An Approach to Annotation of Learning Texts on Programming within a Web-Based Educational System

Vladimír Mihál and Mária Bieliková
Institution of Informatics and Software Engineering
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 3, 842 16 Bratislava, Slovakia
Email: {xmihalv, maria.bielikova}@stuba.sk

***Abstract*—Many approaches have been developed to simplify user navigation and information search in large information spaces. Annotations, i.e. notes, comments, explanations or other types of remarks bring to these activities new dimension. Annotations enrich the content and also support collaboration. This paper presents a method for annotation learning materials on programming. Learning materials in domain of programming consist of text mixed with programming exercises and code samples. The main idea of the annotation is an extension of learning materials by additional information helping students to understand learning subject easier and faster together with minimizing a need of finding supplemental information in external sources such as reference guides. Additionally, user annotations provide a context based communication space for students and their teacher: students add annotations into the learning text to ask, suggest or comment and the teacher evaluates the annotations and responds. To verify value of this approach we tested proposed method for annotation learning texts within an existing learning system in domain of functional and logic programming.**

***Keywords*-learning text annotation; automatic annotation; generic annotation; context annotation; learning programming;**

I. Introduction

The Web provides a large-scale environment for information exchange. As the Web is nowadays available to practically everyone and everywhere, it is suitable and currently widely used for educational purposes. With increasing flexibility and variety of the Web content, educational use of the Web evolved from publishing of static learning texts into a usage of interactive personalized web-based systems, which support learning, advice and examination of students.

Due to mentioned changes, the quantity of information available to every student dramatically increased, making knowledge organizing and finding one certain information much more difficult. Possibilities of content navigation, search and presentation increased too. Many approaches have been developed to simplify user navigation within large information spaces and information search, often based on semantics [1]. We have chosen learning text annotation as an approach, which further improves current state of the art by providing additional context based information related to the annotated text (be it a keyword or phrase) to students, and exact fast content related feedback to teachers.

We present a method for educational texts annotation. We consequently evaluate, whether annotations provide a student additional benefits and simplify the learning process. Proposed approach is applicable in various domains; however, we specifically designed some features considering the domain of learning programming, which served also for an evaluation of proposed approach.

One of the most common problems within the learning programming is understanding particular elements of program code and the ability to generalize patterns from existing code on one side, and specialize existing patterns to the specific solutions on the other side. A student, especially one who is not familiar with the syntax of particular programming language or API used in the presented sample code usually has to consult a reference guide or appropriate supporting documents every time when reading a sample. Modern integrated development environments (IDE) provide several ways to view API documentation or other description of used programming language constructs, e.g., a method, procedure or other element of code, for instance by a tool tip or by separate help window or frame.

The most important advantage of in-text help (or explanation) over separately located documentation is absence of distraction caused by drawing attention away from the main task of the student. Placing in-text help into learning texts (and programming code samples in our case) helps to keep related pieces of information together allowing an educational system to be more context based instead of a hierarchy (chapter, section) based. Of course traditional course structure is also useful and should be available in the educational system.

Another significant aspect of text annotation is communication aspect. If we allow the students to insert their questions or suggestions as annotations, they spread over the learning text and would grow into context based discussion. It also leads to grouping of information within the same context to one place, for instance questions are answered exactly where they have risen. Students can also be guided through the course by annotations towards recommended content by adaptive link annotation [2], where links are augmented with personalized hints informing the student about the current state of learning objects behind the annotated links.

The paper is structured as follows. In Section 2 we describe related work based on examples of currently available systems for the annotation. In Section 3 we discuss principles of document annotation in general. Section 4 describes our approach to learning texts annotation, specifies documents we aim our work at. In Section 5 we present an evaluation of proposed method – developed prototype of the learning text annotator and the experiment we performed. Section 6 concludes our paper with the summary of our work and presents ideas for further work.

II. Related work

The meaning of document annotation varies in current annotation related research. Basically by annotations we mean notes, comments, explanations, or other types of remarks that are attached to presented document (e.g., web page or an educational text). This means that we deal in this paper the content annotation. In fact many existing web-based systems support the annotation in various simple forms such as highlighting or underlining parts of the text or associating free comments with parts of the text. This techniques are successfully employed in adaptive web-based systems for the years [3].

One viewpoint on the content annotation is employing semantics. This approach considers the annotation as a task aimed at mapping annotations to the text based on its semantics [4]. The process of semantic annotation can be automatic, semi-automatic or manual. In the automatic semantic annotation process we map metadata into the text document automatically based on discovered semantics. For example, the annotation part of the KIM project [5] is looking for named entities (e.g., people, organization, location names) and links them to their semantic descriptions.

One example of the manual semantic annotation systems is Ontology-Based Information Extraction Manual semantic annotation (a part of GATE, General Architecture for Text Engineering platform, http://gate.ac.uk/). Its main purpose is a support of manual information extraction from text data. A user manually marks words from a document and then selects appropriate classes from the opened ontology for each selected word (e.g., “day of week” class for word “Friday”). Semantically annotated document is further automatically processed considering added semantics.

An example of automated semantic annotation is Pannda software tool [6]. It online annotates any web document in real time – during its accessing. Pannda finds instances of concepts from given ontology in the text presented on the web page being annotated. Then enriches it with information related to instances found. Server side of the system gets the URL of the web page from the client application, downloads and processes the document, and finally sends the annotation data to the client. On the client side requested web page (which was regularly downloaded by the web browser) is combined with prepared annotation data. This approach assumes existence of well prepared domain ontologies. Moreover, the author does not consider a problem of possible overloading annotations as often seeing all the annotation can be confusing.

Another approach to automatic annotation called OnTea focuses on text processing using regular expressions patterns and detecting concepts according to the domain ontology [7]. However, this method of information extraction is suitable only if the set of concepts to detect is well known.

Entirely different approach to the annotation is the content enrichment by users. Several systems for user webpage annotation already exist. For instance the Co-ment annotation system (http://www.co-ment.net/) provides an annotation of any given text (there are some texts for testing on the project homepage) by any user. Visitors can also react (or reply) to existing annotations by other annotations.

Considering e-learning domain interesting approach is linking Wikipedia’s text to the structural knowledge that provides summary information [8]. It moves further semantic annotation using NLP techniques for basic semantic tagging followed by a mapping the found tags and DBpedia collections (http://www.dbpedia.org).

Most current annotation systems are realized either as automatic annotating engines for text documents or for the user commenting only. We believe that automatic content annotation approach combined with user annotations and interactions with annotations inserted already would be versatile as well as attractive to students. The impact of the shared document annotation in collaborative learning was already proved as positive. The authors in [9] report observed higher learning outcomes and better performance. However, a kind of annotations filtering is necessary in order to lower information overhead when many annotations are going to be presented.

III. Basic principles of annotation process

In general, annotation is a process of document enrichment with additional information especially in form of notes or comments. This information has to be created or collected, stored and then inserted back in some form into the document to be presented to the user. Most significant part of the annotation is choosing right enriching information to insert and finding appropriate place in the document for it. From this point of view we recognize two ways of the annotation: automatic annotation and manual user created annotation.

Annotation process generally works in *read—retrieve—insert* cycle for each document or its part:

1) *Read:* preprocessing, analyzing content of the document.
2) *Retrieve:* selecting appropriate annotation for the document.
3) *Insert:* inserting annotations into determined places in the document.

The automatic annotation processes documents entirely without a human interaction. Information for annotations is inferred obviously using a repository of annotations. Therefore it requires the base of additional information already present (created manually or automatically).

In manual annotation major parts of all three steps of the annotation are performed by a user. The user

- reads the document,
- decides what information to insert and
- which part of the document to annotate.

Inserted annotations are in the manual annotation process often specific to the annotated document and not applicable to other documents.

Considering generality of annotations we distinguish two types of annotations:

- *Generic annotations:* annotations used to explain domain related terms.
- *Context annotations:* annotations strongly related to the document and certain place within the document.

Main purpose of a generic annotation is to explain a meaning of particular concept. In our case the annotation is linked for example the programming language construction. It carries explaining information for given term. Generic annotation is identified by a concept (keyword, token or ontology concept), which is explained or further elaborated in the annotation. Generic annotation always belongs to certain knowledge domain. This type of annotation is used anywhere, where it is possible or suitable according to topic of the document and user preferences (like IDE documentation tool-tip).

Context annotation is a document specific annotation associated to the particular document and particular position within it (where it is initially placed). Example of such annotation is a student question about certain line of programming code sample. Since context annotations are usually in their fixed positions within the document problems may emerge when the document is radically changed, e.g., annotated part of document is completely removed.

Context of inserted annotations can become not appropriate to the modified document and those annotations can become unusable or misleading. One way to solve this problem is to give users an ability to mark annotations as wrong or not appropriate and to check retrieved context annotations for the document whether they are applicable to current version of document and consequently remove wrong annotations from the repository.

It is possible to use document annotation with any kind of document including text, pictures, videos or sound. When creating annotations (semi)automatically we focus on text documents represented as HTML files, since this are the most common document formats for learning texts found on the Web.

IV. Method of annotation learning texts on programming

Automatic annotating of general text documents is rather comprehensive task, which is currently impossible to provide without large-scale annotations repository. We focus on annotating learning texts from particular domain. We have experience in learning programming domain, there are learning texts available and any improvement within this domain would help students to learn.

Texts for learning programming usually consist of two different kinds of content:

- Semistructured texts with possible media included (in particular pictures or videos).
- Programming language samples that have rather well defined structure.

In process of annotation we identify and separate mentioned parts of the educational text as to achieve the best results, we should process them differently. In most of documents about programming the formatting style of programming code samples differs from the formatting style of other text and it is also particularly marked (for instance in HTML it is obviously included within the `<code>` or `<pre>` tags).

Processing programming code is much simpler task. Programming languages are formal, have strict rules, well defined grammar and great amount of information can be extracted using patterns. Source code comments are either excluded from the annotation process or annotated as a free text. As for enriching information, every reserved word, element of programming language or paradigm typically has short explanation.

Annotation of free text is divided into several steps. Text is tokenized and then meaningful keywords or even concepts are recognized. After the recognition, most suitable annotations for recognized entities are chosen.

Our method for annotation learning texts consists of these four steps of document processing:

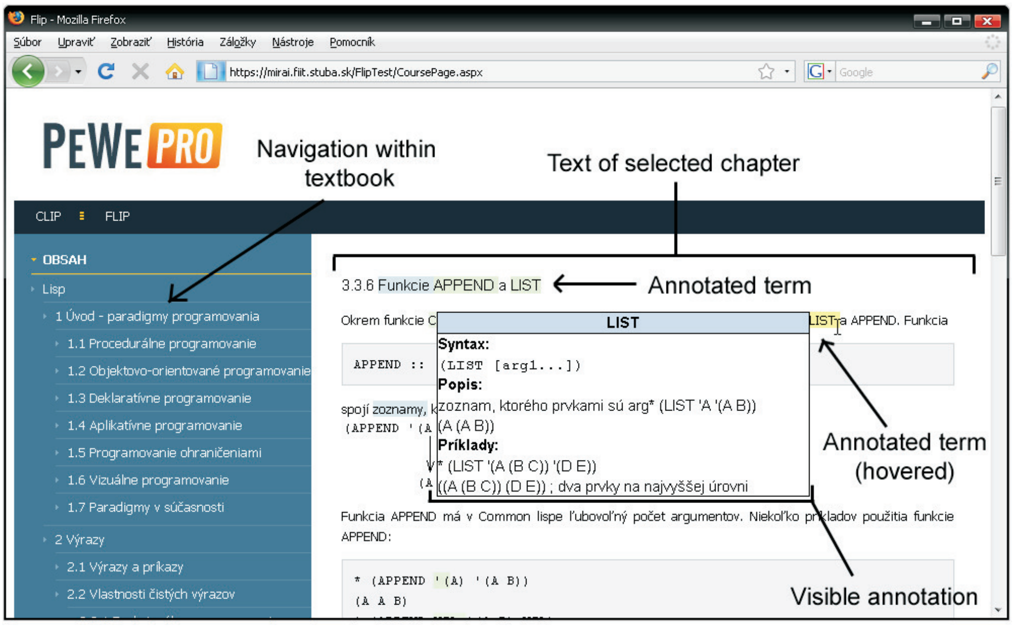
1) Document analysis.
2) Fetching of annotations.
3) Filtering the annotations.
4) Inserting annotations into the document.

Analysis step of document processing divides the source documents into tokens, which are consequently annotated. For document analysis we can use several approaches with different complexity, from rather simple word matching to more advanced syntax based analysis of text.

In annotation fetching step we collect all possibly fitting annotations (if any) from the repository for each recognized token. It is possible that there exist several annotations for one token.

The filtering step is responsible for deciding which annotations are inserted into the document and how they are visualized. Annotation filter decides according to the rules, which are predefined or dynamically changing by actions in the annotator. Filtering step gives us possibility to change the visualization of annotations to be suitable for every student individually (a decision whether particular annotation or set of annotations is displayed for specific user at given time).

Goal of inserting step is to create an output document that contains original content with contextually inserted annotations. The inserting is rather specific to the type of document to be annotated. Since we process text documents in HTML format, resulted document created in this step contains annotated words surrounded with additional HTML tags.


Flip - Mozilla Firefox
Súbor Upraviť Zobraziť História Záložky Nástroje Pomocník
https://mirai.fiit.stuba.sk/FlipTest/CoursePage.aspx
Google
PeWe PRO
Navigation within textbook
Text of selected chapter
CLIP FLIP
OBSAH
Annotated term
Annotated term (hovered)
Visible annotation

- Lisp
  - 1 Úvod - paradigmy programovania
    - 1.1 Procedurálne programovanie
    - 1.2 Objektovo-orientované programovanie
    - 1.3 Deklaratívne programovanie
    - 1.4 Aplikatívne programovanie
    - 1.5 Programovanie ohraničeniami
    - 1.6 Vizuálne programovanie
    - 1.7 Paradigmy v súčasnosti
  - 2 Výrazy
    - 2.1 Výrazy a príkazy
    - 2.2 Vlastnosti čistých výrazov

3.3.6 Funkcie APPEND a LIST

Okrem funkcie C LIST a APPEND. Funkcia

```
APPEND ::
```

spojí zoznamy, k

```
(APPEND '(A
```

LIST

Syntax:

```
(LIST [arg1...])
```

Popis:

zoznam, ktorého prvkami sú arg* (LIST 'A '(A B))
(A (A B))

Príklady:

* (LIST '(A (B C)) '(D E))
((A (B C)) (D E)) ; dva prvky na najvyššej úrovni

Funkcia APPEND má v Common lispe ľubovoľný počet argumentov. Niekoľko príkladov použitia funkcie APPEND:

```
* (APPEND '(A) '(A B))
(A A B)
```

Figure 1. Visualized annotation within learning system Flip.

V. Evaluation

For evaluation of proposed method for annotation we implemented a prototype of learning text annotator. There were several preconditions for the evaluation of our annotator as we decided to integrate it to existing adaptive educational system Flip [10]. Flip is an adaptive web based educational system for learning programming. It is based on evaluation of user knowledge using test questions which are inserted into learning materials. Basically it serves as interactive learning book (see Figure 1) with enhanced functionality including:

- Creating and maintaining student knowledge model reflecting what student (should) know or have learned.
- Advising most appropriate topics of learning texts according a level of students' understanding of current subject.
- Selecting test questions and determining students' level of knowledge within a learning subject.

We knew exactly the document which served as an input, a textbook for Functional and logic programming course. This textbook has been already successfully used in previous experiments with mentioned educational system [11].

We implemented the annotation process as a stream processing of the document:

- In the analysis step, the annotator divides the document into tokens and transforms them into the normalized form (e.g., removing capitalization, diacritic).
- The fetch step handles tokens consequentially, for each token it obtains available annotations from the repository.
- In the final step, the document is recreated from tokens and chosen annotations.

Annotations repository is realized using an XML file for persistent storage of annotations and modified hash map as a memory model of the repository. The repository was designed to map N annotations onto one keyword. Fetching annotations operates in two steps. At first, we look for the exact match of keywords in the incoming token. If no keyword is found, we proceed with the search of the closest match with given token; if sufficiently similar keyword is found the annotation for the keyword is returned.

Visualization of annotations is implemented using separate CSS style for annotated document and nested <span> tag. Various annotated terms are highlighted with different colors; in tested implementation we have used two colors (for explanations of terms and functions of programming language). In Figure 1 we see blue color used for explanations of terms such as "functions" ("funkcie" in Slovak) or "lists" ("zoznamy" in Slovak) and green color for Lisp language functions such as LIST. Colors used for highlighting of annotated terms are pale in effort not to disturb student while reading the document.

Since highlighting every annotated term makes often the document less readable than original (even with barely visible highlighting colors), we proposed a filter controlling visualization of fetched annotations. Highlighting same annotated terms which are close to each other is rather unnecessary. We defined minimal distance measure between identical terms to be highlighted both and implemented the distance filter to control highlighting of terms. Distance filter maintains the count of all processed tokens (current position within the stream) and for every annotated token it holds position where the token was last time highlighted.

The filter checks the last highlighted occurrence of incoming token and if

$$currentPosition - lastPosition \geq minimalDistance,$$

the token is going to be highlighted. Minimal distance is initially set to 70 tokens[1], but it can be changed through web service interface for every annotated document. However, even not highlighted terms, which were annotated are active, the user can interact with them and see attached annotation.

We obtained data for annotations semi-automatically. Main source for the annotations was the documents itself. Textbook used contains the list of the most common Lisp functions, each with a short explanation and syntax. We also noticed that every example of input/output of Lisp interpreter (demonstrating the usage of functions) contains the same sequence of characters, to be exact "*_(" (underscore represents space). Using simple regular expression matching we gathered more than 100 examples and matched them to already obtained list of functions. As a result we got annotations for 65 functions, each with syntax, description and example of use. Additionally we manually created 19 annotations containing explanations of terms related to functional programming and Lisp language.

Client side of the Flip web interface is written JavaScript; server side is developed using ASP.NET technology. Since our annotation system is developed in Java, it was necessary to design a communication interface for successful integration. For this purpose we developed web service interface for the annotator.

Described loose integration with the Flip system allowed us to easily test our annotator without the need of creating separate system just for evaluation.

We tested our annotator in the experiment aimed broader to an analysis of behavior of learners of programming languages using several means for increasing effectiveness of learning mainly based on personalization. The experiment was aimed at programming learning with assistance of interactive content inserted into learning text. We could also compare effectiveness of annotation and adaptive navigation with questions.

For each student participating in the experiment we had his grade point average (GPA), hence we had an estimation of abilities and skills of particular student. Students were divided into three groups (marked as A, B and C). Each of group had different settings:

- A – plain text without any interactive content,
- B – text with annotations,
- C – text with questions and adaptive navigation within textbook.

The students had 90 minutes to learn basics of functional programming using Flip. Then they were given a test focused on practical Lisp forms evaluation and construction. Our hypothesis was that the students from group B achieve better results than students from group A. Results of the experiment are in Table 1.

The experiment showed that average result of the group B was not significantly different (0.05 point difference

[1]Allowing maximum of two the same highlighted terms within one screen; it was tested in the computer lab, where the experiment was held.

Table I
RESULTS OF TEST IN THE EXPERIMENT.

	All students		GPA≥1.8	
	Avg.	Sd.	Avg.	Sd.
A	15	5.25	12	3.92
B	14.95	3.77	13.4	1.71
C	16.59	5.05	15.25	3.05

of total 27 points) from result of group A, while results of group C were considerably better as we can see in Figure 2. At this point we considered that annotations had not affected all students in same way.

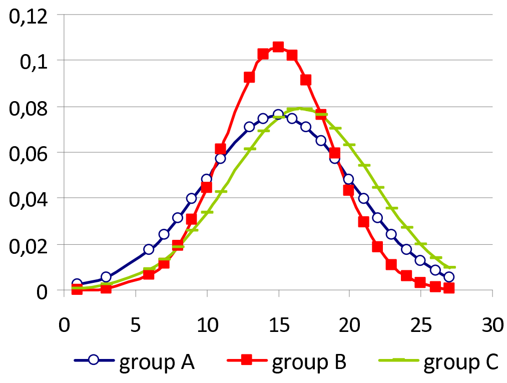


Figure 2. Standard distribution of results.

We filtered out results of students with GPA less than 1.8 points (those with good studying results) and observed visible difference in results of groups a and B (in favor of group B, see Figure 3), while result of group C were even better. In addition we noticed remarkably smaller standard deviation of test results of students in group B which shows another benefit of the annotation, specifically helping weaker students to reach average level of knowledge faster.

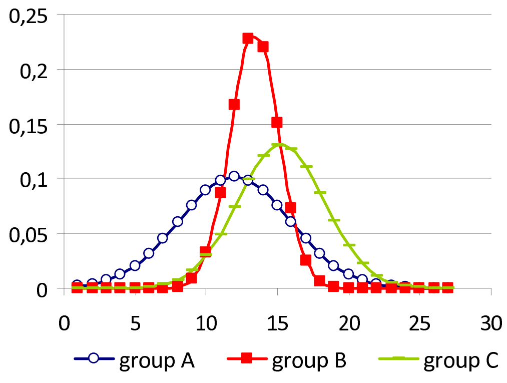


Figure 3. Standard distribution of results of students with GPA ≥1.8.

We did similar filtering of results focused on students with better GPA. It showed us almost same average results for all three groups (results were in 16.5 ± 0.3 point range).

From these results we can assume that annotations give most noticeable benefits to weaker students, while better students are to achieve good results without such assistance (we should have in mind that students were presented with basics of functional programming language; the students did not have any previous knowledge on functional programming, but clever student approaching a quality text is obviously able to better to comprehend the content which is on basic novice level).

VI. Conclusions

Annotations are being more and more widely used in documents on the Web. This brings the content enrichment and interaction to every document on the Web. Content annotation is useful approach also to assist students in web-based educational systems. Annotations are very helpful if they are inserted into documents containing many terms specific to subject. As we saw in the results of our experiment, annotations showed value in assistance to weaker students helping them to achieve results closer to average result of all students.

We experimentally evaluated automatic annotations as one part of the proposed concept. Manual user annotations can also produce considerable advantage over plain document and bring important social aspect. In our future work we will explore more deeply the effect of user annotations within learning system on studying results of students.

Another important part of our future research is making annotations adaptive to a document viewer including adaptive annotations filtering and visualization. It is based also on the fact that user annotations together with tags represent fine source of user interests [12]. Moreover, annotations or tags can serve also for identification of user groups as similar annotations can lead to similar interests [13]. There is promising direction of combining annotations with adaptive navigation [14], which is suitable especially for educational texts and considering user characteristics acquired while he is studying educational materials [15].

Acknowledgment

This work was partially supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07 and by the Slovak Research and Development Agency, the contract No. APVV-0391-06.

We wish to thank Oto Vozár, Michal Barla and Marián Šimko (members of PeWe group, pewe.fiit.stuba.sk) for their invaluable contribution to the integration of our annotator to the Flip system and helping with the experiment.

References

[1] M. Tvarožek and M. Bieliková, "Reinventing the web browser for the semantic web," in *WIRSS'09: Proc. of the WIRSS Workshop at the IEEE/WIC/ACM International Conferencie on Web Intelligence*. IEEE Computer Society, 2009, pp. 113–116.

[2] P. Brusilovsky, S. Sosnovsky, and M. Yudelson, "Addictive links: The motivational value of adaptive link annotation," *New Review of Hypermedia and Multimedia*, vol. 15, no. 1, pp. 97–118, 2009.

[3] P. Brusilovsky, "Methods and techniques of adaptive hypermedia," *User Modeling and User-Adapted Interaction*, vol. 6, no. 2-3, pp. 87–129, 1996.

[4] L. Reeve and H. Han, "Survey of semantic annotation platforms," in *HSAC'05: Proceedings of the 2005 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2005, pp. 1634–1638.

[5] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov, "Kim – semantic annotation platform," *Journal of Natural Language Engineering*, vol. 10, no. 3-4, pp. 375–392, 2004.

[6] M. Adam, "An approach to automated on-line annotation," in *Proc. of research project workshop Tools for Acquisition, Organization and Presenting of Information and Knowledge*, P. N. et al., Ed., 2007, pp. 20–25.

[7] M. Laclavik, M. Šeleng, M. Ciglan, and L. Hluchý, "Ontea: Platform for pattern based automated semantic annotation," *Computing and Informatics*, vol. 28, no. 4, pp. 553–577, 2009.

[8] P. Mika, M. Ciaramita, H. Zaragoza, and J. Atserias, "Learning to tag and tagging to learn: A case study on wikipedia," *IEEE Intelligent Systems*, vol. 23, no. 5, pp. 26–33, 2008.

[9] P. Nokelainen, J. Kurhila, M. Miettinen, P. Floren, and H. Tirri, "Evaluating the role of a shared document-based annotation tool in learner-centered collaborative learning," in *In Proc. ICALT 2003*. IEEE Computer Society Press, 2003, pp. 200–203.

[10] O. Vozár and M. Bieliková, "Adaptive test question selection for web-based educational system," in *SMAP'08: Proc. of the 2008 3rd Int. Workshop on Semantic Media Adaptation and Personalization*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 164–169.

[11] M. Bieliková, "An adaptive web-based system for learning programming," *International Journal Continuing Engineering Education and Life-Long Learning*, vol. 16, no. 1/2, pp. 122–136, 2006.

[12] M. Barla and M. Bieliková, "On deriving tagsonomies: Keyword relations coming from the crowd," in *ICCI'09: Proc. of Int. Conf. on Computational Collective Intelligence*. New York, NY, USA: LNAI 5796, Springer, 2009, pp. 309–320.

[13] S. Bradshaw and M. Light, "Annotation consensus: implications for passage recommendation in scientific literature," in *HT'07: Proc. of the 18th Conf. on Hypertext and hypermedia*. New York, NY, USA: ACM, 2007, pp. 209–216.

[14] R. Farzan and P. Brusilovsky, "Annotated: A social navigation and annotation service for web-based educational resources," *New Review in Hypermedia and Multimedia*, vol. 14, no. 1, pp. 3–32, 2008.

[15] M. Barla, M. Tvarožek, and M. Bieliková, "Rule-based user characteristics acquisition from logs with semantics for personalized web-based systems," *Computing and Informatics*, vol. 28, no. 4, pp. 399–427, 2009.