Tag and Topic Recommendation Systems

Ágnes Bogárdi-Mészöly[1,3], András Rövid[2], Hiroshi Ishikawa[3], Shohei Yokoyama[3], Zoltán Vámossy[2]

[1]Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Magyar tudósok körútja 2. QB207, 1117 Budapest, Hungary, agi@aut.bme.hu

[2]John von Neumann Faculty of Informatics, Óbuda University, Bécsi út 96/B, 1034 Budapest, Hungary, rovid.andras@nik.uni-obuda.hu, vamossy.zoltan@nik.uni-obuda.hu

[3]Department of Computer Science, Shizuoka University, 3-5-1 Johoku, Naka-ku, 432-8011 Hamamatsu, Japan, ishikawa@inf.shizuoka.ac.jp, yokoyama@inf.shizuoka.ac.jp

Abstract: The spread of Web 2.0 has caused user-generated content explosion. Users can tag resources in order to describe and organize them. A tag cloud provides rough impression of relative importance of each tag within the overall cloud in order to facilitate browsing among numerous tags and resources. The size of its vocabulary may be huge, moreover, it is incomplete and inconsistent. Thus, the goal of our paper is to establish tag and topic recommendation systems. Firstly, for tag recommendation system novel algorithms have been proposed to refine vocabulary, enhance reference counts, and improve font distribution for enriched visualization. Secondly, for topic recommendation system novel algorithms have been provided to construct a special graph from tags and evaluate reference counts for topic identification. The proposed recommendation systems have been validated and verified on the tag cloud of a real-world thesis portal.

Keywords: social network; tag cloud; tag analysis; vocabulary; reference count; font distribution algorithm; topic recommendation

1 Introduction

With the appearance of Web 2.0 [1] and the spread of social media sites [2] users became from passive spectators to active content generators. Users can interact and collaborate with each other in virtual communities. Nowadays lots of social sites exist for various purposes: collaborative projects (Wikipedia), blogs (Twitter), content communities (YouTube), social networking sites (Facebook), virtual game worlds (World of Warcraft), virtual social worlds (Second Life), etc.

There are numerous books dealing with this topic denoted for introduction and marketing [3] for research [4] etc. Various significant companies have research groups for social computing: Microsoft [5], IBM [6], HP [7], etc.

Usually on these sites users can assign tags to resources in order to describe and organize them. This tagging process [8] establishes associations between tags and resources, which can be applied to navigate to resources by tags, as well as, to tags based on related tags, etc. With tagging a folksonomy (folk (people) + taxis (classification) + nomos (management)) evolves, which is the vocabulary of tags emerged by the community [9]. The size of these vocabularies may be huge, moreover, they are incomplete and inconsistent. Thus, in connection with social tagging several challenges have emerged [4].

The paper is organized as follows. Section 2 covers the background. Section 3 introduces the experimental environment. Section 4 presents the proposed tag recommendation system. Section 5 shows the provided topic recommendation system. Finally, last section reports the conclusions.

2 Background

In this section, definitions related to tagging are discussed. Tags are user-defined informal and personal strings, short descriptions related to resources, keywords associated with resources. They are helpful in browsing and searching. Resources are such identities which can be tagged, such as text, image, audio, video, document, etc. Tagging is the process of assigning existing and new tags to resources. Tag recommendation systems exist to help users in tagging based on own tags or tags of other community members.

There are lots of different kinds of tags: content-based, context-based, attribute, ownership, subjective, organizational, purpose, factual, personal, self-referential, tag bundles, etc. Furthermore, users have various motivations for tagging: future retrieval, contribution and sharing, attract attention, play and competition, self presentation (self referential tags), opinion expression, task organization, social signaling, money, technology ease, etc. [4]. In our tag clouds, tags are content-based, tagging is motivated by contribution and sharing. With content-based tags the actual content of the resources can be identified. By the contribution and sharing as motivation, tags describe resources, and add them to conceptual clusters or refined categories for known and unknown audience.

Tag clouds are visually depicted tags in order to facilitate browsing among numerous tags and resources. It gives rough impression of relative importance of each tag within the overall cloud. In this paper such tag clouds are investigated.

In some situations to answer various questions browsing in tag clouds are more

useful than searching [10]. Search interface is preferred if the needed information is specific. Tag clouds are preferred if the sought information is more general.

For this reason, the visualization of tag clouds is one of the most important and complicated consideration [11]. Tag clouds have two dimensional representations. Tags can be ordered alphabetically, based on semantic similarity or any kind of clusters [12] [13]. Relevant tags can be visually emphasized using such visual properties as shape, color, position, etc. Each tag cloud is visualized in its own unique way. The basis of the used methods is similar, but there are no two tag clouds whose visualization is the same. Numerous font distribution algorithms exist [14] [15]. In our tag clouds all tags are represented simply ordered by alphabetically and visually weighted by letter size.

In social networks power laws occur many times in many contexts [4]. A random variable is distributed according to a power law when its probability density function is given by $x^{-\gamma}$, where $x \geq x_{\min}$ and $\gamma > 1$ [16]. γ is a constant parameter called exponent or scaling parameter, typically in the range of $2 < \gamma < 3$.

In our previous work, algorithms have been yielded to improve tag clouds [17]. In addition, font distribution algorithms have been investigated [18]. Moreover, algorithms have been provided to recommend topics [19]. In this paper, complete tag and topic recommendation systems are established.

3 Experimental Environment

The Faculty of Electrical Engineering and Informatics of the Budapest University of Technology and Economics has a web portal to manage all theses of the faculty for the whole workflow starting from description to review [20].

This portal has been implemented as a three-tier ASP.NET web site. The presentation layer is in HTML and jQuery. In the business logic layer there are C# classes. In the data access layer LINQ and stored procedures are used mixed. The database is in Microsoft SQL Server. The provided algorithms have been implemented in SQL stored procedures, C# classes using LINQ, and MATLAB functions [21].

On this thesis portal tags are assigned to theses to describe and organize them in order to be helpful in browsing and searching. The portal has tag clouds in Hungarian and English languages.

For fast visualization of tag clouds, the tags are stored in a cache. The cache is refreshed when a new tag is created, an existing tag is modified, deleted. The enhanced reference counts (see Section 4.2) are stored in this cache, as well.

In these clouds the tags are classified to four classes according to their reference counts. The number of classes is an arbitrary parameter, it can be varied. The originally applied font distribution algorithm uses weights based on number of classes, and average, maximum, minimum reference counts. The following proposed systems have been validated and verified on these tag clouds.

4 Tag Recommendation System

The aims of the proposed tag recommendation system are to improve the quality of tags, detect the actually popular tags, and enrich the visualization of tag clouds. The provided system has three main steps. In the first step the vocabulary is refined, namely, certain spelling and clerical errors are corrected, in addition, certain tags are contracted. In the second step the reference counts are enhanced. In the third step the font distribution algorithm is improved.

4.1 Vocabulary Refinement

In this section, two algorithms have been proposed to improve the quality of tags. The purpose of the first algorithm is not to correct all spelling and clerical errors in each tag, but to contract such tags which are the same only with different writing (the algorithm retains not in all cases the grammatically correct version). The aim of the second algorithm is to contract such simple tags of each thesis to a compound tag in which case the created compound tag is used for another thesis. Table 1 summarizes the meaning of the indices used in these algorithms. In addition, the notations of these algorithms can be seen in Table 2.

Table 1
Meaning of indices used in the algorithms of the tag recommendation system

Notation	Description
u	uninvestigated
s	simple
c	compound
a	related to assignment
i	isomorphic
e	elected from isomorphic
$\min_x$	minimum value of a given property in set x
$\max_x$	maximum value of a given property in set x

For all languages function *gc* is case-insensitive. For English and Hungarian languages function *gc* must remove all spaces and dashes from tags, because according to grammatical rules (for example number of vocals, word-class, etc.)

compound words can be written in different ways (into one, or with dashes, or with spaces). If short-length tags (which contain only some letters) exist, then dashes and spaces must be retained. Related to grammatical rules function *gc* can be determined for other languages, as well.

Table 2
Notations of algorithms of the tag recommendation system

Notation	Description
τ	set of tags
π	set of papers
α	set of assignments between tags and papers
σ	set of strings extracted from tags related to grammatical rules
t	a tag
p	a paper
l	length of a tag t
c	reference count of a tag t
s	reference count sum of a tag t
d	date and time when a tag t is created
$t_1 \mapsto t_2$	tags t_1 and t_2 are isomorphic
$t \rightarrow p$	tag t is assigned to paper p
t%	starts with tag t: "t string" (string with at least 1 char)
%t	ends with tag t: "string t" (string with at least 1 char)
%t%	contains tag t: "t" or "string1 t string2" (strings with at least 1 char)
$t_1 + t_2$	concatenates tag t_1 and tag t_2: "t_1 t_2"
gc: $\tau \rightarrow \sigma$	function related to grammatical rules in order to compare
gr: $\tau \rightarrow \sigma$	function related to rules to retrieve correct version

For all languages function *gr* is case-insensitive, as well. It determines related to some simple grammatical rules (for example in case of Hungarian language the number of vocals is bigger than 7) how a given tag may be written correctly.

It is worth to realize in these functions only simple grammatical rules, because the main aim of these algorithms is not spell correction based on edit distance, only vocabulary refinement by contract tags.

4.1.1 Algorithm to Correct Spelling and Clerical Errors

In tag clouds, there exist many misleading tags due to spelling and clerical errors. Spelling errors occur due to grammatical mistakes, while typing mistakes are considered as clerical errors.

In case of clerical errors it is worth to permit only one character difference, because in shorter tags two or more character difference may yield two correct but different tags. Moreover, it can be automated with minimum failure only for longer tags and for tags which do not contain numbers in specific technical environment. See for example tags "motor" and "rotor", or "IPv4" and "IPv6".

Therefore, it is more accurate to use the proposed algorithm only for collecting the potential clerical errors, and correct them under human control.

Therefore, isomorphic function can be defined related to aforementioned comments similar to function *gc*. In addition, clerical errors can be forbidden or can be permitted only for one character. The following algorithm has been provided to correct spelling and clerical errors.

Definition 1. The algorithm for *Correcting Spelling and Clerical Errors in Tag Cloud* is defined by Algorithm 1, and the associated notations are in Tables 1 and 2.

Algorithm 1. Pseudo code of the Correcting Spelling and Clerical Errors in Tag Cloud algorithm

1: $\tau_u = \tau$
2: **while** $\tau_u \neq \emptyset$ **do**
3: for $t \in \tau_u$, where $c = c_{\max_u}$
4: $\tau_i = \{ \forall t_i \in \tau$, where $t_i \mapsto t \}$
5: **if** $|\tau_i| > 1$ **then**
6: **if** $|\{ \forall t_i \in \tau_i$, where $c_i = c_{\max_i} \}| = 1$ **then**
7: $t_e = t_i$, where $t_i \in \tau_i$ and $c_i = c_{\max_i}$
8: **else**
9: **if** $\exists t_i \in \tau_i$, where $t_i = gr(t)$ **then**
10: $t_e = t_i$, where $t_i \in \tau_i$ and $t_i = gr(t)$
11: **else**
12: $t_e = t_i$, where $t_i \in \tau_i$ and $d_i = d_{\min_i}$
13: $c_e = \sum_{\forall t_i \in \tau_i} c_i$
14: $\tau_i = \tau_i \setminus \{ t_e \}$
15: **for all** $t_i \in \tau_i$ **do**
16: $\pi_a = \{ \forall p \in \pi$, where $\exists t_i \rightarrow p \}$
17: **for all** $p \in \pi_a$ **do**
18: $\alpha = \alpha \setminus \{ \mathrm{t_i} \rightarrow \mathrm{p} \}$, $\alpha = \alpha \cup \{ \mathrm{t_e} \rightarrow \mathrm{p} \}$
19: $\tau = \tau \setminus \tau_\mathrm{i}$, $\tau_u = \tau_u \setminus \{ \mathrm{t_e} \}$
20: $\tau_u = \tau_u \setminus \tau_\mathrm{i}$

For a given tag, isomorphic tags are looked for. It is worth to start from the maximum reference counts of tags, because usually there are only some spelling errors compared to the number of the correct tags. So the number of iterations in the loop may decrease, namely, the computational time may become less. If there is at least one more isomorphic tag besides the given tag, firstly, try deciding which one to elect based on the reference counts, secondly, based on grammatical rules, finally, based on the identifiers. (Using automatically generated identifiers, where less identifier means earlier created tag.) The algorithm retains not in all

cases the correct version of the tags, but the tags which are the same with different writing are contracted. In Step 13 of the algorithm, the reference count of this elected tag is updated. In Steps 15 to 18 assignments of the other isomorphic tags are modified for this elected tag. In Step 19 the other isomorphic tags are deleted.

On existing tag clouds this first algorithm has to be executed once in a maintenance phase. After that or for newly created tag clouds, it can be applied in two different ways: the whole algorithm is executed periodically only in maintenance phases, or only the part of Steps 4 to 19 is run when a new tag is created, an existing tag is modified, deleted. Since sets τ_i created in Step 4 of the algorithm are disjunct, it is enough to investigate the set of assignments only once after creating all sets τ_i.

4.1.2 Algorithm to Contract Tags

Some simple tags can be contracted to compound tags. For example, see the following three tags: social, network, social network. If "social network" tag exists, and tags "social" and "network" are assigned to the same papers, then for that papers these tags can be contracted to tag "social network". The contraction of tags can be an important step in improving tag clouds, because reference counts of tags can considerably be changed (see reference counts in detail in Definition 3). A novel algorithm has been provided to contract tags as follows.

Definition 2. The algorithm for *Contracting Tags in Tag Cloud* is defined by Algorithm 2, and the associated notations are in Tables 1 and 2.

Algorithm 2. Pseudo code of Contracting Tags in Tag Cloud algorithm

1: $\tau_u = \tau$
2: **while** $\tau_u \neq \varnothing$ **do**
3: for $t_{s_1} \in \tau_u$, where $l_{s_1} = l_{\min_u}$
4: $\tau_c = \{ \forall t_c \in \tau$, where $gc(t_c) = gc(t_{s_1}\%)$ or $gc(t_c) = gc(\%t_{s_1}) \}$
5: **for all** $t_c \in \tau_c$ **do**
6: **if** $\exists t_{s_2} \in \tau$, where $gc(t_c) = gc(t_{s_1} + t_{s_2})$ or $gc(t_c) = gc(t_{s_2} + t_{s_1})$ **then**
7: $\pi_a = \{ \forall p \in \pi$, where $\exists\ t_{s_1} \to p$ and $\exists\ t_{s_2} \to p \}$
8: **for all** $p \in \pi_a$ **do**
9: $c_{s_1} = c_{s_1} - 1,\ c_{s_2} = c_{s_2} - 1$
10: **if** $c_{s_1} = 0$ **then** $\tau = \tau \setminus \{ t_{s_1} \}$
11: **if** $c_{s_2} = 0$ **then** $\tau = \tau \setminus \{ t_{s_2} \}$
12: $c_c = c_c + 1$
13: $\alpha = \alpha \setminus \{ t_{s_1} \to p,\ t_{s_2} \to p \},\ \alpha = \alpha \cup \{ t_c \to p \}$
14: $\tau_u = \tau_u \setminus \{ t_{s_1}, t_{s_2} \}$
15: **else**
16: $\tau_u = \tau_u \setminus \{ t_{s_1} \}$

For a given tag, such compound tags are looked for, which start or end with the tag, and the other parts of the compound tags are existing tags in the cloud. It is worth to start from minimum length of tags, because the chance to contract shorter tags to a longer tag are higher, so the number of iterations in the loop may be less. In Steps 8 to 13 of the algorithm, the two simple tags are contracted to the compound tag, accordingly reference counts and assignments are updated. In Steps 10 to 11 tags with zero reference count are deleted.

This second algorithm has to be executed similarly like the first one. Recall that on existing tag clouds it has to be executed once in a maintenance phase. After that or for newly created tag clouds, it can be applied in two different ways: the whole algorithm is executed periodically only in maintenance phases, or only the part of Steps 4 to 13 is run when a new tag is created, an existing tag is modified, deleted. Since the algorithm investigates tags in ascending order by length and the set of tags may be decreased during the execution, the algorithm cannot be parallelized.

4.1.3 Experimental Results

The proposed algorithms do not influence considerably the number of tags in classes. Thus, for these intermediate steps only the change in the total number of tags is presented in Table 3.

Table 3
Change in total number

Name of used algorithm	Name of total number	Hungarian	English
Original	Number of tags	5354	4199
Correct spelling and clerical errors	Number of tags Number of decrease	5304 50	4169 30
Contract tags	Number of tags Number of decrease Number of contractions	5269 35 142	4152 17 66
Improve reference counts	Number of tags Number of improvements	5269 923	4152 628

The number of decrease and contractions is not too numerous in numbers, but using Algorithm 1 such annoying spelling errors can be corrected as “Java EE” and “JavaEE", or “model-driven development” and “model driven development”, in addition, such annoying clerical errors as “optimalization” and “optimalisation”, or “activity of daily living” and “activity od daily living”.

Moreover using Algorithm 2 such coherent tags can be contracted as “ASP.NET” and “MVC” to “ASP.NET MVC”, or “IBM” and “WebSphere” to “IBM WebSphere”, or “social” and “network” to “social network”.

4.2 Reference Count Enhancement

In the second step the reference counts are enhanced in order to reflect much more effectively the reality.

4.2.1 Algorithm to Enhance Reference Counts

In tag clouds reference counts of tags are one of the most important consideration, because there is a huge number of tags in the clouds, and popular tags can be emphasized according to their reference counts.

Definition 3. The *reference count* of tag t is $|\{ \forall p \in \pi$, where $\exists\ t \rightarrow p \}|$.

The reference count of a given tag can be improved to the sum of reference counts of all tags which contain the given tag. For example, see the following tags: mobile, mobile application development, mobile communication, mobile network, mobile platform, mobile robot, mobile technology, etc. All mentioned tags contain the word "mobile". Thus, the reference count of tag "mobile" can be improved to the sum of reference counts of all mentioned tags. The following algorithm has been proposed to improve reference counts.

Definition 4. The algorithm for *Improving Reference Counts in Tag Cloud* is defined by Algorithm 3, and the associated notations are in Tables 1 and 2.

Algorithm 3. Pseudo code of Improving Reference Counts in Tag Cloud algorithm

1: **for all** $t \in \tau$ **do**
2: $\tau_c = \{ \forall t_c \in \tau$, where $gc(t_c) = gc(\%t\%) \}$
3: $s = \sum_{\forall t_c \in \tau_c} c_c$

It is worth storing the improved reference counts of tags (in database or cache). On existing tag clouds this third algorithm has to be executed once in a maintenance phase. After that or for newly created tag clouds, it has to be run when a new tag is created, an existing tag is modified, deleted only on tags related to the created, modified, deleted tag.

4.2.2 Experimental Results

The algorithm to improve reference count does not modify the number of tags, however it influences considerably the reference counts. In Table 3 the total number of improvements related to reference counts can be seen. The change in the reference counts can be noticed for the whole interval in Fig. 1, and from reference count 10 in Fig. 2. It can be seen on the figures that with the improvement the count of tags referenced only 1 is substantially decreased, in addition, for larger counts there are more bins (with non zero count), moreover, the counts are considerably increased.

4.3 Enriched Visualization

In the third step the font distribution algorithm is improved to efficiently divide tags to classes and easily identify popular tags.

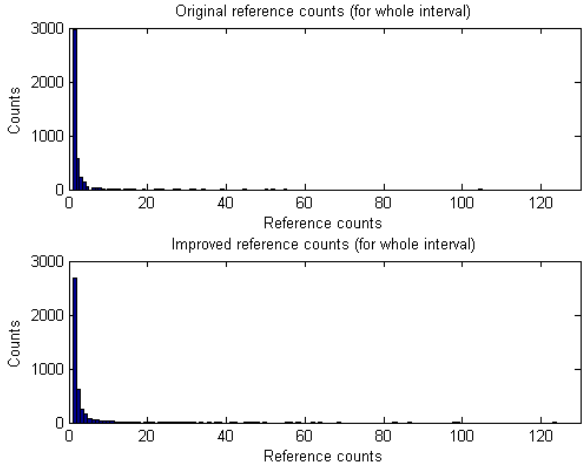

Figure 1
Histogram of original and improved reference counts for whole interval

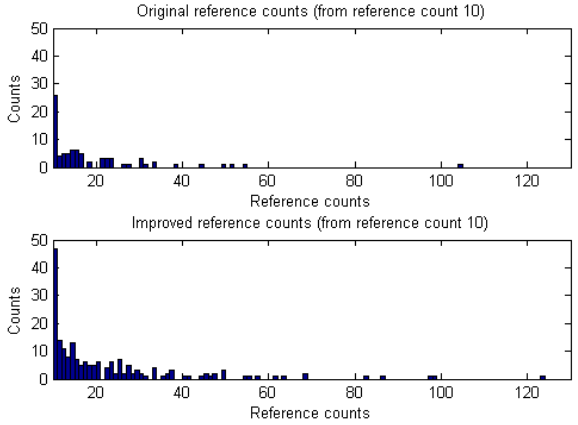

Figure 2
Histogram of original and improved reference counts starting from reference count 10

4.3.1 Distribution of Reference Counts

It seems from the histograms that the improved reference counts obey a power law. The investigation has been performed with MATLAB functions provided by [21]. Firstly, a power law distribution has been fitted to the data set of reference counts, and its parameters have been estimated. The maximum likelihood estimate of the scaling exponent is $\gamma = 2.12$, and the estimate of the lower bound of the power law behaviour is $x_{\min} = 1$. The uncertainties in the estimated parameters are 0.0185 for γ and 0 for $x_{\min}$.

Then, the test whether the power law is a plausible fit to the empirical data set of reference counts has been performed graphically and numerically, as well. The result of the graphical method shows that the empirical data follows a straight line but departs from it at the end. This means that the empirical data has a longer tail than the estimated theoretical power law distribution. The test can be performed numerically with the help of a hypothesis test using Kolmogorov-Smirnov statistic. Since the calculated p-value 0.136 is greater than 0.1 (in addition, $x_{\min} = 1$ and the number of reference counts is more than 4000), the power law is a plausible hypothesis for the data.

4.3.2 Font Distribution Algorithm

In tag clouds the tags are classified according to their reference counts. The number of classes is an arbitrary parameter. In our clouds, tags are classified to four classes, and the improved reference count sums *s* are used instead of reference counts *r*. The result of different distribution algorithms can be depicted in Table 4.

Table 4
Number of tags belonging to classes

Font distribution algorithm	Class 1	Class 2	Class 3	Class 4
Linear for *r*	4151	8	0	1
Linear for *s*	4124	27	6	3
Logarithmic for *s*	3542	471	121	26
Power law and percentage based for *s*	3935 (95%)	142 (3%)	63 (1.5%)	20 (0.5%)

The linear distribution algorithm simply divides linearly the whole range (from minimum count to maximum) count by the number of classes. The logarithmic algorithm divides logarithmically equal intervals.

Since the improved reference counts obey a power law, a power law and percentage based approach can led to correct visual impression. The proposed distribution algorithm divides the area of the given power law function (Eq. 1)

according to arbitrary percentages taking into consideration the bounds of same reference counts.

$$\frac{\max(s)^{-\gamma+1} - \min(s)^{-\gamma+1}}{-\gamma+1} \tag{1}$$

4.3.3 Experimental Results

In Fig. 3 a part of the original tag cloud is illustrated. In Fig. 4 the same part of the resulted tag cloud is depicted.

Figure 3

A part of the original tag cloud

Figure 4

A part of the resulted tag cloud

It can be seen that in the original cloud almost all tags are in unite, however in the resulted cloud many popular tags are emphasized, thus, the resulted tag cloud is much more perspicuous, the popular tags can be indentified easily.

5 Topic Recommendation System

The aim of the proposed topic recommendation system is to detect the most popular topics from the huge tag cloud.

The provided system has three main steps. In the first step a special graph is constructed from tags. In the second step the reference count of each node is evaluated in order to identify topics. In the third step the improved font distribution algorithm of the proposed tag recommendation system is applied and a visualization is introduced. Table 5 summarizes the notations of algorithms.

Table 5
Notations of algorithms of the topic recommendation system

Notation	Description
τ	set of tags
τ_u	set of uninvestigated tags
t	a tag
$t.dn$	display name property of tag t
$t.rc$	reference count property of tag t
$t.wc$	word count property of tag t
$dn.w_i$	i^{th} word of a display name
$dn.w_{i \to j}$	concatenated word starting from i^{th} word and ending at j^{th} word of a display name dn
ν	set of nodes
ε	set of edges
v	a node
v_s	source node of a directed edge
v_t	target node of a directed edge
$e_{i,j}$	an edge from node v_i to node v_j
$v.id$	identifier property of node v
$v.dn$	display name property of node v
$v.wt$	weight property of node v
$v.rc$	reference count property of node v

5.1 Graph Construction

In the first step a special graph is constructed from tags.

5.1.1 Algorithm to Construct Graph from Tags

A directed, weighted graph $G = (\nu, \varepsilon)$ is defined as a set of nodes ν and edges ε. The ij^{th} entry of the adjacency matrix A is 1 if there is a directed edge from node i to node j, and 0 if such edge does not exist. Only the nodes have nonnegative weights, the edges are not weighted. This graph is constructed using Algorithm 4.

Definition 5. The algorithm for *Constructing Graph from Tags* is defined by Algorithm 4, and the associated notations are in Table 5.

Algorithm 4. Pseudo code of Constructing Graph from Tags algorithm

$\tau_u = \tau$
$\upsilon = \varnothing,\ \varepsilon = \varnothing$
$k = 1$
while $\tau_u \neq \varnothing$ **do**
    for $t \in \tau_u$, where $t.wc = \min(t_u.wc),\ \forall t_u \in \tau_u$
    $k = AddNode(k, t.dn, t.wc, t.rc) + 1$

function $AddNode(k, dn, wc, rc) : i$
    if $wc = 1$ **then**
        $v.id = k, v.dn = dn, v.wt = rc, \upsilon = \upsilon \cup \{ v \}$
        return $v.id$
    else
        $i_1 = 0$
        for $i = 1 \rightarrow wc - 1$ **do**
            if $i_1 = 0$ **then**
                if $\exists v_1 \in \upsilon$, where $v_1.dn = dn.w_{1 \rightarrow wc-i}$ **then**
                    $i_1 = v_1.id,$ where $v_1.dn = dn.w_{1 \rightarrow wc-i}$
                    if $\exists v_2 \in \upsilon,$ where $v_2.dn = dn.w_{wc-i+1 \rightarrow wc}$ **then**
                        $i_2 = v_2.id,$ where $v_2.dn = dn.w_{wc-i+1 \rightarrow wc}$
                    else
                        $i_2 = AddNode(k, dn.w_{wc-i+1 \rightarrow wc}, i, 0)$
                        $k = i_2 + 1$
                else if $\exists v_1 \in \upsilon$, where $v_1.dn = dn.w_{i+1 \rightarrow wc}$ **then**
                    $i_1 = v_1.id,$ where $v_1.dn = dn.w_{i+1 \rightarrow wc}$
                    if $\exists v_2 \in \upsilon$, where $v_2.dn = dn.w_{1 \rightarrow i}$ **then**
                        $i_2 = v_2.id,$ where $v_2.dn = dn.w_{1 \rightarrow i}$
                    else
                        $i_2 = AddNode(k, dn.w_{1 \rightarrow i}, i, 0)$
                        $k = i_2 + 1$
        if $i_1 = 0$ **then**
            $v_1.id = k, v_1.dn = dn.w_1, v_1.wt = 0, \upsilon = \upsilon \cup \{ v_1 \}$
            $i_1 = v_1.id$
            $i_2 = AddNode(k + 1, dn.w_{2 \rightarrow wc}, wc - 1, 0)$
            $k = i_2 + 1$
        $v.id = k, v.dn = dn, v.wt = rc, \upsilon = \upsilon \cup \{ v \}$
        $\varepsilon = \varepsilon \cup \{ e_{i_1, v.id}, e_{i_2, v.id} \}$
        return $v.id$

This is a recursive algorithm. The tags are investigated ordered ascending by their word count. If the word count is one, then create a new node for it, and return with their identifier in Steps 9-11.

If the display name of the tag contains more words, then search for a matching node investigating the display name from its begin (Steps 16-17) and its end (Steps 23-24) starting from length of word count minus one until one. If a matching node exists, then investigate whether the other part of the display name exists as a node (Steps 18-19, 25-26).

If there is no matching for the other part, then create appropriate nodes for it by recursive calls (Steps 21-22, 28-29). If there is no matching node investigating the display name from its begin and its end starting from length of word count minus one until one (Step 30), then create a new node for the first word (Steps 31-32), and create appropriate nodes for the other part of the display name by recursive calls (Steps 33-34).

After the shorter parts of the display name exist, then create a new node for the whole display name (Step 35). Moreover, create two directed edges from two nodes with shorter parts to the node of the whole display name (Step 36).

5.1.2 Experimental Results

An example part of the constructed tag graph can be seen in Fig. 5. The tags are the following: data, text mining, data mining, data mining competition.

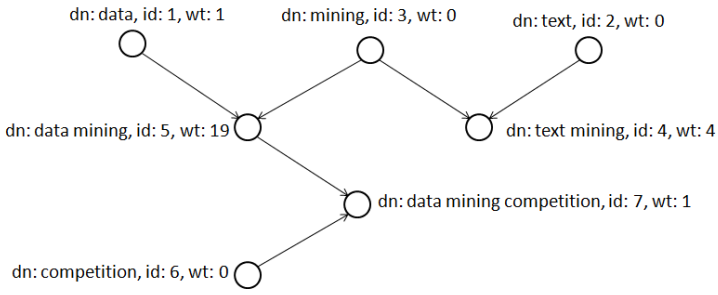

Figure 5
An example part of the tag graph

The reference counts of such nodes which do not correspond to real tags are zero (Steps 21, 28, 33 of Algorithm 4). The in-degree of nodes can be zero or two: zero if the display name of a given node contains only one word, in addition, two if it is a compound word.

The histogram of out-degree of nodes is depicted in Fig. 6. There are numerous nodes whose out-degree is zero, namely, they are not building items of nodes with longer display name. However, there are some nodes which are frequently used building items. The weight and out-degree of nodes influences the reference counts of nodes, namely, the recommended topics.

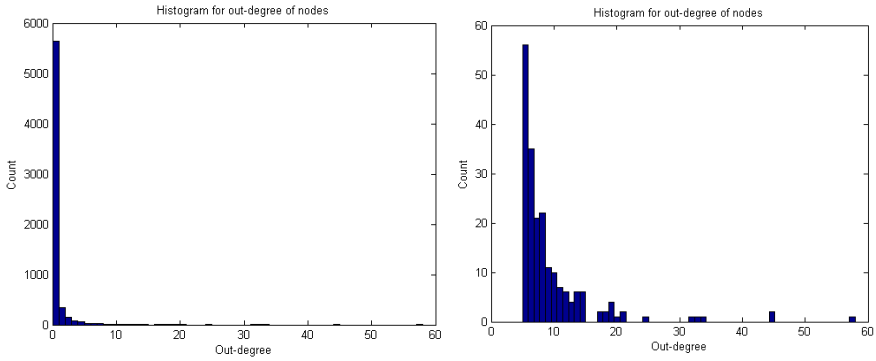

Figure 6
Histogram for out-degree of nodes (left: all, right: only from out-degree 5)

5.2 Topic Identification

In the second step the reference count of each node is evaluated in order to identify topics.

5.2.1 Algorithm to Recommend Topics from Graph

In the second step the reference counts of nodes are calculated by Algorithm 5. The reference count of a node is calculated as the sum of weights of such nodes which can be reached from the given node via directed edges. The proposed special tag graph with the calculated reference counts can be used for topic recommendation. The recommended topics are the nodes with the most reference counts. The limit in reference counts or the maximum number of topics can be chosen arbitrary.

Definition 6. The algorithm for *Calculating Reference Counts of Nodes* is defined by Algorithm 5, and the associated notations are in Table 5.

Algorithm 5. Pseudo code of *Calculating Reference Counts of Nodes* algorithm

1: **for all** $v \in \upsilon$ **do**
2: $v.rc = v.wt + Count(v,0)$
3: **function** $Count(v_s, rc) : rc$
4: $\upsilon_t = \{ v_t, \text{where } \exists e_{s,t} \}$
5: **if** $|\upsilon_t| \geq 1$ **then**
6: $rc = rc + \sum_{\forall v_t \in \upsilon_t} v_t.wt$
7: **for all** $v_t \in \upsilon_t$ **do**
8: $rc = Count(v_t, rc)$
9: **return** rc

5.2.2 Experimental Results

The construction of the tag graph is described in Table 6.

Table 6
Construction of tag graph

	Count
Tags	4152
Nodes Edges	6408 5404
Nodes whose in-degree = 0 Nodes whose in-degree = 2	3706 2702
$rc \geq 10$ $wt \neq rc$ ($rc \geq 10$) $wt = 0$ ($rc \geq 10$)	274 236 70

The nodes whose reference count is greater or equal to 10 are identified as topics. The reference counts and the weights of topics are different numbers in more than 85 percentages, thus, the proposed reference counts of nodes are an important improvement of the reference count of tags. More than 25 percentages of nodes are such topics, which are not existing tags, hence, the provided tag graph is a significant enhancement of the original tag cloud.

5.3 Visualization of Recommended Topics

In the third step the improved font distribution algorithm of the proposed tag recommendation system (see Section 4.3) is applied and a visualization is introduced.

The resulted distribution of nodes among classes is summarized in Table 7. The resulted topic cloud are depicted in Figs. 7 and 8.

Table 7
Distribution of nodes among classes

Class	**Percentage**	**Count of nodes**
Class 1	95	255
Class 2	3	8
Class 3	1.5	4
Class 4	0.5	1

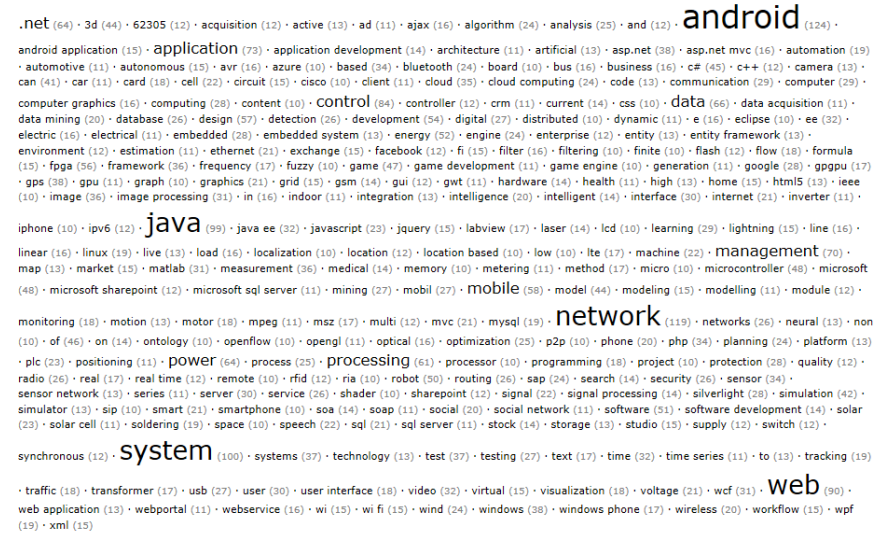

Figure 7
Resulted topic cloud

Figure 8
Resulted topic cloud with tooltip about tags belonging to node of "processing"

The identified topics are visualized as a tag cloud alphabetically ordered and visually weighted by letter size. The calculated reference counts are in brackets after the display name of topics. The tags with their reference counts, which fill parts of the given node, are shown on tooltips. In Fig. 8 see for example the tooltip for node 'processing', who is not in the original tag cloud, who is not an itself existing tag, but identified as an important topic.

Conclusions

The visualization of huge tag clouds is one of the most important and complicated consideration. In our thesis portal tag clouds have a very important role to facilitate browsing and searching among numerous tags and theses. In this paper complete tag and topic recommendation systems have been provided.

The tag recommendation system has three steps. In the first step the vocabulary has been refined. The aim of the algorithm correcting spelling and clerical errors is not to correct all spelling and clerical errors in each tag, but to contract such tags which are the same only with different writing (the algorithm retains not in all cases the grammatically correct version). The purpose of the algorithm contracting tags is to contract such simple tags of each thesis to a compound tag in which case this compound tag is used for another thesis. In the second step the reference counts have been enhanced in order to reflect much more effectively the reality. In the third step the font distribution algorithm has been improved to efficiently divide tags to classes and easily identify popular tags. Using the established tag recommendation system the quality of tags, the detection of the actually popular tags, and the visualization of tag clouds can be improved.

The topic recommendation system has three steps. In the first step a special graph has been constructed from tags. In the second step the reference count of each node has been evaluated in order to identify topics. In the third step the improved font distribution algorithm of the proposed tag recommendation system has been applied and a visualization has been introduced. The resulted topic cloud is a significant enhancement of the original tag cloud, since lots of such topics are identified, which are not existing tags in the original tag cloud, in addition, the popularity of topics is evaluated more properly, furthermore, popular topics can be detected easily.

The proposed systems have been implemented in C# classes using LINQ, SQL stored procedures, and MATLAB functions. They have been validated and verified on tag clouds of a real-world thesis portal.

Acknowledgement

First of all we wish to thank SUZUKI Foundation for sponsoring this research. This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TÁMOP-4.2.2.C-11/1/KONV-2012-0013).

References

[1] T. O'Reilly, What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software, International Journal of Digital Economics, No. 65, pp. 17-37, March 2007

[2] A. M. Kaplan, M. Haenlein, Users of the World, Unite! The Challenges and Opportunities of Social Media, Business Horizons, Vol. 53(1), pp. 59-68, 2010

[3] J. Zimmerman, D. Sahlin, Social Media Marketing All-in-One for Dummies, Wiley Publishing, ISBN 978-0-470-58468-2, 2010

[4] C. C. Aggarwal, Social Network Data Analytics, 1st Edition, Springer, ISBN 978-1-4419-8461-6, 2011

[5] Microsoft Research - Social Computing Group, http://research.microsoft.com/scg

[6] IBM Research - Social Computing Group, http://www.research.ibm.com/haifa/dept/imt/ct st.shtml

[7] HP Labs - Social Computing Lab, http://www.hpl.hp.com/research/scl

[8] S. A. Golder, B. A. Huberman, The Structure of Collaborative Tagging Systems, 2005

[9] D. H. Pink, Folksonomy, New York Times, December 11, 2005

[10] J. Sinclair, M. Cardew-Hall, The Folksonomy Tag Cloud: When is It Useful?, Journal of Information Science, Vol. 34(1), pp. 15-29, 2008

[11] S. Lohmann, J. Ziegler, L. Tetzlaff, Comparison of Tag Cloud Layouts: Task-Related Performance and Visual Exploration, INTERACT, Part I, LNCS 5726, pp. 392-404, 2009

[12] Y. Hassan-Montero, V. Herrero-Solana, Improving Tag-Clouds as Visual Information Retrieval Interfaces, International Conference on Multidisciplinary Information Sciences and Technologies, Merida, Spain, 2006

[13] J. Salonen, Self-Organizing Map Based Tag Clouds, OPAALS Conference, 2007

[14] kentbye's blog, Tag Cloud Font Distribution Algorithm, http://www.echochamberproject.com/node/247, June 24, 2005

[15] K. Hoffman, In Search of ... The Perfect Tag Cloud, http://files.blog-city.com/files/J05/88284/b/insearchofperfecttagcloud.pdf

[16] A. Clauset, C.R. Shalizi, and M. E. J. Newman, Power-Law Distributions in Empirical Data, SIAM Review, Vol. 51(4), pp. 661-703, 2009

[17] Á. Bogárdi-Mészöly, A. Rövid, H. Ishikawa, Algorithms to Improve Tag Clouds, 5th International Conference on Emerging Trends in Engineering & Technology, Himeji, Japan, pp. 303-308, ISBN 978-0-7695-4884-5, 2012

[18] Á. Bogárdi-Mészöly, A. Rövid, H. Ishikawa, An Improved Font Distribution Algorithm for Tag Clouds, 6th International Conference on Soft Computing and Intelligent Systems, Kobe, Japan, pp. 2264-2267, ISSN 1880-3741, 2012

[19] Á. Bogárdi-Mészöly, A. Rövid, H. Ishikawa, Topic Recommendation from Tag Clouds, 2nd International Workshop on Networking, Computing, Systems, and Software, Okinawa, Japan, 2012, pp. 25-29, Bulletin of Networking, Computing, Systems, and Software, ISSN 2186–5140, Vol. 2(1), January 2013

[20] Thesis Portal, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, http://diplomaterv.vik.bme.hu

[21] A. Clauset, C.R. Shalizi, and M. E. J. Newman, Implementation for Article of Power-Law Distributions in Empirical Data, http://tuvalu.santafe.edu/~aaronc/powerlaws/