Distributed Recognition of Content Similarity in a Tele-Teaching Portal

Maria Siebert
Hasso Plattner Institut
University of Potsdam
Potsdam, Germany
maria.siebert@hpi.uni-potsdam.de

Franka Moritz
Hasso Plattner Institut
University of Potsdam
Potsdam, Germany
franka.moritz@hpi.uni-potsdam.de

Christoph Meinel
Hasso Plattner Institut
University of Potsdam
Potsdam, Germany
christoph.meinel@hpi.uni-potsdam.de

Abstract**—**This paper focuses on the problem of finding similarities between complex e-learning objects. For a web portal, which contains a lot of different video objects, different types of meta data for these objects are available. To combine the information of these different data types to a combined similarity measure, a distributed calculation algorithm is introduced.

Furthermore examples for calculating the similarity of different aspects like the object title and tags. Some problems like calculation time are considered and solutions are proposed.

Keywords**—**similarity detection, recognition, e-learning, tagging

I. Introduction and Related Work

When the amount of content is increasing, it is hard for the user to find the content, he is interested in. Especially in tele-teaching, where the main content consists of video material, which cannot easily be searched, keyword based search is difficult. Therefore other methods for supporting learners in finding the appropriate content are needed.

Several approaches to provide support for the users in finding the content they are looking for are researched currently. One method that was evaluated in this context is to utilize common learning paths by tracking the learner's interests [1], [2]. Other methods are based on web 2.0 technologies, like community tagging [3]. Web filtering systems, where the search results can be limited, are also amongst the research fields [4]. Most of these options can be utilized for a personalization of the portal according to the user's needs and preferences [5]. A last and vastly researched field are recommendation systems [6], [7]. In the tele-teaching context those systems could be attached to lectures and provide additional information to a selected video or basic knowledge, which is needed.

Recognition of the similarity of content is a basis for building recommendation systems. Recommendation systems mostly consist of three parts. The first part is the comparison of the users, to find similarities between them. The second part is the analysis of the data, which is used to generate new knowledge of the content. At last it is possible to consider the outside influences on the system, like the date and time of the request or even the temporal closeness to a special event.

This paper focuses on the finding of similarities in the content. Therefore it will give a definition for similarity based on the distance of objects. Then it will show, how the distributed calculation of similarity can be done using a plug-in architecture. Afterwards we describe how the distributed calculated results can be combined using different approaches.

We will also give some examples, how the implementations of the plug-in interface can be done and which algorithm could be used. Furthermore some problems with possible solutions are discussed and possible enhancement are pointed out.

A. About tele-TASK

The tele-TASK system [8] is a portable lecture recording system. It allows to record two video and one audio streams parallel. It is used to capture the screen of the presentation as well as the presenter. This system is used since 2001. More than 3000 recordings in more than 400 series, containing lectures and conferences, have been produced so far.

These recordings are available at the tele-TASK portal (http://www.tele-task.de). As not only students from our institute, but also external persons use the portal, it is an ideal platform to research the content search process with real data and real users. Our research focuses on creating better learning experiences on basis of this available data.

II. Definition of Similarity

When comparing two objects o_i and o_k from the set O of all objects, the similarity of these objects $s(o_i, o_k)$ is a function $s : O^2 \mapsto \mathbb{R}_0^+$.

In this paper we will assume, that the similarity of two objects is the opposite to the distance of these two objects. Therefore we assume, that if two objects are equal, their distance is zero and their similarity is maximized and when they have nothing in common, their distance is maximized and their similarity is zero. For better handling we will assume, that the maximum of all values is 100, which should be a reference to percentage calculation. Therefore we get a function $s : O^2 \mapsto [0, 100]$.

Furthermore we define that the following relation between similarity $s(o_i, o_k)$ and distance $d(o_i, o_k)$ should be used:

$$s(o_i, o_k) = 100 - d(o_i, o_k) \tag{1}$$

So when two objects have the similarity of zero, they have the distance of 100 and when they have the similarity of 100, their distance is zero.

To visualize the distance $d(o_i, o_k)$ between each tuple of $n+1$ different objects, we need a n-th dimensional room, for being able to have the correct distance between all objects. Figure 1 shows this fact for the example of three objects and a two dimensional room.

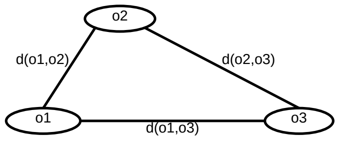

Fig. 1. Distance between three objects

The following facts should be concerned, when talking about the result of the similarity of objects.

- Symmetry: The result of $s(o_i, o_k)$ should be equal to $s(o_k, o_i)$, because the distance of two objects is obviously symmetrical.
- Maximimum of $s(o_i, o_i)$: For every object it is obvious, that the similarity of the object with itself is 100, because each object is equal to itself. Therefore the similarity of the object to itself is equal or higher than the similarity to any other object. It is easily possible to force the value of $s(o_i, o_i)$ to the value 100 by using a stretching factor for all calculations.
- Transitivity of the distance: Because of the triangle inequality the following rule has to be valid:

$$\forall i,j,k \ d(o_i, o_j) \leq d(o_i, o_k) + d(o_k, o_j) \tag{2}$$

Using the connection between distance $d(o_i, o_j)$ and similarity $s(o_i, o_j)$ as defined in formula 1 it should be visible that the following rules has to be followed:

$$\forall i,j,k \ -s(o_i, o_j) \leq 100 - s(o_i, o_k) - s(o_k, o_j) \tag{3}$$
$$\forall i,j,k \ s(o_i, o_j) \geq s(o_i, o_k) + s(o_k, o_j) - 100 \tag{4}$$

In the following section we will show, how the similarity of two objects concerning a subset of their meta data can be calculated, before combining this data to an overall similarity result. All of these calculations have to consider the facts described in this section.

III. Distributed Calculation Algorithm

The idea of the implementation of a distributed algorithm is, that every module of the application knows best, how to treat its own data concerning the fact of comparing different objects. Therefore in this section we will describe, how we are capable of distributing the calculation to the different modules of our application using plug-ins. With the help of two examples it is shown how each module developer can implement his specialized calculation algorithm.

In the next section we give some explanations on how these intermediate results can be combined to a single similarity measure.

A. Plug-in Architecture

The tele-TASK portal is designed using a plug-in architecture as we described in [9]. Plug-ins are a known concept for a distributed application with few connections between the different modules [10]. It results in an inversion of control and a loose coupling between the modules of the application.

A plug-in consists of two parts, the plug-in interface and the plug-in implementation. For a single plug-in interface it is possible to provide no plug-implementation of a huge number of them. This has to be considered, when writing function, which uses the plug-in functionality.

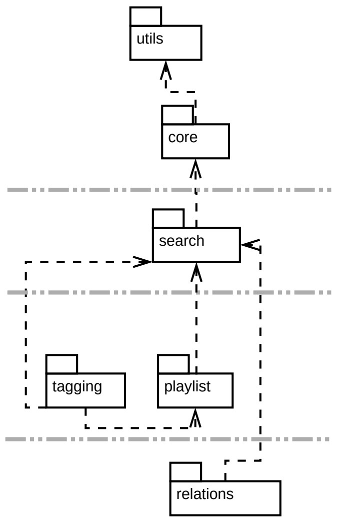

Fig. 2. Excerpt of the structure of the architecture

This architecture allows us to define a global calculation function, which combines the calculation results from the different plug-in implementations provided by different modules.

Therefore the calculation module needs no knowledge about the modules, which will provide the intermediate search results. It only provides the interface, which the modules have to implement and therefore knows, how the classes and functions looks like and how to use them.

```
class RelationScore(PluginBase):
  types = list()
  weight = 3
  description = ''

  def __init__(self, object):
    self.scoreObject = object
    self.type = object.__class__.__name__

  def get_scores(self):
    ''' This function has to be overridden for
        new calculation results '''
    return dict(), list()
```

Listing 1. Plug-in interface for recommendation calculations

The provided plug-in interface for the calculation functions has to describe the parameters of the function and the available variables. It needs at least a function which can be called for generating a list of the compared objects and the result of the comparison. Therefore the function has to know the source object, which should be compared to other objects. A short version of the interface can be found in listing 1.

This version of the plug-in interface provides the *get_scores* function. This function has to be implemented with the calculation algorithm for the specific problem, like the comparison of titles or tags. It returns two representations of the data, a dictionary with the compared object as key and the score as value as well as a list of all compared objects. This is necessary for easier data management.

B. Suggestions for basic calculations

Each calculation function is allowed to generate their own results. To have a better possibility for comparison it is recommended to the plug-in authors to provide results between 0 and 100 points. If they want to provide the special case of downgrading objects (for example because of obvious disconnection between the content), they are allowed to vote between -100 and 100 points. This should not be used in regular cases but for special function like user ratings of similarities.

When comparing two objects o_i and o_k, each can be described as a set of atomic data objects $D(o_i)$ and $D(o_k)$. These atomic data objects are used to calculate the similarity s_{o_i,o_k} between the two objects. A simple calculation algorithm of the similarity of the these objects concerning the data D, which follows the rules is shown in formula 5.

$$s_D(o_i,o_k) = \begin{cases} 0, \text{ if } D(o_i) \cup D(o_k) = \emptyset \wedge o_i \neq o_k \\ 100, \text{ if } D(o_i) = \emptyset \wedge o_i = o_k \\ 100 \cdot \frac{\#(D(o_i) \cap D(o_k))}{\#(D(o_i) \cup D(o_k))}, \text{ else} \end{cases} \quad (5)$$

The data set D can be interpreted as all possible values in the given aspect, for example it can be the set of all words in a title. So $D(o_i) \subseteq D$ is the subset of all values of D which are used in the data of o_i for example all words used in the title of o_i. Because the first two cases are obvious and only interesting if no data exists, we will not write them down any longer, but implicitly use them as fact in every calculation.

This approach does not take into concern, that each data can have a different weight. How to use the number of usages of a special data value is considered, we will describe in the next section.

C. Usage of logarithm

In many cases of comparing data it is a difference, how often the data is used globally. Every term which is used more often will result in a more useless information. For example it would not help to use a tag in every lecture for finding similarities. Every lecture would get the same value added and no new information can be gained. Therefore we define the function $u : D \mapsto \mathbb{N}$, which defines the number of usages of a value $d \in D$ related to the set of all objects.

It is also obvious, that the information that a specific person holds a lecture has more quality if a person holds only a small number of lectures. That is why the overall number of data usage has to be considered and has to decrement the similarity value. If we would use a linear approach, like shown in formula (6), the factor would decrease really fast.

$$s_D(o_i,o_k) = \frac{100}{\#(D(o_i) \cup D(o_k))} \sum_{d \in D(o_i) \cap D(o_k)} \frac{1}{u(d)} \quad (6)$$

Because this could result in lower calculation results even for the calculation of $s_D(o_i,o_i)$, we have to use a stretching factor to increase the results. This stretching factor $s^*_D(o_i,o_k)$ is:

$$s^*_D(o_i,o_k) = \min \left(\frac{1}{\sum\limits_{d \in D(o_i)} \frac{1}{u(d)}}, \frac{1}{\sum\limits_{d \in D(o_k)} \frac{1}{u(d)}} \right) \quad (7)$$

It is more recommendable to use the logarithm like in formular (8) for a smaller decrease when using higher values.

$$s_D(o_i,o_k) = \frac{100}{\#(D(o_i) \cup D(o_k))} \sum_{d \in D(o_i) \cap D(o_k)} \frac{1}{\log(u(d))} \quad (8)$$

Analogically a strechting factor is needed, to ensure that the $s_D(o_i,o_i)$ is 100.

$$s^*_D(o_i,o_k) = \min \left(\frac{1}{\sum\limits_{d \in D(o_i)} \frac{1}{\log(u(d))}}, \frac{1}{\sum\limits_{d \in D(o_k)} \frac{1}{\log(u(d))}} \right) \quad (9)$$

The decision, which basis is used for the logarithm, depends on the highest value of a usage n^* of an element $d \in D$ expected. We think it should not happen, that the decrease factor gets smaller than $\frac{1}{10}$, therefore the basis of the logarithm should be about $\sqrt[10]{n^*}$.

The next two paragraphs will introduce two examples for modules which contributed to the similarity calculation.

D. Example: Comparison of title

The first implementation of a comparison function is the comparison of the title of different lectures. Normally a title consists of a different number of words. So when comparing a title with another title, we have two sets of words $W(o_i) = \{w_{i_1}, \ldots, w_{i_j}\}$ and $W(o_k) = \{w_{k_1}, \ldots, w_{k_l}\}$. We are looking for the set $W(o_i) \cap W(o_k)$, which contains all words, which are used ins both titles.

So in the first step we calculate the value for the similarity of both titles of the objects o_i and o_k in formular 10.

$$s_W(o_i,o_k) = 100 \cdot \frac{\#(W(o_i) \cap W(o_k))}{\#(W(o_i) \cup W(o_k))} \quad (10)$$

Some words are used more often, than other words. This fact should be kept in mind, when calculating a more precise result. Therefore we need the number of usage of the word

w in other lecture titles, which is $u(w)$, as described before. The usage of $u(w)$ as a linear factor would result in a fast decrease of the values, that is why we decided as explained before to use the logarithm instead, as shown in formula 11.

$$s_W(o_i, o_k) = \frac{100 \cdot s^*_W(o_i, o_k)}{\#(W_i \cup W_k)} \cdot \sum_{w \in W(o_i) \cap W(o_k)} \frac{1}{\log(u(w))} \tag{11}$$

With that approach common words have less impact on the calculation. It is also possible to exclude some words completely from the calculation by using stop word lists. Therefore we exclude these words of the title from the list of used words before calculating the similarity of both titles.

E. Example: Comparison of tags

Tagging is a concept know from the web 2.0. Tags are user generated keywords. We implemented this function as one of our user centric functions to allow the users of the portal to actively use the portal for learning.

Tags are terms which are connected to content objects. With the help of these terms, the content of the object is described (see figure 3).

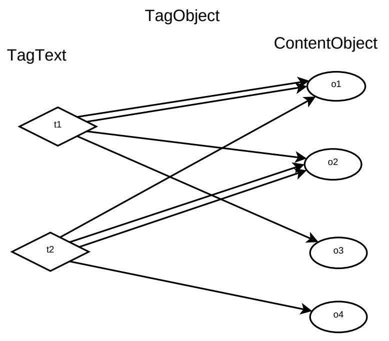

Fig. 3. Example for tags

As can be seen, a tag can be used multiple times for the same object. This happens, when more than one user uses the same tag on the object. This would mean, that more than one user thinks, this is a good tag for the object. So it should of course be treated with higher weight than a single connection between a term and an object.

When calculating the similarity between two objects concerning the tags used for these objects, the weight of each tag has to be considered. Therefore let T be the set of all tags and $c_i(o, t)$ the i-th connection of object o with tag t. Furthermore we define:

$$T_o = \{\text{tags of object } o\} \tag{12}$$

$$w_M(t) = \#\{c_i(o, t) | o \in M\} \tag{13}$$

When comparing two objects o_i and o_k, the easiest approach is to look at the number of tags available in both objects and compare it with the number of tags available in at least one object, as shown in formula 14.

$$s_T(o_i, o_k) = 100 \cdot \frac{\#(T_{o_i} \cap T_{o_k})}{\#(T_{o_i} \cup T_{o_k})} \tag{14}$$

Because of the possibilities of more than one connection between a tag and an object, this approach is a little to easy. The number of usages of a tag should be considered. That is why we need to consider that fact as well.

$$s_T(o_i, o_k) = 100 \cdot \frac{\sum\limits_{t \in (T_{o_i} \cap T_{o_k})} w_{\{o_i, o_k\}}(t)}{\sum\limits_{t \in (T_{o_i} \cup T_{o_k})} w_{\{o_i, o_k\}}(t)} \tag{15}$$

Combined with the idea of decreasing the value of often used tags a calculation algorithm is shown in formula 18. The part $4s^*_T(o_i, o_k)$ is the stretching factor for the calculation, which depends on the structure of p_2 as described in the section before.

$$p_1 = \frac{\sum\limits_{t \in (T_{o_i} \cap T_{o_k})} w_{\{o_i, o_k\}}(t)}{\sum\limits_{t \in (T_{o_i} \cup T_{o_k})} w_{\{o_i, o_k\}}(t)} \tag{16}$$

$$p_2 = \frac{1}{\max\left(1, \sum\limits_{t \in (T_{o_i} \cup T_{o_k}) \setminus (T_{o_i} \cap T_{o_k})} w_{\{o_i, o_k\}}(t)\right)} \tag{17}$$

$$s_T(o_i, o_k) = 100 \cdot p_1 \cdot p_2 \cdot s^*_T(o_i, o_k) \tag{18}$$

The problem is, that this formular will have small values in most cases, because the number of equal tags between two objects is small in most cases. That is why it is better to use a smaller decreasement than linear, for example a logarithmic one (see exchange of the second part of the formula in formula 19).

$$p_2 = \frac{1}{\max\left(1, \sum\limits_{t \in (T_{o_i} \cup T_{o_k}) \setminus (T_{o_i} \cap T_{o_k})} \log(w_{\{o_i, o_k\}}(t))\right)} \tag{19}$$

This results in good similarity results concerning tags for different types of objects. It is important to have a huge basis of tags for better results, but when tags are available, they are a good indicator, so the tag calculation should get a high weight.

F. Saving results for faster calculations

Calculating the similarity of all objects every time an object is requested will result in a big overhead. Therefore it is useful to save intermediate results of a calculation in the database.

In our project we have a database table, which contains most of the text fields, which should be compared. Therefore it is useful, to have an extra table, which contains the similarity measure of the entries in this table. This table can be updated, each time a text is added, changed or deleted.

Such an additional data table would reduce the cost of calculation, because like for most web sites, there are more data reading than data changing tasks.

IV. Combining Calculations with Different Means

In the following section it is shown, how the result of the implementations of every single calculation algorithm for the different aspects of data can be combined to get a single result for each comparison.

For combining the values generated by the different calculation functions, different possibilities of calculating mean values are available. These possibilities will therefore be introduced and compared.

A. Arithmetic Mean

The arithmetic mean is the best known algorithm for calculating average values. It is calculated through summing up all the values and dividing the result through the number of values (see formula (20)).

$$\bar{x}_{\mathrm{a}} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n} \tag{20}$$

The arithmetic mean is useful, if you want to combine a lot of values in the same range. It is also used in combination with negative values and it is the mean, which is used the most often. But it is not always the best choice, so for different kinds of scenarios other mean algorithms should be used.

B. Geometric Mean

The geometric mean can be used for the calculation of the mean growth. If n is the number of values, it is the nth root of the product of all values (see formula (21)).

$$\bar{x}_{\mathrm{g}} = \sqrt[n]{\prod_{i=1}^{n} x_i} = \sqrt[n]{x_1 \cdot x_2 \cdot \ldots \cdot x_n} \tag{21}$$

A big problem with the geometric mean is the usage of the value zero. This value will result in the overall value of zero and will therefore eliminate all other values. That is why we decided, to change the calculation algorithm by setting all zero values to the value one. This does not influence the overall result that much, because the values are stretched to 100 and the difference between zero and one is not that big.

C. Harmonic Mean

The harmonic mean is typically used for calculating mean values of percentage values. Therefore the number of values is divided by the sum of the reciprocal of every value (see formula (22)).

$$\bar{x}_{\mathrm{h}} = \frac{n}{\sum\limits_{i=1}^{n} \frac{1}{x_i}} = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}} \tag{22}$$

The harmonic mean also has a big problem with zero values. The problem is, that $\frac{1}{0}$ is not defined, therefore if the value zero exists, the harmonic mean cannot be calculated. Therefore it is defined, when looking at the limit value towards zero, that the harmonic mean becomes zero, if one value is zero.

That is a problem, because for most of our objects, at least one calculation will have a value of zero, but other calculations will have higher values. This would result in a big number of results of zero, which we do not want to have.

D. Root Mean Square

For calculating the root mean square, the root of the sum of the squares of every value divided by the number of values is calculated (see formula (23)).

$$\bar{x}_{\mathrm{r}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \cdots + x_n^2}{n}} \tag{23}$$

The root mean square is used for calculations where the top values are more important, because they have a higher influence on the overall result.

The root mean square has a problem with negative numbers. When using the square of a value the algebraic sign get lost. Therefore it is not possible to use it for values in the interval of $[-100, 100]$ without modifications.

E. Comparison of the Different Means

Each mean has its own purpose. When comparing the results of the mean calculation the following fact is known:

$$\min(x_1, \ldots, x_n) \leq \bar{x}_{\mathrm{h}} \leq \bar{x}_{\mathrm{g}} \leq \bar{x}_{\mathrm{a}} \leq \max(x_1, \ldots, x_n) \tag{24}$$

We are still evaluating, which mean will be generating the best results. It is not decided which mean will become the final candidate. Because of missing implementations of some possibilities to calculate intermediate results, we have not done a complete evaluation, which mean produces the best results. Therefore we implemented all of these calculation algorithms and are able to switch between them easily.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Case 1	0	0	0	10	10	10	20	40	80	100
Case 2	0	0	0	0	0	0	0	80	90	100
Case 3	0	0	20	20	20	20	20	20	70	80

	Case 1	Case 2	Case 3
arithmetic mean	27	27	27
geometric mean	9,56	3,85	14,3
harmonic mean	2,94	1,42	4,3
root mean square	43,24	49,05	37,02

TABLE I
Example for calculation of different means

When looking at the table I of possible data and considering the problems of the harmonic and the geometric mean, it is more likely that at the end either the arithmetic mean or the geometric mean is used. The effect of the geometric mean of preferring higher values makes it the better candidate for the final calculation algorithm.

F. Adding Weights to the Mean Calculation

It is obvious that different functions will provide a different quality of results. Therefore we use the possibility to weight the result of the calculation function.

For the arithmetic mean, a weighted mean, which could use any real number as weight is defined. But to allow the usage of weight for every type of mean calculation it is better to allow only natural numbers.

When using natural numbers as weight, each algorithm can be extended with this weights easily by duplicating the values for the calculation. For example if we have the values 20 with the weight of 2 and 40 with the weight of 3, we calculate the mean of the values 20, 20, 40, 40, 40 instead.

We decided to set the weight of 3 as default weight for all individual similarity calculations. So it is possible to degrade a function if it has only weak results. To grade a good function higher it is possible to use higher weights.

V. Conclusion and future work

We showed in this paper, how it is possible to combine single calculation algorithm to an overall similarity calculation result and how a single calculation can be done.

The next task is to implement functions for different types of meta data that complement the recommendation algorithm with further data. This meta data can include transcripts from optical character recognition [11], transcripts from audio analysis [12] as well as administrator or user generated meta data. For the user generated meta data several concepts known from social web portals [13] can be exploited for tele-teaching as well.

Afterwards we have to evaluate, which mean calculation algorithm creates good results. Therefore we have to find good relations manually and to check if these relations are found automatically as well. We will also allow the users of our portal to rate the similarity of different objects, to get a better overall opinion, which objects are similar to each other.

As pointed out in [14] it could also be possible, that it is better to use a subset of similarity calculation functions for generating the overall result. We have to evaluate, if this effect will also appear with our data.

We also want to separate these relations in different classes. In tele-teaching context, we have two big groups of relations. On the one hand there are lectures which can be seen as preparation for the actual lecture, like basic knowledge, which is used in the lecture. On the other hand there are lectures which are more detailed and therefore provides more knowledge. Both types are interesting in a tele-teaching portal, because both scenarios could occur.

To get better results we also want to provide better functions to compare words. It is obvious that a simple comparison will miss a lot of similarities, because the same word is often written differently, for example when using singular and plural form of the word. Therefore it is useful to stem the word and compare the stems.

On the other hand, there are the problems of homonyms and synonyms. Therefore the semantic meaning of the word is important. It is possible to enhance the data with semantic data to get better results.

When the similarity algorithm will produce good results it should be included in a recommendation system to provide better user experiences.

References

[1] L. Wang and C. Meinel, "Detecting the Changes of Web Students' Learning Interest," pp. 816–819, 2007.

[2] L. H. P. I. Wang and C. H. P. I. Meinel, "X-Tracking the Changes of Web Navigation Patterns," pp. 772–779.

[3] T. Gruber, "Collective knowledge systems: Where the Social Web meets the Semantic Web," *World Wide Web Internet And Web Information Systems*, vol. 6, pp. 4–13, 2007.

[4] M. G. Noll and C. Meinel, "Design and Anatomy of a Social Web Filtering Service," *Cooperative Internet Computing - Proceedings of the 4th International Conference (CIC 2006)*, no. July, pp. 69–93, 2006.

[5] A. Lau and E. Tsui, "Knowledge-Based Systems," *Knowledge-Based Systems*, vol. 22, pp. 324–325, 2009.

[6] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[7] F. Fouss and M. Saerens, "Evaluating Performance of Recommender Systems: An Experimental Comparison," *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 735–738, Dec. 2008.

[8] V. Schillings and C. Meinel, "tele-TASK - Teleteaching Anywhere Solution Kit," in *Proceedings of ACM SIGUCCS*, Providence, USA, 2002.

[9] M. Siebert, F. Moritz, and C. Meinel, "Enriching E-Learning Meta Data with User Generated Playlists," in *5th International Conference for Internet Technology and Secured Transactions (to appear)*. London, UK: IEEE Computer Society, 2010.

[10] D. Birsan, "On Plug-ins," *Queue*, no. March, 2005.

[11] H. Sack, "Automated Annotation of Synchronized Multimedia Presentations," in *In Workshop on Mastering the Gap: From Information Extraction to Semantic Representation, CEUR Workshop Proceedings*, Berkeley, CA, USA, 2006.

[12] S. Repp and C. Meinel, "Automatic Extraction of Semantic Descriptions from the Lecturer's Speech," in *Proc. 3rd ICSC*, I. Press, Ed., Berkeley, CA, USA, 2009, pp. 513–520.

[13] C. Brooks, S. Bateman, J. Greer, and G. Mccalla, *Lessons Learned using Social and Semantic Web Technologies for E-Learning*. IOS Press, 2009, ch. 14, pp. 260–278.

[14] M. Hirota, S. Yokoyama, N. Fukuta, and H. Ishikawa, "Constrained-based Clustering of Image Search Results Using Photo Metadata and Low-level Image Features," in *Ninth IEEE/ACIS International Conference on Computer and Information Science Article*. Yamagata, Japan: IEEE Computer Society, 2010.