Markups for Knowledge Wikis

Joachim Baumeister, Jochen Reutelshoefer, Frank Puppe
Institute of Computer Science
University of Würzburg
Würzburg, Germany
baumeister/reutelshoefer/puppe@informatik.uni-wuerzburg.de

ABSTRACT

Knowledge wikis extend normal wikis by the representation of explicit knowledge. In contrast to semantic wikis the defined knowledge is mainly used for knowledge–intensive tasks like collaborative recommendation and classification. In this paper, we introduce a prototype implementation of a knowledge wiki, and we discuss appropriate markups for problem–solving knowledge to be used in knowledge wikis. The main aspect of the proposed markups is their simplicity and compactness, since it is aimed that these markups are used by normal wiki users. Case studies report on practical experiences we have made with the development of various knowledge wikis.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

1. INTRODUCTION

Many successful examples have shown that wiki systems are a reasonable basis for building information systems in a collaborative way, i.e., the creation and evolution of the content is accomplished by the community itself. In general, wiki systems [6] are web-based content–management systems that allow for the simple construction of new content and the evolution of existing content, i.e., mostly text, figures and pictures. Here, the content is viewed by a web browser, but is also changed by editing the particular wiki page in an edit pane. In order to simplify the interface for the users, the formatting syntax is commonly less complex than HTML. The arguably most prominent example of a successful wiki system is the encyclopedia Wikipedia.

In this paper we focus on *knowledge wikis* to be used in the context of classification systems, e.g., systems built for consultation, selection, and recommendation tasks. Knowledge wikis extend regular wikis by the possibility to create and use explicit knowledge together with the classic wiki content. The creation and evolution of the knowledge is done within the normal edit pane of the wiki, i.e., developers insert and change problem–solving knowledge using a textual knowledge markup. Examples for knowledge markup are the definition of models, rules and the semantic annotation of wiki text. The simplicity and the intuitive representation of the markup is very important, since we address normal but "experienced users" to be the developers of the knowledge wiki and not knowledge engineers that can be trained thoroughly beforehand. In consequence, the markup should support the formation of "ad-hoc knowledge engineers". The formalized knowledge is also used through the wiki interface: the user can enter findings into the wiki system and retrieves appropriate recommendations as a solution for the given input.

We describe different markups to be defined in knowledge wikis to capture and maintain knowledge for the creation of recommendation systems. As already emphasized before, it is important to provide simple and intuitive markups in order to enable standard users to adapt this representation. All proposed markups have been implemented in the prototype system KnowWE (Knowledge Wiki Environment).

Wikis are an already proven technology that is well–adapted by large communities, and therefore the approach of extending wikis by knowledge markup appears to be quite feasible. Due to its open and immediate access to the knowledge a wiki serves as a natural platform for knowledge communities. In consequence, we also expect a knowledge wiki to be a suitable development tool for "closed communities" [2], i.e., a community of interest with a particular goal that is comparable to open–source software projects. Here, the resulting software is actually used by a wide range of people, but the development process itself is carried out by a smaller (and often distributed) group of specialists. These developers have the access and the possibility to extend and modify the system, since they are recognized by the rest of the community. For example, this clearly differs from systems like Wikipedia that are mainly built for "open communities".

The following is organized as follows: in Section 2 we give a brief overview of the knowledge wiki KnowWE applicable for recommendation tasks. Appropriate markups to be used as explicit problem–solving knowledge are introduced in Section 3. We discuss the possible integration of explicit problem–solving knowledge and normal wiki text in Section 4, and we provide case studies in Section 5. The paper is concluded with a summary and outlook in Section 6.

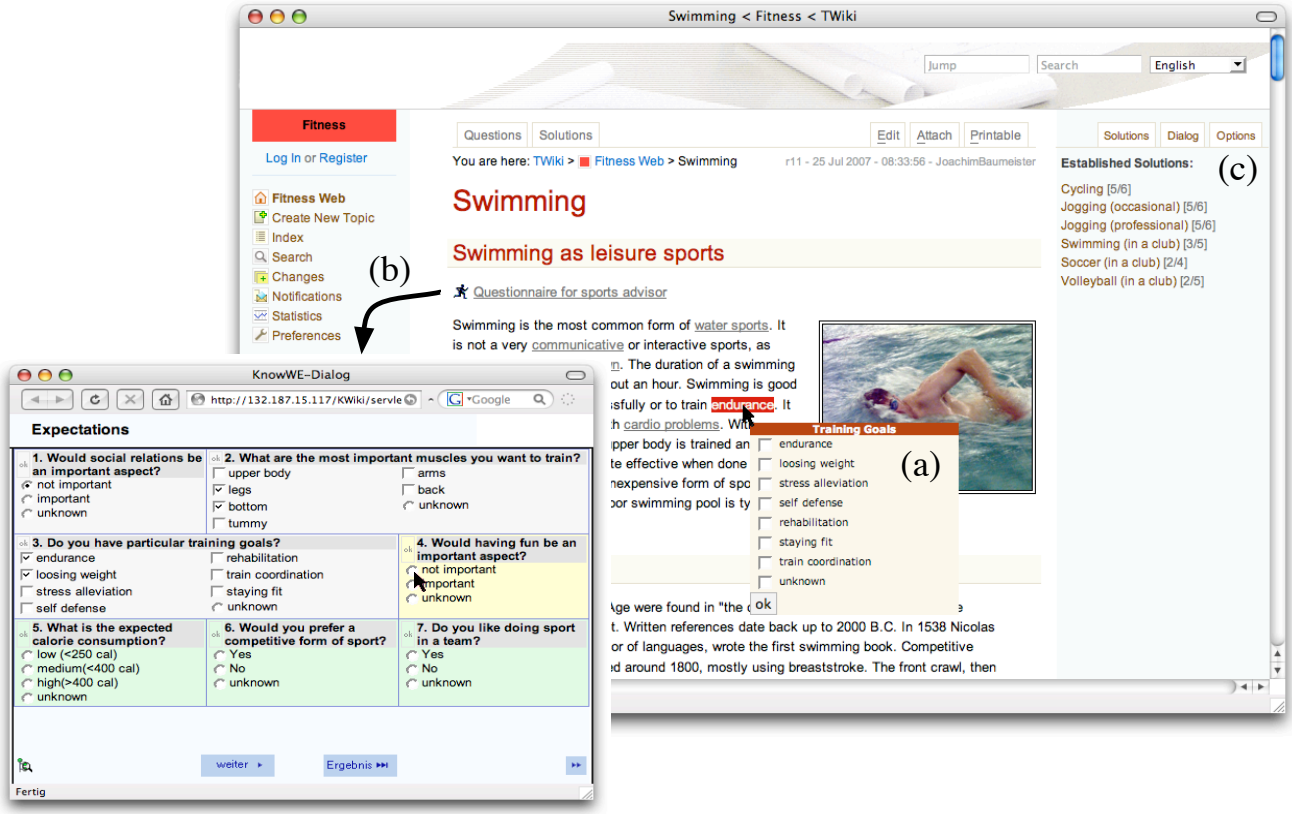

Figure 1: The user interface of the knowledge wiki KnowWE: user inputs are provided by (a) in–place answers or by (b) starting a structured interview; (c) currently derived solutions are displayed immediately.

2. KNOWWE – A KNOWLEDGE WIKI

The knowledge wiki *KnowWE* is an extension of the classic wiki engine TWiki[1] in order to support classification tasks in a knowledge–based manner. For the classification task and its corresponding problem–solving, respectively, we identify two main ontological classes: *user inputs* represented by questions to the user and *solutions* (recommendations) derived by the system for a set of given inputs. We support different types of user inputs, namely one–choice, multiple–choice and numeric inputs. In the presented system the reasoning engine d3web [1, 8] is integrated, which provides a variety of problem–solving methods like heuristic scoring rules, decision trees, set–covering models, and case–based reasoning.

In this section, we briefly introduce the basic concepts of KnowWE, i.e., the creation of new knowledge, the integration of explicit knowledge into the normal wiki context, and the extensions of the wiki for a structured problem–solving process. Throughout the sections we use an exemplary sport recommendation system, that tries to select appropriate forms of sports as solutions for given user inputs (i.e., the entered preferences).

In Figure 1 the standard interface of KnowWE is shown: user inputs can be given by (a) inplace–answers of an annotated text or by (b) activating a knowledge–based interview provided by a link. The right pane of the wiki displays currently derived recommendations for the given inputs (c).

[1]http://www.twiki.org

2.1 Knowledge Creation

The acquisition and maintenance of knowledge in a knowledge wiki is done within the edit pane, i.e., by entering and changing text in a browser window. Since we propose an extension of classic wikis the problem–solving knowledge is managed together with the text contained in the normal wiki. Thus, the knowledge is embedded "in–text", i.e., the textual representation of the actual knowledge is jointly edited together with the textual content of a wiki page.

Figure 2 shows a screenshot of the edit pane of the wiki page "Swimming" of a knowledge wiki for the consultation and selection of a form of sport (see case study in Section 5 for more details). Thereafter, explicit knowledge is added to the normal wiki text surrounded by the special tag *Kopic* (for knowledge topic). In the example, we see the textual definition of a set–covering model for the solution "Swimming (occasional)", i.e., the listing of typical preferences/findings that would derive this solution as an appropriate recommendation. Besides set–covering models the knowledge wiki is able to process further types of knowledge like decision trees and (heuristic scoring) rules. We describe the possible knowledge representations in Section 3 in more detail.

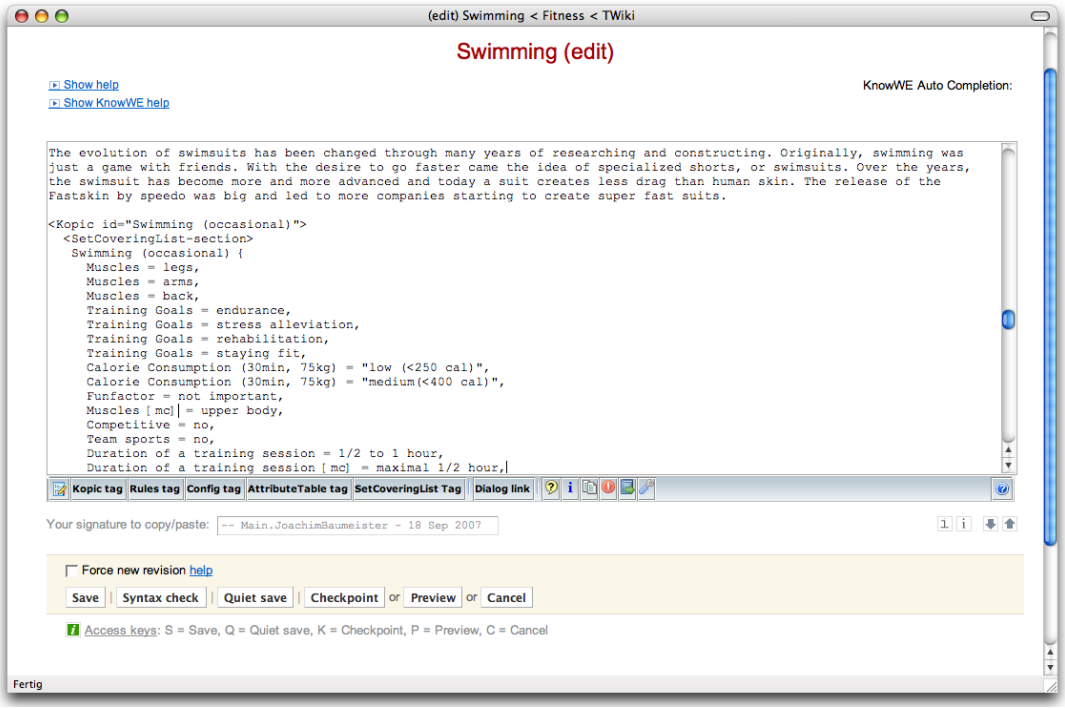

Figure 2: Creating an article on "Swimming" with the definition of set–covering knowledge.

The user is supported by a help page describing the syntax of the knowledge wiki extensions and by an auto completion feature. The auto completion suggests findings and solutions, that are defined in the current or other wiki pages, and thus encourages the reuse of already known ontological concepts. A syntax check gives verbose feedback in the case of an incorrect use of definitions.

2.2 Knowledge Integration

The entered text and knowledge, respectively, is stored by simply using the "Save" button, and usually the page is then filed to the repository. KnowWE additionally extracts the distinguished parts tagged by "Kopic" in order to compile an executable knowledge base. The knowledge base is then stored in the knowledge repository. Both, the repository of wiki pages and the repository of knowledge bases, are defining the *knowledge wiki repository*, which is depicted in Figure 3.

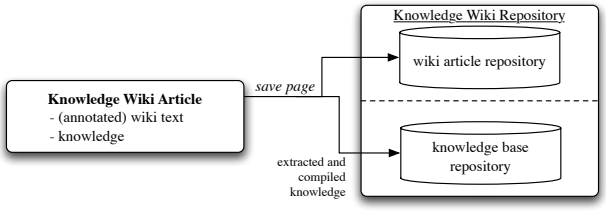

Figure 3: Workflow for saving articles of the knowledge wiki.

In the context of recommendation wikis we commonly define one wiki page for one solution (or a coherent group of solutions). The corresponding knowledge is captured on this page within the tag "Kopic". In consequence, we usually get a large number of knowledge bases for the entire wiki. These knowledge bases are loosely coupled by an alignment process that automatically aligns user inputs with the same name and type. The alignment process can be simplified by limiting the allowed inputs to a pre–defined terminology. Then, new knowledge bases to be inserted are bound to use this fixed set of possible user inputs. In this case, the inputs of the particular knowledge bases are trivially aligned to the corresponding inputs pre-defined in the global ontology.

2.3 Knowledge Consumption

With *knowledge consumption* we basically consider the process of (interactive) problem–solving when the defined knowledge is used. In contrast, today's problem–solving in the web typically comprises the search of appropriate web sites and browsing some of them in order to retrieve sufficient information for solving the actual problem. We call this procedure *manual problem–solving*. Common places for manual problem–solving are knowledge clusters in the web: bulletin boards and wikis are prominent sources of collaborative knowledge clusters, i.e., places where many people capture and share their knowledge. Examples for wikis can be found in various documentation projects of open–source software, the encyclopedia Wikipedia and many enterprise wikis. In knowledge wikis we try to enhance the manual problem–solving process described above by knowledge–based techniques: The wiki still provides the content to be read by the user, but adds interactive elements and offers structured dialogs to solve the problem in a more formalized way.

In such an interactive problem–solving session the user en-

ters findings into the knowledge wiki in order to derive appropriate solutions. User inputs are collected by *knowledge services*, that contain the corresponding knowledge base but are also responsible for the data acquisition strategy. Inputs from a particular knowledge service are propagated to a broker that aligns the respective inputs to globally known ontological concepts and subsequently stores the aligned concepts in a central blackboard. In consequence, all further knowledge services are notified of the newly aligned concepts and their particular value is propagated to the corresponding knowledge bases. Thus, an entered finding of a user is not only given to the currently active knowledge base, but also to all other knowledge bases in the wiki that share the particular finding (see Section 2.2). Subsequently, reasoning results of every registered knowledge base are again returned to the blackboard for further broadcasting and interpretation.

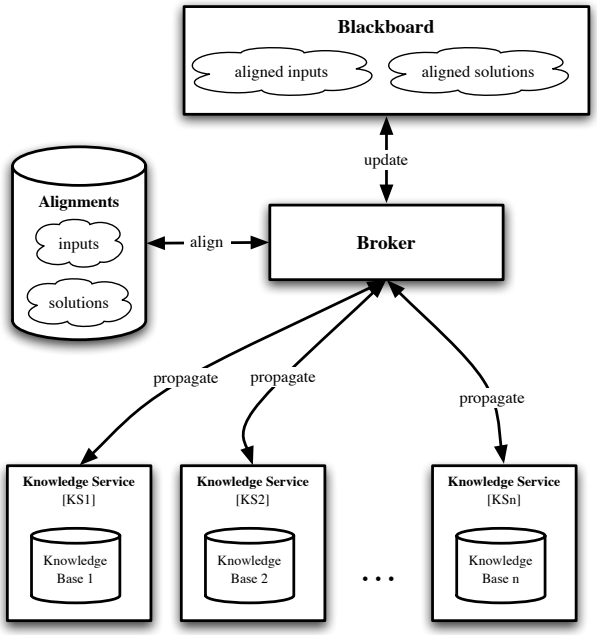

Figure 4: Blackboard architecture for the distributed problem–solving of the knowledge wiki KnowWE.

Figure 4 depicts the communication paths during a distributed problem–solving session using a blackboard architecture with a simple broker for alignments. Within the proposed knowledge wiki we distinguish two possible ways for the acquisition of user inputs. We discuss them in more detail in the following.

2.3.1 Structured Data Acquisition

The developer can add links to any wiki page that reference to a structured questionnaire for a solving a problem in a classic knowledge–based manner, e.g., Figure 1(b) displays a standardized interview generated from a knowledge base.

In the example, the user clarifies the appropriateness of the specific sports type by answering a couple of questions, e.g., the training goals of the user that should fit with the questioned form of sports.

Since the given user inputs are not only submitted to the currently active knowledge base but also to all other relevant knowledge bases contained in the wiki, further solutions (e.g., forms of sport) are derived as possible solutions. This possibility for generating differential solutions during the clarification process enhances the problem–solving capabilities of a knowledge wiki when compared to a traditional wiki application.

2.3.2 In–Place Answers

Often, users are not comfortable with answering structured questions but are more used to solve a specific problem by browsing (mostly appropriate) texts, i.e., by manual problem–solving. For this reason, knowledge wikis additionally offer the possibility to make *in–place answers*, which allows for a seamless acquisition of findings during the browsing process. Here, tagged text phrases with a distinct semantics provide pop–up menus to answer questions during the browsing process. For example, in Figure 1 the click on the text phrase "endurance" activates a pop–up asking for the "Training Goals" of the user, where "endurance" is a possible answer. As the structured problem–solving approach described above, the entered findings are also given to the problem–solving engine in order to derive suitable solutions. In summary, in–place answers provide a technique for the smooth integration of unstructured browsing and structured problem–solving.

In this section, we briefly described the main ideas of the knowledge wiki KnowWE that supports knowledge–based recommendation tasks in a collaborative way. In the following section, we introduce a selection of knowledge markups of KnowWE in more detail.

3. KNOWLEDGE MARKUPS FOR WIKIS

Knowledge markups are textual representations of "knowledge" in arbitrary forms. In this paper, we propose textual representations for different types of knowledge representations, that can be easily embedded into the normal wiki text. Here, the syntax of the knowledge markup should be as simple as possible in order to allow for an intuitive creation and evolution of the knowledge together with the normal wiki text. In the best case, typical wiki users are capable to understand and use the knowledge wiki syntax without a thorough training.

In general, we enclose the definition of a knowledge base in a wiki page with the tag `<Kopic id="theID"> ... </Kopic>`, where *theID* is the unique identifier of the knowledge base. In this Kopic tag further sub–tags are used for the particular knowledge representations.

An important issue is the understandability of the knowledge representation itself: we provide (heuristic scoring) rules, decision trees, and set-covering models as possible knowledge representations. Furthermore, we plan to include case–based reasoning in the system, since cases are probably the most intuitive representation to formalize experience knowledge. In the following, we introduce the particular knowledge representations and their markup, respectively, in more detail.

3.1 Rules

Rules are certainly the most applied knowledge representation for building intelligent systems. A rule $r = c_r \rightarrow a_r$ derives facts as defined in its consequent (rule action) a_r, if the specified rule condition c_r is satisfied. Facts derived by the rule can be interpreted as solutions or further inputs that are used in conditions of other rules. The rule condition typically consists of an arbitrary combination of conjunctions and/or disjunctions constraining the user inputs. For sake of simplicity we distinguish two basic types of rules, that can be used in the knowledge wiki: *abstraction rules* for deriving new input facts and *scoring rules* for deriving a new state of a solution. Then, the consequent of the rule either assigns a value to an input (abstraction rule), or derives a new state for a particular solution (scoring rule).

Scoring Rules. Scores are used to represent a qualitative approach for deriving solutions with symbolic confirmation weight. These weights state the degree of confirmation or disconfirmation of a particular solution. In this way, a symbolic weight c expresses the strength for which satisfied rule condition will confirm/disconfirm the solution s. The definition and semantics of scoring rules goes back to the INTERNIST/QMR project [7]. Analogous to the representation of the d3web rule system [1] we distinguish seven positive weights (P1, ..., P7) and seven negative weights (N1, ..., N7). Here, the weight P7 stands for the categoric derivation of a solution, the counter–weight N7 yields the categoric exclusion of a solution. The remaining weights are defined in a way, so that the aggregation of two equal weights results in the weight of the next higher weight, e.g., N3 + N3 = N4; two equal weights with opposite sign nullify, e.g., N3 + P3 = 0.

In the context of knowledge wikis the readability and intuitiveness of the markup is crucial for the success and its application, respectively. Therefore, we decided to *not* use an already existing standardized markup for (horn clause) rules like SWRL/RuleML [4], but to promote a more compact and human–readable notion.

In the following example two rules are given: the abstraction rule r1 derives the body-mass index *BMI* as a new fact, if the two inputs *Height* and *Weight* are defined and greater than zero. The *scoring rule* r2 adds the new scoring weight P4 to the score of the solution *Running*, if the user has entered the input *Training Goal is endurance* but has not answered the question *Physical Problems* with the value *knees*.

```
<Rules-section>
  // Abstraction rule r1
  IF   (Height > 0) AND (Weight > 0)
  THEN BMI = (Weight / (Height * Height))

  // Scoring rule r2
  IF   ("Training goals" = endurance)
       AND ("BMI" < 30)
       AND NOT("Physical Problems" = knees)
  THEN Running = P4
</Rules-section>
```

The surrounding tag `<Rules-section>` is used as a sub-tag of `<Kopic>` and collects all rules defined for the corresponding knowledge base. The order of rules is not meaningful, e.g., with respect to a conflict resolution strategy of rule engines.

3.2 Decision Trees

The representation of classification knowledge using decision trees is very popular in machine learning research, but is also widely used for building knowledge systems manually. A tree–like structure cannot be represented directly in a textual notion. Therefore, we have decided to use hyphens ("-") in order to express the depth of a particular tree node. The root of a tree has no preceding hyphen, whereas the ancestors of a root are depicted with an increasing number of hyphens. The following example shows the two decision trees *Naive Sport Advisor* and *Muscle Questions* (the line numbers are only added for describing the particular lines).

```
 1|  <Questions-section>
 2|  Naive Sport Advisor
 3|  - Training goals [oc]
 4|  -- endurance
 5|  --- Physical problems [oc]
 6|  ---- knees
 7|  ----- Swimming (!)
 8|  ---- no problems
 9|  ----- Running (!)
10|  -- increase muscles
11|  --- Muscle Questions
12|
13|  Muscle Questions
14|  - Favorite regions of muscles [mc]
15|  -- arms
16|    ...
17| </Questions-section>
```

Inputs. Inner nodes of a decision tree define the user inputs with their type information in one line. In the example above, three questions are defined in the lines 3, 5, and 14. The type of an input is defined in brackets right after its name, e.g., `[oc]` stands for a one–choice input, `[mc]` stands for a multiple–choice input, and `[num]` stands for a numeric input. The possible answers of the particular inputs (with indent i) are defined in the subsequent lines using an indent increased by one, i.e., $i + 1$. For example, the question *Training goals* (line 3 with indent $i = 1$) defines the possible answers *endurance* (line 4) and *increase muscles* (line 10), both with indent $i = 2$. Besides the creation of the user inputs a dialog strategy is also defined by decision trees: possible answers of an input are denoted in an extra line of the markup. If such an answer is followed by a user input with an indent increased by one, then this input is interpreted as a follow–up input to be presented when the previous input was answered with the given value.

In knowledge wikis we also use the markup of decision trees to define new user inputs. Then, the attachment of solutions and indications at the leafs of a tree are optional.

Solutions/Indication. The leafs of a decision tree are lines that have no immediate subsequent line with a larger number of hyphens, e.g. lines 7, 9, and 11. Typically, solutions

are derived in the leafs of a decision tree by writing a scoring weight in parentheses right after the name of the solution. We use scoring weights that were already introduced for scoring rules (cf. Section 3.1) Alternatively, solutions can be derived categorically by using a exclamation mark, e.g., in line 9 the solution *Running* is derived categorically by "(!)". Besides solutions the leaf of a decision tree can also contain the indication of another decision tree, e.g. as the decision tree *Muscle Questions* is called in line 11. Then, another decision tree is activated at this point of interview instead of deriving a solution. The activation of other decision trees allow for the modularization of larger decision trees into components, as for example proposed by the pattern *heuristic decision tree* [9].

The most prominent advantage of decision trees is their intuitive definition of a dialog control, i.e., the structuring and order of inputs to be presented to the user. In the example above, the input *Physical problems* is only presented if the preceding input *Training goals* was answered with *endurance*. Thus, a compact and meaningful interview is defined very easily. Decision trees having the shown size are very easy to understand and to maintain. However, for larger decision trees it is recommendable to partition the tree logic in a set of sub–trees and refer to these sub–trees from a main tree. This is sketched with the two exemplary trees *Naive Sport Advisor* as the main tree and *Muscle Questions* as a sub–tree.

The surrounding tag `<Questions-section>` is used as a sub-tag of `<Kopic>` and collects all decision trees defined for the corresponding knowledge base. Since trees can be easily formulated using XML it would have been an obvious approach to also represent the decision tree by an XML structure. Although this would yield a reduced compilation effort due to the existence of well–established XML parsers the resulting markup appears to be too verbose and complex for standard wiki users. In contrast, the proposed markup is by far more compact and less vulnerable to syntax errors made by the user when defining the decision tree.

3.3 Set–Covering Models

The knowledge markups introduced above considered deductive knowledge representations, that are commonly useful for defining classification knowledge. However, in the context of recommendation systems it is sometimes preferable to implement an abductive knowledge representation.

Set–covering models [10] are a prominent and intuitive example of an abductive knowledge representation. Solutions are described by features that can be typically observed when the solution is present. When the user enters inputs the best matching solution is selected, i.e., the solution that *best covers* its expected features with the entered findings. As a special feature, such models can be incrementally extended by background knowledge in order to improve the expressiveness of the knowledge [3].

For example, the features can be weighted with respect to their importance for the respective solution or partial similarities can be defined to refine the abductive matching process. In Figure 5 a visual notion for describing the sport "Running" is depicted; e.g., for a present solution "Running" we would expect to observe the feature "Favorite regions of muscles" with value "legs" or "bottom" etc.

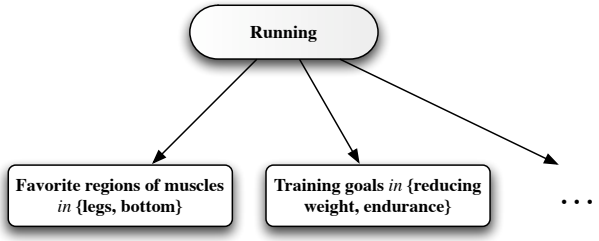

Figure 5: A visual notion of set–covering models.

For the application of set–covering models in knowledge wikis we introduce the simplified version *set-covering lists*: here, for each solution a collection of inputs is defined that are typically observed/entered when the solution is appropriate; the inputs are enclosed in braces. The example below shows a set–covering list for the solution *Running*:

```
<SetCoveringList-section>
  Running {
    Favorite regions of muscles = legs,
    Favorite regions of muscles = bottom,
    Training goals = reducing weight,
    Training goals = endurance,
    Calorie consumption (30min, 75kg) = high,
    ...
  }
</SetCoveringList-section>
```

Typical user inputs are listed that were expected to be observed for the sports form *Running*, e.g., the preferred muscle parts to be trained would be answered with *legs* and *bottom*, and the goals of doing the sport are (amongst others) *reducing weight* and *endurance*.

The surrounding tag `<SetCoveringList-section>` is used as a sub-tag of `<Kopic>` and collects all set–covering lists defined for the corresponding knowledge base.

3.4 Reuse of Terminologies

For every solution we commonly create a new wiki page, that describes its characteristics as text and multimedia (as done in normal wikis), and also defines explicit problem–solving knowledge to derive the particular solution for possible user inputs.

With the growth of the wiki a number of knowledge bases is defined distributed over the available wiki pages of the entire wiki. The particular knowledge bases are connected with each other by using user inputs with the same names. The reuse of inputs can be encouraged by providing a pre–defined terminology, that is referenced in every wiki page. In a particular knowledge base we reference to a pre–defined terminology by adding the optional attribute `terminology` to the `Kopic` tag, e.g., in the following markup

```
<Kopic id = "myKopic"
       terminology = "WikiPage..kopicID">
   ...
</Kopic>
```

the knowledge base with ID `myKopic` references to the terminology of another knowledge base with the ID `kopicID` having the namespace `WikiPage`.

4. SEMANTIC ANNOTATION OF KNOWLEDGE WIKIS

In the previous section, the inclusion of explicit problem–solving knowledge was described. Ontological concepts (inputs, solutions) and the corresponding derivation knowledge is easily defined in a textual manner together with the text of the corresponding wiki page. For each knowledge base a link is created in the view mode of the wiki; the link initiates an interview to collect user inputs in a structured way. In this section, we describe a more integrative approach for collecting user inputs: by annotating particular phrases of the wiki text the user is able to answer distinguished inputs in–place during browsing a wiki page.

User inputs are used to annotate text phrases, when they were previously defined in knowledge bases of an arbitrary wiki page. Similar to the annotation techniques known from semantic wikis, e.g. [5, 11], in knowledge wikis the semantic meaning of text phrases can be annotated by concepts of the input terminology. In contrast to semantic wikis the annotation in knowledge wikis is mainly used for problem–solving, i.e., by creating interactive menus to enable data entries of the user within a wiki page. The presented knowledge wiki introduces the markup

```
%TAG{ns="a" input="b" text="c"}%
```

in order to annotate the displayed wiki text `c` with an input concept `b` known from the knowledge base with the namespace `a`. If the optional namespace `ns` is not specified, then the knowledge base contained in the corresponding wiki article is used to align the annotated input. In the example below, an extract of the normal wiki text of the article *Swimming* is annotated in order to link text phrases with corresponding user inputs, i.e., questions.

```
Swimming is good for
 %TAG{input="Training goals"
      text="reducing weight"}%
successfully or to train
 %TAG{input="Training goals" text="endurance"}%.
```

Here, the input “Training goals” defined in the wiki page “Swimming” and its included knowledge base “Swim” is annotated twice: first, with the text “reducing weight” and second with the text “endurance”.

In the view mode of an wiki article for the tagged text phrases pop–up menus are generated that enable the user to answer the annotated inputs *in–place*. For example, Figure 1 shows the view mode of the wiki page “Swimming”; the click on the text phrase “endurance” activates a pop–up asking for the “Training Goals” of the user, where “endurance” is a possible answer. As described in Section 2.3 the acquired user inputs are mainly used for problem–solving.

5. CASE STUDIES

The applicability of knowledge wikis and their markup, respectively, was evaluated in two preliminary case studies.

5.1 Sport Recommendation Wiki

The first case study was intended as a “proof–of–concept” experiment: here, an exemplary recommendation system for forms of sport was implemented by three students, that were already familiar with the wiki system and the underlying knowledge representation. The recommendation system comprises 56 knowledge bases for 31 of the most common forms of sports (some sports are redundantly defined reflecting different opinions of the students), and defines in total 38 user inputs. The knowledge bases mostly use set–covering knowledge for the definition of the derivation knowledge, but some decision trees are also included.
With this first case study the general applicability of the knowledge wiki and its markup was evaluated. An initial table–like representation of set–covering knowledge for defining knowledge for multiple solutions was discarded, since the markup proved to be not very comfortable for the acquisition and maintenance of the relations. Because the knowledge bases in a knowledge wiki mostly define knowledge for a single solution, a table representation could not show its actual power. Furthermore, tables are often difficult to maintain in the textual edit pane of a wiki. Instead a list–like markup of set–covering knowledge as presented in this paper was introduced. This was experienced to be more compact and intuitive for the definition of the knowledge for single solutions. In summary, the results were encouraging and a second, now larger case study was initiated.

5.2 General Recommender Wikis

The second case study considered the creation of knowledge wikis with a larger group of initially untrained persons. In total, 45 students started to implement 11 knowledge wikis with varying sizes. The students formed groups, where each group undertook the development and evolution of one knowledge wiki. For example, knowledge wikis for meal selection, recommenders for holidays and recreation, a movie advisor, and a wine selection were created. The untrained students were initially introduced into the concepts of the knowledge wiki and its markup (about 1h). Then, the students started to build initial versions of their systems. A trivial movie recommender was included as a demo knowledge wiki and helped the new users to get familiar with the system. In fact, most of the new knowledge bases were implemented by simply copying, pasting, and adapting the demo markup. The set–covering list appeared to be the predominant knowledge markup for the second case study, since it was experienced to be intuitive and compact in most of the cases. A significant disadvantage of set–covering models are their inability of representing negative (exclusion) knowledge, i.e., the presence of user inputs for which a solution should be discarded from the recommendations. This problem was tackled by the introduction of a hybrid rule–extension: here, the users are able to define rule

conditions under which a solution is excluded even when the set–covering result was positive. The second case study produced about 700 knowledge bases spread over 11 knowledge wikis; we counted about 2500 edits of the particular knowledge bases not including the initial creation of the projects (the logs were not available from the beginning of the case study).

In summary, we experienced the proposed knowledge markup as an intuitive and compact representation to enable nearly untrained users to capture and maintain knowledge in a collaborative way. Whereas the markup for implementing derivation knowledge (rules, set–covering lists) was successfully applied in both case studies, the semantic annotation of text phrases was only evaluated in the first case study. Since, the manual annotation of text appeared to be very time–consuming the simplification of this approach, e.g., by semi–automatic methods, is an important issue for future work.

6. CONCLUSIONS AND OUTLOOK

The development of larger knowledge systems and especially their maintenance is still a complex and open problem. Even before the success of Web 2.0 applications with the collaborative creation of information and knowledge, e.g., by tagging and writing, many researchers have started to think about the collaborative capture and update of knowledge systems. The presented work introduced a collaborative approach for knowledge capture and its necessary markup tightly integrated in a wiki system. Here, wiki systems allow not only for the development of unstructured content like text and multimedia, but also provide the intuitive acquisition of explicit knowledge. From our point of view, the most important aspect for the success of such a proposal is the presence of intuitive markups, in order to enable even untrained/little trained users to become knowledge wiki developers. We have presented different possible knowledge markups that are easy to understand and to apply, and that are used to collaboratively build knowledge wikis for recommendation tasks. The case studies showed the applicability of the presented approach and especially the proposed knowledge markups.

Most related to knowledge wikis are the approaches proposed for the development of semantic wikis. For example, Semantic MediaWiki [5] enhances a normal wiki semantically in order to annotate wiki texts with explicit ontological concepts. Annotations are placed in–text and are based on the top concept the actual wiki page is describing. Properties for that concept can be easily formalized within the normal wiki text. Semantic wikis usually provide markups for ontological relations and attributes using a web ontology language like OWL. In contrast, the ontological expressiveness of the proposed knowledge wikis is limited to two basic objects, i.e., input concepts and solution concepts. However, we additionally support various knowledge markups to formalize problem–solving knowledge in a simple manner. Semantic wikis support the manual problem–solving process, i.e., the search for appropriate information for a given problem, by semantically enhanced search engines and dynamic linking between related concepts. Knowledge wikis exploit explicit problem–solving knowledge to handle stated problems in a more knowledge–based way.

In the near future we are planning to exploit existing learning methods for automatic annotation: Since the manual annotation of text phrases in the wiki text appeared to be very time–consuming, we want to simplify this approach by (semi–)automatic methods. Besides the introduction of appropriate tools attached to the edit pane of the wiki to insert annotation templates by a mouse click, we are also planning to suggest annotations based on the results of learning methods.

7. REFERENCES

[1] J. Baumeister. *Agile Development of Diagnostic Knowledge Systems*. IOS Press, AKA, DISKI 284, 2004.

[2] J. Baumeister, J. Reutelshoefer, K. Nadrowski, and A. Misok. Using Knowledge Wikis to Support Scientific Communities. In *Proceedings of 1st Workshop on Scientific Communities of Practice (SCOOP)*, Bremen, Germany, 2007.

[3] J. Baumeister, D. Seipel, and F. Puppe. Incremental Development of Diagnostic Set-Covering Models with Therapy Effects. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11(Suppl.):25–50, November 2003.

[4] I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. OWL Rules: A Proposal and Prototype Implementation. *Journal of Web Semantics*, 3(1):23–40, 2005.

[5] M. Krötzsch, D. Vrandecic, and M. Völkel. Semantic MediaWiki. In *Proceedings of the 5th International Semantic Web Conference (ISWC06), LNAI 4273*, pages 935–942, Berlin, 2006. Springer.

[6] B. Leuf and W. Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley, New York, 2001.

[7] R. A. Miller, H. E. Pople, and J. Myers. INTERNIST-1, an Experimental Computer-Based Diagnostic Consultant for General Internal Medicine. *New England Journal of Medicine*, 307:468–476, 1982.

[8] F. Puppe. Knowledge Reuse among Diagnostic Problem-Solving Methods in the Shell-Kit D3. *International Journal of Human-Computer Studies*, 49:627–649, 1998.

[9] F. Puppe. Knowledge Formalization Patterns. In *Proceedings of PKAW 2000*, Sydney, Australia, 2000.

[10] J. A. Reggia, D. S. Nau, and P. Y. Wang. Diagnostic Expert Systems Based on a Set Covering Model. *Journal of Man-Machine Studies*, 19(5):437–460, 1983.

[11] S. Schaffert. IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In *1st International Workshop on Semantic Technologies in Collaborative Applications (STICA'06)*, 2006.