The Semantic GrowBag Algorithm: Automatically Deriving Categorization Systems

Jörg Diederich and Wolf-Tilo Balke

L3S Research Center and Leibniz Universität Hannover, Hanover, Germany
{diederich,balke}@l3s.de

Abstract. Using keyword search to find relevant objects in digital libraries often results in way too large result sets. Based on the metadata associated with such objects, the faceted search paradigm allows users to structure and filter the result set, for example, using a publication type facet to show only books or videos. These facets usually focus on clear-cut characteristics of digital items, however it is very difficult to also organize the actual semantic content information into such a facet. The *Semantic GrowBag* approach, presented in this paper, uses the keywords provided by many authors of digital objects to automatically create light-weight topic categorization systems as a basis for a meaningful and dynamically adaptable *topic facet*. Using such emergent semantics enables an alternative way to filter large result sets according to the objects' content without the need to manually classify all objects with respect to a pre-specified vocabulary. We present the details of our algorithm using the DBLP collection of computer science documents and show some experimental evidence about the quality of the achieved results.

Keywords faceted search, category generation, higher-order co-occurrence

1 Introduction

Due to today's sophisticated ranking techniques, the simple keyword search paradigm has been remarkably successful in finding relevant resources in huge data collections, such as digital libraries or even the world wide web. One remaining problem, however, is that users are often unsure which actual keywords to choose so that finding a particular resource often involves several search queries, which then have to be manually refined step-by-step according to the result set of the previous keyword search.

The *faceted search* [1–4] paradigm makes this process of refining queries explicit and presents the results along with several orthogonal facets, which characterize the result set (e.g., a 'publication type' facet might reveal that there are only two videos among possibly 10,000 relevant results) and thus allow the user to restrict the result set in an easy and intuitive way by exploiting metadata.

This paper is focused on facets based on the actual content of the objects, which allow to restrict the result set to a specific topic. To limit the size of such a *topic facet*, a hierarchical system is typically used to structure the facet, for example, using the Dewey Decimal Classification System, the ACM curriculum or any other cataloguing system (cf. Fig. 1). Such a topic facet can be used in several different ways:

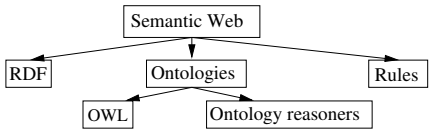

Fig. 1. Example categorization system for topics

- To filter results according to a whole subtree (e.g., if a user selects 'ontologies' in a facet based on the categories in Fig. 1, all objects about 'OWL' and 'Ontology reasoners' can be automatically included in the result set, too).
- To find communities (e.g., for a query 'XML', the facet might show a large number of objects in 'RDF' and in 'OWL', but not in 'Rules').
- To characterize authors or groups of authors (e.g., showing that the objects authored by a particular person are mainly about the topics 'OWL' and 'Rules').
- To characterize the object collection as a whole (e.g., if a query 'rules' only provides results related to 'Semantic Web', but not about 'business rules').

The main problem, however, is creating and maintaining the underlying topic categorization system, which is often done manually and especially difficult for very dynamic domains such as computer science research.

We have already presented first demonstrations of the basic applicability of our approach [5, 6] for topic categorization, but not yet disclosed and evaluated our algorithm's specifics. This paper details our *Semantic GrowBag approach* to automatically organize topics in light-weight hierarchical categorization systems (so-called *GrowBag graphs*), starting from the author keywords available in large digital object collections such as DBLP for the computer science domain, or Medline for the medical domain. In Sect. 2 we briefly investigate related work for faceted searches and taxonomy generation. Section 3 presents our Semantic GrowBag algorithm and its basic characteristics. Using the DBLP data set we explain our algorithm in detail in Sect. 4 along practical use cases and evaluate the derived GrowBag graphs. In this way, we give a good intuition about the algorithm's practical impact and show that the (rather limited) tagging with keywords in today's collections is already useful to derive good and intuitive topic facets. The paper closes with a short summary and outlook.

2 Related Work

The basic idea of faceted search was built on the scatter/gather clustering approaches [7] for document browsing. The hierarchical organization of result documents according to certain faceted categories has been shown to enable intuitive user interfaces superior to the presentation of ranked lists [8]. But for the actual creation of good faceted categories some degree of manual interaction is still needed and all categories of interest must always be known in advance, thus, important emerging trends in the data may not be shown in faceted interfaces [1–3]. We will see in the practical use cases that the Semantic GrowBag algorithm overcomes some of these problems by automatically deriving faceted categories on the fly and can even reflect trends in the taggings by selecting only parts of the underlying document base.

Another related technology are so-called topic maps [9] that were defined as an XML-based data format (standardized in 1999 as ISO/IEC 13250) to formulate simple structures for knowledge representation. The topic maps defined 'associations' between topics and 'occurrences' that linked topics to documents, e.g., on the WWW. In contrast to our approach, topic maps therefore base on the idea of reflecting more or less static thesauri and indexes like e.g., the topic hierarchy of the Open Directory Project (ODP).

For the (to some degree) automatic creation of ontologies there are several approaches mostly relying on (supervised) learning techniques based on natural language processing, e.g., using language models or syntactic contexts [10, 11]. These approaches identify synonyms, sub-/superclass hierarchies, etc. from full texts by relying on the sentence structure, where phrases like 'such as…' imply a certain hierarchy between terms. Moreover, the belief in the correctness of derived classes and/or hierarchies can be supported by comparison to general ontologies like WordNet or counting co-occurrences, e.g., in documents retrieved by Google. In contrast to our approach these techniques aim to understand the whole information space concerned with a topic and not the most discriminating facets as given by a collection.

Sanderson and Croft [12] have proposed a basic approach to automatically create categorization systems by exploiting keyword co-occurrences. Keyword Y is defined as subsuming keyword X if at least 80% of the objects tagged with X are also tagged with Y, and Y occurs more frequently than X. Although manually determined subsumption relations usually have a much stronger semantics, already this simple subsumption definition has shown a nice performance for navigational purposes [12] when using keywords extracted from full text. However, for manually specified keywords, authors tend to use only 'leaf' categories they have in mind, so two subsuming keywords hardly co-occur in more than 80% of the cases.

3 The Semantic GrowBag Algorithm

The basic idea of the Semantic GrowBag algorithm is to automatically create categorization systems from a corpus of digital objects (like documents, images, etc.) annotated with keywords. This is done by exploiting *higher-order co-occurrence*, known from computational linguistics [13]. As shown in Fig. 2, keywords X and Z are associated with one object (C), which is a *first order co-occurrence* or simply *co-occurrence*. Keywords Y and Z are not associated with the same object, but there may still be a subsuming keyword (X) which is associated with objects (B and C) that are tagged by both keywords Y and Z, respectively. Such *second order co-occurrences* (or generally higher-order co-occurrences) occur more often than first-order occurrences alone

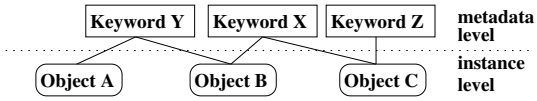

Fig. 2. Example {first|second}-order co-occurrence

and, hence, reduce the sparsity of the co-occurrence dataset. They have also been found to be more robust than first-order co-occurrences, for example, improving word sense disambiguation algorithms [13].

Including higher-order co-occurrences has two main effects: (1) Finding additional ('hidden') relations between keywords which cannot be found using first-order co-occurrences only. (2) Changing the 'strength' (i.e., the number) of existing first-order co-occurrences to include higher-order co-occurrences.

The Semantic GrowBag algorithm uses a biased PageRank algorithm [14] for the computation of the higher-order co-occurrences, as the properties of PageRank are well understood, it can be computed very efficiently and converges to a stable solution for appropriate input data (e.g., in the SimRank approach [15], PageRank has been used in a similar way to compute transitive similarities). Our algorithm comprises the following three steps, which are explained in more detail in the upcoming sections.

I. Compute a new co-occurrence metric including higher-order co-occurrence.
II. Find relations between keywords, based on the new co-occurrence metric.
III. Construct for each keyword i a single GrowBag graph to present a limited view on the 'neighborhood' of keyword i instead of having one manually crafted graph as for legacy classifications/thesauri which covers the whole object collection.

3.1 Part I: Higher-Order Co-occurrences

In a nutshell, computing the co-occurrence metric including higher-order co-occurrences comprises the following three basic steps:

1. Create an (nxn) matrix M based on weighted (first-order) co-occurrence relations for the n keywords.
2. For each keyword i, determine the most often co-occurring keywords (the *direct neighbors*).
3. For each keyword i, compute the Biased PageRank scores using matrix M and biasing on the direct neighbors of keyword i.

The resulting n PageRank score vectors are complementary to the n lists of most often co-occurring keywords, but additionally include higher-order co-occurrences due to the flow of the scores in the PageRank graph. Specifically, the ranking imposed by the scores in the PageRank score vector of keyword i now determines the *hidden related keywords* to keyword i and is, hence, an enhancement of the legacy 'related keywords' notion.[1] The following sections provide more details.

Co-occurrence Matrix M The elements $m(j,i)$ of matrix M are defined as follows:

$$m(j,i) = \frac{cooc(i,j) * ICF(i)}{\sum_j cooc(i,j) * ICF(i)} \tag{1}$$

with $cooc(i,j)$ being the (first order) co-occurrences of keywords i and j, i.e., the number of objects that are tagged with both keywords[2]. The inverse co-occurrence frequency

[1] Latent semantic analysis [16, 17] is very related here, which basically performs a singular value decomposition on the co-occurrence matrix to find 'latent' topics, i.e., abstract topics given by linear combinations of keywords. However, it cannot find subsumption hierarchies.

[2] Using Dice or Jaccard similarity instead of co-occurrences has been found inferior as they give a rather high weight for rarely occurring terms, which does not support the notion of 'emergent' semantics.

(ICF) much resembles the inverse document frequency as known from information retrieval [18] and is defined as follows:

$$ICF(i) = \log(\frac{\text{Overall \# of keywords}}{\text{total \# of keywords co-occurring with keyword } i}) \quad (2)$$

Multiplying the ICF of i to $cooc(i, j)$ in eq. (1) has the effect that $m(j, i)$ is decreased for those keywords i that co-occur with many other keywords. These have, thus, a less discriminating power.

The above defined matrix M has the following properties: It is symmetric in terms of links, i.e., if $m(j, i) \neq 0$ then $m(i, j) \neq 0$. Hence, the PageRank graph defined by M does not contain dangling nodes or rank sinks. Though symmetric in terms of links, it is **not** symmetric in terms of weights: The normalization in the denominator of eq. (1) ensures that $m(j, i) \neq m(i, j)$ since the sum of all co-occurrences is typically not the same for all pairs of keywords. Therefore, the PageRank graph defined by M is a **directed** graph so that the PageRank scores do not converge to the number of co-occurring keywords for each keyword. Hence, matrix M is stochastic, irreducible, and primitive because of normalization and because we apply the Random Jump vector as defined by Page and Brin [14] (with the usual 15% random jump probability). Therefore, the PageRank computation converges to the principal Eigenvector [19].

Direct Neighbors: Top-X To find the direct neighbors for a given keyword t_i, we first sort the vector with all co-occurring keywords for t_i. We then define the direct neighbors as the *top-X elements* of the sorted vector which accumulate the first $P\%$ of the sum of all vector values (i.e., the integral of the co-occurrence graph). P is the essential parameter of our algorithm that controls to which degree higher-order co-occurrence should be included. A typical value found useful in practice is $P \in [10 : 30]\%$.

Biasing on Top-X For computing the biased PageRank scores, we initialize the start vector and the random jump vector with 1 to provide each node in the PageRank graph with some default score. This step is the most costly part of our approach in terms of time complexity, but as the overall set of author keywords will remain stable (they are never changed and changes are relatively small for large object collections), our algorithm can be computed off-line and re-run periodically to update the results according to the added author keywords.

3.2 Part II: Finding Hidden Relations

To find relations between two keywords i and j, the PageRank score vectors for the keywords i and j are used as follows:

1. Sort the power-law distributed PageRank score vectors and cut the tail to filter out all keywords with too low scores.
2. Compare the scores of keywords i and j in those two sorted and filtered score vectors. If both i and j exist in both vectors and either i or j has a higher score in *both* vectors, the one with the higher score is a candidate for subsuming the one with the lower ranks.

3. Post-filter candidates based on their rank in both score vectors and determine the confidence of the final subsumption relation.

The resulting list contains triples (keyword i, keyword j, confidence), denoting that keyword i subsumes keyword j with the given confidence.

As the PageRank scores are power-law distributed for power-law distributed graphs (and co-occurrences are typically also power-law distributed), the tail of the PageRank score vector comprises many keywords which are only very weakly related to the keyword, on whose neighborhood we biased the PageRank computation. Hence, we can safely apply *tail cutting* in step 1 and keep only those elements in the sorted PageRank score vector, which accumulate 80% of the overall score in the score vector (following the well-known 80-20 rule).

In the third step, we apply the following rules to filter too weak relations and determine the confidence of the remaining subsumption relations:

1. If neither keyword i is among the top-X_j elements of the PageRank score vector of keyword j nor keyword j is among the top-X_i elements of the vector of keyword i, then both are assumed to be too weakly related and are deleted from the list of subsumption candidates.
2. If the subsumed keyword is among the top-X elements of the score vector of the subsuming keyword, then we set the confidence in the subsumption relation to 'low' (*weak relation*).
3. If both keywords are among the top-X elements of both score vectors, then we set the confidence in the subsumption relation to 'high' (*strong relation*).

3.3 Part III: Creating GrowBag Graphs

In the third and final part, the Semantic GrowBag algorithm finally uses the hidden related keywords of i as 'seed nodes' and 'grows' the set of nodes to create a specific GrowBag graph for keyword i using the following steps:

1. Add those keywords as nodes that subsume the latent related keywords of i directly.
2. Add recursively all keywords as nodes that are subsumed by already collected keywords.
3. Add all relations as edges where both involved keywords have already been collected. Use different edge visualizations to account for the 'strength' of the relation (dashed lines for weak relations and two-headed arrows for strong relations).

The intention is to visualize only the most important *related neighbors* and the keywords subsumed by them to limit the size of the graph.

4 Use Case and Experimental Evaluation with Practical Data

This section explains the Semantic GrowBag algorithm in detail along one use case for the start keyword 'RDF' and additionally shows how the GrowBag graph for 'RDF' changes over time. We used a subset of the computer science publications listed in

DBLP and extracted the author keywords from the web, post-processed them using acronym replacement and Porter stemming [20] and removed those, which are mentioned less than five times, resulting in $13,200$ keywords. The relation DocumentID $\rightarrow$ keywords comprises about $500,000$ entries for $93,000$ publications.[3]

4.1 Creating a GrowBag Graph for 'RDF'

This use case shows how to find the GrowBag graph for the keyword 'RDF' for the period 2001–2005. The description follows the three main parts of the algorithm to (1) create a co-occurrence matrix including higher order co-occurrences, (2) find relations between keywords, and (3) create a GrowBag graph for the keyword 'RDF'. In out GrowBag demonstrator,[4] all graphs are periodically updated (so the version in the Web might differ from the graphs presented here).

Part I: Top-X and Biased PageRank After creating the co-occurrence matrix M, the algorithm first extracts the corresponding 'RDF' row of M, comprising the weighted co-occurrences of 'RDF' (using the ICF as weight) with all other keywords. This matrix row is sorted according to the matrix values (cf. left part of Table 1; the plain co-occurrence values are shown for comparison.

Rank	Keyword	$m(j,i)$	$cooc(i,j)$
1	RDF	259.8	55
2	Semantic Web	112.1	31
3	Metadata	55.3	15
4	XML	41.4	15
5	Annotation	26.3	6
6	Ontology	26.1	8
7	DAML	23.1	4
8	RDF Schema	20.6	3
9	DAML+OIL	18.0	3
10	OWL	16.0	3
...	...	...	

Rank	Keyword	Score
1	Semantic Web	828.1
2	Metadata	755.0
3	RDF	746.4
4	Ontology	127.1
5	XML	120.2
6	Web Service	74.6
7	Information Retrieval	50.2
8	Data Mining	49.3
9	Clustering	49.3
10	Annotation	49.0
...	...	...

Table 1. Direct neighbors (left) / the PageRank score vector (right) for 'RDF' (top-$X = 3$)

In this example, the direct neighbors of 'RDF' are 'RDF' itself (the start keyword is included by definition), 'Semantic Web' and 'Metadata' as they accumulate $P = 20\%$ of the total sum of matrix values of all keywords in that list.[5]

Afterwards, the algorithm uses PageRank (biasing to 100% on the three direct keywords of 'RDF') to find a new ranking. The resulting *PageRank score vector* includes higher-order co-occurrences, which are not reflected in the sorted co-occurrence vector. Practically speaking, the main objective of the first part is to see whether the start keyword 'RDF' itself remains the 'top keyword' in the score vector after running the

[3] Titles, authors, citations etc. are planned to be included in future versions.

[4] `http://dblp.l3s.de/GrowBag`

[5] This heuristics to find P was confirmed by manual inspection of a large set of resulting graphs but is subject of further research.

Biased PageRank or if other keywords are more relevant and 'overtake' (in the sorted co-occurrence vector, the start keyword stays always on top). As shown in the right part of Table 1, indeed 'Semantic Web' and 'Metadata' achieve a higher score than 'RDF' and are candidates for subsuming 'RDF'.

Part II: Deriving Relations The basic idea of the second part to identify 'subsumption' relations between 'RDF' and its direct neighbors is to do a pairwise comparison of the scores of two vectors: the PageRank score vector of 'RDF' and of its direct neighbors (cf. Table 2).

Rank	Keyword	Score
1	Semantic Web	407.7
2	Ontology	373.0
3	XML	358.4
4	Web Service	330.6
5	RDF	311.4
6	Metadata	303.5
7	Description Logics	288.5
8	OWL	284.3
9	Data Mining	47.8
10	Security	45.6
...	...	...

Rank	Keyword	Score
1	Metadata	262.6
2	XML	125.1
3	Semantic Web	207.1
4	Digital Libraries	199.5
5	Ontology	197.5
6	Interoperability	172.3
7	Annotation	171.8
8	RDF	167.1
9	web	151.1
10	OAI	149.7
...	...	...

Table 2. PageRank score vectors for 'Semantic Web' (left; $X = 8$) & 'Metadata' (right; $X = 16$)

For the direct neighbor 'Semantic Web', the score of 'RDF' is lower than the score of 'Semantic Web' in both the PageRank list of 'Semantic Web' (cf. Table 2 (left)) and the PageRank list of 'RDF' (cf. Table 1 (right)). Hence, 'RDF' is defined to be subsumed by 'Semantic Web' (RDF achieves a sufficiently high score in both lists not to be affected by tail cutting). Since 'RDF' is among the top-$X = 8$ elements of the PageRank score vector of 'Semantic Web', the confidence in this relation is defined as 'strong'. Analogously, 'RDF' is found to be subsumed by 'Metadata'.

Part II is repeated to find subsumption relations between all pairs of keywords.

Part III: Combining the Relations into a Graph For the final graph, the set of keywords N related to 'RDF' and the corresponding edges between the keywords in N have to be found. An excerpt of the resulting graph when starting from 'RDF' is depicted in Fig. 3:

To determine N, the Semantic GrowBag algorithm includes all direct neighbors of 'RDF' in N as initial 'seed', i.e., the start keyword 'RDF' itself (depicted as box with a black background), and the keywords 'Semantic Web', and 'Metadata' (grey boxes). This set is grown by recursively collecting all subsumed keywords in N (transparent boxes). Finally, the immediate 'parents' of the direct neighbors are added to the final set N to put these direct neighbors into their immediate context (which does not add keywords to the graph in this example). To connect the nodes, all known subsumption relations involving keywords from N are added.

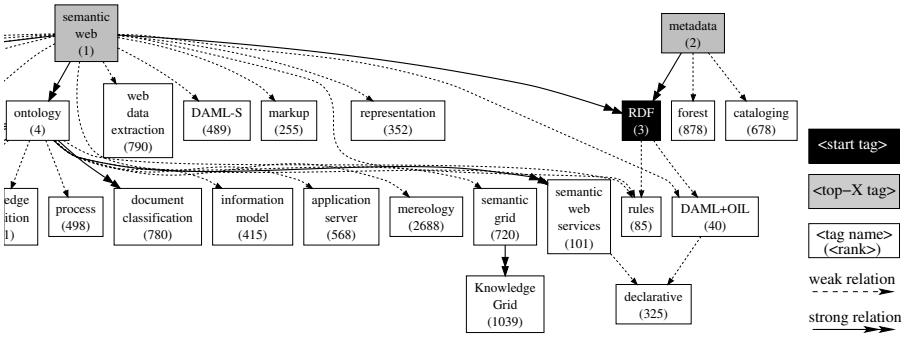

Fig. 3. Excerpt of the GrowBag graph for 'RDF' in the period 2001–2005

Two additional pieces of information are also shown in the graph: First, the Semantic GrowBag algorithm additionally provides a 'confidence' for an edge that depends on the underlying data. Second, all keywords in the graph are also associated with their rank in the PageRank score vector of the start keyword 'RDF', which gives evidence on how closely related the keyword is to 'RDF'. As an example, 'ontology' (rank 4; sibling of 'RDF') has a closer relation to 'RDF' than 'web data extraction' (rank 790; also a sibling of 'RDF').

4.2 Graph Development over Time

Whereas facets are usually considered to be static, the Semantic GrowBag algorithm can also show the development of the categorization over time for our input data. The previous figure depicted the GrowBag graph using all documents in the main period of 'RDF' (2001-2005). In contrast, figures 4 and 5 show the relations restricted to 2002–2003 and 2003–2004 respectively[6].

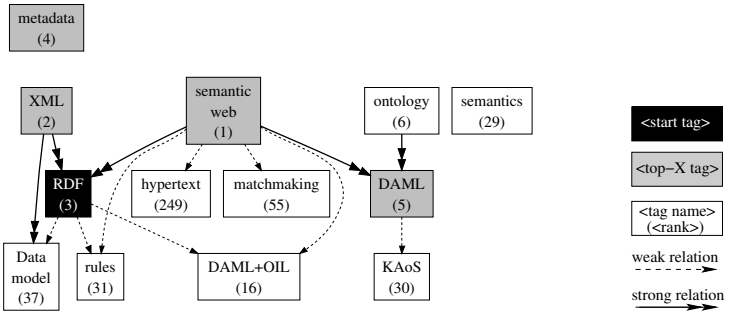

Fig. 4. GrowBag graph for 'RDF' (2002-2003)

In 2002/03, for example, 'Semantic Web' subsumes 'DAML' whereas in 2003/04, 'Semantic Web' subsumes 'OWL'. Very interesting is the relation between 'Semantic Web' and 'Alloy' in 2003/04: Alloy is a modelling language from the Formal Methods community, that has been used to validate Semantic Web ontologies for consistency.

[6] We coupled the data from two years, since the data was too sparse in single years. We omitted the children of XML for space reasons.

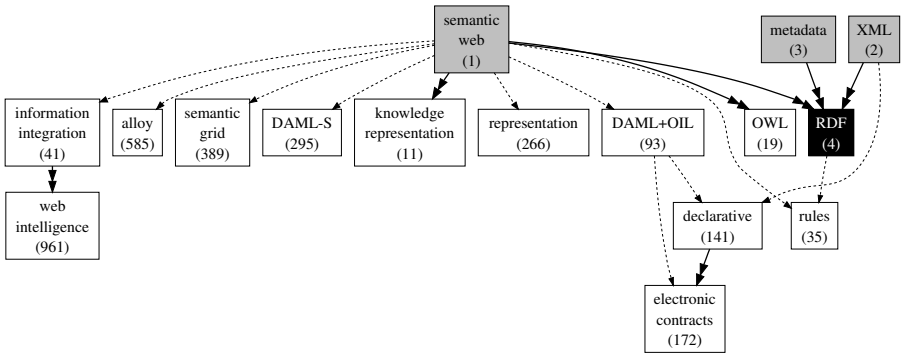

Fig. 5. GrowBag graph for 'RDF' (2003-2004)

4.3 Comparison with the Baseline Approach

In general, the quality of categorization systems is a very subjective issue and can hardly be evaluated automatically. One objective of the Semantic GrowBag approach is finding additional relations between keywords based on higher-order co-occurrences. Hence, we compared the number of relations found by our approach with the relations found using the baseline approach [12] (cf. Sect. 2, to which we also applied the 'tail cutting' to eliminate too shallow relations). Table 3 shows the number of found relations in all GrowBag graphs and for the baseline approach for three different periods.

Approach	2001–2005		1996-2000		2004-2005	
	Relations	Overlap	Relations	Overlap	Relations	Overlap
GrowBag	2150 (929)	n.a.	2494 (988)	n.a.	853 (413)	n.a.
Baseline 51%	2297	96 (63)	2101	414 (168)	1427	63 (35)
Baseline 55%	2297	80 (50)	2101	394 (152)	1427	59 (31)
Baseline 60%	1793	60 (41)	1859	334 (120)	1245	41 (18)
Baseline 70%	727	29 (18)	605	130 (49)	458	19 (8)
Baseline 80%	324	8 (7)	352	58 (20)	232	3 (2)
Baseline 100%	260	5	270	40	215	1

Table 3. Identified relations and overlap for 1996-2000, 2001–2005, and 2004–2005

For the originally proposed 80% threshold in the baseline approach, only about $230 - 350$ relations are found ($15 - 30\%$ compared to GrowBag), of which $8 - 58$ overlap with the relations identified by our approach. Hence, our approach finds additional relations using higher-order co-occurrences compared to the original baseline approach, which assumes that subsuming keywords and the keywords they subsume are used together pretty often (e.g., $> 80\%$) to annotate one resource. This is true for automatically extracted keywords from full text, but in general happens less often for manually assigned keywords.

Furthermore, most of those relations being found by both approaches are actually 'strong' relations as found by the Semantic GrowBag scheme (shown in parentheses in Table 3). This indicates that the distinction between 'strong' and 'weak' relation is reasonable, though there is a high variance between the different periods. As an alternative

way to increase the identified number of relations in the baseline approach, the threshold could be decreased to $51 - 60\%$ (cf. Table 3). However, the overlap in the identified relations between GrowBag and the baseline approach remains still very small.

We also examined the depth of the hierarchies created by GrowBag and the baseline approach with 60% and found that less than 10% of the nodes have a depth larger than one (i.e., have 'grand-children' nodes) with an average depth of 1.1 as opposed to 22% and an average depth of 1.3 in the GrowBag approach. Hence, the introduction of higher-order co-occurrences helps in creating deeper hierarchies as expected.

While this section does not allow to judge the quality of the relations found by our approach, it shows that the same results cannot be achieved with a (simpler) system based on first-order co-occurrences only.

4.4 The Top-X Threshold

Finding an appropriate top-X is important for high quality graphs. In this context, the threshold $P = 20\%$ for the accumulated sum of the PageRank scores was determined empirically for our specific data set. We varied this threshold between 10% and 30%, but already in the 15% case very many keywords have two or less keywords as direct neighbors, so that too few PageRank lists can be computed. In the 25% case, the maximum size of the top-X list becomes too high (up to 49 compared to 34 in the 20% case). This threshold P will be subject of further research and always has to be adapted to the underlying corpus. This is because it influences the quality of the identified relations and depends on characteristics of the particular object collection such as the average number of keywords per object, the average number of co-occurring keywords, etc.

5 Summary and Future Work

In this paper we presented the Semantic GrowBag approach for the automatic creation of hierarchical categorization systems for usage in topic facets. Exploiting existing author keyword annotations, we have strong evidence that the automatically derived facets generally are indeed semantically meaningful. Even at this preliminary stage of applying the algorithm to real world collections (which are often not yet thoroughly annotated) the existing annotations allow to get sufficiently good facets for result presentation. Moreover, we have shown that also the evolution of topics over time can be derived and may provide an added value for collection browsing. This benefit for result presentation has already been demonstrated in the FacetedDBLP prototype [6] that also can be accessed online[7]. It provides a faceted view on the DBLP data using GrowBag graphs to create topic facets (with respect to a chosen time interval).

As future work, we want to further improve the quality of the topic facets, e.g., using a more sophisticated detection of the direct neighbors (i.e., the top-X). Furthermore, we envision a change detection scheme to automatically detect keywords in quickly changing environments. We have also started to look at different input data sets, such as the Medline Database, or data sets stemming from collaborative tagging, which in general require a much stronger pre-processing stage to clean the corpus of keywords/tags.

[7] `http://dblp.l3s.de`

Acknowledgments

This work was partially funded by the European NoE Knowledge Web and the Emmy-Noether Program of the German Research Foundation DFG. The authors gratefully acknowledge the many fruitful discussions with our colleagues.

References

1. Hearst, M.A.: Clustering versus faceted categories for information exploration. Commun. ACM **49**(4) (2006) 59–61
2. Rodden, K., Basalaj, W., Sinclair, D., Wood, K.: Does organisation by similarity assist image browsing? In: Proc. of SIGCHI conference. (2001) 190–197
3. Ross, K., Janevski, A.: Querying faceted databases. In: Proc. of SWDB. (2004) 199–218
4. Weber, A., Reuther, P., Walter, B., Ley, M., Klink, S.: Multi-layered browsing and visualization for digital libraries. In: Proc. of the ECDL. (2006)
5. Diederich, J., Thaden, U., Balke, W.T.: The semantic growbag demonstrator for automatically organizing topic facets. In: Proc. of the SIGIR Workshop on Faceted Search. (2006)
6. Diederich, J., Thaden, U., Balke, W.T.: Demonstrating the Semantic GrowBag: Automatically Creating Topic Facets for FacetedDBLP. In: Proc. of the JCDL. (2007)
7. Cutting, D.R., Karger, D.R., Pedersen, J.O., Tukey, J.W.: Scatter/gather: a cluster-based approach to browsing large document collections. In: Proc. of SIGIR conference. (1992) 318–329
8. Pratt, W., Hearst, M.A., Fagan, L.M.: A knowledge-based approach to organizing retrieved documents. In: Proc. of the AAAI conference, Menlo Park, CA, USA (1999) 80–85
9. Park, J., Hunting, S.: XML Topic Maps: Creating and Using Topic Maps for the Web. Addison-Wesley (2002)
10. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: Proc. of the Conference on Computational Linguistics. (1992) 539–545
11. Cimiano, P., Völker, J.: Text2onto - a framework for ontology learning and data-driven change discovery. In: NLDB conference. (June 2005) 227–238
12. Sanderson, M., Croft, B.: Deriving concept hierarchies from text. In: Proc. of the SIGIR conference. (1999) 206–213
13. Schütze, H.: Automatic word sense discrimination. Comput. Linguist. **24**(1) (1998) 97–123
14. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University (1998)
15. Jeh, G., Widom, J.: SimRank: A Measure of Structural-Context Similarity. In: Proc. of the SIGKDD conference. (July 2002)
16. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. JASIS **41**(6) (1990) 391–407
17. Hofmann, T.: Probabilistic latent semantic indexing. In: Proc. of SIGIR conference. (1999) 50–57
18. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval. ACM Press / Addison-Wesley (1999)
19. Langville, A., Meyer, C.: Deeper inside pagerank. Internet Mathmatics **2**(1) (2004) 335–380
20. Porter, M.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137