Multi-objective Frequent Termset Clustering

Andreas Kaspari and Michael Wurst
University of Dortmund
Computer Science VIII
44221 Dortmund, Germany
andreas.kaspari@uni-dortmund.de
michael.wurst@uni-dortmund.de

Abstract

Large, high dimensional data spaces, are still a challenge for current data clustering methods. Frequent Termset (FTS) clustering is a technique developed to cope with these challenges. The basic idea is to first find frequent termsets and then to transform the resulting directed acyclic graph into a tree by deleting edges and termsets. While this technology was originally developed for document clustering, it can be applied in many other scenarios as well. Existing approaches to FTS clustering apply different heuristics to convert a set of frequent termsets into a final cluster set. In this work, we explore another approach. We first make the desirable properties of an FTS clustering explicit by stating different objective functions. We then show, how these functions are related to each other and that, in general, they are conflicting. This leads directly to the formulation of FTS clustering as a multi-objective optimization problem. We explore the ability of this approach to produce different, pareto-optimal solutions on a social bookmarking data set.

1 Introduction

Data clustering is a key technology to access large data and information spaces in a structured way. While significant progress was made in this area, there are still many open challenges when facing complex, high dimensional data spaces, such as text collections. Traditional clustering approaches, e.g. agglomerative clustering, are not well-suited for such scenarios. A first issue is that it is hard to find clusters in a high dimensional data space, as clusters often only exist in subspaces of this space. In order to cluster publications in the area of artificial intelligence, only terms that concern this topic or its subtopics are relevant, other terms will blur the result. Second, traditional methods often lead to clusters that are hard to interpret for the user. Comprehensible results are, however, essential for clustering, which is in large parts an explorative task. Third, many traditional clustering approaches require the user to set several parameters, a task that is far from being trivial. Finally, many clustering approaches scale poorly with an increasing number of data points and dimensions.

These observations led, among others, to several new clustering approaches, that can be summarized as frequent termset or itemset clustering (Fung *et al.* [2003]; Beil *et al.* [2002]). The basic idea is to first find frequent itemsets in the underlying data. This task can be accomplished even on very large data sets with many dimensions (Agrawal *et al.* [1993]). The actual clustering step is then performed on the resulting frequent itemsets and not on the original data. As clusters are represented by frequent itemsets, the resulting cluster structure automatically contains cluster descriptions that are comprehensible for the user. Also, the resulting clusters are by definition clusters only in subspaces of the original data space. This technique has been particularly applied to text clustering, where the number of dimensions is extremely high. In this case, frequent itemsets are sets of terms, that often co-occur in the documents to be clustered. The resulting clustering is a tree consisting of combinations of terms arranged according to the subset relation (see figure 1).

While frequent termset clustering was successfully applied to different areas, there are still some open points. Existing approaches use a number of heuristics to derive a cluster structure from a set of frequent termsets, implicitly trading-off diverse diserable criteria of such cluster structures. These criteria include maximal coverage, minimal overlap, simplicity of the overall structure etc. This implicit, heuristic merging of criteria makes it hard for the user to control the clustering process. This is particularly true, as these criteria are often conflicting. The overlap, for instance, can often be reduced by incorporating additional clusters in the final cluster structure. This, on the other hand, makes the final structure more complex and harder to overlook.

In this work we therefore choose another approach. We first analyze which properties are desirable for an FTS clustering and then derive several, partially conflicting objective functions. Instead of merging them in a heuristic manner, we first analyze their mutual relations. We then use a subset of actually conflicting criteria in an explicit, multi-objective optimization procedure. In general multi-objective optimization delivers more than one solution, as several criteria are involved, that may be in conflict with each other. This gives the user the opportunity to choose a desired solution from a set of pareto-optimal solutions, instead of having to search a large space of parameters manually in a laborous trial-and-error process.

This principle has been applied to other clustering tasks as well. In particular there are several approaches that apply multi-objective optimization to the task of feature selection for clustering. Since initial approaches, as Kim *et al.* [2000, 2002] or Morita *et al.* [2003], showed some weaknesses, a new sound framework was proposed in Mierswa and Wurst [2006a]. The proposed criteria trade off the cluster quality against the similarity of the resulting feature space to the original one. This approach is denoted as information preserving feature selection. The rationale behind this idea is, that clustering is basically an explorative task, that should describe or summarize the given data in an adequate and unbiased way. Information should only be omitted if this leads to a better and simpler cluster structure. The user can then decide in an interactive way, which features carry useful information and which of them are noise by inspecting the set of pareto-optimal solutions. In Mierswa and Wurst [2006b] this idea was extended to feature construction as well.

In this work, we propose a multi-objective framework for frequent termset clustering as well as several objective functions for this task. We analyze the mutual relation of these objective functions and the soundness of the framework on a real world social bookmarking data set. The work is structured as follows. In section 2 we give a brief introduction to frequent termset clustering and argue, that finding an optimal clustering implies finding a trade-off between different, conflicting criteria. In section 3 we then give a short introduction to multi-objective optimization. This is the point of departure for the formulation of several, partially conflicting objective functions for frequent termset clustering and for the analysis of their mutual relations in section 4. In section 5 we discuss the application of the approach to the problem of clustering tags in a social bookmarking system and present empirical results. In section 6 we summarize the results and point out the future direction of research.

2 Frequent Termset Clustering

In the following we assume a set of uniquely identified resources R and a set of terms T. We can then define the notion of a termset C as a set of terms in T, thus $C \subseteq T$.

We further assume a function $g : T \times R \rightarrow \mathbb{N}$ that assigns term occurrences to resources. In the simplest case, this function is binary, stating whether the term is assigned to the resource or not. In general, it can express the relevance of the term to the resource, as known from the vector space model.

Note that while this terminology suggests an application in text clustering applications, frequent termset clustering can be applied to other areas as well. In this case resources are general transactions and terms are items.

Based on the function g we define a cover relation $\nabla \subseteq R \times \mathcal{P}(T)$ for which the following holds:

$$r \nabla C \equiv \forall t \in C : g(t, r) > 0 \tag{2.1}$$

Thus a resource is covered by a termset, if all terms in the termset are assigned to the resource. The support of a termset is defined as the fraction of resources it covers.

Algorithms as Apriori (Agrawal *et al.* [1993]; Agrawal and Srikant [1994]) or FPGrowth (Han *et al.* [2000]) allow to efficiently find the set of frequent termsets, that have a support that exceeds a given minimal support. For an overview on frequent itemset mining refer to Goethals [2003].

This set of frequent termsets can then be arranged into a directed acyclic graph (DAG), by using the subset relation. Such a structure is however often not very well suited to access and navigate a complex information space. This is especially true, if the number of frequent termsets is very high. Several approaches have been proposed to derive flat or hierarchical cluster structures from frequent termsets. A first, simple approach to exploit the idea of frequent terms for clustering is presented in Wang *et al.* [1999]. While this algorithm does not make direct use of frequent itemset mining, it exploits the same underlying idea. Clusters consist of resources and are characterized by terms. For each term, the support within each cluster is calculated. Based on a threshold value, terms concerning a cluster are separated into large and small terms. Resources are then assigned to clusters in such a way, that each cluster contains as few small terms as possible and that the overlap of large terms among clusters is minimized. The complexity of the clustering is controlled by setting the support parameter appropriately.

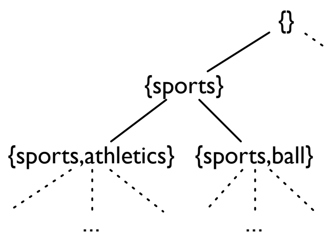


Figure 1: Excerpt from a clustering produced by the methods proposed in Fung et al. *[2003]*

A natural extension of this idea is presented in Beil *et al.* [2002]. First a frequent itemset algorithm is applied to identify frequent termsets. Then a subset of this set of frequent termsets is selected, that minimizes the overlap among the selected frequent termsets, while covering all resources. This method is extended to produce hierarchical clusters by first applying it to all frequent termsets of size one, then to all corresponding frequent termsets of size two, and so on.

In Fung *et al.* [2003] an alternative method is proposed, that converts the DAG of frequent termsets into a simpler, hierarchical cluster structure. First, a frequent itemset mining algorithm is used to identify frequent termsets. These frequent termsets form an initial clustering by arranging them according to the subset relation. Each resource (in this case text documents) is assigned to exactly one frequent termset using a centroid-based approach. Then, in a bottom-up way, the algorithm selects a parent node for each frequent termset by calculating a centroid resource of all resources in a cluster. Both steps aim to minimize the overlap among nodes on the same level of the cluster tree. The resulting tree is pruned applying several heuristics, as to reduce its complexity. Figure 1 shows an example.

While this approach can lead to good results, a general issue is the use of several heuristic steps that partially contain hidden parameters. These make it unclear which criteria are actually optimized. Also, the user has no influence on these criteria and on their combination.

In this work we propose a systematic approach to identify different desirable criteria of frequent termset clusterings. Implicitly, the above approaches aim to achieve the following:

- *Coverage*: The number of resources that cover at least one of the termsets should be as high as possible, because any resource for which no such termset exists is not represented in the cluster structure. The approaches above achieve full coverage, which is however not suitable in the presence of outliers. Therefore coverage should be considered a continuous value.
- *Overlap*: Traditional clustering algorithms, such as k-means, usually try to make clusters as separated as

possible. This principle is also applied by the FTS Clustering approaches presented above. We will discuss in section 5, whether this is always necessary and desirable.

- *Detailedness*: The resulting clustering should be as detailed as necessary, as the task of clustering is to describe the underlying data set, leaving the exploration to the user. This usually implies that the cluster tree should be as deep as possible, allowing fine grained distinctions.
- *Simplicity*: The resulting structure should be easy to navigate and overlook by a human user.

These criteria are clearly in conflict with each other. Adding additional clusters makes the clustering more complex and harder to navigate, will however usually increase the coverage and decrease the overlap. A deeper and thus more complete clustering will usually introduce additional overlap, etc.

Instead of using heuristic procedures to combine these criteria, we formalize objective functions that reflect these properties and apply them in a multi-objective optimization procedure. This allows to make the trade-off between different criteria explicit and will help to gain insight in the ways in which the criteria are related.

In the following we give a general definition for a frequent termset clustering. In contrast to Fung *et al.* [2003] we do not require the resulting cluster structure to be a tree, but a DAG.

Definition 2.1. (Frequent Termset-Clustering)
A termset clustering orders a set of (frequent) termset in a hierarchical way.

- $\mathsf{C} \subseteq \mathcal{P}(T)$ is an non-empty, finite set of termsets. Each termset represents a cluster. C is denoted as the cluster set.
- The relation $\prec$: $\mathcal{P}(T) \times \mathcal{P}(T)$ defines a hierarchical order on all clusters in C. For all pairs of clusters $C, D \in \mathsf{C}$ the following holds:

$$C \prec D \Leftrightarrow (C \subset D) \wedge (|C| = |D| - 1) \tag{2.2}$$

We require some additional constraints on a frequent termset clustering.

Definition 2.2. (Frequent Itemset Clustering Conditions)
A cluster set $\mathsf{C} \subseteq \mathcal{P}(T)$ must fulfill the following constraints:

$$\emptyset \in \mathsf{C} \tag{2.3a}$$
$$\forall D \in \mathsf{C} \text{ with } D \neq \emptyset : \exists C : C \prec D \tag{2.3b}$$
$$\forall C \in \mathsf{C} : \exists r \in R : r \nabla C. \tag{2.3c}$$

Condition (2.3a) states that the empty set must be contained in each cluster set. Condition (2.3b) ensures that there is a path from each cluster to the empty set (thus the cluster set is a connected graph). Condition (2.3c) ensures that each cluster contains at least one resource.

The set $\mathfrak{C}$ will denote all possible cluster structures that can be derived from T, considering only termsets that meet a certain minimal support.

$$\mathfrak{C} = \{\mathsf{C} \mid \mathsf{C} \subseteq \mathsf{FTS} \text{ and } \mathsf{C} \text{ valid}\} \tag{2.4}$$

where FTS is the set of frequent termsets with a minimal support of σ.

Based on this definition, we present a multi-objective optimization algorithm that selects several cluster sets as subsets of the set of all frequent termsets. These are then presented to the user. The selection process will be governed by several optimization criteria that will be presented in section 4.

3 Multi-objective Optimization

In the last section we argued, that frequent termset clustering is an inherently multi-objective problem. In the following, we will give a more precise definition of this notion.

Definition 3.1. (Multi-objective optimization problem)

$$\max \vec{f}(x) \text{ with } x \in S \tag{3.1}$$

The set S describes the set of valid solutions. $\vec{f}$ assigns to each element $x \in S$ a solution vector from R^k, where R is a totally ordered set. Each element of this vector represents one criterion and k is the number of criteria.

Definition 3.2. (Pareto-dominance)
A solution vector $\vec{u}$ dominates a solution vector $\vec{v}$ (short $\vec{u} \succeq \vec{v}$), iff:

$$\forall i \in \{1, ..., k\} : u_i \geq v_i \tag{3.2a}$$

as well as

$$\exists i \in \{1, ..., k\} : u_i > v_i \tag{3.2b}$$

A solution vector is called non-dominated, if it is not dominated by any other solution vector.

Definition 3.3. (Pareto-optimal set)
For a multi-objective optimization problem, the set of pareto-optimal solutions is defined as

$$P^* := \{x \in S | \nexists y \in S : \vec{f}(y) \succeq \vec{f}(x)\} \tag{3.3}$$

i.e. P^* contains only non-dominated solution vectors.

The task of multi-objective optimization is to determine a set of pareto-optimal solutions.

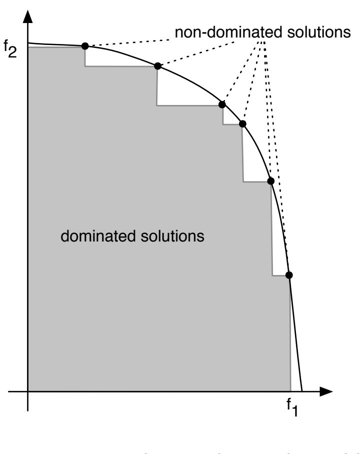


Figure 2: The pareto-optimal set is depicted as a black line. This line is often referred to as the pareto front.

Multi-objective optimization is very powerful, as it allows users to choose from a set of results, instead of a single result (Zitzler and Thiele [1999]; Coello Coello [1999]). While it would be in principle possible, to combine two or more criteria by a linear combination, this would work only, if these criteria are indeed in a linear relation to each other. In general, pareto fronts, as the one shown in figure 2, are highly non-linear and sometimes of very complex shape. The non-linear shape of the pareto front often guides the user in the process of selecting an appropriate solution, as it is very easy to visually identify interesting points, such as "elbows".

A desirable property of pareto-optimal solutions is that they should cover the space of possible solutions well, thus that the dots on a pareto front should be distributed as homogeneously as possible. This allows the user to choose from a wide variety of different solutions.

An important prerequisite for achieving such solution sets is that the criteria are in conflict. If two criteria are correlated, for instance, the solution set will contain only a single solution that dominates all other solutions.

4 Multi-objective Frequent Termset Clustering

4.1 Fitness Criteria for Frequent Termset Cluster Structures

As described above, the essential step of FTS clustering is to select the subsets of frequent termsets that are presented to the user. There are several possible criteria to assess the quality of each such subset. In general we define the fitness of a cluster structure in the following way.

Definition 4.1. Fitness of a cluster structure
A cluster structure fitness measure is a function $\mathfrak{C} \to \mathbb{R}$, that assigns to each valid cluster set $\mathsf{C} \in \mathfrak{C}$ a real-valued quality measure.

In section 2, several criteria were discussed that are implicitly used in most frequent termset clustering algorithms. These criteria are coverage, overlap, simplicity and detailedness. In the following we will derive formal definitions for these criteria and discuss whether they are well-suited for deriving comprehensible and complete cluster structures.

To ease the presentation of the individual criteria, we introduce several properties. The function $\text{res} : \mathcal{P}(\mathcal{P}(T)) \to \mathcal{P}(R)$ calculates for a set M of clusters the set of all covered resources $r \in R$:

$$\text{res}(\mathsf{M}) = \bigcup_{M \in \mathsf{M}} \{r \mid r \nabla M\} \tag{4.1}$$

The function $\text{depth} : \mathfrak{C} \to \mathbb{N}$ calculates the depth of a cluster set:

$$\text{depth}(\mathsf{C}) = \max_{C \in \mathsf{C}} |C| \tag{4.2}$$

The term $\mathsf{M}_{|i}$ denotes all sets in M that contain i elements, thus $\mathsf{M}_{|i} = \{M' \in \mathsf{M} \mid i = |M'|\}$, where $i \in \mathbb{N}$.

A property that is very popular is overlap. Algorithms as k-means aim to maximize the inter-cluster dissimilarity making individual clusters as disjoint as possible. In the context of ontologies, disjointness of concepts plays an important role for achieving sound definitions.

In the following we capture the average overlap among frequent termsets on the same level of a cluster set by the following expression.

Definition 4.2. (Overlap)
Given a clustering C, then $\text{overlap} : \mathfrak{C} \to \mathbb{R}$ is defined as

$$\text{overlap}(\mathsf{C}) = \frac{1}{\text{depth}(\mathsf{C})} \cdot \sum_{i=1}^{\text{depth}(\mathsf{C})} \frac{|\text{res}(\mathsf{C}_{|i})|}{\sum\limits_{C \in \mathsf{C}_{|i}} |\text{res}(\{C\})|} \tag{4.3}$$

For $\text{depth}(\mathsf{C}) = 0$, we assume $\text{overlap}(\mathsf{C}) = 1$.

The function overlap measures the average overlap among clusters on each level. In the best case, each resource in R appears only in one cluster on each level. This criterion is very much akin to the overlap criterion proposed in Beil *et al.* [2002]. In section 5 we will argue, why it is not always desirable to optimize this criterion explicitely.

A second property of a frequent termset clustering is coverage. Many existing clusterings algorithms require to cluster all resources, thus to achieve a full coverage. In application areas, in which we can expect a high number of outliers, this can lead to poor results. It is therefore often desirable to ignore some resources in the clustering process, which however leads to a smaller coverage. Therefore, as second criterion, we define the coverage of a cluster set.

Definition 4.3. (Coverage)
Given a cluster set $\mathsf{C} \in \mathfrak{C}$, then the function $\text{cover} : \mathfrak{C} \to [0, 1]$ is defined as follows:

$$\text{cover}(\mathsf{C}) = \frac{|\text{res}(\mathsf{C}_{|1})|}{|R|} \tag{4.4}$$

Overlap and coverage are usually in conflict with each other. Given a clustering that does not cover all resources, additional resources can only be covered by adding more frequent termsets. In this process the overlap can never decrease and is likely to increase, if the set of possible choices (frequent terms not yet chosen) is limited.

A third criterion is the simplicity of the clustering from a user perspective. The FTS clustering algorithms presented above achieve this by optimizing the inner cluster similarity and by applying heuristic pruning procedures.

We can however capture the simplicity of a cluster tree from a user perspective in an explicit way. For a flat clustering, the number of clusters should be chosen as small as possible, while achieving a cluster quality that is as high as possible. The underlying idea is that inspecting each cluster is connected with certain costs for the user. This idea can be transferred to hierarchical cluster sets as well. As such sets are often navigated top-down, we use the number of child nodes at the root and each inner node as indicator of the complexity of a cluster set.

Definition 4.4. (Child count)
Given a cluster set C, we define $\text{succ} : \mathsf{C} \to \mathcal{P}(\mathsf{C})$ as

$$\text{succ}(C) = \{D \in \mathsf{C} \mid C \prec D\} \tag{4.5}$$

and thus the set $\mathsf{C}' \subseteq \mathsf{C}$ as

$$\mathsf{C}' = \{C \in \mathsf{C} \mid |\text{succ}(C)| > 0\} \tag{4.6}$$

Based on this, we can define

$$\text{childcount}_{max}(\mathsf{C}) = \max_{C \in \mathsf{C}'} |\text{succ}(C)| \tag{4.7}$$

Thus the complexity of a cluster structure is given as the most complex node.

The maximal child count will usually increase with increasing coverage, as to cover more resources, additional clusters are needed.

Another criterion is the depth of a cluster structure. As users often navigate a cluster structure top-down, the depth does not contribute to the complexity of the cluster tree as does the child count. In contrary, a high depth allows for more fine grained distictions at a lower level. The depth should therefore be maximized.

This is captured in the following criterion.

Definition 4.5. (**Depth of a cluster set**)

Let the set $\mathsf{C}' \subseteq \mathsf{C}$ be defined as

$$\mathsf{C}' = \{C \in \mathsf{C} \mid \nexists D : C \prec D\} \tag{4.8}$$

We can then define the following criteria:

$$\text{depth}_{avg}(\mathsf{C}') = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} |C| \tag{4.9}$$

$$\text{depth}_{max}(\mathsf{C}') = \max_{C \in \mathsf{C}'} |C| \tag{4.10}$$

$$\text{depth}_{var}(\mathsf{C}') = \frac{1}{|\mathsf{C}'|} \sum_{C \in \mathsf{C}'} (|C| - \text{depth}_{avg}(\mathsf{C}')) \tag{4.11}$$

How is depth related to the other criteria? Per definition, coverage is mostly independent of the depth of the cluster tree, as only the resources covered by first level nodes are regarded. The same holds for child count. Overlap is likely to increase with increasing depth in many cases, as the distictions on top-level are stronger than on a more detailed level. This is also confirmed in our experiments, showing that if we do not explicitely optimize for depth, the algorithm produces rather shallow cluster structures.

One possibility to solve this problem is to use depth as an additional criterion in multi-objective optimization. This is however not fully satisfying, as it makes the optimization process more complex and as depth is not in strong conflict with any of the other criteria.

We therefore rather solve this problem by replacing coverage by another concept, namely completeness. The idea of completeness is, that the selected clusters should represent the given frequent termsets as good as possible.

Definition 4.6. (**Completeness**)
Given two cluster sets C and C_{ref}. We assume $\mathsf{C} \subset \mathsf{C}_{ref}$. Then the function compl : $\mathfrak{C} \times \mathfrak{C} \rightarrow \mathbb{R}$ is defined as:

$$\text{compl}(\mathsf{C}, \mathsf{C}_{ref}) = \frac{|\,\mathsf{C}\,|}{|\,\mathsf{C}_{ref}\,|} \tag{4.12}$$

Thus the more of the original frequent termsets are contained in the final clustering, the higher the completeness. This combines coverage and cluster depth in one straighforward criterion.

This criterion is in conflict with child count and with overlap. In section 5 we will analyze empirically, how the criteria are related to each other.

Beside the criteria presented here, there are many other possible criteria, such as to what grade the cluster structure resembles a tree or the number of paths by which a resource can be reached from the root. These criteria are however beyond the scope of this presentation.

4.2 Deriving Pareto-optimal solutions

Several algorithms were proposed for multi-objective optimization, almost all of them based on evolutionary computation (Coello Coello [1999]; Zitzler and Thiele [1999]). In this work we use the genetic algorithm NSGA-2 (Deb *et al.* [2000]) to approximate the set of pareto-optimal solutions. Individuals are represented as binary vectors, such that each element of the set of frequent termsets corresponds to one position in the vector. The cluster conditions are enforced by post-processing each individual.

The algorithm approximates the set $\mathfrak{C}^* \subseteq \mathfrak{C}$ of pareto-optimal cluster sets.

$$\mathfrak{C}^* = \{\mathsf{C} \in \mathfrak{C} \mid \nexists \mathsf{D} \in \mathfrak{C} : \vec{f}(\mathsf{D}) \succeq \vec{f}(\mathsf{C})\} \tag{4.13}$$

where $\vec{f}(\mathsf{D}) \succeq \vec{f}(\mathsf{C})$ states that there is no cluster set D that pareto-dominates the cluster set C with respect to the fitness functions $\vec{f}$. Thus $\mathfrak{C}^*$ contains all non-dominated cluster sets.

$\mathfrak{C}^*$ is also referred to as the pareto front. As mentioned before the algorithm should produce solutions that are equally spread across the pareto front.

5 Application and Evaluation

5.1 Social Bookmarking and Automatic Tag Clustering

Social bookmarking systems allow users to annotate resources on the internet with arbitrary textual descriptions called tags (Hammond *et al.* [2005]). These systems are extremely popular, as assigning tags is very simple (Shirky [2005]). In contrast to predefined keywords or categories, tags are very flexible and dynamic, allowing to capture the views of even rather small niche communities. Based on the “everything is a link” paradigm, users can navigate the hypergraph of user ids, tags and resources (Golder and Huberman [2006]).

While this is convenient for few tags and resources, it quickly becomes chaotic as the number of tags and resources grows. An important challenge is therefore to transform the user assigned tags into a navigation structure that is simple to overlook but on the other side reflects as many of the underlying resources as possible. Such a structure combines the best of both worlds: the relative simplicity of predefined taxonomies and the flexibility and subjectivity of user assigned tags.

Traditional clustering methods are not well suited for this task, as the data space is extremely complex and sparse. Apparently frequent termset clustering is designed to deal with exactly this kind of data.

There are several approaches that cluster tags as to make the resulting structure easier to overlook and navigate. In Hassan-Montero and Herrero-Solana [2006], tags are selected that show a high degree of diversity by applying a *tf/idf*-like measure. These tags are then clustered using Bisecting k-means and Jaccard-Similarity. Begelman *et al.* [2006] and Kaser and Lemire [2007] represent tags as graphs on which they apply graph clustering algorithms to obtain sets of similar tags. These methods suffer from the same problem as other traditional clustering algorithms, namely the extremely high number of dimensions and the high number of tags to be clustered. Also, by making tag clusters as dissimilar as possible to each other, they implicitly minimize the overlap, which is not always appropriate, as will argued below. Finally, these approaches include partially complicated parametrization (e.g. the right choice of a similarity measure, number of clusters, etc.). This leads to a laborious trial-and-error procedure in practice. This problem is even more severe, as the resulting tag clusters do not contain cluster descriptions, making them harder to interpret by the user. In contrast, the parameterization of our multi-objective clustering is controlled by the optimization procedure. Instead of trying out different parameter settings, the algorithm directly proposes to the user several promising results.

An approach that applies frequent itemset mining to tag structures is described in Schmitz *et al.* [2006]. The authors use different kinds of projections to map tag assignments to transactions. They then use frequent itemset mining to derive association rules. These association rules can then be

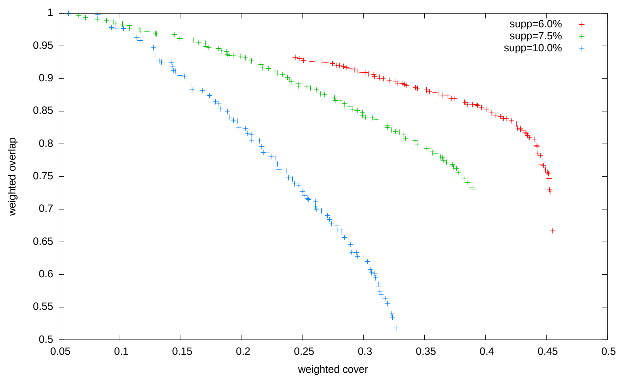


Figure 4: Pareto front for Overlap vs. Coverage for different supports

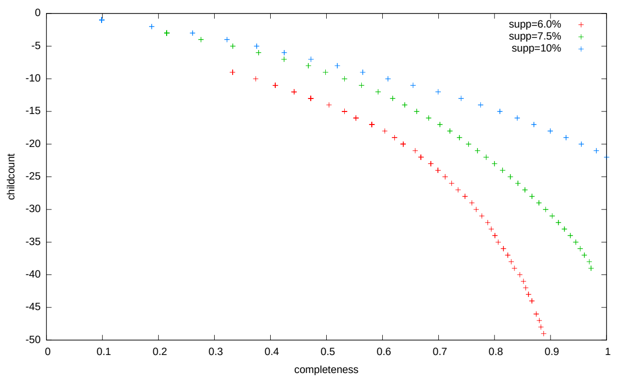


Figure 5: Pareto front for child count vs. completeness for different minimal supports

visualized as a cluster tree. The association rules are however not filtered or post-processed in any way. This, however was identified as a crucial step that frequent termset clustering achieves.

5.2 Experimental Results

We applied the approach proposed above to the freely available social bookmarking data set derived from the Bibsonomy system (Hotho *et al.* [2006]). This data set contains the tag assignments of about 780 users. The number of resources tagged by at least one user is about 59.000. The number of tags used is 25.000 and the total number of tag assignments is 330.000.

In order to find a set of frequent tagsets with respect to some minimal support σ, we must first define our notion of a tag's frequency. Although tag frequency can be defined in several ways, we consider a tag to be frequent if a certain number of users have assigned it to arbitrary resources.

We performed two experiments. In a first experiment we tried to optimize the overlap against the coverage of a cluster structure. This corresponds to the traditional idea of a cluster structure as being a level-wise disjoint structuring of entities in a domain of interest.

The resulting pareto front is shown in figure 4.

A more detailed analysis of the individual results shows the following:

- Cluster sets that fulfill the overlap criterion well are quite narrow and show a bad coverage.
- Cluster sets that fulfill the coverage criterion well are very broad and contain a lot of overlap.
- All resulting cluster structures are very shallow, as neither of the criteria forces the selection of deep clusters. Both, high coverage and low overlap can be achieved with clusters of level one.

This supports our claim, that optimizing overlap and coverage leads to not very detailed cluster structures that often resemble rather a flat partition than a hierarchical clustering.

There is also another interesting observation. Minimizing overlap removes the natural heterogeneity from the data. Some users may for instance have tagged news articles with country names (e.g. germany, france) and other users may have tagged them thematically (e.g. ecology). Now, if ecology and germany has an overlap that is very strong, then probably one of both tags will be removed, if the overlap is minimized. This is however not desirable, as both tags represent valid access paths to tagged resources.

Therefore we argue, that at least for tag clustering, minimizing the overlap is not only not an important criterion, it could even be counter-productive. In many other application scenarios, this holds as well.

In a second experiment, we used the two other criteria proposed in this work, namely maximal child count and completeness with respect to the frequent termsets. The corresponding pareto front is depicted in figure 5.

A nearer inspection of the pareto optimal results yields the following:

- Clusterings with a small maximum child count are narrow, but deep. This effect can be explained, as deep clustering yield on average a higher completeness.
- Clusterings with high completeness are broader, but still deep and contain much of the heterogeneity contained in the original data. They also show a very high coverage.

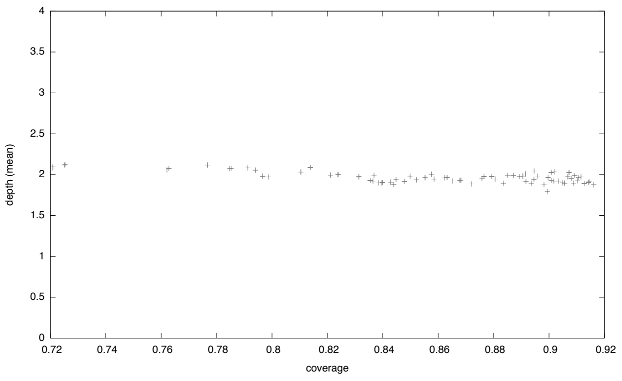
In this way we can actually optimize three criteria at once: the simplicity of the cluster structure, its detailedness in terms of cluster depth and the coverage of resources. These criteria are furthermore not biased to remove heterogeneity from the data, which is essential in many explorative applications.

Figure 3 shows how the different optimization criteria introduced in this work are related to each other. Figure 6 shows exemplarily the most simple tag structures produced by overlap vs. coverage and child count vs. completeness respectively.

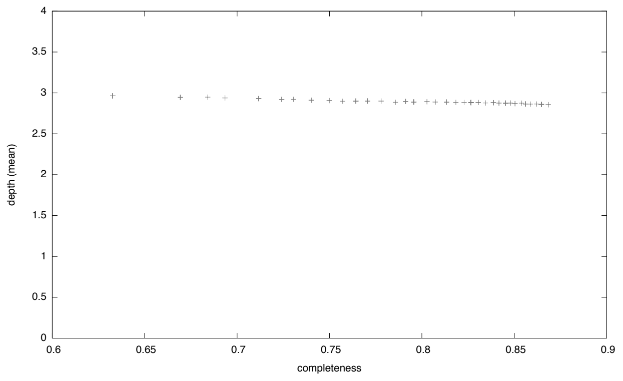
6 Conclusion

In this work we presented an approach to frequent termset clustering that makes use of multi-objective optimization. This enables the user to choose from a set of promising results instead of having to search a complex parameter space in a trial-and-error way or having to rely on heuristic procedures. It also makes desirable properties of frequent termset clusterings explicit and allows to explore the relation among different optimization criteria in a systematic way.

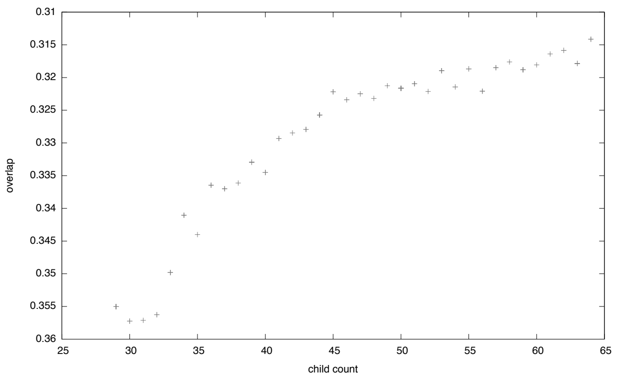
We applied the algorithm in a social bookmarking scenario. The aim was to simplify the complex structure of user assigned tags in order to make this structure easier to navigate. We pointed out, that the overlap criterion applied by many clustering algorithms is not satisfying in this scenario, as it is likely to destroy the natural heterogeneity in the underlying data. Optimizing for small complexity and high completeness with respect to the selected frequent



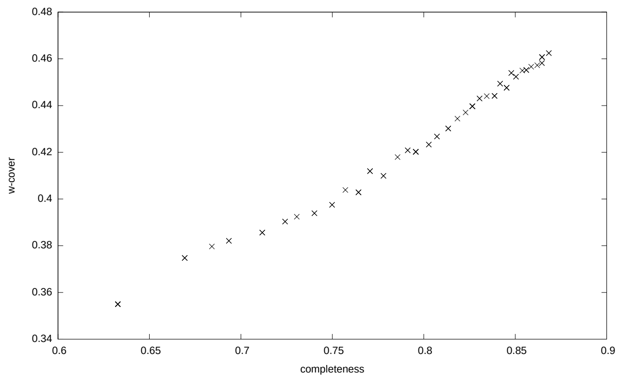
(a) Coverage vs. cluster depth when using overlap and coverage as optimization criteria



(b) Completeness vs. cluster depth when using completeness and child count as optimization criteria

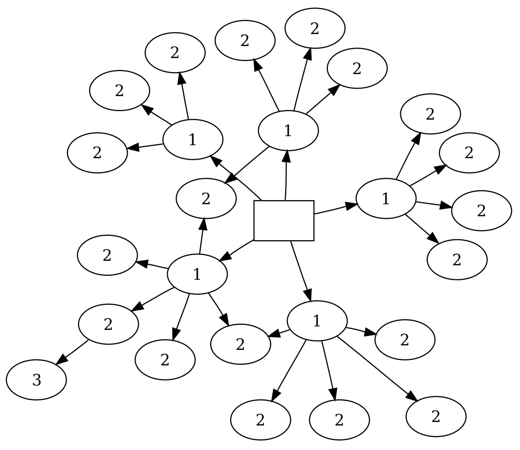


(c) Child count vs. overlap when using completeness and child count as optimization criteria



(d) Completeness vs. coverage when using completeness and child count as optimization criteria

Figure 3: These plots show how the different criteria are related to each other.



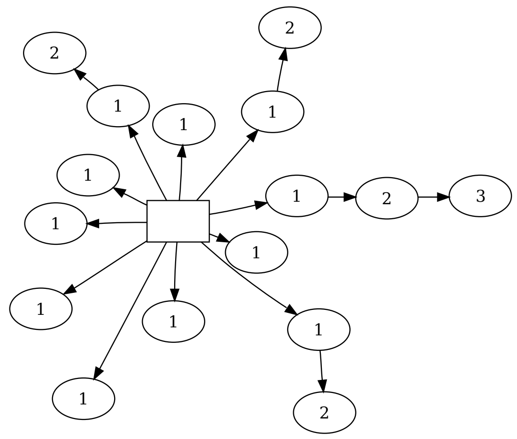


Figure 6: The simplest tag structures for child-count vs. completeness (on the left) and for coverage vs. overlap (on the right). The numbers in the nodes denote their depth in the tree. The tree on the left is deeper and better balanced than the tree on the right.

termsets on the other hand led to sound results that implicitly optimized other criteria as the cluster tree depth as well.

In our future work we plan to analyze additional criteria and their relationship. Also, we will explore the suitability of the approach in other application areas, such as customer segmentation. High-dimensional, complex data spaces are still challenging for clustering algorithms. We think that both, multi-objective optimization and frequent item based approaches will play an important role to solve these challenges in the future.

References

R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the International Conference on Very Large Data Bases*, 1994.

R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1993.

G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop*, 2006.

F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):129–156, 1999.

K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature Conference*, 2000.

B. C. M. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM International Conference on Data Mining*, 2003.

B. Goethals. Survey on frequent pattern mining. `http://www.adrem.ua.ac.be/bibrem/pubs/fpm_survey.pdf`, 2003.

S. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

T. Hammond, T. Hannay, B. Lund, and J. Scott. Social Bookmarking Tools (I): A General Review. *D-Lib Magazine*, 11(4), 2005.

J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000.

Y. Hassan-Montero and V. Herrero-Solana. Improving Tag-Clouds as Visual Information Retrieval Interfaces. In *Proceedings of the International Conference on Multidisciplinary Information Sciences and Technologies*, 2006.

A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. BibSonomy: A social bookmark and publication sharing system. In *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the International Conference on Conceptual Structures*, 2006.

O. Kaser and D. Lemire. Tag-cloud drawing: Algorithms for cloud visualization. In *WWW Workshop on Tagging and Metadata for Social Information Organization*, 2007.

Y. Kim, W.N. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2000.

Y. Kim, W. N. Street, and F. Menczer. Evolutionary model selection in unsupervised learning. *Intelligent Data Analysis*, 6:531–556, 2002.

I. Mierswa and M. Wurst. Information preserving multi-objective feature selection for unsupervised learning. In *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, 2006.

I. Mierswa and M. Wurst. Sound multi-objective feature space transformation for clustering. In *Workshop on Knowledge Discovery, Data Mining, and Machine Learning*, 2006.

M. Morita, R. Sabourin, F. Bortolozzi, and C.Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, 2003.

C. Schmitz, A. Hotho, R. Jäschke, and G. Stumme. Mining association rules in folksonomies. In *Proceedings of the IFCS Conference*, 2006.

C. Shirky. Ontology is overrated: Categories, links, and tags. `http://www.shirky.com/writings/ontology_overrated.html`, 2005.

K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *Proceedings of the International Conference on Information and Knowledge Management*, 1999.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.