Recommending Structure in Collaborative Semistructured Information Systems

Eva Zangerle Wolfgang Gassler Günther Specht

Databases and Information Systems
Institute of Computer Science
University of Innsbruck, Austria
{firstname.lastname@uibk.ac.at}

ABSTRACT

Semistructured data provides the users of a community-based information system with the flexibility to store information without having to adhere to any predefined, rigid schema. However, such flexibility needs to be used with caution as it can lead to a very heterogeneous data structure and is therefore not feasible in terms of unified data access and search functionality. We present an approach which avoids such proliferation of substructures and provides the inserting user with recommendations, which are responsible for the creation of a commonly used structure. The presented recommendation algorithm adapts the recommendations to the stored information and its structure created by the community.

Categories and Subject Descriptors

H.3.5 [**Storage and Retrieval**]: Online Information Services; H.4.m [**Information Systems**]: Miscellaneous

General Terms

Algorithms, Design, Human Factors, Experimentation

Keywords

Semistructured Data, Recommendations, Structure, Human Interaction, RDF

1. INTRODUCTION

Storing information in an online, collaborative system is often realized by wiki systems. These systems provide means to easily add, modify and delete information, which does not have to adhere to any predefined schema or structure. In contrast, traditional (relational) databases are strictly-structured. They would not be able to cope with such a large amount of collaboratively created information with very heterogeneous structures and schemata like for example Wikipedia does. Nevertheless, strictly-structured databases provide the big advantage of structured access, which enables complex query capabilities. Traditional wiki systems only support full-text search which is not feasible for complex queries such as "Which Austrian cities have more than 10.000 inhabitants and have a female mayor who has a doctoral degree?". Weikum *et. al.* [10] observed that information systems have to be able to support both structured and unstructured data to combine the advantages of both worlds and be able to answer such questions.

Semistructured data provides mechanisms for the combination of both unstructured and structured storage of data. Such semistructured data features a structure without having to specify a fixed schema. The most popular example is RDF, which consists of triples containing a *subject*, a *property* and a *value*. Infoboxes within Wikipedia articles are perfect examples of such semistructured data, which can also be extracted as RDF triples. These infoboxes are manually created, tabular aggregations of the most important facts within an article and consist of multiple properties and according values. For example an infobox about New York City contains the property-value pairs `area metropolis: 468.9 sq mi` and `elevation: 33 ft`. These property-value pairs – together with the article URL itself – constitute RDF-triples. Such triples are machine-processable, thus allowing for structured access and structured queries.

The problem of collaborative semistructured information systems without any restrictions is the proliferation of substructures and schemata. Every single user has his own view of structuring information and uses his own terminology. Furnas *et. al.* [3] showed that two people would spontaneously choose the same word for an object with a probability of less than 20%. The proliferation of structures, schemata and vocabulary results in a very heterogeneous schema and therefore impedes a common schema, which is essential for structured access and complex queries. Even Wikipedia has to cope with this problem. According to Boulain *et. al.*, only 35% of all edits within Wikipedia are related to content, whereas all other edits are concerned with the structure of articles to avoid proliferation. Moreover, Wu and Weld [12] showed that even schemata of template-based infoboxes – which are supervised and enforced by the community – are divergent and noisy. Wu and Weld evaluated Wikipedia infoboxes and found that 25% of all templates have less than five instances, 11% have only one instance. Additionally, only 46% of all attributes are used by at least 15% of template-instances. These facts imply that even with the support of a huge commited community, the proliferation of schemata within a multi-user system cannot be prevented.

RecSys2010, September 26–30, 2010, Barcelona, Spain.

We present the recommendation algorithm of *SnoopyDB*, a novel information system, which is based on semistructured data and is able to avoid the proliferation of schemata by providing structure recommendations to the user during the insertion process.
The remainder of the paper is organized as follows. Section 2 describes the basic Snoopy concept. Subsequently, the algorithm for the computation recommendations is described in Section 3. Section 4 covers the conducted evaluations. Section 5 describes related work and Section 6 concludes this paper with an outlook.

2. SNOOPY CONCEPT

The Snoopy concept [4] offers the same flexibility as wiki systems but at the same time provides the possibility to structure information like (relational) databases. In the Snoopy concept, information about a certain subject is stored as a *collection*, similar to a wiki page. A collection consists of an arbitrary number of property-value pairs, which can be specified by the user without restrictions of any kind. To cope with the mentioned proliferation of structures in such semistructured data, the Snoopy concept pushes the part of alignment of data and schemata to the inserting user and supports the user during the insertion and alignment process using a self-adapting schema system and recommendations. The Snoopy concept "snoops" as much information as possible from the user (structure, relations, semantics, etc.), who has extensive knowledge about the subject and can therefore resolve many issues on the fly during the insertion process. One of the main parts of the concept is the structure recommendation algorithm, which adapts itself to the already stored information in the system and guides the user in the alignment process to a common, homogeneous schemata.

3. STRUCTURE RECOMMENDATIONS

The goal of structure recommendations is to provide the user with appropriate property recommendations when adding property-value pairs to a certain subject. Structure recommendations guide the user to apply an implicit, common schema and therefore enable a homogeneous schema throughout the whole system. For the computation of these recommendations all previously entered properties are taken into account and common substructures, which consist of multiple properties which are frequently used together on the same subject, are detected. Consider the example of a user entering information about cities. The property-value pairs describing the *name* of the city and its *mayor* have already been entered. Multiple other users, who already entered information about cities, also used these properties and additionally specified the *population* and *elevation* of the respective city. Therefore, the recommendation mechanism provides the user with recommendations concerning further properties, in this case e.g. the *population* and *elevation* of the city. Such a recommendation system points the user to frequently used properties and – given the acceptance of such recommendations – leads to a common schema within the subjects.

3.1 Definition

Recommendations are based on all already stored information: Let $\mathcal{P} = \{p_1, p_2, p_3, ..., p_n\}$ denote the set of properties and $\mathcal{S} = \{S_1, S_2, S_3, ...S_m\}$ the set of all subjects defined within the system. For each subject $S_i \in \mathcal{S}$, the set of properties occurring within this specific subject is $\mathcal{P}_{S_i} \subseteq \mathcal{P}$, where $\bigcup_{S_i \in \mathcal{S}} \mathcal{P}_{S_i} = \mathcal{P}$. In our case, the recommendation to add a certain property to a certain subject can be seen as a collaborative filtering system, where for each subject S_i, recommendations of properties are computed based on all subjects stored within the system.

3.2 Recommendation Computation

The computation of recommendations is based on all pairs (p_a, p_b) where $p_a, p_b \in \mathcal{P}$ and both p_a and p_b occur within the same subject S_i. These pairs are stored as *rules*, which have the form $r = (p_a, p_b, c)$ and are contained in the set $\mathcal{R}$. Such a rule denotes that property p_a and property p_b co-occur on c subjects. This way of storing pairs as rules eliminates duplicate pairs and provides a compact and fast storage facility. Based on these rules, the recommended properties for a certain subject S_i are computed by selecting a subset $\mathcal{C} \subseteq \mathcal{P}$ of property recommendation candidates by determining all rules which feature properties occurring on the subject S_i, as shown in Algorithm 1. It is important to note that the computation of recommendations is an iterative process – as soon as a new property is added to the subject, rules are created or updated based on this new set of properties.

Algorithm 1: Rec. Candidate Computation

Input: $\mathcal{P}_{S_i}, \mathcal{R}$
Output: set $\mathcal{C}$ of all recommendation candidates for S_i
$\mathcal{C} \leftarrow \varnothing$
$\mathcal{T} \leftarrow \varnothing$
foreach $p_i \in \mathcal{P}_{S_i}$ **do**
  foreach $(p_a, p_b, c) \in \mathcal{R}$ **do**
    if $(p_a == p_i \wedge p_b \notin \mathcal{P}_{S_i})$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \{(p_a, p_b, c)\}$
    end
  end
end
foreach $(p_a, p_b, c_i) \in \mathcal{T}$ **do**
  if $(\exists (p_x, c) \in \mathcal{C},\ where\ p_x == p_b)$ **then**
    $\mathcal{C} = \mathcal{C} \backslash \{(p_x, c)\}$
    $\mathcal{C} = \mathcal{C} \cup \{(p_x, c + c_i)\}$
  else
    $\mathcal{C} = \mathcal{C} \cup \{(p_b, c_i)\}$
  end
end
return $\mathcal{C}$

3.3 Ranking

The execution of the recommendation candidate computation algorithm listed in the previous section results in a set of recommendation candidates $\mathcal{C}$. These candidates are basically pairs (p_b, c) where p_b is the recommended property and c is the number of co-occurrences with properties in the same subject. This probably large set $\mathcal{C}$ cannot be shown to the user in its entirety. Due to this fact, the use of a subset of $\mathcal{C}$ of about 10 properties has proven to be reasonable as it can be perceived completely by the user. To present the 10 most appropriate properties of $\mathcal{C}$, a ranking has to be computed for all elements of $\mathcal{C}$. Basically, the count value c which is computed for all recommendation candidates (see Algorithm 1), is used to rank each candidate. The higher

the count value, the higher the frequency of occurrence of the rule and therefore the co-occurrences of properties p_a and p_b and therefore the higher the rank of the property.

4. EVALUATION

The presented recommendation algorithm is evaluated using a test set based on Wikipedia infoboxes. The SnoopyDB system uses semistructured information and guides the inserting user by recommendations. However, as SnoopyDB cannot provide a large data set yet, we used the Wikipedia infobox set, as it is very large and features similarities to data sets of a semistructured and guided information system. Wikipedia does not impose any restrictions on structure or schemata, the better part of the data adhere to homogeneous schemata which are enforced by a committed community. The schemata, which have been grown with the Wikipedia system, can be compared to recommendation based evolved schemata. However, these schemata are not completely homogeneous [12] and feature a noisy collaborative style and therefore are best suited for our experiments.
For the evaluation of this approach, we used a DBpedia dump (2010-03-16) [2] containing all 4,000,000 infobox instances (about 41 million triples) of the English Wikipedia page. We randomly chose 150,000 instances (10%) out of about 1.5 million instances which have at least 6 distinct properties for our test set. Based on the remaining set of 3,850,000 instances (38 million triples), we computed about 486 million rule instances as described in Section 3.2. The reduced rule set of 7.8 million distinct rules (about 250 MB disk space including the primary index, MySQL 5.0 InnoDB engine) provided a basis for all further recommendation computations. We conducted a leave-one-out test on the 150,000 infobox instances by randomly choosing three properties within every infobox and removing all other properties from the infobox. Based on these three properties, Algorithm 2 was applied to each subject of our test set.

Algorithm 2: Test Algorithm (user simulation)

$\mathcal{P}_{cur} \leftarrow \{p_1, p_2, p_3\}$ //*remaining three properties*
$\mathcal{P}_{rem} \leftarrow \{p_4, p_5, \ldots, p_n\}$ //*removed properties*
$\mathcal{P}_{rec} \leftarrow top10rec\,(\mathcal{P}_{cur})$ //*get top* 10 *rec. properties*
while $((\mathcal{P}_{rec} \cap \mathcal{P}_{rem}) \neq \varnothing)$ **do**
  $p \leftarrow random\ c \in (\mathcal{P}_{rec} \cap \mathcal{P}_{rem})$
  $\mathcal{P}_{cur} \leftarrow \mathcal{P}_{cur} \cup \{p\}$
  $\mathcal{P}_{rem} \leftarrow \mathcal{P}_{rem} \setminus \{p\}$
  $\mathcal{P}_{rec} \leftarrow top10rec\,(\mathcal{P}_{cur})$
end

The test algorithm computes the top-10 ranked property recommendations based on the three remaining properties. If the top-10 recommendations contain a previously removed property, the recommendation is accepted and the property is added to the set of current properties, which is then the basis for further recommendation computations. This step is repeated until the top-10 recommendation set does not contain removed properties anymore or the subject is fully reconstructed with respect to the original infobox. Properties occurring on less than five subjects – the so-called long tail of the property distribution – are neither considered for recommendations, nor for the evaluation. Based on this algorithm, we evaluated the following metrics on the test set.

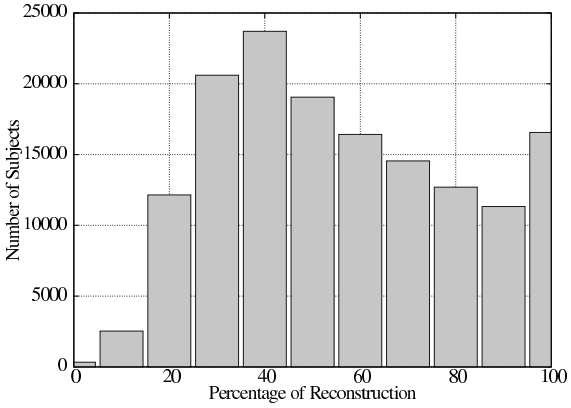

Figure 1: Reconstruction of Subjects (top-*n*, *n*=10)

Reconstruction denotes to which extent an infobox instance can be reconstructed using Algorithm 2 (see Figure 1). Out of the 150,000 subjects, 16,565 subjects were fully reconstructed. Overall, 90,630 subjects were reconstructed to more than 50%, which amounts to 60% of all subjects.
Precision and *Recall* were calculated using the first top-10 recommendation set, which is based on the remaining three properties (denoted by *first*). Furthermore, the average precision and average recall of all iteratively computed top-10 recommendation sets (denoted by *all*) was calculated. Precision and recall are defined as $precision\,(\mathcal{P}_{rec}) = \frac{|\mathcal{P}_{rec} \cap \mathcal{P}_{rem}|}{|\mathcal{P}_{rec}|}$ and $recall\,(\mathcal{P}_{rec}) = \frac{|\mathcal{P}_{rec} \cap \mathcal{P}_{rem}|}{|\mathcal{P}_{rem}|}$ respectively, where $\mathcal{P}_{rem}$ are the previously removed properties and $\mathcal{P}_{rec}$ is the set of top-10 recommendations. We conducted our evaluations using the top-n recommendations with n = 5, 10, 15, 20 and 25 on all 150,000 subjects. The resulting *reconstruction*, *precision* (first and all) and *recall* (first and all) values are shown in Figure 2. The results show that the reconstruction rate cannot be increased by setting the number of recommendations to a value higher than 10. The higher the number of recommendations, the more rises the recall (first) value and the more decreases the precision (first). The overall recall and precision stay constant, which can be led back to the higher precision of recommendations based on subjects, which have already been reconstructed for the most part.
Our experiments and results show that only three properties inserted by the user are sufficient to guide the user to a common schema by recommendations. Even in a very large and collaboratively created data set, like our Wikipedia dataset, the algorithm is stable and features a precision of over 40% for all 800,000 top-10 recommendation sets in the test run. This means that *at least 4 out of 10* recommendations are appropriate, the remaining 6 can be inappropriate or also adequate but are note used in the respective Wikipedia article.

The presented experiments evaluate the automatic recommendation of structure without any user interaction. However, the Snoopy concept provides additional recommendations while the user is typing. Consider a user who specifies the first character of a property, e.g. "A". This information can cut down the number of suitable recommendations dramatically. Furthermore, by using a thesaurus, synonyms can be matched and a more commonly used synonym can be recommended to the user. Consider a user who enters

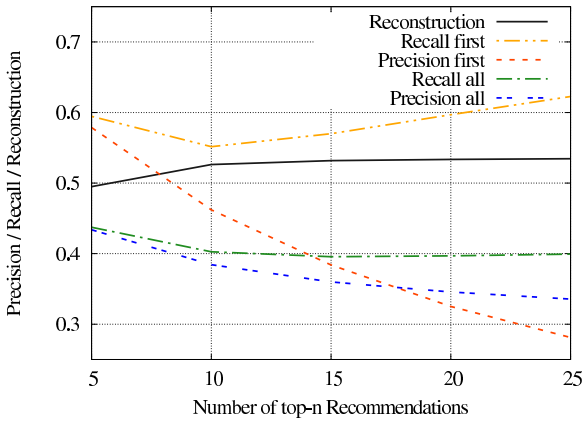

Figure 2: Precision, Recall and Reconstruction

citizens as a property. The system recommends the usage of *population* and by accepting this recommendation, the user contributes to a more homogeneous schema. All these user input-based recommendations heavily increase the recall and precision values but cannot be tested within the presented test scenario, as real interaction of a human user is required. Some preliminary results of such user-based tests can be found in [4] and show the acceptance of such recommendations and a first prototype of the concept.

5. RELATED WORK

Various approaches by the recommendation, data mining and semantic web communities are relevant for our approach.

The detection of frequent patterns within data has been studied extensively [6, 5]. The detection of substructures within semistructured information has been studied in [1, 7]. Such common substructures can be used for unified access. However, the structure of data is not changed or improved in any way. The computation of CF-based recommendations for web personalization based on Association Rules has been facilitated in [8]. However, our approach does not compute Association Rules based on an already existing data basis, the rules are stored during the insertion process.

There are approaches aiming at the storage of semantic information within wiki systems. Semantic Wikipedia [9] extends the wiki markup language and introduces annotated links and attributes within wikis to increase the machine-readability of data. However, the user is not supported during the specification of this additional information in any way. The "Intelligence in Wikipedia project" [11] aims at extending infoboxes in Wikipedia. Missing attributes within infoboxes or even missing infoboxes are detected and after this step, the system attempts to complete this missing information. Therefore, the project features Kylin, a self-supervised information extraction system which gathers information from various sources. This information is then verified by explicit community feedback.

To the best of our knowledge, SnoopyDB is the only approach which has tried to contribute to a common schema within triple-like data by using recommendations already during the insertion process.

6. CONCLUSION

We presented a self-adapting recommendation algorithm to ensure common and homogeneous structures within collaborative semistructured information systems. The recommendations are provided to the user during the insertion process and guide the user to a common schema without restricting the user or the structure of information. These recommendations are computed by detecting common substructures in already collaboratively created information. The evaluation on a very large dataset of 4 million subjects showed that at least 4 out of 10 recommendations are appropriate and the algorithm is able to guide the user to a common schema after having inserted only three properties. The resulting homogeneous data contributes to a common data access and therefore enables complex queries on collaboratively created information.

7. REFERENCES

[1] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proceedings of the Second SIAM International Conference on Data Mining*, pages 158–174, 2002.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. *Lecture Notes in Computer Science*, 4825:722, 2007.

[3] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):971, 1987.

[4] W. Gassler, E. Zangerle, M. Tschuggnall, and G. Specht. Snoopydb: narrowing the gap between structured and unstructured information using recommendations. In *HT '10: Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 271–272, New York, NY, USA, 2010. ACM.

[5] B. Goethals. Survey on frequent pattern mining. Manuscript, 2003.

[6] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

[7] T. Miyahara, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tree structured patterns in semistructured web documents. In *PAKDD '01: Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 47–52, London, UK, 2001. Springer-Verlag.

[8] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on Web information and data management*, page 15. ACM, 2001.

[9] M. Voelkel, M. Kroetzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic wikipedia. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 585–594, New York, NY, USA, 2006. ACM.

[10] G. Weikum, G. Kasneci, M. Ramanath, and F. Suchanek. Database and information-retrieval methods for knowledge discovery. *Commun. ACM*, 52(4):56–64, 2009.

[11] D.S. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, and M. Skinner. Intelligence in wikipedia. In *Twenty-Third Conference on Artificial Intelligence (AAAI'08)*, 2008.

[12] F. Wu and D. S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 635–644, New York, NY, USA, 2008.