Chapter

Tagging and Tag Recommendation

Fabiano M. Belém, Jussara M. Almeida and Marcos A. Gonçalves

Abstract

Tagging has emerged as one of the best ways of associating metadata with objects (e.g., videos, texts) in Web 2.0 applications. Consisting of freely chosen keywords assigned to objects by users, tags represent a simpler, cheaper, and a more natural way of organizing content than a fixed taxonomy with a controlled vocabulary. Moreover, recent studies have demonstrated that among other textual features such as title, description, and user comments, tags are the most effective to support information retrieval (IR) services such as search, automatic classification, and content recommendation. In this context, tag recommendation services aim at assisting users in the tagging process, allowing users to select some of the recommended tags or to come up with new ones. Besides improving user experience, tag recommendation services potentially improve the quality of the generated tags, benefiting IR services that rely on tags as data sources. Besides the obvious benefit of improving the description of the objects, tag recommendation can be directly applied in IR services such as search and query expansion. In this chapter, we will provide the main concepts related to tagging systems, as well as an overview of tag recommendation techniques, dividing them into two stages of the tag recommendation process: (1) the candidate tag extraction and (2) the candidate tag ranking.

Keywords: tagging, folksonomies, Web 2.0, tag recommendation, keyword extraction, tag ranking

1. Introduction

Web 2.0 applications are characterized by the central role played by users in the creation and sharing of their own content. Tagging has become a common feature available in these applications, consisting in associating freely created tags (keywords) to objects (e.g., videos, images, texts). In comparison with a fixed taxonomy, tags are simpler, cheaper, and a more natural way of organizing content. In fact, taxonomies with a controlled vocabulary do not suit the increasing and evolving Web 2.0 environment [1].

Moreover, various studies have demonstrated that, among other textual features such as title, description, and user comments, tags are the most effective to support information retrieval (IR) services such as search [2], automatic classification [3], and content recommendation [4].

The tagging process can benefit a lot from a tag recommendation service. This type of service supports users in the selection of some of the recommended tags or in the creation of new ones. With that in mind, tag recommendation benefits are not limited

to the improvement of the user experience: there is a high potential of improving the quality of the generated tags by, for example, reducing the amount of misspellings and nondescriptive keywords. Thus, the quality of the IR services that rely on tags as data sources can be indirectly improved by tag recommendation. Other examples of the benefits that tag recommendation can bring to IR services include the direct application of the recommended tags in search [5] and on query expansion [6]. In search, the recommended tags can be exploited to measure the similarity between queries and documents, improving the quality of the retrieved documents. Query expansion, in turn, aims at suggesting more specific and unambiguous queries to the user, which also allows the achievement of better search results. Further examples include researcher profile summarization [7] and search result summarization [8].

Tag recommendation brings specific challenges that other kinds of recommendation services do not: in the tag domain, we are interested not only in matching the interests of the target user but also in describing, summarizing, and organizing Web content. Thus, the design of tag recommenders demands specific solutions which greatly differ from methods proposed for item recommendation tasks in general. For instance, text mining, knowledge extraction, and semantics play a substantial role in the tag domain. In sum, the recommendation effectiveness affects not only user satisfaction but also the performance of various IR services that rely on tags as data source.

The goal of this chapter is to present the concepts of tagging systems and to provide an overview of tag recommendation techniques, explaining the two main steps of these methods: the candidate tag generation and the candidate tag ranking.

The rest of this chapter is organized as follows. In Section 2, we define tags, objects, folksonomies, and other basic concepts related to tagging systems. In Section 3, we state the tag recommendation problem, while we explain the main tag candidate extraction and ranking techniques in Sections 4 and 5, respectively.

2. Tags and Web 2.0 objects

A Web 2.0 *object* or *resource* (e.g., a textual document, audio image, or video) is defined as the main content of a Web 2.0 page. There are various sources of data related to this object, here referred to as its *features*, which we can classify as content features, textual features, user profile features, and social features.

Content features are attributes that can be extracted from the main content of the Web 2.0 object, such as the color histogram of an image. *Textual features*, in turn, comprise the self-contained textual blocks that are associated with an object, usually with a well-defined functionality, such as title, description, categories, tags, and user comments [3]. Note that these two sets of features may not be disjoint (e.g., when the main object is a textual document).

In particular, *tags* are keywords freely created by users and associated with objects. Tags are not necessarily unigrams (unless the application automatically splits them by whitespaces). Thus, tags may be composed by two or more words, sometimes separated by spaces, hyphenated, or joined.

Figure 1 illustrates a MovieLens page containing textual features assigned to an object (a movie, in this case).

User profile features include characteristics of the users who created or interacted with the content, while *social features* refer to interactions among users (e.g., explicit friendship links, subscriptions, "likes," etc.). The social connections among users may be explicitly represented by friendship links or implicitly indicated by subscriptions (connections established among users that show interests in one another's content), and endorsements (e.g., "likes"). **Figure 2** illustrates examples of these features.

DOI: http://dx.doi.org/10.5772/intechopen.82242

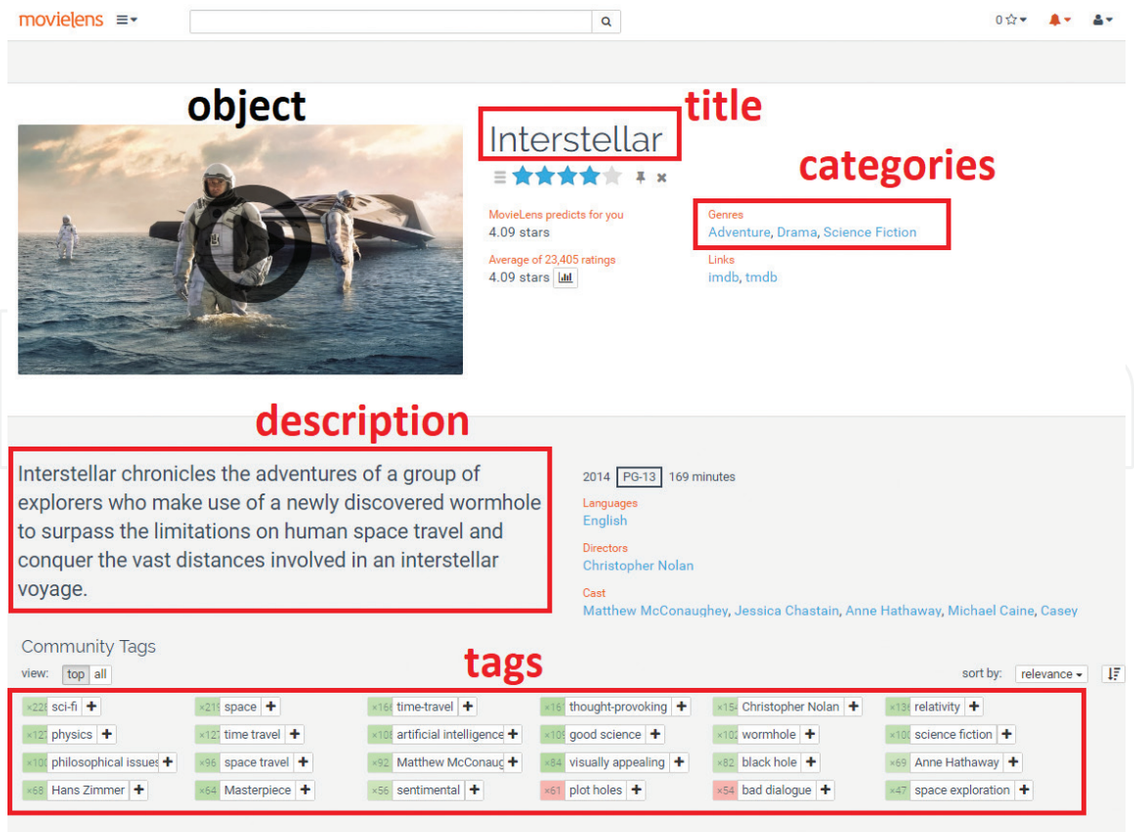

Figure 1.
A Web 2.0 page and some of its textual features.

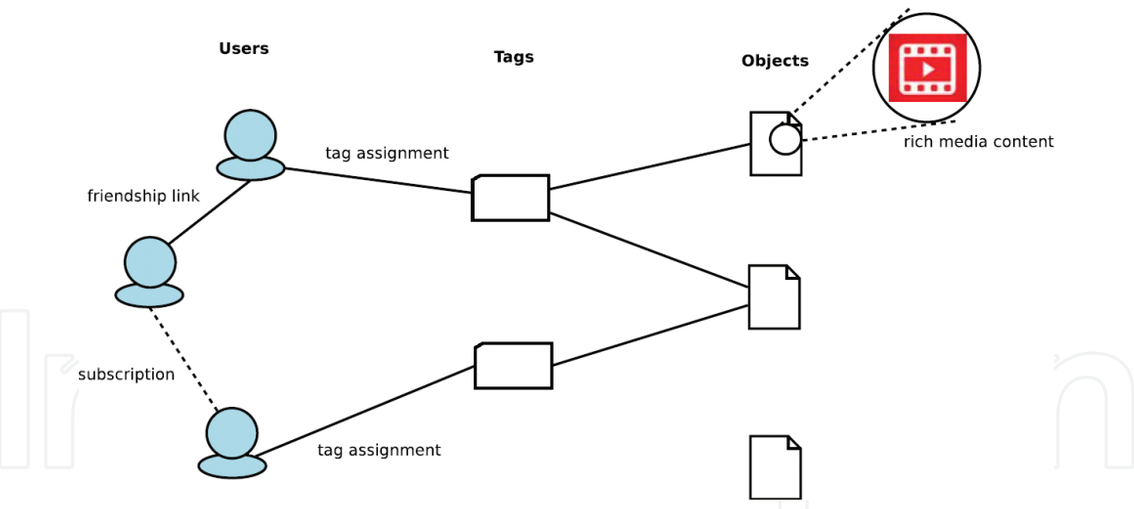

Figure 2.
Features commonly found in Web 2.0 pages. Friendship and subscription links are representative examples of social features. The set of tags a user assigned to objects in the applications is taken as one of the user profile features. Features extracted from the content of the main object (e.g., color histogram) are examples of content features [9].

The Web 2.0 tags, objects, and users form the basic structure of the *folksonomies*, which are defined as the categorization of objects using freely chosen keywords by users. Unlike a taxonomy, which provides a hierarchical categorization with well-defined classes, a folksonomy establishes categories (as tags) without imposing a hierarchical structure [10].

More formally, a folksonomy is defined as a relation $F = (U, T, O, P)$, where U, T, and O are finite sets composed by users, tags, and objects, respectively, and P, the set of postings, is a ternary relation between these elements, that is, $P \subseteq U \times T \times O$ [11].

Thus, each element $(u, t, o) \in P$ indicates that a user u associated a tag t to an object o (this is illustrated as the edges connecting users, tags, and objects in **Figure 2**).
In [12], folksonomies are classified in *broad* and *narrow* folksonomies. A broad folksonomy occurs when multiple users can apply the same tag to an object, while a narrow folksonomy occurs when only one user (typically the target object's creator) can tag a given object.

Examples of broad folksonomies include the online radio station LastFM (http://www.last.fm/) and the publication sharing application Bibsonomy (http://www.bibsonomy.org). The photo sharing site Flickr (http://www.flickr.com/) is an example of narrow folksonomy. While both broad and narrow folksonomies have common goals, a broad folksonomy can be further exploited to rank tags by their popularity and visualize the most important tags by means of tag clouds, which also provide an easy way to navigate the tags, objects, and users of a folksonomy.

Examples of tagging datasets available online for experimentation include MovieLens and Bibsonomy snapshots (https://grouplens.org/datasets/movielens and http://www.kde.cs.uni-kassel.de/bibsonomy/dumps, respectively) and our LastFM, YouTube, and YahooVideo crawled data (https://figshare.com/articles/data_tar_gz/2067183).

3. The tag recommendation problem

As in [13], we define two tag recommendation tasks: the *object-centered* problem and the *personalized* problem. In the former, the goal is to generate and rank candidate tags according to their relevance to the target object, that is, the extent to which the tag is related to or describes the target object. *Object-centered tag recommendations,* which do not vary according to the target user, aim at improving tag quality and indirectly improving the effectiveness of information retrieval services, such as searching, classification, and item recommendation, which exploit tags as data sources.

On the other hand, *personalized* tag recommendation takes not only the target object but also the target user into account, aiming at suggesting tags that are relevant to both the target object and the user. Thus, personalized tag recommenders might provide different results for different users, which may better capture the user interests, profile, and background. According to [13], "in applications where multiple users can assign tags to the same object, such as Last.FM, a personalized tag recommender is not only useful for the individual user (e.g., for content organization) but also in a collective sense. This is because, jointly, the tags recommended to different users may provide a more complete description of the object, benefiting search and recommendation services."

In more formal terms, the tag recommendation tasks are defined in [13] as:

> "**Object-Centered Tag Recommendation**. *Given a set of input tags I_o associated with the target object o, generate a list of candidate tags C_o, sorted according to their relevance to object o, and recommend the k candidates in the top positions of C_o.*
>
> **Personalized Tag Recommendation**. *Given a set of input tags I_o, associated with the target object o, generate a list of candidate tags $C_{o,u}$, sorted according to their relevance to both user u and object o, and recommend the k candidates in the top positions of $C_{o,u}$."*

Note that possibly there are no tags available in the target object, that is, $I_o = \varnothing$. This is a variation of the "cold start" problem, a well-known problem in recommender systems generally defined as a scenario in which there is an insufficient amount of information about the target user or object, making it difficult to provide effective recommendations.

These definitions focus on relevance as the only objective to be maximized. However, other aspects of the problem, such as novelty and diversity, have been considered as important, in recommendation systems in general and also in the specific tag recommendation domain [14].

According to the traditional definition of relevance or accuracy, the relevance of each tag in a recommendation list is independent of the relevance of the other tags in the list. However, in the general recommendation context, given that a recommendation satisfied the user need, the usefulness of similar recommendations is arguable. This occurs in the tagging context when, for example, only synonyms or strongly similar words are provided as recommendations. To deal with these issues, concepts of novelty and diversity have been introduced.

In tag recommendation, the novelty of a tag has been defined from the perspective of its popularity in the application. In [14], tag novelty is calculated as the inverse of the frequency at which the tag is used in the collection. The rationale is that frequently used tags tend to be more "obvious" recommendations (if relevant), thus being of little use to improve the description of the target object. We note that, according to this definition, noisy terms such as typos may be considered highly novel. However, novelty and diversity must be considered jointly with relevance in order to provide effective tag recommendations. It is worth mentioning that this definition of novelty is closely related to tag *specificity* [15], since rare words tend to be more specific (less general). For example, the word "feline" is less specific than "cat" or "tiger," and thus it is expected that "feline" would be used to describe a larger number of objects than these more specific terms. Therefore, specificity can be interpreted as a statistical property of the term use, being estimated as an inverse function of the frequency of the tag in the collection [14].

The *diversity* of a list of recommended tags, in turn, can be interpreted as the *exhaustivity* of these tags, which is defined in [15] as the coverage they provide for the topics of the associated object. Two approaches to estimate diversity in tag recommendation have been proposed. The implicit approach exploits properties of the recommended items (tags in our case), estimating diversity as the average pairwise semantic dissimilarity between the top recommended tags. In this context, a list of synonyms or semantically related words presents low diversity [14]. The explicit diversification approach, on the other hand, exploits properties of the target of recommendations, such as a set of explicit topics (e.g., categories) related to the target object. The goal of the explicit diversifiers is to cover as many topics related to the target object as possible, and as early in the ranking as possible, minimizing redundancy, that is, focus on a single topic.

Table 1 summarizes the tag recommendation problem and its aspects.

	Personalized	**Object-centered**
Input	I_o: set of input tags associated with the target object *o*	
Target	Object *o*	Pair object-user <*o*,*u*>
Output: ranked list of candidate tags	C_o: sorted according to relevance and other aspects related to *o*	$C_{o,u}$: sorted according to relevance and other aspects related to the pair <*o*,*u*>
Other aspects		
Novelty/specificity: capacity of recommending more rare tags **Diversity/exhaustivity**: capacity of recommending tags related to the different topics of the target object or user		

Table 1.
Tag recommendation: problem statement.

4. Candidate tag generation

Tag recommendation methods can be divided in two steps: (1) the generation of a set of candidate tags and (2) the ranking of the candidate tags produced in step (1). In this section, we introduce the main techniques to tackle the first step, while in Section 5, we discuss methods to perform the second step.

The candidate tag generation depends on the data sources available in the target application. As summarized by [9], previous tag recommendation strategies have exploited as data sources: (1) the folksonomy (history of tag assignments); (2) textual features (other than tags), such as title, description, and user comments; (3) rich media content, that is, image, audio, or video; and (4) social features, such as friendship links in social networks and other interactions among users as illustrated in Section 2.

Based on these data sources, we can name three main groups of techniques to extract or generate candidate tags: (1) extraction of terms from the textual features associated with the target object, (2) tag co-occurrences with terms in these textual features (possibly including previously assigned tags) or other features (e.g., visual features for rich media content), and (3) tags extracted from neighbors, that is, objects that are similar to the target object or users that are similar to the target user. These three groups of techniques will be the subject of Sections 4.1–4.3, respectively.

4.1 Keyword extraction from texts

The simplest strategy to extract candidate tags from a given text is to consider each (whitespace) separated word as a candidate, after removing punctuation and other special characters. After this, a basic post-processing step is to remove *stop words* (i.e., words such as articles, prepositions, and conjunctions, which carry little semantics and thus are not adequate as keywords) from the list of generated candidates. Finally, corpus-oriented statistics of these individual words are evaluated to select the most promising candidates. These statistics are also exploited to rank candidate tags, and thus they will be discussed in Section 5.

However, this simple strategy is only capable of generating single words as tags, although it is common to use expressions containing two or more words (e.g., "information systems," "digital image processing") as tags. Thus, alternative keyword extraction techniques first generate all word n-grams obtained from a sliding window through the text, for n ranging from one to, let us say, three or four words. For example, for the following sentence:

> "The tagging process can benefit a lot from a tag recommendation service."

A sliding window of size $n = 3$ would produce the following initial terms as keywords:

> The tagging process – tagging process can – process can benefit – can benefit a – benefit a lot – a lot from – lot from a – from a tag – a tag recommendation – tag recommendation service.

To filter out meaningless or uninformative candidate tags such as "benefit a lot" or "from a tag," some authors, such as [7, 16] exploit a selection approach based on part-of-speech (PoS) labels, which captures the idea of keywords having a certain syntactic property. Besides that, this approach is based on empirical evidence obtained

in training data. First, the most frequent PoS patterns of keywords that occur in a given training dataset are identified. For example, the three most frequent PoS patterns for keywords found in [15] are:

- ADJECTIVE + NOUN (singular or plural)
- NOUN + NOUN (both singular or plural)
- ADJECTIVE + NOUN (plural)

Thus, only sequences of words that match the top-x (let us say, x = 50) most frequent patterns are selected as candidate tags. For the aforementioned example and considering n = 2 and x = 3, the selected candidate tags would be "tagging process," "tag recommendation," and "recommendation service", all three of them matching the ADJECTIVE + NOUN pattern.

Unlike the PoS-based approach, which is a supervised, language-dependent approach that processes a training dataset, the *Rapid Automatic Keyword Extraction* (RAKE) [17] relies only on the target text to generate keywords, being known as a "document-oriented" approach, as opposed to the "corpus-oriented" methods. RAKE is based on the observation that keywords frequently contain multiple words but rarely contain standard punctuation or stop words. Instead of using an arbitrarily sized sliding window, RAKE splits the text using stop words and punctuation as delimiters. In our sentence example, "**The** tagging process **can** benefit **a lot from a** tag recommendation service," the stop words (in bold) would be discarded, generating the following candidate tags:

tagging process – benefit – tag recommendation service

After extracting candidate keywords, RAKE builds a graph of word co-occurrences, in which there is an edge between two words if they appeared in the same keyword. The score of each word w is calculated as $deg(w)/freq(w)$, where $deg(w)$ is the degree of w in the co-occurrence graph and $freq(w)$ is the number of occurrences of w in the text. The score of a given candidate keyword is defined as the sum of the scores of its containing words. Finally, in order to consider keywords that contain stop words (e.g., "set *of* natural numbers"), pairs of candidate keywords that appear in consecutive positions of the text at least twice are adjoined.

4.2 Tag co-occurrences

Another strong source of candidate tags is the history of tag assignments of the application (folksonomy). Tags that the target user frequently used in previous tagging events are good candidates to recommend for this user, especially in a personalized recommendation task. Still more interesting, we can exploit tag co-occurrences in these previous posts, recommending to an object o, with an initial set of tags I_o, tags that frequently co-occur with the tags in I_o in a training folksonomy dataset D, as performed by [2, 13, 14].

Tag co-occurrences are usually computed by exploiting association rules, which are employed in general to describe frequently co-occurring item sets. For tag recommendation, association rules assume the form $X \rightarrow y$, where X (the antecedent) is a set of tags and y (the consequent) is a candidate tag for recommendation. The main metrics that estimate the strength of an association rule are the *support*,

defined as the number of co-occurrences of X and y in the training set, and the confidence, calculated as the conditional probability that y is assigned as a tag to an object given that all tags in X are also associated with it. Considering that the number of rules extracted from the training set can be very large and some of them may not be useful for recommendation, minimum support and confidence thresholds are used as lower bounds to select only the most important and/or reliable rules. This selection can improve both effectiveness and efficiency of the recommender.

To recommend tags for an object o, we select rules $X \rightarrow y$ in which X is a subset of I_o, the set of initial tags in o. For each term c appearing as consequent of any of the selected rules, we usually estimate its relevance as a tag for the object (and for the user in the personalized case), given the initial tag set I_o, as the sum of the confidences of all rules containing c. In the absence of an initial tag set, words occurring in other textual features of the target object, such as title and description, can be used as I_o, as performed by [18]. Another alternative is to compute co-occurrences between tags and visual features extracted from images or other rich media content associated with the target object [19].

4.3 Tags from neighbors

Another form of obtaining candidate tags that are external to the target object, besides exploiting co-occurrences, is extracting tags from the neighborhood of the target object o, that is, the set of most similar objects with relation to o. Similarly, we can generate candidate tags for a target user u from similar users or users that have some kind of connection in the application (e.g., explicit friendship links, endorsement links, etc.). The rationale is that similar objects or users are usually associated with similar tags.

Thus, the neighborhood-based tag generation approaches exploit a graph in which the nodes correspond to objects or users, and there is an edge between two objects (or two users) if they are similar (e.g., share tags or other words in common). Alternatively, visual features extracted from image and video objects can be used to estimate content similarity [19, 20], although they may face scalability issues and a larger semantic gap [20].

To identify similar objects or users, each object (or user) is usually modeled as a bag of terms (extracted from the textual features of the object or from the vocabulary of the users). These terms receive a TFIDF weight, and a similarity measure such as the cosine of these term vector representations is exploited to estimate the similarity between objects or users [21].

5. Candidate tag ranking

After generating a set of candidate tags, it is necessary to rank them, showing the most relevant tags first, in order to provide effective tag recommendations. Some tag candidate generation strategies already provide a measure to estimate the candidate tag relevance, such as the degree/frequency ratio in RAKE, as defined in Section 4.1. In this section, we first discuss various tag quality attributes that can be used to estimate tag relevance (Section 5.1), isolated or combined with other attributes. Then, we discuss methods that can automatically combine various attributes exploiting a learning-to-rank approach (Section 5.2).

5.1 Tag quality attributes

Tag quality attributes can be grouped into the following categories, based on the aspect they try to capture regarding the tag recommendation task [13]:

- *Tag co-occurrence attributes*: estimate how relevant a candidate tag c is given a set of input tags that often co-occur with c in the data collection.

- *Descriptive power attributes*: estimate how accurately a candidate tag describes the object's content based on statistics of the occurrence of the tag in the textual features of the target object.

- *Discriminative power attributes*: estimate the capability of a candidate to distinguish the target object from others.

- *Term predictability*: indicates the likelihood that a word can be predicted as a tag.

- *User interest attributes*: used for personalization, these attributes estimate the interest of a target user in certain tags.

5.1.1 Tag co-occurrence attributes

As mentioned in Section 4.2, tag recommenders select association rules in which antecedents are included in I_o, the set of tags already available in the target object, or terms that can be used as proxy for these initial tags. For each tag c appearing as consequent of any of the selected rules, the relevance of c as a tag for the object o, given the initial tag set I_o, can be estimated by sum, which sums up the confidences of all rules that point to c, i.e.:

$$Sum(c, I_o) = \sum_{X \subseteq I_o} confidence(X \rightarrow c), (X \rightarrow c) \in R, |X| \leq l, \quad (1)$$

where R is a set of association rules generated from the training set and l is the size limit for the association rules' antecedents, usually limited to 1 or 2 words, due to performance issues.

Sum was proposed by [2], which also proposed several other attributes related to tag co-occurrences. For example, *Vote* (c, I_o) can be defined as the number of association rules whose antecedents are tags in I_o and whose consequent is the candidate tag c. In other words, it is the number of "votes" a candidate tag has received from related tags associated with the target object.

5.1.2 Descriptive power attributes

Descriptive power attributes usually estimate the descriptive capacity of candidate tags based on statistics of their occurrence in the textual features of the target object. We [13] proposed the use of four of these attributes for tag recommendation. We start by defining the *Term Spread* of a candidate c in an object o, $TS(c, o)$, as the number of textual features (except tags if we desire to recommend only "new" tags for that object) of o that contain c [3].

The rationale behind $TS(c, o)$ is that the larger the number of textual features of o containing c, the more related c is to o's content. For example, if the term "X-men" appears in all features of a video, there is a high chance that the video is related to the famous comics. Our results in [3] indicate that, in isolation, TS provides better tag recommendations than the traditional TF in most datasets.

TF or *term frequency*, in turn, is the total number of occurrences of the candidate tag c in all textual features of the target object o and thus considers these textual features as a single bag of words. In contrast, TS takes into account the multiple textual blocks that compound the structure of the target object.

However, neither TS nor TF consider that some textual features may describe the content of the target object more accurately than others. For example, the title is usually the most representative textual feature of the object's content [3]. Thus, we proposed in [13] two other attributes, which extend TF and TS, weighting a candidate tag based on the average descriptive powers of the textual features in which it appears.

To define these new attributes, we need first to automatically estimate the descriptive power of a textual feature F_i using the *average feature spread* (AFS) metric [3]. Let the *feature instance spread* of a feature $F_{i,o}$ associated with an object o, $FIS(F_{i,o})$, be the average TS over all terms in $F_{i,o}$. We define $AFS(F_i)$ as the average $FIS(F_{i,o})$ over all instances of F_i associated with objects in the training set D. Thus, we define weighted TS (wTS) and weighted TF (wTF) as

$$wTS(c,o) = \sum_{F_{i,o} \in o} \boldsymbol{I}(c,F_{i,o}) \times AFS(F_i), \quad where\ \boldsymbol{I}(c,F_{i,o}) = \begin{cases} 1, if\ c \in F_{i,o} \\ 0, otherwise \end{cases}$$
$$wTF(c,o) = \sum_{F_{i,o} \in o} tf(c,F_{i,o}) \times AFS(F_i) \qquad (2)$$

where $tf(c, F_{i,o})$ is the number of occurrences of the candidate tag c in textual feature $F_{i,o}$ of the target object o.

5.1.3 Discriminative power attributes

Discriminative power attributes promote more infrequent terms as tags, since they may better *discriminate* objects into different categories, topics, or levels of relevance, particularly considering that several services (e.g., classification, searching) often perform IR on multimedia content by using the associated tags as data sources. This aspect is captured by the *inverse feature frequency* (IFF) attribute [3], directly derived from the traditional *inverse document frequency* (IDF), considering, however, the term frequency in a specific textual feature (tags, in this case), instead of the full set of terms associated with the objects in the training dataset D. Given the number of elements in the training set $N = |D|$, the IFF of a candidate tag c in a textual feature i (tags in this case) is defined as $IFF(c, i) = log((N + 1)/(f_i(c) + 1))$, where $f_i(c)$ is the number objects in the training set in which c appears in the textual feature i. In our case, $f_i(c)$ is the number of training objects that are tagged with c.

We note that the value 1 is added to both numerator and denominator, without harming the tag specificity estimation, to deal with the value 0 in the denominator, which occurs for new terms that do not appear as tags in the training data.

IFF may have privilege terms from other textual features that do not appear as tags in the training data or noisy terms such as typos. Nevertheless, this attribute can be combined with the other attributes into a function, using, for example, learning-to-rank algorithms. Thus, its relative weight can be adjusted in order to avoid negative impacts in tag recommendation effectiveness.

Considering that both too general and too specific or noisy terms may not be ideal tag recommendations, [2] propose the s*tability* attribute, which promotes terms with intermediate frequency values.

5.1.4 Term predictability

Another important aspect for tag recommendation is term predictability. Heymann et al. [22] measure this characteristic through the term's *entropy*.

If a term occurs consistently with certain tags, it is more predictable, thus having lower entropy. Terms that occur indiscriminately with many other tags are less

predictable, thus having higher entropy. Term entropy can be useful particularly for breaking ties, as it is better to recommend more "consistent" or less "confusing" terms.

Another predictability attribute, called *Pred* [13, 18], measures the probability that a term is used as a tag in an object given that it was used in another textual feature of the same object.

5.1.5 User interest attribute

The user frequency (UF) attribute was used in [13, 18] in order to estimate the relevance of a candidate tag for a target user and thus provides personalized recommendations. *UF(c, u)* is simply the frequency at which the target user u assigns a candidate tag c to objects in a training collection. The idea is that the more frequently a user u assigns a candidate tag c to other objects in the application, the more relevant c is for u.

It is also common to exploit the temporal dynamics of tagging, particularly in user frequency-based tag attributes. From the observation that the temporal decay of the users' word choices follows a power-law function, the authors in [23] integrate a time component that gives more weight to tags that have been used more recently.

5.2 Learn-to-rank-based tag recommendation

Observing that recommendation is usually modeled as a ranking problem (i.e., we want to recommend the most relevant items first), learning-to-rank (L2R) techniques constitute an appropriate approach to tackle it. L2R-based methods are supervised approaches that automatically "learn" a ranking function from "previously seen" data known as training instances. Such training examples usually consist of candidate tags, their tag quality attribute values, and their relevance labels, which indicates their relevance levels. These labels can be assigned either manually or by exploiting previous tag assignments as ground truth. The objective of L2R approaches is to generate a model (function) that maps the tag quality attributes into a relevance score or rank.

More formally, for each candidate tag c for each object o (or pair object-user $<o, u>$ for personalized recommendation), we associate a vector $X_{c,o} \in \mathbb{R}^m$ (or $X_{c,o,u} \in \mathbb{R}^m$), where m is the number of considered tag quality attributes (e.g., each metric defined in Section 4). For training instances, we also assign a relevance label $y_{c,o}$ (or $y_{c,o,u}$), indicating the relevance level of candidate tag c to the object o (and user u). For example, we can define two relevance levels: 1 for relevant tags, and 0 for nonrelevant tags. In the offline training step, this data is exploited to generate the recommendation model. In the online recommendation step, in which we have new objects or users as input, the $y_{c,o}$ (or $y_{c,o,u}$) values are unknown, and the model learned in the training step is applied in order to predict these values.

Various L2R-based algorithms have been proposed for tag recommendation in the literature, including RankSVM, RankBoost, Genetic Programming, Random Forest (RF), Multiple Additive Regression Trees (MART), Lambda-MART, AdaRank, ListNet, Ranknet, and Coordinate Ascent. In [24] we can find a brief description of each of these algorithms and experimental results of the comparison of these methods using the RankLib tool (https://sourceforge.net/p/lemur/wiki/RankLib/). According to our results, RF, MART, and Lambda-MART are found to be the best performing strategies for the tag recommendation problem.

In [25], the author reviewed existing L2R algorithms in the context of document ranking, categorizing them into three approaches: pointwise, pairwise, and listwise. The *pointwise* approach associates a numerical score to each query-document pair and thus approximates the ranking problem by a regression problem. *Pairwise* approaches, in turn, transform the ranking problem into binary classification: given a pair of

documents (or tags, in our case), we need to predict which one is the most relevant. Finally, the *listwise* approaches try to directly optimize a given evaluation measure.

Finally, it is also worth mentioning that, instead of adopting an attribute engineering approach, exploiting various handcrafted attributes like those described in Section 4, some recent works focus on investigating techniques that can learn attribute interactions from raw data, such as deep learning and factorization machines (FM) [26, 27]. The most representative method of this group is pairwise interaction tensor factorization (PITF). In this method, the tensor (i.e., a "tridimensional matrix") that models the pairwise interactions among users, items, and tags (i.e., the ranking preferences of the tags for each pair user object, which is obtained from the folksonomy relation data) is factored into lower-dimensional matrices to reduce noise [27]. The PITF model is learned from an adaption of the Bayesian personalized ranking (BPR) criterion. More recently, [26] exploit not only the folksonomy but also visual features of images, such as the objects appearing in the image, colors, shapes, or other visual aspects, into factorization machine models.

6. Conclusions

In this chapter, we have reviewed the main concepts related to tags and tag recommendation. There are various sources of data associated with Web 2.0 objects that can be used to extract and rank tags. Candidate tags can be extracted from the textual features associated with the target object using keyword extraction techniques, from mining co-occurrences with other tags, or other textual and content features, and from the neighborhood of the target object and/or target user. We also have briefly discussed various tag quality attributes that can be exploited to rank candidate tags. An effective way to combine these attributes is by means of learn-to-rank techniques, which can automatically "learn" tag recommendation functions from training examples.

Acknowledgements

Our research group is partially funded by Google, the Brazilian National Institute of Science and Technology for Web Research (MCT/CNPq/INCT Web Grant Number 573871/2008-6), and the authors' individual grants from CNPq, CAPES, and FAPEMIG.

Author details

Fabiano M. Belém*, Jussara M. Almeida and Marcos A. Gonçalves
Computer Science Department, Federal University of Minas Gerais, Belo Horizonte, Minas Gerais, Brazil

*Address all correspondence to: famube@gmail.com

References

[1] Gupta M, Li R, Yin Z, Han J. Survey on social tagging techniques. SIGKDD Explorations. **12**(1):58-72

[2] Sigurbjörnsson B, Zwol R. Flickr tag recommendation based on collective knowledge. In: Proceedings of WWW Conference. 2008. pp. 327-336

[3] Figueiredo F, Belém F, Pinto H, Almeida J, Gonçalves M. Assessing the quality of textual features in social media. Information Processing & Management. 2012;**49**:222-247

[4] Zuo Y, Zeng J, Gong M, Jiao L. Tag-aware recommender systems based on deep neural networks. Neurocomputing. 2016;**204**(C):51-60

[5] Hsu M, Chen H. Efficient and effective prediction of social tags to enhance web search. Journal of the Association for Information Science and Technology. 2011;**62**(8):1473-1487

[6] Oliveira V, Gomes G, Belém F, Brandão W, Almeida J, Ziviani N, et al. Automatic query expansion based on tag recommendation. In: Proceedings of the International Conference on Information and Knowledge Management. 2012. pp. 1985-1989

[7] Ribeiro I, Santos R, Gonçalves M, Laender H. On tag recommendation for expertise profiling: A case study in the scientific domain. In: Proceedings of the International Conference on Web Search and Data Mining. 2015. pp. 189-198

[8] Venetis P, Koutrika G, Garcia-Molina H. On the selection of tags for tag clouds. In: Proceedings of the ACM Conference on Web Search and Data Mining; 2011. pp. 835-834

[9] Belém F, Almeida J, Gonçalves M. A survey on tag recommendation methods. Journal of the Association for Information Science and Technology. 2016;**68**(4):830-844

[10] Spiteri L. Structure and form of folksonomy tags: The road to the public library catalogue. Webology. **4**(2):13-25

[11] Jäschke R, Marinho L, Hotho A, Schmidt-Thieme L, Stum G. Tag recommendations in folksonomies. In: Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases. 2007. pp. 506-514

[12] Wal V. Explaining and Showing Broad and Narrow Folksonomies. Retrieved on Oct 8th, 2015: http://www.vanderwal.net/random/entrysel.php?blog=1635

[13] Belém F, Martins E, Almeida J, Gonçalves M. Personalized and object-centered tag recommendation methods for web 2.0 applications. Information Processing & Management. 2014;**50**(4):524-553

[14] Belém F, Batista C, Santos R, Almeida J, Gonçalves M. Beyond relevance: Exploiting novelty and diversity in tag recommendation. ACM Transactions on Intelligent Systems and Technology. 2016;7:26:1-26:34

[15] Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Boston, USA: Addison-Wesley; 2011

[16] Hulth A. Improved automatic keyword extraction given more linguistic knowledge. In: Conference on Empirical Methods in Natural Language Processing. 2003. pp. 216-223

[17] Rose S, Engel D, Cramer N, Cowley W. Automatic keyword extraction from individual documents. In: Text Mining: Applications and Theory. Wiley; 2010. pp. 1-20. Available at: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470689646.ch1

[18] Lipczak M, Milios E. Efficient tag recommendation for real-life data.

ACM Transactions on Intelligent SystemsTechnology. 2011;**3**(1):2:1-2:21

[19] Wu L, Yang L, Yu N, Hua X. Learning to tag. In: Proceedings of the 18th International Conference on World WideWeb. 2009. pp. 361-370

[20] Zhu X, Nejdl W, Georgescu M. An adaptive teleportation random walk model for learning social tag relevance. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2014. pp. 223-232

[21] Martins E, Belém F, Almeida J, Gonçalves M. On cold start for associative tag recommendation. Journal of the Association for Information Science and Technology. 2016;**67**(1):83-105

[22] Heymann P, Ramage D, Garcia-Molina H. Social tag prediction. In: Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval. 2008. pp. 531-538

[23] Kowald D. Modeling cognitive processes in social tagging to improve tag recommendations. PHD-Symposium paper. 2018. Available at: https://arxiv.org/pdf/1805.11878.pdf

[24] Canuto S, Belém F, Almeida J, Gonçalves M. A comparative study of learning-to-rank techniques for tag recommendation. Journal of Information and Data Management. 2013;**4**:453-468

[25] Liu T-Y. Learning to rank for information retrieval. Foundations and Trends in Information Retrieval. 2009;**3**(3):225-331

[26] Nguyen H, Wistuba M, Schmidt-Thieme L. Personalized tag recommendation for images using deep transfer learning. In: Ceci M, Hollmen J, Todorovski L, Vens C, Dzeroski S, editors. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. 2017. pp. 705-720

[27] Rendle S, Schmidt-Thieme L. Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining. 2010. pp. 81-90