FaSet: A Set Theory Model for Faceted Search

Dario Bonino, Fulvio Corno, Laura Farinetti
Dipartimento di Automatica ed Informatica
Politecnico di Torino, 10129 - Torino, Italy
Email: {dario.bonino, fulvio.corno, laura.farinetti}@polito.it

Abstract

Faceted classification is a technique originated and refined in the library science field, that recently gained a lot of attention for creating efficient search interfaces for web databases. Faceted search requires the definition of a formal representation model, a search algorithm and a responsive user interface. This paper proposes FaSet, a representation model and search algorithm supporting the implementation of faceted search engines. FaSet relies on set theory, and strikes a good balance between expressive power and ease of implementation on web architectures. The paper presents the formal definition of the model, search and ranking algorithms, and a relational mapping of data structures and algorithms that enables its efficient implementation.

1. Introduction

Faceted classification and search systems were firstly investigated in early 60’s, originating from the Library Science research field [1]. Nowadays, they are gaining a new momentum thanks to the increasing need for effective management of overwhelming quantities of information exposed to web users everyday. Faceted search interfaces are proving to be effective and usable, with sensible advantages over traditional keyword-based search engines [2]. The strength of this re-discovered interaction paradigm lies in the ability to assign multiple classifications to the same object, supporting many different and interwoven ways to access the same information unit.

Although the validity of the faceted model is known from decades, the availability of new interaction technologies such as the Web 2.0 and the ever increasing amounts of informations directly accessible through the Internet has fostered new, intense research on the field, resulting in several interesting approaches. Research efforts are directed in two main areas: user interfaces and formal models. The former attracts most of researchers, while the latter had fewer contributions and, apart from proposals for integrating Semantic Web technologies and Faceted Search [3], [4], very little significant innovation can be found in the last 10 years. This inequality of effort distribution is clearly reflected by state of art systems such as Flamenco [2], Castanet [5], Facet Folders [6], Stuff I’ve Seen [7], Elastic Lists [8] where the main concern involves Usability and Human-Computer Interaction rather than navigation and retrieval models. While recognizing the importance of user centered design, there is a sensible lack of formal modeling of faceted search systems.

This paper presents the FaSet model supporting the generation of faceted user interfaces in a versatile, efficient, and flexible way. The fundamental concepts of other faceted representation models (see Section 4 for an overview), such as hierarchical facets, multiple classification, query refinement, etc., are supported in our model. The main characteristics of FaSet lay in striking a balance between the formal complexity of the model, and its suitability for integration in web applications. In fact, FaSet relies on a sound formalization, based on simple set theory, and allows all the required representation and search flexibility. At the same time, FaSet can be directly mapped to relational database operations, thus guaranteeing a painless integration into new or existing web applications, immediately compatible with current web architectures.

The two aspects of FaSet are described in the two main sections of the paper: Section 2 describes the set-theoretical formalization for the general hierarchical faceted classification and search problem, while Section 3 shows the direct mapping of FaSet definitions and operators into a relational database implementation. Related works, and their relationship with FaSet, are outlined in Section 4, while Section 5 concludes the paper.

2. The FaSet Logic Model

This section describes the formalization adopted for the FaSet information retrieval model. Set theory was chosen as it offers the necessary flexibility and is sufficiently simple to be mapped to relational database operations.

2.1. Definitions

Definition 1 (Facet) A facet $F$ is an independent point of view for representing the content of a resource.
be used to describe the resource, according to the specified point of view. The elements of \( F \) never need to be explicitly mentioned, since \( F \) just acts as a “container” of the hierarchy of interesting categories. The concept of independence of the point of view implies that, when multiple facets are used, they must be disjoint. As a consequence:

**Definition 2 (Facet)** A facet \( F \) is a set of items. When more facets are defined, they are disjoint: \( F_a \cap F_b = \emptyset \).

**Definition 3 (Facet space)** Given a set of facets \( F_a, F_b, \ldots \), the facet space \( U \) is the universe set defined by the Cartesian product of all facets:

\[
U = F_a \times F_b \times F_c \times \ldots
\]

From this point on, an (arbitrary) ordering of facets is defined, and coherently adopted in all successive definitions and algorithms.

**Definition 4 (Focus)** A focus \( L \) (plural: foci) of a facet \( F \) is a named subset of \( F \): \( L \subseteq F \).

In other terms, faceted classification associates a resource to a subset of the universe represented by the facet \( F \). We restrict our attention to named subsets, only, that represent the interesting concepts or terms in our classification. The numeric naming of the foci, defined below, is used to represent a nesting hierarchy, such as the one represented in the upper part of Figure 1.

**Definition 5 (Focus name)** The “name” of a focus is a variable-length, possibly empty, list of integer numbers (called indexes): \( L(i, j, k, \ldots) \).

As an example, \( L() \) is a valid focus name (empty list of indexes), and \( L(3) \) and \( L(3, 2) \) are also valid distinct names. For brevity, we use \( i, j, k, \ldots \) for denoting indexes, and \( p \) as any (possibly empty) sub-sequence (usually a prefix) of the list of integers.

**Definition 6 (Hierarchical Foci)** Different foci in the same facet are organized as a tree hierarchy. The hierarchy is induced by the integer numbers in the list \( L(...) \). The hierarchy depth is the length of the list. By definition, \( L() \equiv F \).

The interpretation of the tree hierarchy, as induced by the integer lists, is clarified in the two following corollaries.

**Corollary 1 (Disjointness)** All foci whose names share an initial prefix are disjoint if the first index after the common prefix is different:

\[
\forall i, j, \forall p : i \neq j \Rightarrow L(p, i, \ldots) \cap L(p, j, \ldots) = \emptyset
\]

**Corollary 2 (Nesting)** Given a focus with prefix \( p \), all foci represented by a longer list starting with the prefix \( p \) are subsets of the first focus:

\[
\forall p, j : L(p, j, \ldots) \subseteq L(p)
\]

### 2.2. Resource modeling

The set theoretical model of facets and foci applies uniformly to resources (documents) and queries: they are both modeled by representing the list of relevant foci, as detailed in the following set of definitions.

**Definition 7 (Classification)** The classification of a resource \( r \) with respect to a (single) facet \( F \), denoted\(^1\) as \( r \perp F \), is the subset of \( F \) that is relevant to the resource \( r \).

Such a definition is highly general and allows for any subset of \( F \) to be used for describing \( r \). In a faceted search system, we are only interested in classifications that can be expressed as combinations (union) of defined foci. We call them sharp classifications.

**Definition 8 (Sharp classification)** A classification \( r \perp F \) is called sharp when it can be expressed as a union of foci:

\[
\exists P : \forall r \perp F = \bigcup_{p \in P} L(p)
\]

where \( P \) is a set of focus names.

A sharp classification may be simply represented by the corresponding set of focus names \( P \). Example: \( P = \{ ⟨1, 2⟩, ⟨2⟩, ⟨3, 1, 2⟩, ⟨3, 1, 3⟩ \} \). A special case is the “root” classification \( P = \{ ⟨⟩ \} \), representing the whole facet \( F \).

Without loss of generality, we may assume that \( P \) is prefix-minimal, i.e., if \( ⟨p⟩ \) is present in \( P \), then \( ⟨p, q⟩ \) is absent, for any non-empty \( q \).

**Definition 9 (Multi-dimensional classification)** The multi-dimensional [sharp] classification of a resource \( r \) with respect to a facet space \( U = F_a \times F_b \times \ldots \) is defined as the cartesian product of the [sharp] classification of \( r \) with respect to each of the facets.

\[
r \perp U \overset{\text{def}}{=} (r \perp F_a) \times (r \perp F_b) \times (r \perp F_c) \times \ldots
\]

By combining (1) with (2), we obtain

\[
r \perp U = \bigcup_{p \in P_a} L_a(p) \times \bigcup_{p \in P_b} L_b(p) \times \bigcup_{p \in P_c} L_c(p) \times \ldots
\]

that implies that a multi-dimensional sharp classification may be simply represented by the list, one for each facet \( F_x \), of the sets of focus names \( P_x \). As an example, \( P_U = (P_a, P_b) = (\{ ⟨1, 2⟩, ⟨2⟩ \}, \{ ⟨3, 1, 2⟩, ⟨3, 1, 3⟩ \}) \). This simple representation allows a flexible and efficient implementation in relational databases.

### 2.3. Facet types in search applications

Although this paper is not concerned with user interfaces, we should take into account application requirements, and

1. For brevity of notation, in this paper we redefine the symbol \( \perp \) to denote classification
Table 1. Facet types and examples

<table>
<thead>
<tr>
<th>Facet type</th>
<th>Flat</th>
<th>Hierarchical</th>
<th>Nested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single value</td>
<td>Type SF</td>
<td>Type SH</td>
<td>Type SN</td>
</tr>
<tr>
<td>Nobel Prize category</td>
<td>Geographical region</td>
<td>PEGI rating system</td>
<td></td>
</tr>
<tr>
<td>Multiple values</td>
<td>Type MF</td>
<td>Type MH</td>
<td>-</td>
</tr>
<tr>
<td>Language</td>
<td>Subject</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Sample FaSet representation

2.4. Search algorithm

The search algorithm, starting from a query \( q \), represented by the list of focus names \( P_U(q) \), evaluates the classified resources, in two successive steps:

1) **Filtering:** determines which resources \( r \) are compatible with the query \( q \). In this paper we propose a stateless filtering model, where the result is uniquely determined by the current value of the query, thus allowing an easier cognitive model and the possibility of horizontal navigation in the query space.

2) **Ranking:** determines a suitable sort order of the filtered resources. Unlike most other faceted search approaches, where ranking is neglected or is inherited from a bootstrap full-text search, we explicitly model ranking as a measure of matching quality between the expressed query, and the underlying classification.

2.4.1. Filtering.

**Definition 10 (Filtering)** Given a query \( q \) and a resource \( r \), under the multi-dimensional classification \( U \), the resource is compatible with the query iff:

\[
C = (q \perp U) \cap (r \perp U) \neq \emptyset
\]  

In sharp multi-dimensional classifications, the intersection \( C \) in Definition 10 can be computed by applying (3) to \( q \) and \( r \):

\[
C = \left( \bigcup_{p \in P_a(q)} L_a(p) \times \bigcup_{p \in P_b(q)} L_b(p) \times \ldots \right) \cap \left( \bigcup_{p \in P_a(r)} L_a(p) \times \bigcup_{r \in P_b(r)} L_b(p) \times \ldots \right)
\]

The last expression can be further simplified, by distributing the cross product over the intersection, yielding:

\[
C = \left[ \bigcup_{p \in P_a(q)} L_a(p) \right] \cap \left[ \bigcup_{r \in P_a(r)} L_a(p) \right] \times \ldots
\]
The hierarchical structure of check whether an index list in reduces to a simple comparison of integer lists where we therefore the condition

\[ P \subset L(2) \]

\[ P \not\subset L(2) \]

\[ (1, 3) \]

\[ (2) \]

\[ \emptyset \]

\[ \emptyset \]

Figure 1. Sample filtering computation

A Cartesian product is null iff one of its factors is empty, therefore the condition \( C \neq \emptyset \) in equation (4) is equivalent to:

\[
\begin{cases}
    \left( \bigcup_{p \in F_a(q)} L_a(p) \right) \cap \left( \bigcup_{p \in F_a(r)} L_a(p) \right) \neq \emptyset \\
    \left( \bigcup_{p \in F_a(q)} L_b(p) \right) \cap \left( \bigcup_{p \in F_a(r)} L_b(p) \right) \neq \emptyset \\
    \ldots
\end{cases}
\]

The intersections can be checked efficiently by exploiting the hierarchical structure of \( L_a(p) \). In fact, the computation reduces to a simple comparison of integer lists where we check whether an index list in \( F_a(q) \) is equal to, or is a prefix of, a list in \( F_a(r) \), or vice-versa. This is formalized in the following definition.

**Definition 11 (Prefix compatibility)** Given two foci \( L_a(p_1) \) and \( L_a(p_2) \) of a same facet \( F_a \), we say that the foci are prefix-compatible, and we write \( L_a(p_1) \preceq L_a(p_2) \) iff one of the following properties holds:

- \( p_1 = p_2 \), or
- \( p_1 \) is a prefix of \( p_2 \), i.e., \( p_2 = (p_1, p') \) for some \( p' \), or
- \( p_2 \) is a prefix of \( p_1 \), i.e., \( p_1 = (p_2, p') \) for some \( p' \)

The above analysis leads to a simple and effective implementation of the filtering problem (Section 3.1).

**Theorem 1 (Filtering)** The computation of equation (4) in Definition 10 is equivalent to checking equation (5) for each facet \( F_a \).

\[
\exists p_q \in P_a(q), \exists p_r \in P_a(r) : L_a(p_q) \preceq L_a(p_r) \quad (5)
\]

**Example.** To better clarify the filtering computation let us analyze the example classification reported in Figure 1. In this case we observe that:

- the query is \( q = \{ (1, 3), (2) \} \);
- resource \( r_1 = \{ (1) \} \) is compatible with \( q \) because \( 1 \) is a prefix of \( (1, 3) \);
- resource \( r_2 = \{ (2, 2) \} \) is compatible with \( q \) because \( 2 \) is a prefix of \( (2, 2) \);
- resource \( r_3 = \{ (1, 2), (1, 3) \} \) is compatible with \( q \) because \( (1, 3) \) is equal to \( (1, 3) \);
- on the other hand, resource \( r_4 = \{ (1, 1), (1, 2) \} \) is not compatible with \( q \).

**2.4.2. Ranking.** The filtering algorithm described above includes all resources that are compatible with the given query in the result set. Filtering, by definition, is a Boolean operation, and generates an unordered set of interesting resources. Classically, the ordering of relevant results has been neglected, and the many powerful approaches available in the Information Retrieval (IR) field are not used.

The ordering of the results may be improved by taking into account the degree of similarity of each result with the query, measured by a suitable metric, thus borrowing the concept of similarity metric from the IR field. In this section, we introduce a notion of relevance of a match that allows us to define an ordering (relevance ranking) of compatible results.

The notion of similarity that is proposed in this paper assumes that the relevance of a match increases if (a) a higher number of foci are prefix-compatible and/or (b) the match is discovered at higher levels of detail (i.e., deeper in the facet hierarchy), because that means more specific information has been found. The similarity measure we propose therefore increases with the number of matches and with their depth.

The approach we propose for defining a similarity metric is based on 3 phases:

1. Start with measuring the Focus Similarity \( S \) of individual foci (Definition 13).
2. Aggregate focus-level similarities into facet-level similarities \( S \) by combining their weights (Definition 14), and normalize them (\( S \), Definition 15).
3. Intersect the similarities of each facet \( S \) into an overall similarity \( S \) (Definition 16), by evaluating the joint similarity with respect to the Cartesian space of all facets.

**Definition 12 (Focus Depth)** Given a focus \( L_a(p) \) of a same facet \( F_a \), we define the depth \( D(L_a(p)) \) of the focus as the number of hierarchy levels that compose the name of the focus.

This simple definition implies that \( D(L_a(\{ \}) = 0 \), \( D(L_a(2)) = 1 \), \( D(L_a(3, 1)) = 2 \), \( D(L_a(2, 4, 1)) = 3 \), etc.

**Definition 13 (Focus Similarity)** Given two foci \( L_a(p_1) \) and \( L_a(p_2) \) of a same facet \( F_a \), we define their focus similarity \( S(L_a(p_1), L_a(p_2)) \) as:

- \( 0 \), if \( L_a(p_1) \not\preceq L_a(p_2) \)
- \( W(\min\{D(L_a(p_1)), D(L_a(p_2))\}) \), if \( L_a(p_1) \preceq L_a(p_1) \)

where \( W \) is a vector of increasing weights that maps each depth level to a real number into the interval \((0, 1]\) (e.g., as in Table 3).

The definition and computation of Focus Similarity for some example couples of foci is illustrated in Table 4.

The values of focus similarities are defined on each matching focus, and must now be aggregated at the facet level.
This aggregation behaves like a union operator, and adds up the similarity values of individual foci. This sum must be restricted to the interval [0, 1] therefore the “algebraic sum” (or “probabilistic OR”) operator adopted in fuzzy systems [9] is used here: \( a \oplus b = a + b - a \cdot b \). If several foci of the resource match the same focus of the query, the deepest matching prefix is used, only (see the max operator).

**Definition 14 (In-Facet Similarity)** Given a query \( q \) and a resource \( d \), we define the similarity along facet \( F_a \), \( S_a(q,r) \) as:

\[
S_a(q,r) = \bigoplus_{p.q \in (q \land F_a)} \max_{p.r \in (r \land F_a)} S(L_a\langle q \rangle, L_a\langle r \rangle)
\]

Since different facets may have varying numbers of foci, and the query itself can be represented by a varying number of foci, the above similarity metric is not normalized. To combine the values of different facets, we must first normalize them over their maximum value. Due to the increasing nature of vector \( W \), the maximum value is equal to the similarity of the query with itself.

**Definition 15 (Normalized In-Facet Similarity)** Given a query \( q \) and a resource \( d \), we define the normalized similarity along facet \( F_a \), \( S_a^*(q,r) \) as:

\[
S_a^*(q,r) = \frac{S_a(q,r)}{S_a(q,q)}
\]

This normalized measure is always in the interval [0, 1].

We can now combine the effect of the various facets, by “intersecting” their similarity values with the “probabilistic AND” operator [9], defined as \( a \otimes b = a \cdot b \).

**Definition 16 (Similarity)** Given a query \( q \) and a resource \( r \), we define the similarity \( S(q,r) \) as:

\[
S(q,r) = \bigotimes_{a \in \text{facets}} S_a^*(p,r)
\]

### Example

The ranking computation may be better illustrated with an example. Let us define:

- a query \( q = \{(2, 1), \{(3)\}, \{(1, 2), \{(2)\}\}\) \)
- a resource \( r = \{(2)\}, \{(1), (2, 2)\}\).

By comparing the foci of the first facet, and applying Definition 13, we get:

- \( S(\{(2, 1), (2)\}) = W(1) \)
- \( S(\{(3), (2)\}) = 0 \).

For the second facet, we have 4 combinations:

- \( S(\{(1, 2), (1)\}) = W(1) \)
- \( S(\{(1, 2), (2, 2)\}) = 0 \)
- \( S(\{(2), (1)\}) = 0 \)
- \( S(\{(2), (2, 2)\}) = W(1) \).

We aggregate (Definition 14) and normalize (Definition 15), and we obtain:

- \( S_1 = W(1) \)
- \( S_1^* = \frac{W(1)}{W(1) + W(2)} \)
- \( S_2 = W(1) \otimes W(1) \)
- \( S_2^* = \frac{W(1) \otimes W(1)}{W(1) \otimes W(1)} = 0.5 \)

Using the example weights of Figure 3, we obtain:

- \( S_1 = 0.33 \)
- \( S_1^* = \frac{0.33}{0.33 + 0.33} = 0.52 \)
- \( S_2 = 0.33 \otimes 0.33 = 0.5511 \)
- \( S_2^* = \frac{0.33 \otimes 0.33}{0.33 \otimes 0.33} = 0.87 \)

Finally, \( S = S_1^* \otimes S_2^* = 0.52 \otimes 0.87 = 0.45 \) is the measure of the overall similarity between \( q \) and \( d \). \( \square \)

### 3. FaSet Relational Implementation

The proposed search and ranking model has been designed to be efficiently implemented in any modern relational database, thus easing the integration of faceted search interfaces in any new or existing web application. A first implementation of the FaSet model is currently empowering the Freeable web site, a faceted search engine for software and disability aids\(^2\).

The FaSet reference implementation is based on creating 3 additional tables in the application database, and on performing most of the search work at the database (SQL) level. The mapping between the abstract model and its relational implementation relies on a simple mapping of facet names and focus names into strings. In particular, each facet is identified by a unique facet prefix of type string and of fixed length (e.g., a single capital letter A-Z, or, in the unlikely event that more than 26 facets are used, a 2-letter combination). On the other hand, each focus is identified by a focus number, defined as the concatenation of the facet prefix with the hierarchical name of the focus (e.g., "R132", "A2", "B"). Focus names are again interpreted as fixed-lengths string representations of the abstract integer lists. If

\(^2\) http://www.freeable.eu
all foci have less than 10 sub-foci, then a single digit per hierarchical level is sufficient; otherwise, two or more digits per level will be used (e.g., "R010302").

The 3 additional tables are called Facet, Focus, and SIndex (short for Search Index). The first two represent the static information describing the classification system, and may be used in generating the user interface. In particular, the Facet table (Tab. 5) lists the facets in the search engine, and the Focus table (Tab. 6) contains all defined foci for each facet.

The SIndex table (Tab. 2) holds all classification information about indexed resources, and is the only one needed by the search algorithm at run-time. This information is dynamic (it changes when new resources are added or updated). The foreign key fkResource is the only field linking the classification system to the web application database, since it stores a unique identifier (integer, or string) pointing to the resource being described.

Example. The encoding of the classification information in the SIndex table is illustrated by the example in Table 8, where we reported the relational mapping of the hypothetical resource number 42, whose faceted classification was already shown in Table 2.

The search algorithm that operates on the relational model is split in two phases:

1) Filtering: given the query q and the contents of the SIndex table, derive a list of idResource that are compatible with q, according to Definition 10, implemented as in Theorem 1.

2) Ranking: for each relevant idResource, compute its ranking value.

CREATE FUNCTION compatible
(Lj varchar(99), Lk varchar(99))
RETURNS INTEGER
DETERMINISTIC
RETURN (
SUBSTRING(Lj FROM 1 FOR MIN(LENGTH(Lj),LENGTH(Lk))) =
SUBSTRING(Lk FROM 1 FOR MIN(LENGTH(Lj),LENGTH(Lk))));

Figure 2. Computing prefix-compatibility in SQL

The final result of the search algorithm is an in-memory table (idResource, relevance), returned to the web application for user interface generation. In the following, a possible relational implementation in shown, that completely implements the FaSet search model; in the presentation, clarity is preferred over efficiency, and several database-level optimization are not shown.

3.1. Filtering

According to Theorem 1, a resource is compatible if and only if, for each facet, there is at least one prefix-compatible focus. Prefix-compatibility may be computed with a user-defined SQL function (shown in Figure 2). The list of compatible resources may be therefore computed with the nested query shown in Figure 3. The query has as many AND clauses as the number of total facets. For each facet, a nested query is run to determine resources compatible on that facet, with a set of OR sub-clauses for the various foci present in the query.

3.2. Ranking

The ranking algorithm works on a single query q and a single resource d (from the previous list of relevant resources). The algorithm computes a real number in the interval [0, 1]. The algorithm is a straightforward implementation of Definition 16, and is composed of 3 steps: computing the ⊕-sum for each facet; normalizing the ⊕-sum for each facet; and computing the ⊗-product giving the final ranking value. It is worth noticing that computing the ranking values
is an in-memory operation that only requires the faceted representation of the resource and of the query, with no additional database access.

4. Comparison with Related Works

Very few formal models for faceted navigation have been defined in the literature, and they often date to late 90’s. Among available solutions, the most interesting works are Sacco’s Dynamic Taxonomies [10], [11] and Priss’ Facet Knowledge Representation [12], [13].

Sacco [10] introduces the Dynamic Taxonomy concept, as a new taxonomic model for accessing large information bases. Dynamic Taxonomies are based on the systematic, structured partitioning of information bases and exploit interaction paradigms known since Aristotele’s works. Sacco’s model is based on a Set Theory interpretation of the taxonomic relationships (isA or PartOf) traditionally used to define the “conceptual” structure of a given knowledge domain. Relationships between information units (i.e., documents) are inferred by the fact that some documents are classified under the same taxonomy entries (concepts). According to Sacco, Dynamic Taxonomies are tree taxonomies allowing to systematically summarize search results, to effectively restrict the search domain without using keywords, and to quickly reduce the information space exposed to users, thus enabling them to manually inspect the outcome of the search process.

The Set Theory model proposed in FaSet shares many aspects with Dynamic Taxonomies. They both use a Set Theory foundation, they allow implementing all set operators during the search process and they interpret taxonomic inheritance with the same set inclusion semantics. However, several differences between the two models must be highlighted:

- Dynamic Taxonomies implicitly assume the existence of a single taxonomy, where the root concept is always very abstract and artificial, since it must “summarize” many different aspects considered in the classification.

In the FaSet approach, instead, different facets are described by different taxonomies, implicitly intersected in the search process (see Section 2).

- In Sacco’s model queries are composed by iteratively selecting taxonomy entries and by performing subsequent zoom operations. While this filtering paradigm allows to quickly build complex queries (with few documents satisfying the query), it substantially prevents exploratory search and horizontal browsing of the information base. In fact, all the horizontal navigation steps basically require undoing zoom operations up to the desired abstraction level, and subsequently selecting new taxonomy concepts on which to zoom. On the converse, FaSet allows to easily change the search topics by applying isolated changes on single facets.

- Dynamic Taxonomies formalization is only targeted at the classification side of the retrieval process. No specification is provided for the query model and nothing is said about the ranking of selected resources. Sacco always assumes that search results have a human-explorable size. FaSet, instead, defines a model for results ranking, making it easier to reach desired results, even before reducing the amount of selected documents to a human-manageable size.

- Finally, Sacco does not provide any information on the implementation of Dynamic Taxonomies in relational databases, only highlighting the low-efficiency achieved in these systems with respect to the proprietary Universal Knowledge Processor³ he uses. FaSet adopts a completely different approach and defines a formal retrieval model based on sub-string matching, directly implementable on any state-of-art relational database.

Nearly at the same time of Dynamic Taxonomies, Uta Priss defined the so-called Faceted Knowledge Representation (FKR) model [12]. This model formalizes faceted classification, and search, with a completely new approach based on the notion of units, facets and interpretations. The FKR model has several advantages over Dynamic Taxonomies as it is more general, it already supports the disjoint representation of different facets and it is able to model both the Set-based inheritance notion applied in Sacco’s model (facet aggregation) and some more complex taxonomy formation primitives (through the composition operator). However it is more abstract and no direct, real world application is shown in which the model effectiveness could be assessed. This lack of direct applicability has lately been overcome through the definition of a direct mapping of the FKR model on Formal Concept Analysis (FCA) [13]. With respect to FaSet several differences can be noted:

- The FKR formalization is backed on FCA while Set Theory is used in FaSet. FKR is more general but more difficult to apply than FaSet: the former requires niche

software/algorithms whereas the latter can be easily implemented in standard web application environments, based on relational databases.
- The FaIR system described in [13] uses lattices for representing facets, while FaSet simply uses trees, that is more consistent with current user interfaces.
- FKR is based on a unique name assumption for facets, which often requires facet renaming during the application of the FKR operators of composition and aggregation. FaSet, instead, keeps a clear separation between human-readable facet names and focus names, which are unique by construction.
- The classification in FaIR only supports single-valued facets (at most one concept per facet may be assigned), thus preventing more sophisticated resource representations.
- FKR also does not define any model for documents ranking.

5. Conclusions

This paper presented FaSet, a formal model for describing faceted classification systems, complete with a search and ranking algorithm. The model is based on Set Theory, and is able to express various kinds of facet types, depending on the nature of metadata. The model is designed to be efficiently integrated with web applications, and an implementation of the data model and the search algorithm in terms of relational database operations is shown to prove it. With respect to competing approaches, FaSet sacrifices some expressive power (arbitrary and-or queries with respect to Dynamic Taxonomies, and lattice-structured facets with respect to the Faceted Knowledge Representation), rarely needed in web applications, for the sake of an easy and efficient implementation in real-world applications.

The model has been implemented, as a module for the open source Drupal content management system, and will power a website aggregating expert reviews for accessible software. Future work will involve performance evaluation of the FaSet implementation on realistic datasets, and the evaluation of user interface design for improving usability.

Acknowledgements

This work has been partially supported by Fondazione CRT under the VivoMeglio program. The authors wish to thank Federico Benedetto and Luca Schiatti for their skills and commitment in the implementation of the FaSet engine.

References