Future Generation Computer Systems 31 (2014) 111–119

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Guided curation of semistructured data in collaboratively-built knowledge bases

Wolfgang Gassler *, Eva Zangerle, Günther Specht

Databases and Information Systems, Institute of Computer Science, University of Innsbruck, Technikerstrasse 21A, 6020 Innsbruck, Austria

HIGHLIGHTS

- Avoid proliferation of structures in the knowledge base.
- Semantic refinement by resolving homonyms and avoiding synonyms.
- Exploit user's extensive and valuable knowledge.
- Increase the quantity of information contained in the knowledge base.
- Increase the quality of information in the knowledge base.

ARTICLE INFO

Article history:
Received 18 January 2012
Received in revised form
3 May 2013
Accepted 17 May 2013
Available online 28 May 2013

Keywords:
Collaboration
Semistructured data
RDF
Recommendations
Schema proliferation
Mixed-initiative

ABSTRACT

The collaborative curation of semistructured knowledge has become a popular paradigm on the web and also within enterprises. In such knowledge bases a common structure of the stored information is crucial for providing efficient and precise search facilities. However, the task of refining, extending and homogenizing knowledge and its structure is very complex. In this article we present two paradigms for the simplification of this task by providing guidance mechanisms to the user. Both paradigms aim at combining the power of automated extraction algorithms with the semantic awareness of human users to accomplish this refinement task.

1. Introduction

Large knowledge bases have always been built in a collaborative fashion to collect and archive the common knowledge of groups. Especially with the enormous growth of the internet and the web 2.0 movement, collaboration has been lifted to a new level—online mass-collaboration. One of the most relevant and popular knowledge bases is Wikipedia, which is also based on this mass-collaboration paradigm and is the most important representative of the wiki-concept. The wiki-concept constitutes one of two traditional major paradigms to store information as it stores information as fulltext without any explicit structure. The second paradigm is the storage in (relational) databases which forces the user to store knowledge or information according to a strict and predefined schema. The advantage of such an approach is that all information has to adhere to the same schema and can be searched and presented in a very uniform and hence efficient way. This paradigm is still used in information systems which are focused on one single domain where the contained items are all structured the same way, e.g., a database of movies and reviews. The big disadvantage of this storage paradigm is the unsatisfactory flexibility regarding new structures or content. Changes to the schema are a very time-consuming, tedious and complex task, as it has to be performed manually and the already stored information has to be adapted to the new structure. Thus, the end-user is fixed to a given schema and cannot insert additional information not matching the given schema. For flexible knowledge bases, especially in the area of mass-collaboration, such a restrictive approach is hardly suited, as it can result in a significant loss of information because of the fact that the user cannot insert all information she might want to. This problem was solved by wikis and was one key of success of wiki-systems. Wiki-systems store content in an entirely unstructured manner and can therefore hold any textual information regardless of its structure. Therefore, users do not need to adapt their knowledge to any predefined

* Corresponding author. Tel.: +43 512 507 96892; fax: +43 512 507 96955.
E-mail addresses: wolfgang.gassler@uibk.ac.at (W. Gassler), eva.zangerle@uibk.ac.at (E. Zangerle), guenther.specht@uibk.ac.at (G. Specht).

0167-739X/$ – see front matter
http://dx.doi.org/10.1016/j.future.2013.05.008

schema and are able to insert all information and knowledge they want to. The shortcoming of this structure-less paradigm is its limited search capability. Consider a complex query such as "Which Austrian cities have more than 10,000 inhabitants and have a female mayor who has a doctoral degree?". It is not possible to answer such a query through full-text search which is provided by most wiki-systems. Weikum et al. [1] observed that modern 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.

The rest of the article is organized as follows. In Section 2, we describe the characteristics of semistructured data. Section 3 contains a description and motivation of the problem which is tackled in this paper. In Section 4, approaches dealing with the problem of refinement after the data has been inserted are described. Section 5 outlines the main idea behind the Snoopy Concept which is a representative of an alignment and refinement of knowledge and structure during the insertion of data. Section 6 concludes the article and describes future work.

2. Semistructured data

The *semistructured data model* incorporates both the paradigm of structured and unstructured storage. The need for such a new storage paradigm already arose in the 90s [2,3]. Back then it became clear that it would be increasingly important to be able to store mostly unstructured data while at the same time providing efficient and structured querying facilities. With the advent of the World Wide Web, which currently forms the largest unstructured knowledge base, it became obvious that such data cannot be fitted into a predefined schema in order to be able to query it. The application of traditional retrieval and extraction techniques to query such unstructured data reached unsatisfactory results as the formulation of structured and precise queries was not possible due to the lack of structure. Thus, the semistructured data model combines both the structured and the unstructured data model and provides a highly flexible way of storing data as it supports the storage of information in a structured way without the need of specifying a predefined schema.

Throughout the last two decades, various models for semistructured data have been developed, like e.g. [4,5]. Currently, the most popular example of the semistructured data model is RDF (Resource Description Framework, W3C recommendation[1]) [6]. RDF basically models knowledge as triples consisting of a subject, a predicate and an object. The subject (also called the resource) is described by multiple pairs of predicates and according objects. The resource is uniquely identified by a URI (Uniform Resource Identifier[2]). Important facts about the University of Innsbruck within a knowledge base can be stored using triples as e.g. in Listing 1.

```
<http://dbpedia.org/../University_Innsbruck>
  <numberOfStudents><26626>
<http://dbpedia.org/../University_Innsbruck>
  <established><1669>
```

Listing 1: Triples.

These predicate–object pairs – in combination with the article URI itself (the resource, in this case University_Innsbruck) – constitute triples. The subjects, predicates and objects are not restricted in any way and can therefore hold any information while at the same time providing structure due to the triple concept as all objects are given context by the according predicates. The triples are machine-readable and processable and thus provide the base for structured access and complex structured queries. In order to query such semistructured RDF knowledge bases, the standard query language is SPARQL (SPARQL Protocol and RDF Query Language) [7].

As RDF is very flexible and is able to link to even external resources by specifying an external URI as the object of a triple, the interlinking between knowledge bases has become very popular. Sir Tim Berners-Lee has coined the term Linked Open Data (LOD) [8] for such linked and semistructured data.

It is important to note that within semistructured systems, users can arbitrarily choose the predicate used for storing information. This fact is very beneficial as it provides a huge amount of flexibility to the users of the system while at the same time – due to the predicate–object format – still features a certain amount of structure. This fact is crucial in online, mass-collaboration information systems, as there are thousands of different users who come from different social levels, backgrounds and edit information of different domains and contexts.

Beside many other approaches aiming at raising the amount of structural information in wiki-systems, RDF increasingly gained ground in many wiki-systems or plugins [9] in order to lift wikis from unstructured black holes to structured knowledge bases. Even Wikipedia articles already contain semistructured data. The tabular aggregation of the most important facts about an article – so-called infoboxes – which are located on the right hand side of many articles were originally not intended for structured and computer-readable access. However, these are now extracted and used as a base for the most important open semistructured knowledge base DBpedia [10], which consists of more than one billion triples extracted from Wikipedia's huge collaboratively built knowledge base.

3. Problem description

The flexibility of the previously described schemaless semistructured storage in combination with collaborative data curation leads to a massive problem. Every single user has her own view of structuring knowledge and information and uses her own terminology. Furnas et al. [11] already showed in the 80s that two people would spontaneously choose the same word for an object with a probability of less than 20%. This suggests that collaboratively built knowledge based on the semistructured model shows a very high proliferation of structures, schemata and vocabulary. The resulting heterogeneous schema impedes the search facilities as a common schema is essential to answer complex structured queries. For example, a user who searches for the value of *numberOfStudents* cannot find information which was stored using the properties *students*, *numberStudents* or *num_students*. Therefore, especially in collaborative knowledge systems, the creation of a common schema without restricting the domain, type or amount of information is desired. Wikipedia is fighting such heterogeneity by introducing collaboratively created templates and the supervision by the committed community. The task of creating structure in Wikipedia is very demanding, which is shown by Boulain et al. [12]. The authors analysed the edits in Wikipedia and identified that only 35% of all edits within Wikipedia are related to content, whereas all other edits aim at enhancing the structure within the Wikipedia knowledge base. Additionally, Wu and Weld [13] showed that infoboxes which adhere to predefined templates are still divergent and noisy.

Another problem in collaboratively built knowledge bases is the barrier for new users to insert information to knowledge bases. In Wikipedia most of the content is created by a very small group of users. Furthermore, articles have to conform to many policies and other regulations which increase the barrier for contributions by newcomers [14,15]. But not only in public knowledge bases, moreover and especially in enterprise knowledge bases or wikis,

[1] http://www.w3.org/RDF/, last accessed 2012-10-09.
[2] http://www.w3.org/TR/uri-clarification/, last accessed 2012-10-09.

the poor adoption rate of content constitutes a major problem. Besides social group phenomena, especially the high costs for users to contribute and manage wikis (e.g. wiki syntax, complex user interfaces, etc.) is the main reason for a bad contribution rate [16].

Thus, the main goal is to provide a very intuitive and guided process to the user to curate semistructured knowledge with common and normalized schemata in collaborative systems. This goal can be achieved by using self-learning and optimized guidance systems which improve the following two processes (also sketched in Fig. 1):

- Guided refinement and enrichment of stored knowledge (see the right box in Fig. 1).
- Guided insertion of knowledge (see the left box in Fig. 1).

The traditional approach to enrich, refine and extend knowledge is to apply information extraction algorithms to the content of external sources to extract new semistructured knowledge. As this task is very error-prone, new mixed-initiative approaches combine extraction technologies with the knowledge of the collaborating users. Another new approach encourages the user to create semistructured knowledge already during the insertion of information. The inserting user is supported during the insertion process to create semistructured data which adheres to a common schema by providing recommendations. These recommendations are based on already collaboratively created knowledge and guide the user without restricting her in regard to the amount and type of information. Both approaches are described in detail in the following sections and common representatives of the approaches are introduced.

4. Guided refinement and enrichment

The main idea of the following approaches is the collaborative refinement of already existent semistructured data. The possible refinements, alignments and extensions of already stored knowledge are mostly computed based on external sources. Such sources can be other structured or semistructured knowledge bases. But even the web – the biggest available knowledge base – can be exploited to gather new refinements of already stored knowledge. Especially the exploitation of the web is very challenging due to its size, the unstructured format and the associated scaling and performance issues. Approaches which use external sources to compute refinements and subsequently rely on a collaborative review of the detected refinements (sketched in Fig. 1 (right box)) are described in the following sections.

4.1. Alignment and refinement

As already carved out in the introductory section, collaboratively curated data may also be refined and aligned *after* the insertion of data. This is a crucial step in order to provide efficient querying facilities. Such an alignment mostly aims at homogenizing the set of predicates to a common schema.

During the last years, various approaches for aligning and matching RDF data have been developed. Hausenblas and Halb use a manual approach [17] that enables users to collaboratively interlink resources within RDF data. Beside such a manual alignment, also automatic alignment methods have been developed. The method proposed by Horrocks [18] is based on common naming schemata, namely the comparison of properties within the datasets. Further approaches only rely on string-matching, e.g., for the DBpedia Lookup service.[3] Such alignment services may also be based on context information of the entities which have to be aligned, e.g., based on the geographic coordinates, data types, etc. The Silk project [19] aims at combining these different alignment techniques to interlink RDF datasets.

4.2. Knowledge harvesting and information extraction

The process of information extraction or knowledge harvesting aims at scanning the unstructured text of arbitrary documents and extracting structured knowledge to extend or build large knowledge bases. The retrieval of facts from natural language sources, such as unstructured web documents, is mostly based on NLP-techniques. Natural Language Processing (NLP) [20] is basically concerned with how a computer system can understand natural language in order to gather the sense of a sentence. By doing so, computers are able to summarize texts, detect entities and to extract certain facts from a text.

The most popular semistructured dataset is DBpedia [10] which is built up by extracting information from Wikipedia articles. Especially the pieces of information in infoboxes, which are located on the right hand side of many Wikipedia articles and contain tabular summaries of the most important facts about the article, are extracted and converted to a semistructured format. DBpedia provides collections of RDF triples which are also interlinked with many other LOD datasets. The current version of DBpedia contains more than 1,000,000,000 triples[4] describing 3.7 mio. entities.

Furthermore, DBpedia is enhanced by information taken from the YAGO knowledge base. The YAGO knowledge base [21,22] is a huge ontology which contains semantic information in the form of subject–predicate–object triples. The ontology is built up by extracting information from Wikipedia and enhanced by the taxonomy of WordNet [23]. The YAGO ontology aims at reaching high confidence due to a hand-picked set of rules and therefore is constrained to roughly 100 manually defined types of relations. The current version of YAGO, YAGO2 [22] is further enhanced with spatial and temporal information. It contains 120 million triples describing about 10 million entities.

Although YAGO and DBpedia extract information from the plain text of articles, the extraction method is very limited as it is based on fixed patterns which are focused on a very small amount of the available information in an article—either infoboxes or category information. Therefore, the following approaches extend the simple pattern matching to more complex matching algorithms or NLP-techniques, which analyse the grammatical structure of sentences.

For an automated creation of large knowledge bases, the manual creation of patterns is not feasible. Therefore, ground truth data or other knowledge bases are used to create patterns which are learned and refined automatically. Suchanek et al. [24] introduced SOFIE which uses the knowledge base YAGO to find new patterns. This is accomplished by searching already known information (trusted facts) in natural language documents. Consider the known fact that Einstein was born in Ulm which is stored in a triple <Einstein><bornIn><Ulm>. If many documents contain the sentence "Albert Einstein was born in Ulm" the pattern "X was born in Y" can be derived for finding values for the property *bornIn*. After this step, the pattern can be used to harvest new *bornIn*-information of already known persons (X) and cities (Y) in arbitrary natural language documents. Nakashole et al. [25] showed that such an approach is also feasible for the web in terms of scalability.

Wu et al. [26] introduced an approach to find missing values of infoboxes in the natural language text of Wikipedia articles as a part of their "Intelligence in Wikipedia" project. The extraction process is accomplished by the Kylin extractor. This extraction process is started by a preprocessing step which is responsible for the creation of a training set for the extractor. This is done by

[3] http://lookup.dbpedia.org/.

[4] http://blog.dbpedia.org/2011/09/11/dbpedia-37-released-including-15-localized-editions/.

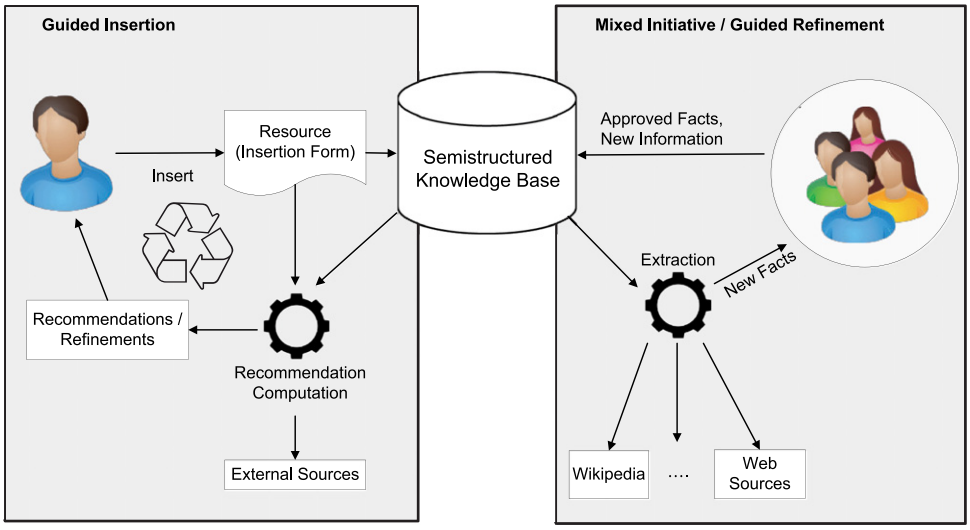

Fig. 1. Overview of the two presented approaches.

retrieving the most popular attributes from pages which make use of the same infobox template. For each of these attributes, standard NLP techniques are used to label a matching sentence which contains the attribute's value. The extraction of facts for Wikipedia infoboxes is done by learned extractors based on conditional random fields.

The result of all described approaches consists of new semi-structured facts. These facts feature different confidence values and, therefore, require an additional review process to dismiss invalid or wrong facts and confirm true facts. This review process is described in the following section.

4.3. Review process/mixed-initiative

Knowledge bases, such as YAGO or DBpedia described in the section above, provide reliable knowledge with a very high confidence as they are based on very limited patterns which are optimized for a single source like Wikipedia infoboxes. To go beyond this scope, new approaches introduce more flexible patterns and extractors such as Kylin or SOFIE described above. However, this flexibility implies higher error rates and lower confidence which have to be tackled. This can be realized by automatic reasoning or by a collaborative review system which exploits the human resources and knowledge of a large community. Automatic reasoning is able to dismiss many wrong facts by using logical rules [25,24]. For example, it is not possible that a doctoral advisor is more than 100 years older than the Ph.D. student. The precision (correct found facts) of reasoning enhanced systems can be up to 98% [25,24] if the underlying knowledge base is comprehensive and the searched relation is clear and without ambiguity. Especially semantic meanings and ambiguities decrease precision dramatically. For example, the ambiguity of city names in the United States for a *bornIn* relation is very misleading (e.g. there are over 40 different cities called "Washington" in the USA).

Especially the disambiguation of such synonyms or the review of more complex relations requires collaborative involvement of human users. Such approaches lead to a higher number of correct facts [27,26,28]. Approaches combining the human and machine intelligence are called mixed-initiative approaches [29]. They basically try to exploit the advantages of both the user and automatic information extraction processes as the extracted chunks of information are verified by human users before this information is finally published. The "Intelligence in Wikipedia" [30] project uses the extraction framework described in the section above to compute new infobox entries. Subsequently, these candidates are shown to the users of the system who are then able to decide whether the candidate infobox entry was successfully and correctly extracted from the text. This is of crucial importance especially in the case of ambiguities which cannot be resolved by automated processes and can only be resolved by users. This mixed-initiative approach features the advantage that automatically extracted information is still reviewed by humans and therefore (i) verified information is published as it is subsequently added to the according infobox and (ii) the training sets of the information extraction system are refined and reviewed and, as such, the extraction process as a whole is improved. It was also shown that the acceptance of such an system is very high, as the tasks that have to be fulfilled by the users are small and easy to accomplish. The users only have to decide whether an extracted fact is correct or not. By doing so, normal users (not just active contributors to Wikipedia) are encouraged to contribute to the mixed-initiative approach.

Even very simple approaches which support the user can lead to a significant increase of data in the knowledge base. That is, Kong et al. [16] recommend relevant content from email and RSS feeds on every page of their proposed wiki system for corporate environments. Emails and RSS feeds are analysed and if there are similarities between a wiki page text and the external source (email or RSS), the external content is shown below the wiki page to encourage the user to import knowledge from the external sources to the respective wiki page. Especially in enterprise environments, where wikis are only one of many communication and documentation systems, a system which integrates different sources is very helpful.

All described approaches combine the intelligence of both humans and machines, as the algorithms can filter extensive data sources by mining algorithms in the first step. Subsequently the human user can review the small amount of recommended knowledge and import the knowledge by accepting the recommendation or resolve ambiguities and other errors. Furthermore, the recommendations and guidance mechanisms help to lower the barrier to contribute to a system and encourage the user to increase the quality and quantity of information with the knowledge base.

5. Guided insertion

All approaches described in the previous section aim at enhancing the stored knowledge after it was stored to the system. In this section we introduce the Snoopy Concept which enables

collaborative, semistructured knowledge bases to exploit the extensive knowledge of the collaborating users already during the insertion process.

5.1. The Snoopy Concept

The main idea of the Snoopy Concept is to incorporate the user already during the insertion process. Considering the fact that a user who inserts new knowledge to the system is a professional in her domain, it is very important to use the opportunity of direct communication with the inserting user already during the insertion process to exploit her expertise. For example, ambiguities within information origination from specialized domains can hardly be resolved by other, non-specialized users of the community. If the system already encounters ambiguities during the insertion process, the disambiguation by the expert user can be completed before the knowledge is stored to the system. Furthermore, the Snoopy Concept is focused on supporting the users in the creation of a common, homogeneous structure. This is realized by incorporating the user into the alignment process by suggesting highly suitable structure to the user during the insertion process. Additional recommendations enable the user to insert information as simple and efficient as possible. By using the user-centric insertion approach – which is also sketched in Fig. 1 (left box) – the following benefits can be achieved:

- Avoid proliferation of structures.
- Avoid synonyms in the system.
- Semantic refinement by resolving homonyms.
- Exploit user's extensive and valuable knowledge.
- Increase the quantity of information contained in the system.
- Increase the quality of information in the system.

All recommendations aim at supporting the user, exploiting the knowledge of the user and therefore "snooping" as much information as possible. The underlying measures and approaches of the Snoopy Concept enabling these benefits are discussed in the following section.

5.2. Data model

The Snoopy Concept is based on the semistructured data model. Essentially, the Snoopy Concept proposes to model information and knowledge as subject–property–value triples which is based on the concept of RDF. This format enables the users to store information about a certain subject as property–value pairs, e.g. the number of students at a certain university could be stored as *numberOfStudents: 20,000*. Storing information about a certain subject, e.g. the University of Innsbruck, forms a so-called *collection* (a resource in traditional RDF) and could be structured as shown in Listing 2.

```
Collection: University of Innsbruck
country: Austria
numberOfStudents: 21,001
numberOfFaculties: 15
established: 1669
```

Listing 2: Example of a Collection.

By using such triples to represent information, all information stored is machine-readable and therefore can, e.g., further be used for automatic reasoning tasks and complex structured search facilities. Furthermore, the data can be exported as valid RDF at any time and ensure high interoperability.

5.3. Recommendations

The key enabler for the described benefits within a knowledge base based on the Snoopy Concept is a recommender system [31]. Essentially, a recommender system analyses all information stored within the system to subsequently provide its users with useful recommendations. Traditionally, recommender systems are used in online shops where clients are pointed to further products. Another scenario is the recommendation of movies for the users of a movie database. Such recommendations are usually computed by applying similarity measures to the stored data in order to either find items similar to those the user bought before or to find users with similar preferences to further deduce item recommendations.

In the context of the Snoopy Concept, the recommender system suggests suitable structures the user might want to use. Also, not only properties (structures) are recommended to the user, but the Snoopy Concept also proposes to recommend values, links, types, input formats and other refinements to the user. These recommendations and their benefits are described in detail in the next sections.

5.3.1. Structure recommendations

Structure recommendation refers to the recommendation of additional properties during the insertion process and is the most important feature of the Snoopy Concept as it significantly contributes to a common and homogeneous schema within the system.

Consider the example of a user who, e.g., specifies a subject consisting of the properties *numberOfFaculties* and *numberOfStudents*. The system computes a set of appropriate properties which occur on collections with a similar structure and recommends this set of properties to the user. To contribute to a common structure, an additional ranking mechanism is proposed. This ranking assures that more popular properties are preferred in the list of recommendations in order to efficiently use the limited visual space in a user interface as well as the limited cognition of the user.

In the mentioned example within the domain of universities, e.g., the properties *rector*, *established*, *location*, *website*, etc., are recommended to the user. These recommendations are computed on the fly and are based on the just entered properties by the user and all already stored properties in the knowledge base. Any additional specified property or accepted recommended property during the insertion process results in the recomputation and refinement of all structure recommendations for the current collection.

The common schemata in the system are very dynamic, as they are based on all stored collections and therefore are influenced by every newly stored collection and its properties. Every newly stored property is automatically taken into the set of possible property recommendations and can influence further recommendations to other users who want to enter information about a similar collection. The similarity of collections is solely defined by the predicates used on these collections, as collections do not feature an explicit, predefined type (e.g. the type "University"). The type of a subject is rather solely defined by its properties and, hence, the type or category system within Snoopy is neither predefined nor rigid. Instead, types within Snoopy are dynamically computed based on the information/properties stored about a subject which is also a well known approach [32] for classification or object identification in the domain of schemaless data such as Linked Open Data.

This flexibility and the fact that users are free to modify the recommended structure prevent the system from creating a completely unified and aligned schema. Thus, the user is guided to a common schema without restricting her in her way of structuring information or extending existing schemata. Therefore, the Snoopy Concept does not require any schema matching [33] after the insertion of data. The alignment is done by the user with her extensive knowledge and is therefore always more powerful than any automated alignment algorithm. Furthermore, in the case

of multiple semantically similar properties which are all used within the system, the community decides which one is more appropriate by using the according property and, hence, 'voting' for the property. This way, a property becomes more popular, gets recommended and hence used.

Furthermore, the recommendation of structure increases the quantity of information as the recommendations indicate "missing" bits of information. In the mentioned example of the domain of universities, the system could recommend the property *rector*. By providing such additional property recommendations, the user is encouraged to enter more information than she originally intended to insert and the valuable knowledge of the user is once more exploited. Such information gathered by the "snooping" process would be lost without recommendations and cannot be completed by any machine afterwards and therefore enhance the information system dramatically.

5.3.2. Avoiding synonyms by recommendations

During the specification of further content, the user is supported by an intelligent auto-completion feature. The system suggests suitable properties to the user which have already been used within the information system. Consider a user who started entering the property *number*. The system subsequently suggests all previously used properties in the system which are related to the term *number*. In this case, the system would suggest *numberOfStudents* and *numberOfFaculties*. In most cases the user accepts such a recommended property if it is suitable in the respective context. In this example the user would be prevented to insert a new synonymous property, such as *numberFaculties*. A more severe challenge in information systems lies in coping with syntactically different synonyms as it cannot be solved by string-based matching approaches. Consider the example of a user entering *Faculty*, a string based matching approach would not recommend the already entered property *Academic Staff*. By using a thesaurus, *Faculty* can be matched with the semantically equivalent, already existent property *Academic Staff* which can then be suggested to the user.

5.3.3. Semantic refinement and value recommendations

The guidance of the user is not limited to properties; also possible values can be suggested. This mechanism features the advantage of providing the user with values in an already aligned form, which prevents the user from entering synonyms of already existing values. This can be achieved by using the same mechanisms used for the recommendation of properties previously described.

Furthermore, not only the value itself can be recommended but the system can additionally suggest semantic links between values and other subjects. This measure copes with the common challenge of preserving semantic "correctness" of homonyms. If the user, e.g., specifies the city Freiburg as a twin city of Innsbruck, it is not clear whether the user refers to Freiburg in Germany or Freiburg in Switzerland. Within the Snoopy Concept, this problem is resolved by recommending possible semantic links from the entered value *Freiburg* to already existent, semantically equivalent collections (e.g. Freiburg, Germany and Freiburg, Switzerland). The user is then able to specify the meaning just by accepting the appropriate link-recommendation. As the user has extensive knowledge about the content to be inserted, she can provide more semantic information than any automated extraction process can do afterwards. Using these measures, homonyms are further semantically enhanced by humans, which leads to a high confidence of semantic data in the system.

5.3.4. Validation and recommendations of types

Furthermore, the concept proposes a validation process, which includes determining a data type for each newly entered value, e.g., the value for the property *numberOfStudents* is asserted to be an integer value. Vice versa, if a property is added that already exists, has a data type assigned and is used by the majority of property instances, the user is prompted to enter values according to this data type. The detection of data types, especially in the case of numeric types, is very important as it is crucial for queries based on numeric evaluation. For example, the query "List all universities having more than 10,000 students" is only possible if the value of the property *numberOfStudents* is stored as a numeric value. Furthermore, also other data types like, e.g., date, time, HTML, file, image, audio or video are possible and lead to special behaviour according to the data type (e.g. date picker, calendar views, content-based image search, mp3 metadata search, etc.). Additionally, the syntactic correctness is validated according to the respective data type (e.g. correct date format).

All these measures enable the user to enter information fast and efficiently by just accepting recommendations while at the same time additional, semantically equivalent properties and values are avoided and the amount of unified information and the confidence of semantic data in the system are increased.

5.4. Prototype

The Snoopy Concept was implemented in a first prototype which is called "SnoopyDB". A screenshot of the SnoopyDB prototype can be seen in Fig. 2. This figure shows the screen of a user entering information about the University of Innsbruck. The user already entered four property–value pairs about the foundation year, the founder, the number of professors and the official website of the University of Innsbruck. The three additional rows displayed in grey font mark the properties which were recommended by the system. The value fields corresponding to these properties already contain exemplary values. This way the user can immediately recognize that the value of the property employees is normally entered as a numeric value. The box on the right side of the screenshot contains further suitable properties for the current subject. These properties can easily be added to the input form by clicking on the arrow icon. The screenshot also shows how the system automatically detects the data type of the entered information. In line 4, a link for the website of the university is created. The system automatically detected that a big percentage of the values belonging to the property website were stored as links and, therefore, the system suggests the insertion of a link to the user. In this case, the user accepted this suggestion and entered the URL of the official website of the University of Innsbruck. The underlying algorithms and implementation of SnoopyDB are described in the following section.

5.4.1. Recommendation algorithm

The SnoopyDB approach is based on a recommendation algorithm (see Algorithm 1), which takes all collections (subjects) and properties occurring on these subjects into account. Formally, the set of properties occurring within the whole system can be denoted as $\mathcal{P} = \{p_1, p_2, p_3, \ldots, p_n\}$ and the set of all subjects occurring can respectively be denoted as $\mathcal{S} = \{S_1, S_2, S_3, \ldots, S_m\}$. The properties belonging to a certain Subject S_i can be identified as $\mathcal{P}_{S_i}$.

Based on these definitions, recommendations can be computed by firstly determining all pairs of properties (p_a, p_b) occurring on the same subject. If a user, e.g., specified the properties name, location and numberOfStudents on the same collection, the pairs (*name*, *location*), (*name*, *numberOfStudents*) and (*location*, *numberOfStudents*) are formed. These pairs are computed for all subjects within the system and subsequently stored together with the total number of occurrences of the respective pair of properties within the whole system. This set of rules (pairs) is denoted by

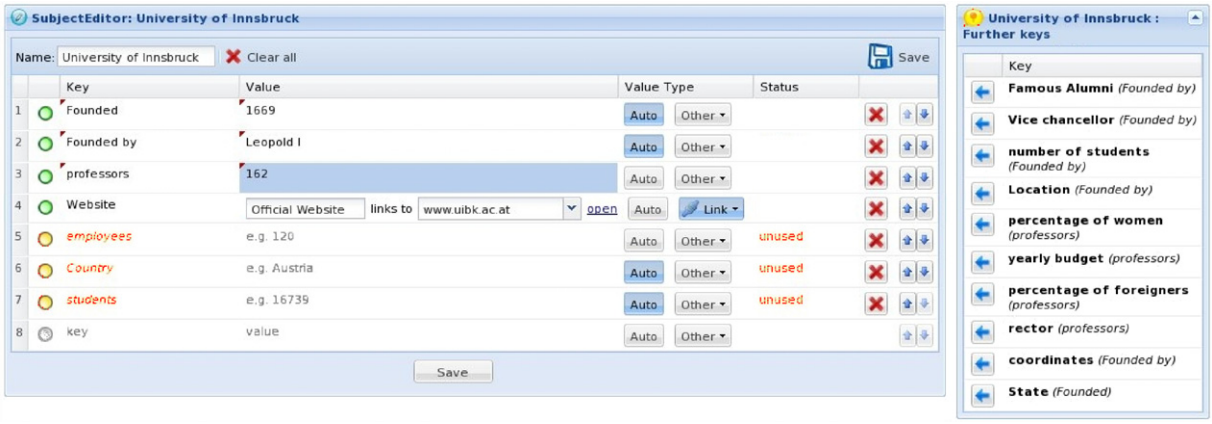

Fig. 2. Screenshot of the SnoopyDB prototype.

Algorithm 1: Recommendation 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 $\mathcal{C}$

$\mathcal{R}$ and serves as input for the computation of recommendations during the insertion process. If a user enters new information about a certain subject, the properties already contained in this subject also serve as input for the computation of recommendations. For each property p_i occurring on the input subject S_i, all triples where p_i is contained within the property pair are detected. These triples basically form the set of recommendation candidates. After having detected these recommendation candidates, the most important and therefore the most useful recommendations for the user have to be extracted. This is accomplished by determining the most popular properties within the recommendation candidates. Therefore, the number of occurrences of each recommendation candidate is summed up. These properties are subsequently ranked by their popularity and the top-k items are recommended to the user.

5.4.2. Evaluation

The SnoopyDB approach was evaluated by a test-user experiment [34]. The goal of the evaluation was to assess the user's acceptance of the recommendation and guidance mechanisms provided by the system. As such an evaluation cannot be simulated or performed artificially, test-users were asked to take part in an experiment for the evaluation of the approach.

In total, 24 test users took part in the experiment. These users stemmed from different backgrounds, 2/3 of all test users were computer scientists and 1/3 of the participating users were standard computer users without any special computer knowledge or experiences with handling semistructured data.

For the evaluation the test users were presented with two different systems: one system was supporting the users with all recommendation and guidance features described in the previous sections (in the following referenced as system A). The other system was not supporting the users at all and, thus, was not providing any recommendations for neither properties nor value entries (in the following referenced as system B). System A was bootstrapped with data created within the previous evaluation of the system [34] which contained subjects originating from two domains (cities and musicians) in order to be able to provide basic recommendations and to be able to assess how the system and the provided recommendations adapt if subjects stemming from a new, unknown domain are added.

In the course of the experiment, users were asked to fulfill the following tasks in order:

1. Insert data about an arbitrary university firstly into system B (no support provided to the user).
2. Insert data about an arbitrary subject related to the motor vehicle industry into system B.
3. Insert data about an arbitrary university into system A (guidance by recommendations).
4. Insert data about an arbitrary subject related to the motor vehicle industry into system A.

The subjects the users had to enter were not specified, as well as the actual information the users had to enter about a certain subject. This information was solely chosen by the test users themselves. Also, no minimum number of property–value pairs was specified and, hence, the amount of information about a certain subject that was added was solely decided by the user. The only restriction was that users were only allowed to use the English language for the names of the properties. Furthermore, users were not allowed to use the English Wikipedia for seeking information as already aligned infobox properties could possibly influence the German-speaking test users and the resulting property names. During the experiments, all actions of the participating test users were logged and stored in order to be able to evaluate the differences in the performance of the two systems in regard to the homogeneity of the resulting vocabulary and the acceptance of the provided recommendations. The results of the analysis of the information gathered during the user experiments are discussed below.

As for the acceptance of the proposed recommendations, the evaluations showed that 22% of all recommended properties were accepted by the users. This seemingly low number can be led

back to the fact that the system always proposes five additional properties. Each time a user accepts a recommendation or adds new information, the recommendation list is recomputed. Hence, the total number of recommended properties is high during each edit-session. In total, 49% of all newly added property–value pairs were added by accepting a property recommendation. In 62% of all user edit sessions (including re-editing of subjects), at least one property recommendation was accepted. Furthermore, also the auto-completion feature provided as an additional guidance and support mechanism was used frequently. 23% of all properties and 17% of all values were entered by accepting the entries proposed by the auto-completion mechanisms. A spellchecker was also implemented in the SnoopyDB prototype. The corrections proposed by the spellchecker were accepted by the participating test users in 37% of all cases. This acceptance rate is low, which can be led back to the fact that the spellchecker web service is based on the information extracted from the web corpus. Hence, the spellchecker was not only suggesting corrections for simple typos but also suggested, e.g., 'Formula 1' instead of 'Formula1' or 'Leopold II' instead of 'Leopold I' which were not accepted by the users. Considering only simple typos (e.g. one missing character), applicable spellchecker recommendations were accepted in 100% of all cases.

The evaluations showed that the schema entered into the recommendation-providing system (A) was 33% more homogeneous in regard to the set of properties entered than without supporting the user (system B). Homogeneity within a set of properties describes how many synonymous terms were used for the description of the same subject, i.e., how many property names were directly reused and, hence, no synonym was used instead. In this evaluation the resulting vocabulary in system A is 33% smaller than the vocabulary of system B. Despite the reduction of properties, the users entered 31% more information into the system (A) when supported by recommendations. This implies that by guiding the user during the insertion process and furthermore, enabling the user to easily add more information and also to point the user to bits of information which she still might want to enter, the total amount of useful and structured information can be increased. This value is biased by the fact that the users already dealt with subjects in the first two tasks of the evaluation (system B). However, as we wanted to simulate guidance and motivation of domain expert users, who already have extensive knowledge about the subject, such a high percentage can also be possible in real-world environments. Another important finding of the evaluations was that the introduction of the new domains did not result in a dramatic increase of newly added properties. This fact implies that most of the properties were reused.

Furthermore, the described algorithm was evaluated on a large dataset in [35]. A leave-one-out test was conducted on a DBpedia set of 41 million triples which tried to reconstruct missing infobox properties by recommendations. The evaluations showed that at least 4 out of 10 recommendations are appropriate (precision is 40%) and the algorithm is able to guide the user to a common schema after having inserted only three properties. 60% of all 150,000 infoboxes were reconstructed to more than 50% and about 11% were completely reconstructed. The rule based computation of recommendations on large datasets [35] also scales very well as the top-k recommendations can be computed within less than a fraction of a second on commodity hardware. Thus, the suitability, strength and the stability of the Snoopy algorithm was shown by offline and online tests.

6. Conclusion and future work

In this article we presented approaches aiming at collaboratively curating semistructured data with the goal of increasing the quality and quantity of stored knowledge. This is achieved by providing guidance and recommendations which are based on collaboratively created knowledge and the according structure. We presented a set of approaches which aim at refining and aligning semistructured data using extracted facts from other knowledge bases or natural language texts. The second approach deals with the refinement, alignment and the schema proliferation already at the time of insertion as it supports the user by recommendations already during the insertion of data. Both approaches aim at simplifying the curation procedure and increasing the quality of the stored information. The Snoopy Concept, furthermore, encourages the user to insert more information and, therefore, increase the amount of stored data while at the same time refining the data and homogenize the structure. However, the limitation of such an approach clearly lies in the fact that the system's performance in regard to the quality of the provided recommendations heavily relies on the information already contained in the system. If not a single subject of the same or a similar type has already been stored in the system, the recommendations are not suitable for a new subject (except for very general properties like, e.g., "name" which matches most of the subjects). However, future work in this direction also contains the incorporation of further data sources for an initial bootstrapping of the database. For example data retrieved from LOD sources may be used for a first bootstrapping. Furthermore, the automatic interlinkage of the date entered by users is a very desirable feature. By, e.g., using a probabilistic approach like in [36], an automated linkage to entities existing in the LOD cloud could be achieved. This way, further information from the LOD cloud about a certain entity could be recommended to the user. Furthermore, also conflicts within the data stored within SnoopyDB could be resolved by using LOD data as a reference point for information.

The presented approaches showed that a mixed-initiative approach, which combines the computational power of computer algorithms with the semantic knowledge of human users, is able to increase the quality and confidence of semistructured knowledge and simplify the curation process in collaboratively-built knowledge bases.

References

[1] G. Weikum, G. Kasneci, M. Ramanath, F. Suchanek, Database and information-retrieval methods for knowledge discovery, Communications of the ACM 52 (4) (2009) 56–64.
[2] P. Schäuble, SPIDER: a multiuser information retrieval system for semistructured and dynamic data, in: Proc. of the 16th Annual Intl. ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1993, pp. 318–327.
[3] P. Buneman, Semistructured data, in: Proc. of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, ACM, 1997, pp. 117–121.
[4] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J.L. Wiener, The Lorel query language for semistructured data, International Journal on Digital Libraries 1 (1) (1997) 68–88.
[5] R.G.G. Cattell, D.K. Barry, D. Bartels, M. Berler, J. Eastman, S. Gamerman, D. Jordan, A. Springer, H. Strickland, D. Wade, The Object Database Standard: ODMG 2.0, Vol. 131, Morgan Kaufmann Publishers, 1997.
[6] G. Klyne, J.J. Carroll, B. McBride, Resource Description Framework (RDF). W3C Recommendation, 2004.
[7] E. Prud'hommeaux, A. Seaborne, SPARQL query language for RDF. Technical Report, W3C, January 2008.
[8] C. Bizer, T. Heath, T. Berners-Lee, Linked data—the story so far, International Journal on Semantic Web and Information Systems 4 (2) (2009) 1–22.
[9] M. Voelkel, M. Kroetzsch, D. Vrandecic, H. Haller, R. Studer, Semantic Wikipedia, in: WWW'06: Proc. of the 15th Intl. Conference on World Wide Web, ACM, 2006, pp. 585–594.
[10] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, Z. Ives, DBpedia: a nucleus for a web of open data, Lecture Notes in Computer Science 4825 (2007) 722.
[11] G.W. Furnas, T.K. Landauer, L.M. Gomez, S.T. Dumais, The vocabulary problem in human–system communication, Communications of the ACM 30 (11) (1987) 971.
[12] P. Boulain, N. Shadbolt, N. Gibbins, Hyperstructure maintenance costs in large-scale wikis, in: SWKM, CEUR Workshop Proceedings, vol. 356, 2008. www.CEUR-WS.org.
[13] F. Wu, D.S. Weld, Automatically refining the Wikipedia infobox ontology, in: WWW'08: Proceeding of the 17th Intl. Conference on World Wide Web, ACM, 2008, pp. 635–644.

[14] R. Priedhorsky, J. Chen, S.K. Lam, K. Panciera, L. Terveen, J. Riedl, Creating, destroying, and restoring value in Wikipedia, in: Proc. of the 2007 Intl. Conference on Supporting Group Work, ACM, 2007, pp. 259–268.
[15] B. Suh, G. Convertino, E.H. Chi, P. Pirolli, The singularity is not near: slowing growth of Wikipedia, in: Proc. of the 5th Intl. Symposium on Wikis and Open Collaboration, ACM, 2009, p. 8.
[16] N. Kong, B. Hanrahan, T. Weksteen, G. Convertino, E. Chi, VisualWikiCurator: human and machine intelligence for organizing wiki content, in: Proc. of the 16th Intl. Conference on Intelligent User Interfaces, ACM, 2011, pp. 367–370.
[17] M. Hausenblas, W. Halb, Interlinking of resources with semantics, in: Poster at the 5th European Semantic Web Conference, 2008.
[18] I. Horrocks, Semantic web: the story so far, in: Proc. of the 2007 Intl. Cross-Disciplinary Conference on Web Accessibility (W4A), ACM, 2007, pp. 120–125.
[19] J. Volz, C. Bizer, M. Gaedke, G. Kobilarov, Silk—a link discovery framework for the web of data, in: Proc. of the 2nd Linked Data on the Web Workshop, 2009.
[20] D. Jurafsky, J. Martin, Speech and Language Processing, second ed., in: Prentice Hall Series in Artificial Intelligence, Prentice Hall, 2008.
[21] F. Suchanek, G. Kasneci, G. Weikum, YAGO: a large ontology from Wikipedia and WordNet, Web Semantics: Science, Services and Agents on the World Wide Web 6 (3) (2008) 203–217. World Wide Web Conference 2007, Semantic Web Track.
[22] J. Hoffart, F. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, G. Weikum, YAGO2: exploring and querying world knowledge in time, space, context, and many languages, in: Proc. of the 20th Intl. Conference on World Wide Web, ACM, 2011, pp. 229–232.
[23] G. Miller, WordNet: a lexical database for English, Communications of the ACM 38 (1995) 39–41.
[24] F. Suchanek, Ma. Sozio, G. Weikum, SOFIE: a self-organizing framework for information extraction, in: Proc. of the 18th Intl. Conference on World Wide Web, ACM, 2009, pp. 631–640.
[25] N. Nakashole, M. Theobald, G. Weikum, Scalable knowledge harvesting with high precision and high recall, in: Proc. of the Fourth Intl. Conference on Web Search and Data Mining, ACM, 2011, pp. 227–236.
[26] F. Wu, R. Hoffmann, D.S. Weld, Information extraction from Wikipedia: moving down the long tail, in: Proceeding of the 14th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 731–739.
[27] R. Hoffmann, S. Amershi, K. Patel, F. Wu, J. Fogarty, D. Weld, Amplifying community content creation with mixed initiative information extraction, in: Proc. of the 27th Intl. Conference on Human Factors in Computing Systems, ACM, 2009, pp. 1849–1858.
[28] F. Wu, D.S. Weld, Autonomously semantifying Wikipedia, in: Proc. of the Sixteenth ACM Conference on Information and Knowledge Management, ACM, 2007, pp. 41–50.
[29] E. Horvitz, Principles of mixed-initiative user interfaces, in: Proc. of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 1999, pp. 159–166.
[30] D. Weld, F. Wu, E. Adar, S. Amershi, J. Fogarty, R. Hoffmann, K. Patel, M. Skinner, Intelligence in Wikipedia, in: Proc. of the 23rd National Conference on AI, AAAI Press, 2008, pp. 1609–1614.
[31] P. Resnick, H.R. Varian, Recommender systems, Communications of the ACM 40 (3) (1997) 58.
[32] J. Noessner, M. Niepert, C. Meilicke, H. Stuckenschmidt, Leveraging terminological structure for object reconciliation, in: The Semantic Web: Research and Applications, in: Lecture Notes in Computer Science, vol. 6089, Springer, Berlin, Heidelberg, 2010, pp. 334–348.
[33] P. Shvaiko, J. Euzenat, A survey of schema-based matching approaches, Journal on Data Semantics 4 (2005) 146–171.
[34] W. Gassler, E. Zangerle, M. Tschuggnall, G. Specht, SnoopyDB: narrowing the gap between structured and unstructured information using recommendations, in: HT'10, Proc. of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Canada, 2010, pp. 271–272.
[35] E. Zangerle, W. Gassler, G. Specht, Recommending structure in collaborative semistructured information systems, in: Proc. of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 261–264.
[36] E. Ioannou, C. Niederée, W. Nejdl, Probabilistic entity linkage for heterogeneous information spaces, in: Proc. of the 20th Intl. Conference on Advanced Information Systems Engineering, CAiSE, in: LNCS, vol. 5074, Springer, 2008, pp. 556–570.

Wolfgang Gassler is a Ph.D. student at the University of Innsbruck, Austria. He is a member of the Databases and Information Systems group at the Institute of Computer Science and works in the field of creating, editing and storing knowledge in large-scale environments. Especially upcoming challenges in the area of semistructured collaborative environments are the main focus of his research. Together with Eva Zangerle he is the leader of the Snoopy Concept project. His primary research is concerned with how to store RDF data in a fast and distributed way and simultaneously provide efficient user-centred interfaces and high usability.

Eva Zangerle is a postdoctoral researcher and a member of the scientific staff of the research group Databases and Information Systems at the Institute of Computer Science at the University of Innsbruck. During her Master studies, she specialized in the field of databases and information systems. Throughout her Ph.D. studies, Eva Zangerle gained expertise and years of experience in the fields of recommender systems, personalization and graph stores, where she is also the author of several papers. Eva Zangerle's research is concerned with how recommender systems are able to create a common structure and schema within heterogeneous information systems like microblogging platforms.

Günther Specht is the head of the research group Databases and Information Systems at the University of Innsbruck. He obtained his Ph.D. at the Technische Universität München and since then, he has gained extensive knowledge in numerous fields of research concerned with databases and information systems. Günther Specht has been engaged in topics concerning the storage of large graphs for several years. Numerous cooperations on national and international level deepen the research in different focuses and projects.