainly by recommender systems to suggest new songs to users, either by directly matching the user query to the stored annotations (content-based recommendation [48]), or by combining the latter with additional feedback of a user community (collaborative filtering) [30].

However, the focus on low or mid-level descriptors to retrieve song is not always adequate. Indeed, as a recent survey on user needs in the song domain reveals that the

[2] http://www.lastfm.fr/
[3] http://grooveshark.com/
[4] http://www.songsterr.com/

majority of users expressed a strong interest in contextual song metadata, as these are reflected through lyrics or additional background knowledge [25]. The works focusing on context-based information are a few and mainly focus on modeling the semantics of lyrics to discover patterns of emotions [35, 42] or clusters of mid-level descriptors, such as artist similarity [27] or genre [24, 16]

Finally, regarding the present work, the method of FCA is used in information-retrieval related tasks. Specifically, the work of [36] uses FCA to improve the representation of a document collection by merging it with information from a thesaurus and thus creating a multi-faceted extended context. The work of [9] uses a similar approach to improve the representation of web results. Specifically, in this work FCA is used to create a faceted web browsing system that enables users to query search engines, such as Google, and then organize the results in a concept lattice. However the above works do not use the concept lattice as a semantic index but to create the facets that will facilitate users in browsing the retrieved results.

We introduce FCA to create a semantic index of a given song dataset and a novel method to query this index, thus contributing to both the area of context-based information retrieval on the music domain, as well as to the broader area of FCA-based information indexing and retrieval.

4 Data sources and dataset preparation

The song dataset used in our experiments is based on two main sources of data, namely musiXmatch and WordNet , briefly introduced hereafter.

4.1 musiXmatch

The musiXmatch dataset[5] is a database that contains the lyrics of approximately 700K songs, in the form of bags-of-words. The musiXmatch dataset was recently released (April 2011) as the official lyrics collection of the Million Song Dataset [6]. The musiXmatch dataset is used as the source to provide the lyrics for each song in our dataset, and this choice is made on the basis that the lyrics it provides are: i) already in the format of bag-of-words and ii) already preprocessed using stemming, in order to eliminate morphologically-related word duplications.

4.2 WordNet

WordNet[6] is a semantic dictionary widely used in the domain of computational linguistics and natural language processing [29]. WordNet can be used to relate a word to a set of synonym terms, called synsets, each of which corresponds to one meaning of the specific word. For example the word “concert” corresponds to three synsets inside WordNet: i) “a performance of music by players or singers not involving theatrical staging”, ii)“contrive (a plan) by mutual agreement” and iii)“settle by agreement”.

Synsets inside WordNet are also mutually related through semantic or lexical hierarchies. The most interesting, for our problem, is the semantic hierarchy, which connects synsets in hypernym-hyponym relationships, with hypernyms representing broader and hyponyms narrower semantic terms. The hypernym-hyponym hierarchy of WordNet can therefore be used to retrieve the semantic distance of a pair of given

[5] http://labrosa.ee.columbia.edu/millionsong/musixmatch
[6] http://wordnet.princeton.edu

Table 1: Synsets retrieved for the word "soldier"from Wordnet

Synset	Synonyms	Definition
soldier.n.01	soldier	an enlisted man who serves in an army
soldier.n.02	soldier	a wingless sterile ant or termite having...
soldier.v.01	soldier	serve as a soldier in the military

synsets, i.e. how "far" or "close" these synsets are in terms of semantic meaning. WordNet is used, through its API, as a resource to provide: i) the synsets of every word inside each song's lyrics and ii) the semantic distances among the synsets of each song. A more detailed description of the use Wordnet's API is provided in section 5.1.

For carrying out our experiments, using musiXmatch and Wordnet, we constructed a dataset of 357 songs in total, where each song is represented by its title and lyrics, in the form of bag-of-words, and each word is connected to a set of synsets.

5 Semantic indexing based on FCA

Current song retrieval systems work by exact matching strings of the user query to song attributes, such as title, genre or artist [49]. However, if the user is interested in searching for songs related to an idea or a concept, the above approach may not be sufficient, since it cannot find a direct connection between the content of a song (what a song is about) and its attributes. For instance, in case the user is interested in songs about *War*, current song retrieval systems would retrieve only songs that contain this word in their title, a fact which would, in turn, lead to incomplete or even inexact results. Trying to match the keyword *War* directly to the lyrics would also not be adequate, since this word can be used in several contexts (consider for instance the love song by Blue Nile, titled "War is Love"). Nevertheless, the lyrics of a song are still a valuable source of information regarding its content, able to complement current attributes.

The problem is then defined as finding songs the lyrics of which are *related* to a set of user-provided keywords, through a sufficient "closeness of meaning". Our goal is to construct a semantic index, from a given set of songs and their lyrics and a closeness relation, which will support successfully context-based song retrieval.

In the following, a song s_i is defined as a pair $\{t_i, L_i\}$ where t_i denotes the title and L_i the lyrics of the song in the form of a bag-of-words, as provided by musiXmatch.

5.1 Task 1: Lyrics Annotation

Given a song $s_i = \{t_i, L_i\}$, the first task is obtaining from WordNet the set of synsets that corresponds to its lyrics L_i. For this purpose we query the API of WordNet[7] using each word $w \in L_i$ (which may comprise multiple words). Each retrieved synset has a definition and a set of synonyms. Word w is part of the synonyms. As an example, Table 1 illustrates the synsets retrieved from WordNet for the word "Soldier".

From the above example it can be understood that not every synset retrieved through WordNet is valuable in the context of a song. For instance, in the context of a war, a reference to a "soldier" would be clearly related to the definition of an *enlisted man*

[7] `http://nltk.googlecode.com/svn/trunk/doc/howto/wordnet.html`

who serves in an army and not to the definition of a *soldier ant*. The third definition can also be disregarded since it is associated with a verb.

Therefore, to annotate each song, we firstly need to obtain the WordNet synsets that correspond to its lyrics and secondly to filter the retrieved synsets, in order to keep only those that correspond to the actual context of the song. Thus, each song was associated with a set of WordNet synsets in the following way: For a particular song s_i and its lyrics L_i, we first used the WordNet API to retrieve the list $ss(L_i)$ of synsets that are associated with each word in L_i.

Then, a filtering process took place as follows: a widespread similarity metric, namely the Wu-Palmer Similarity Measure [52] was used to measure the semantic similarity between every pair of synsets in the $ss(L_i)$ set. The Wu-Palmer similarity measure $wp(ss_1, ss_2) = [0, 1], ss_1 \in ss(L_i) \wedge ss_2 \in ss(L_i)$ is provided by the WordNet API and measures semantic similarity using path distance and the difference of levels in the synset tree. Then for each synset ss_j we calculate the average distance with every other synset $ss_k \in ss(L_i)$ as defined in equation 1.

$$avg\_sim(ss_j) = \frac{\sum\limits_{j \neq k} wp(ss_j, ss_k)}{|ss(L_i)|} \tag{1}$$

The synset with the lowest avg_sim is deleted from $ss(L_i)$. The filtering is repeated until we reach to a threshold of 20 most similar elements in $ss(L_i)$, which will be considered to constitute the so-called *semantic core* of the song s_i, denoted as $core(s_i)$. The threshold of 20 synsets was selected heuristically, since it was found to represent the semantic core of the songs, in the specific dataset, in a concise and non redundant way.

The outcome of the filtering process is the set $core(s_i)$, which is considered as the final set of semantic annotations of the song s_i since it refers to a well-defined semantic schema, i.e. WordNet, where each annotation contains a definition and relations with other annotations.

5.2 Task 2: Semantic Index Creation

One particular aspect of this work is to build a semantic index as a concept lattice using Formal Concept Analysis (FCA) following the lines of [31, 9, 36].

We define the formal context $\mathcal{K} = (G, M, I)$, where G is the set of objects containing all the considered songs while the set of attributes M includes all WordNet synsets constituting the semantic cores of the songs in G. The set I maintains the relations between songs and synsets where gIm stands for "song g has synset m in its semantic core". Table 2 shows an example of a formal context created from 11 songs and 6 synsets.

The concept lattice obtained from the context presented in Table 2 is illustrated on Figure 2. The concept lattice is presented in its *reduced notation* where objects (songs) and attributes (synsets) are shown only next to their object/attribute-concept, i.e. the most general concept for attributes which are inherited from higher to lower levels, and the most specific concept for objects which in turn are shared from lower to higher levels.

Table 2: Context example.

	bolshevik.n.01	son.n.01	man.n.03	white.n.01	serviceman.n.01	buddy.n.01
song1		x	x		x	
song6						x
song10		x	x		x	
song14		x	x		x	
song16	x	x	x	x	x	
song18		x	x		x	
song24			x		x	
song27	x	x				x
song32		x	x		x	x
song33			x	x		
song39	x		x	x	x	x

5.3 Task 3: Semantic Index querying

A simple query to the constructed semantic index (i.e. the concept lattice) is a pair $q = (A_q, B_q)$ where A_q denotes an empty extent to be filled and $B_q = \{ss\}$ is a synset to be searched for. Actually, the retrieval is based on two steps. The first one corresponds to "exact matching" (as in [31]) and the second corresponds to "partial matching" based on the closeness relation introduced hereafter. The first step is based on the search for the attribute-concept (A_{ss}, B_{ss}) of attribute ss in the concept lattice, i.e. the most general concept where ss appears in an intent (also denoted by $\mu(ss)$ in [15]). The extent of A_{ss} contains the list of all songs which are directly associated with the synset ss. This *direct answer* constitutes only part of the answer. The second step is related to partial matching based on the closeness relation defined in the "hypothesis of closeness".

Hypothesis of closeness. Two concepts (A_1, B_1) and (A_2, B_2) which are not comparable for $\leq_{\mathcal{K}}$ are said to be *close* iff there exists (A_3, B_3) such that $(A_3, B_3) \leq_{\mathcal{K}} (A_1, B_1)$ and $(A_3, B_3) \leq_{\mathcal{K}} (A_2, B_2)$. Intuitively, this means that (A_1, B_1) and (A_2, B_2) do not subsume each other and that (A_3, B_3) can be either the lower bound or be subsumed by the lower bound $(A_1, B_1) \sqcap (A_2, B_2)$ (where $(A_1, B_1) \sqcap (A_2, B_2)$ denotes the lower bound of (A_1, B_1) and (A_2, B_2). Actually, (A_3, B_3) represents songs related to both (A_1, B_1) and (A_2, B_2): two songs are related if their semantic cores share some elements, which is the case here, as $A_3 \subseteq A_1 \cap A_2$ and $B_3 \subseteq B_1 \cup B_2$. For example, in figure 2, concept 3 is close to 2 because of concept 8, concept 11 is close to concept 12 because of concept 16 and so on.

For a given attribute concept (A_{ss}, B_{ss}), the querying algorithm extracts all *close concepts* (A_i, B_i) of the synset ss, and then moves down the concept lattice, repeating the same extraction level by level. It should be noticed that the original synset query ss is not present in any of the intents of the *close concepts* B_i, this is why we can speak of "partial matching". Every close concept (A_i, B_i) is ranked according to the intersection that its extent has with the extent of the original attribute concept using the

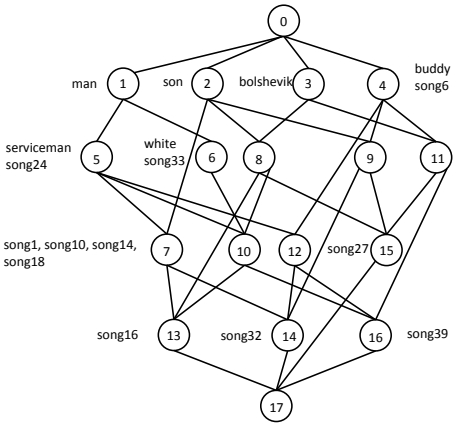

Figure 2: The semantic index as a concept lattice obtained with FCA. Each concept is labelled with a unique identifier.

following metric:

$$rank(A_i, A_{ss}) = \frac{|A_i \cap A_{ss}|}{|A_i|} \tag{2}$$

This metric is two-fold since it allows the detection of concepts (A_i, B_i) which are far from the original concept and share no common objects with the extent of (A_{ss}, B_{ss}) ($A_i \cap A_{ss} = \emptyset$ and rank = 0) and those that are too abstract and describe too many objects ($|A_i| \gg |A_{ss}|$ and rank $\sim$ 0).

Hereafter, we give details on the steps of the algorithm through the use of an example, graphically illustrated in Figure 2. Let us consider a user query for songs related to the synset "bolshevik.n.01" (concept 3 on Figure 2).

1. Find the attribute-concept (A_{ss}, B_{ss}) for the synset $\{ss\}$: concept 3.

2. Find the sub-hierarchy of (A_{ss}, B_{ss}) in the concept lattice, i.e. all concepts subsumed by (A_{ss}, B_{ss}) and order them by levels: concepts 8, 11, 10, 15, 13, 16, 17 (solid arrows in Figure 2).

3. For each concept in this sub-hierarchy, find the super-concepts which are close concepts of (A_{ss}, B_{ss}) (and then for the descendants of (A_{ss}, B_{ss})): concept 2 is close to 3 because of 8, 4 is close to 3 because of 11, 6 is close to 8 because of 10, etc. The final list is ordered by levels: concepts 2, 4, 5, 6, 9, 7, 12, 14 (dashed arrows in Figure 2).

4. Calculate the rank value of each close concept (according to Eq. 2) and sort these close concepts in descending order: concepts 6, 12, 9, 4, 2, 5, 7, 14.

5. The result is composed of the songs in the extent of the attribute-concept (A_{ss}, B_{ss}) and the extents of the close concepts.

The final result, in terms of the retrieved close concepts, their rankings and the songs in their extents, is shown in Table 3.

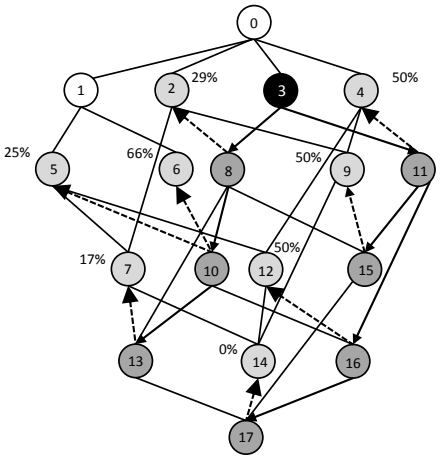

Figure 3: Querying the FCA lattice. The values of the rank metric for each close concept are presented in a percentage form. In this example, concept 3 is the attribute concept, solid arrows show the subhierarchy and dashed arrows show the close concepts of concept 3.

Table 3: The list of retrieved songs in the final order.

Close Concept	$rank$	Songs
3 (attribute-concept)		Song16, Song27, Song39
6	66%	Song33
12	50%	Song32
9	50%	
4	50%	
2	29%	Song1, Song10, Song14, Song18
5	25%	Song24
7	17%	
14	0%	

It is important to note that the basic target of the algorithm proposed above is semantic retrieval. The order in which the algorithm presents the retrieved groups of songs to the user is a "reccomendation decision", which could depend on user preferences and imply the use of a threshold to filter the final list of songs. Semantic retrieval does not necessarily imply the use of a threshold, which is why we present all the songs retrieved.

6 Validation of the approach

As described in the previous section, the lattice is queried using a synset ss (e.g. *bolshevik.n.01*). This synset is directly related to a set of songs ($direct\_answer_{ss}$), i.e. those in the extent of the attribute concept (A_{ss}, B_{ss}) of that synset (in the case of *bolshevik.n.01*, they are the songs 16, 27 and 39). These songs will be retrieved along with the set of songs found in the extents of the *close concepts* ($indirect\_answer_{ss}$)

of (A_{ss}, B_{ss}) (songs 1, 10, 14, 18, 24, 32 and 33).

Regarding the songs in $direct\_answer_{ss}$, we are interested in examining whether our approach can find them if we apply it on a modified formal context where their relations with the synset ss have been eliminated. Of course, in this case, these songs cannot be retrieved as directly related songs, but only as songs found in the extents of *close concepts*. For example, if we eliminate the relation between song 16 and the synset *bolshevik.n.01* we want to know if this song can be retrieved by querying the new lattice using the synset *bolshevik.n.01*.

Regarding the set $indirect\_answer_{ss}$, we are interested in examining how it changes after the application of our approach on the modified formal context since the elimination of a *(song,synset)* relation will affect the structure of the concept lattice and hence the output of the proposed retrieval algorithm. Small variations in the content of this set will indicate robustness.

6.1 Leave-one-out cross validation

To evaluate the above and subsequently the hypothesis of closeness presented in section 5.3 we used and adapted the leave-one-out cross validation (LOOCV) methodology, which is a special type of cross validation [37]. Our adaptation consists of intentionally removing a single *(song,synset)* relation from the original formal context and construct its concept lattice. We call this modified concept lattice a *scenario*. Therefore, each scenario is identified by a pair synset (ss_{scn}) and song (s_{scn}), the relation of which was eliminated for the scenario's construction.

For a given scenario, if song s_{scn} can be retrieved by querying for synset ss_{scn} we mark the scenario as a *success* (e.g. querying for *bolshevik.n.01* and retrieving song 16 for scenario with ss_{scn} = *bolshevik.n.01* and s_{ss} = song 16). It is worth noticing that the *total number of scenarios* for a single synset is determined by the number of songs where the synset appears in (i.e. the songs in the set $direct\_answer(ss)$), since we only eliminate at each time only a single *(song,synset)* relation from the formal context (e.g. for the synset *bolshevik.n.01* we construct 3 scenarios for songs 16, 27 and 39).

For a given synset ss we define a $success\_rate(ss)$ as illustrated in equation 3, where $total\_scenarios(ss)$ refers to the total number of scenarios constructed using synset ss and the songs in $direct\_answer_{ss}$ and $success\_scenarios(ss)$ refers to how many of them were marked as a *success*.

$$success\_rate(ss) = \frac{success\_scenarios(ss)}{total\_scenarios(ss)} \tag{3}$$

High values of *success_rate* indicate that a song can be retrieved for a synset query even if the song is not related to the synset itself.

To evaluate the changes in the set $indirect\_answer_{ss}$, we compare the full set of retrieved songs from each scenario with the respective set of songs retrieved from the original concept lattice. We calculate precision defined in Eq. 4 as the proportion of true positives over the retrieved list of songs in a scenario. Accordingly, we calculate recall, in Eq. 5, as the proportion of true positives over the retrieved list of songs in the original concept lattice, i.e. the concept lattice without any removal. The expression $Ret(scn, ss)$ denotes the total set of songs retrieved from scenario scn ($direct\_answer_{ss} \cup indirect\_answer_{ss}$) querying for synset ss while $original$ denotes the original concept lattice.

$$precision(scn, ss) = \frac{Ret(scn, ss) \cap Ret(original, ss)}{Ret(scn, ss)} \quad (4)$$

$$recall(scn, ss) = \frac{Ret(scn, ss) \cap Ret(original, ss)}{Ret(original, ss)} \quad (5)$$

6.2 Results

For each synset we calculate the mean precision and recall from all their scenarios. From our test set of 357 songs and 1848 synsets we selected 192 synsets and simulated 1027 scenarios (working with approximately 1000 scenarios allows a lower volume of computation and more different trials). Table 4 shows the values of this measures for 10 synsets. For example, it can be seen that synset anteroom.n.01 has relations with 4 songs. A *success_rate* of 1 means that all simulations were successful. Recall of 0.9 and Precision of 0.96 mean that for an elimination of 25% of the relations for a synset (1 over 4 songs), still 90% of the information was retrieved and 96% of the information was correct. There is a positive relation between the number of songs in which the synset appears and the *success_rate* measure. This is not strange since synsets appearing in a few songs will be in less concepts in the lattice and hence the simulation affects them in the worst manner. For example, for the synset bar.n.03, the elimination of one relation with a song leads to the elimination of 50% of its relations (1/2), while for the synset battle.n.01 the elimination of one relation with a song leads to the elimination of only 5% of its relations (1/20).

Figure 4 shows the distribution of *success_rate*, recall and precision (in the interval of $[0, 1]$ in axis y) over the number of songs where synsets appear in (in axis x). The *success_rate* maintains a growing tendency showing that better results are obtained with synsets which appear in a greater number of songs. In a wider sense, precision and recall maintain their values over 70% over all the samples. This is especially important in values of songs per synset below 5 since losing a single connection could disconnect songs more significantly. In the case of the first point (2.5 songs per synset) losing one connection means losing 40% of the information available, however over 70% of the original set of songs is retrieved.

It should be noted that a certain degree of bias, caused by the inclusion of the directly related songs in the measures of precision/recall, is to be expected. That is, given that for each scenario we are eliminating only one *(song, synset)* relation, the remaining directly related songs will be present in both sets retrieved when querying the scenario and the original concept lattice. Therefore, the precision/recall measures are meant to be used as a means of examining how the set of close concepts is affected for each synset, and they should not be considered as a medium of comparison with other information retrieval approaches.

Finally, even if more experiments have to be completed, we can conclude that the *hypothesis of closeness* is valuable and can be used to exploit the use of a concept lattice as a semantic index to retrieve objects not directly related to a query.

7 Discussion, extension and conclusion

Semantic indexing and retrieval based on FCA as introduced above allow us to retrieve a set of songs w.r.t. the content of their lyrics. However, lyrics do not always depict the

Table 4: The results of the simulations.

synset	songs	success_rate	recall	precision
anteroom.n.01	4	1.0	0.9	0.964
bustle.n.01	3	0.333	0.564	0.611
ambition.n.01	9	0.888	0.888	0.938
child.n.03	13	0.923	0.945	0.982
arrest.n.02	4	0.25	0.75	0.807
battle.n.01	20	0.9	0.956	0.989
champion.n.02	2	0.0	0.083	1.0
better.n.03	3	0.0	0.641	0.694
attack.n.01	2	1.0	0.730	0.791
bar.n.03	2	0.0	0.083	1.0

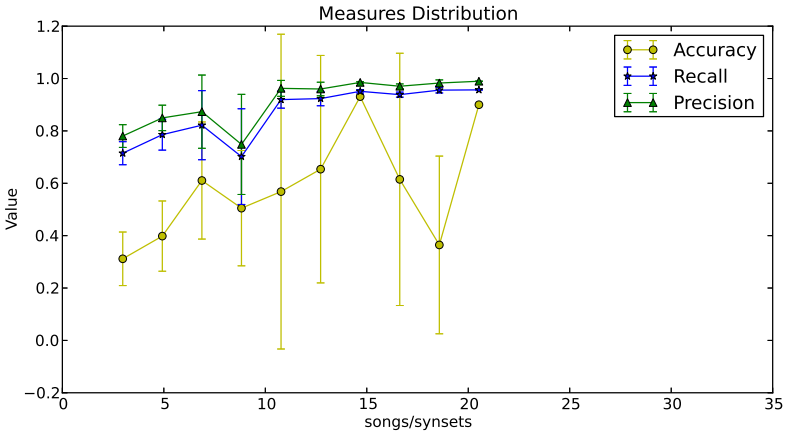

Figure 4: Distribution of measures over songs per synset.

full context of a song. As an example, in the dataset, none of the songs that are known to be about the historical event of the Vietnam war actually contains a reference to the synset "vietnam_war.n.01". This may happen because of two main reasons. Firstly, a song may not refer directly to its full context, but to the circumstances that surround it (a song may not refer directly to Vietnam, but to the story of a soldier). Secondly, a synset important to describe the full context of a song, such as the "Vietnam" synset, may be ignored during synset filtering, because the specific synset was not found to belong to the semantic core of the song. Despite not being directly present in the lyrics, this information is necessary to complete the full context of a song and is extremely valuable. It can help to gain a deeper understanding of the song data and thus to provide the user with richer retrieval results with regard to the context of songs.

There are many sources that could be used to obtain this additional information about songs. A very interesting one is DBpedia, a large-scale effort to provide semantics to the content of Wikipedia. DBpedia categorizes songs in contextual "topics" (e.g. "songs about the Vietnam War") which can be used as *categorical knowledge* to enrich our understanding of the meaning of a song. For taking advantage of categorical knowledge, it is possible to extend the proposed FCA-based approach with *Relational Concept Analysis* [40], which allows to take into account relations between objects in the framework of FCA. It becomes then possible to create a semantic index as a "relational concept lattice", where songs are related not only through their lyrics but also through their categories. In the RCA framework, it is then possible to search for a set of songs which are indexed under the same or related categories (a category can be more or less general in the hierarchy of categories of DBpedia). At present, first experiments were made with RCA. The retrieval process shows similar performances as with FCA but provides alternative lists of results. These experiments have still to be completed and analyzed.

Concluding, in this document we propose a novel contribution to the field of semantic indexing and retrieval, which is based on Formal Concept Analysis. Specifically, we use the concept lattice as a semantic index and propose a novel algorithm to traverse it in order to match user queries with semantically relevant information items. The approach was tested on a song dataset and the obtained results show good capabilities.

8 Acknowledgements

The work of Ioanna Lykourentzou in this research is supported by the National Research Fund, Luxembourg, and cofunded under the Marie Curie Actions of the European Commission (FP7-COFUND). The work of Victor Codocedo and Amedeo Napoli in this research is funded by Quaero project `http://www.quaero.org`. This technical report was presented as a contractual deliverable for Quaero project under the title "Formal representation of knowledge for guiding recommendations".

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press / MIT Press, 1996.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Exploration Newsletter*, 2(2):66–75, 2000.

[4] R. Bendaoud, A. Napoli, and Y. Toussaint. Formal concept analysis: A unified framework for building and refining ontologies. In A. Gangemi and J. Euzenat, editors, *Knowledge Engineering: Practice and Patterns – Proceedings of the 16th International Conference EKAW*, LNCS 5268, pages 156–171, 2008.

[5] R. Bendaoud, Y. Toussaint, and A. Napoli. Pactole: A methodology and a system for semi-automatically enriching an ontology from a collection of texts. In P. Eklund and O. Haemmerl, editors, *Proceedings of the 16th International Conference on Conceptual Structures (ICCS 2008)*, LNCS 5113, pages 203–216, 2008.

[6] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval*, 2011.

[7] R.J. Brachman and T. Anand. The Process of Knowledge Discovery in Databases. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 37–57. AAAI Press / MIT Press, 1996.

[8] Donald Byrd and Tim Crawford. Problems of music information retrieval in the real world. In *Information Processing and Management*, pages 249–272, 2002.

[9] Claudio Carpineto, Giovanni Romano, and Fondazione Ugo Bordoni. Exploiting the potential of concept lattices for information retrieval with credo. *Journal of Universal Computer Science*, 10:985–1013, 2004.

[10] M. d'Aquin, F. Badra, S. Lafrogne, J. Lieber, A. Napoli, and L. Szathmary. Case base mining for adaptation knowledge acquisition. In M.M. Veloso, editor, *IJCAI 2007*, pages 750–755. Morgan Kaufman, 2007.

[11] B. Diaz-Agudo and P.A. Gonzales-Calero. Formal Concept Analysis as a support technique for CBR. *Knowledge-Based Systems*, 14:163–171, 2001.

[12] S. Dzeroski and N. Lavrac, editors. *Relational Data Mining*. Springer, Berlin, 2001.

[13] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A survey of audio-based music classification and annotation. *Multimedia, IEEE Transactions on*, 13(2):303 –319, april 2011.

[14] B. Ganter and S.O. Kuznetsov. Pattern structures and their projections. In H.S. Delugach and G. Stumme, editors, *Conceptual Structures: Broadening the Base, Proceedings of the 9th International Conference on Conceptual Structures (ICCS 2001)*, LNCS 2120, pages 129–142. Springer, 2001.

[15] Benrhard Ganter and Rudoph Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.

[16] Xiao Hu, J. Stephen Downie, and Andreas F. Ehmann. Lyric Text Mining in Music Mood Classification. In *10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 411–416, 2009.

[17] M. Kaytoue, Z. Assaghir, N. Messai, and A. Napoli. Two Complementary Classication Methods for Designing a Concept Lattice from Interval Data. In S. Link and H. Prade, editors, *Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, LNCS 5956, pages 345–362. Springer, 2010.

[18] M. Kaytoue-Uberall, S. Duplessis, S. Kuznetsov, and A. Napoli. Two FCA-Based Methods for Mining Gene Expression Data. In S. Ferr and S. Rudolf, editors, *Proceedings of the 7th International Conference on Formal Concept Analysis (ICFCA 2009)*, LNAI 5548, pages 251–266. Springer, 2009.

[19] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49 – 79, 2004.

[20] Mika Kuuskankare and Mikael Laurson. Mir in enp - rule-based music information retrieval from symbolic music notation. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, pages 699–704. International Society for Music Information Retrieval, 2009.

[21] S.O. Kuznetsov. Pattern structures for analyzing complex data. In H. Sakai, M.K. Chakraborty, A.E. Hassanien, D. Slezak, and W. Zhu, editors, *Proceedings of the 12th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, volume 5908 of *LNCS 5908*, pages 33–44. Springer, 2009.

[22] S.O. Kuznetsov and S.A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2/3):189–216, 2002.

[23] S.O. Kuznetsov and M.V. Samokhin. Learning closed sets of labeled graphs for chemical applications. In S. Kramer and B. Pfahringer, editors, *Proceedings of 15th International Conference on Inductive Logic Programming (ILP 2005)*, LNCS 3625, pages 190–208. Springer, 2005.

[24] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. Multimodal music mood classification using audio and lyrics. In *In Proceedings of the 7th International Conference on Machine Learning and Applications (ICMLA)*, 2008.

[25] Jin Ha Lee and J. Stephen Downie. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *In ISMIR Proceedings*, pages 441–446, 2004.

[26] J. Lieber, A. Napoli, L. Szathmary, and Y. Toussaint. First Elements on Knowledge Discovery guided by Domain Knowledge (KDDK). In S. Ben Yahia, E. Mephu Nguifo, and R. Belohlavek, editors, *Concept Lattices and Their Applications (CLA 06)*, LNAI 4923, pages 22–41. Springer, 2008.

[27] B. Logan, A. Kositsky, and P. Moreno. Semantic analysis of song lyrics. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 2, pages 827 –830 Vol.2, 2004.

[28] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[29] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[30] Leandro Balby Marinho and Lars Schmidt-Thieme. Collaborative tag recommendations. In *Data Analysis, Machine Learning and Applications*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 533–540. Springer, 2008.

[31] Nizar Messai, Marie-Dominique Devignes, Amedeo Napoli, and Malika Smal-Tabbone. Querying a bioinformatic data sources registry with concept lattices. In *Proceedings of ICCS 2005*, pages 323–336. LNCS 3596, Springer, 2005.

[32] A. Napoli. A smooth introduction to symbolic methods for knowledge discovery. In H. Cohen and C. Lefebvre, editors, *Handbook of Categorization in Cognitive Science*, pages 913–933. Elsevier, Amsterdam, 2005.

[33] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In C. Beeri and P. Buneman, editors, *Database Theory - ICDT'99 Proceedings, 7th International Conference, Jerusalem, Israel*, LNCS 1540, pages 398–416. Springer, 1999.

[34] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Pruning closed itemset lattices for association rules. *International Journal of Information Systems*, 24(1):25–46, 1999.

[35] Michael Kai Petersen, Lars Kai Hansen, and Andrius Butkus. Computer music modeling and retrieval. genesis of meaning in sound and music. chapter Semantic Contours in Tracks Based on Emotional Tags, pages 45–66. Springer, 2009.

[36] Uta Priss. Lattice-based information retrieval. *Knowledge Organization*, 27:132–142, 2000.

[37] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009.

[38] Fuji Ren and David B. Bracewell. Advanced information retrieval. *Electronic Notes in Theoretical Computer Science*, 225(0):303 – 317, 2009.

[39] M. Rouane-Hacene, A. Napoli, P. Valtchev, Y. Toussaint, and R. Bendaoud. Ontology Learning from Text using Relational Concept Analysis. In P. Kropf, M. Benyoucef, and H. Mili, editors, *International Conference on eTechnologies (MCETECH 08)*, pages 154–163. IEEE Computer Society, 2008.

[40] Mohamed Rouane-Hacene, Marianne Huchard, Amedeo Napoli, and Petko Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In *Proceedings of ICFCA 2007*, pages 51–65. LNAI 4390, Springer, 2007.

[41] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. van de Velde, and B. Wielinga. *Knowledge Engineering and Management: the CommonKADS Methodoloy*. The MIT Press, 1999.

[42] Govind Sharma and M. Narasimha Murty. Mining sentiments from songs using latent dirichlet allocation. In *Proceedings of the 10th international conference on Advances in intelligent data analysis X*, IDA'11, pages 328–339. Springer, 2011.

[43] S. Staab and R. Studer, editors. *Handbook on Ontologies (Second Edition)*. Springer, Berlin, 2009.

[44] T. Stahl, M. Voelter, and K. Czarnecki. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006.

[45] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, pages 4:2–4:2, 2009.

[46] L. Szathmary, P. Valtchev, A. Napoli, and R. Godin. Constructing iceberg lattices from frequent closures using generators. In J.-F. Boulicaut, M.R. Berthod, and T. Horváth, editors, *Discovery Science*, LNCS 5255, pages 136–147. Springer, 2008.

[47] L. Szathmary, P. Valtchev, A. Napoli, and R. Godin. Efficient Vertical Mining of Frequent Closures and Generators. In N. Adams, J.-F. Boulicaut, C. Robardet, and A. Siebes, editors, *Proceedings of the 8th International Symposium on Intelligent Data Analysis (IDA-2009)*, LNCS 5772, pages 393–404. Springer, 2009.

[48] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):467 –476, 2008.

[49] Rainer Typke, Frans Wiering, and Remco C. Veltkamp. A survey of music information retrieval systems. In *Proceedings of the 6th International Conference on Music Information Retrieval*, 2005.

[50] Avery Wang. The shazam music recognition service. *Communications of the ACM - Music information retrieval*, 49:44–48, 2006.

[51] R. Wille. Why can concept lattices support knowledge discovery in databases? *Journal of Experimental & Theoretical Artificial Intelligence*, 14(2/3):81–92, 2002.

[52] 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.

[53] M.J. Zaki. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):462–478, 2005.