Associative Tag Recommendation Exploiting Multiple Textual Features*

Fabiano Belém, Eder Martins, Tatiana Pontes, Jussara Almeida, Marcos Gonçalves

Computer Science Department
Universidade Federal de Minas Gerais, Brazil
{fmuniz, ederfm, tpontes, jussara, mgoncalv}@dcc.ufmg.br

ABSTRACT

This work addresses the task of recommending relevant tags to a target object by jointly exploiting three dimensions of the problem: (i) term co-occurrence with tags pre-assigned to the target object, (ii) terms extracted from multiple textual features, and (iii) several metrics of tag relevance. In particular, we propose several new heuristic methods, which extend state-of-the-art strategies by including new metrics that try to capture how accurately a candidate term describes the object's content. We also exploit two learning-to-rank (L2R) techniques, namely RankSVM and Genetic Programming, for the task of generating ranking functions that combine multiple metrics to accurately estimate the relevance of a tag to a given object. We evaluate all proposed methods in various scenarios for three popular Web 2.0 applications, namely, LastFM, YouTube and YahooVideo. We found that our new heuristics greatly outperform the methods on which they are based, producing gains in precision of up to 181%, as well as another state-of-the-art technique, with improvements in precision of up to 40% over the best baseline in any scenario. Further improvements can also be achieved with the new L2R strategies, which have the additional advantage of being quite flexible and extensible to exploit other aspects of the tag recommendation problem.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.5 [**Information Storage and Retrieval**]: Online Information Services

General Terms

Algorithms, Experimentation

Keywords

Tag Recommendation, Relevance Metrics

*This work is supported by the INWeb (grant 57.3871/2008-6) and by the authors grants from CNPq and FAPEMIG.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR'11, July 24–28, 2011, Beijing, China.
Copyright 2011 ACM 978-1-4503-0757-4/11/07 ...$10.00.

1. INTRODUCTION

The act of associating *tags* to objects has become paramount in Web 2.0 applications, mainly due to the strong stimuli and easiness for user participation in content creation. This content, typically multimedia (e.g., images and videos), brings challenges to current multimedia Information Retrieval (IR) methods, not only due to the scale of object collections and upload rate, but also due to the (usually) poor quality of user-generated material [5]. Tags, among other textual features such as title and description, commonly associated to Web 2.0 objects, offer a good alternative for supporting content organization, dissemination and retrieval services. In fact, recent studies have demonstrated that tags are one of the best textual features to be exploited by various IR services, such as automatic object classification [7].

In this context, tag recommendation services aim at improving the quality of the available content, and ultimately of information services that rely on them, by suggesting terms that are related to the content of a target object. Many existing strategies exploit term co-occurrence patterns in previous tag assignments in the collection, expanding an initial tag set $\mathcal{I}_o$ of a target object o with other tags that frequently co-occur together with the terms in $\mathcal{I}_o$ [11, 22, 9, 19]. Other methods do not assume the existence of such tags in the target object, using, instead, terms extracted from other textual features [16, 25]. However, various textual features, including tags, are created by the end users, and thus, may contain a lot of *noise* (e.g., misspellings or unrelated terms) [7, 14]. Thus, it is important to filter such terms out of the list of recommendations or reduce their importance, favoring terms that are more "relevant" for the target object. By relevant, we mean terms that are good descriptors of the object's content and/or that help discriminate it from others, for supporting services such as searching and classification which typically use tags as data sources. With that in mind, some previous methods [22, 16, 25, 4] exploit metrics of relevance, such as Term Frequency (TF), either to filter out irrelevant candidates or to boost candidates with more potential. In sum, many previous methods address the problem of recommending a tag for a target object by exploiting (i) term co-occurrences with tags pre-assigned to the object, (ii) terms extracted from multiple textual features, and (iii) metrics of relevance. However, to our knowledge, most of them exploit at most two of these three dimensions[1].

In contrast, we here address the task of recommending

[1]Other dimensions can also be exploited (e.g., user-object-tag relationships [10], and tag usage history [12]). Extending our methods to include them is left for the future.

tags for a given Web 2.0 object by jointly exploiting all three dimensions. In other words, we extend traditional tag co-occurrence based approaches to include not only tags that have been previously assigned to the objects (including the target object), but also terms contained in other textual features, such as title and description. The contents of these textual features are used to extract candidate terms. We also exploit several heuristic metrics to try to capture the *relevance* of each such term as a recommendation for the target object. Some of these metrics, to our knowledge, have never been applied to tag recommendation before. Our approach is to model the tag recommendation problem as a multiple term candidate *ranking* problem. That is, we develop functions to estimate the relevance of a candidate term as a tag recommendation for a given object, thus enabling us to rank the candidates according to such estimations and recommend the most relevant ones as tags for the object.

Specifically, we propose ten tag recommendations strategies. Eight of them are heuristics built as extensions of two state-of-the-art techniques that exploit tag co-occurrence and some metrics of relevance. Our heuristics extend these previous methods by including new metrics of relevance that try to capture how accurately a candidate term *describes* the object's content, and by exploiting multiple textual features. They are simple, easy to compute, and quite efficient, producing gains of up to 181% in precision over the original techniques on which they are based.

However, a number of heuristics can be devised to combine multiple metrics of relevance into a final tag recommendation function. Finding the "best" heuristic is not an easy task due to the potential large size of the search space, which, in our case, consists of all possible tag recommendation functions that can be built using the suggested metrics.

Thus, we also investigate the benefits of applying learning-to-rank (L2R) techniques for tag recommendation purposes, proposing two new strategies. One strategy exploits the traditional RankSVM method [13], whereas the other is based on Genetic Programming (GP) [3]. RankSVM and GP are here treated as *meta-heuristics* to generate ranking functions that combine all given metrics to accurately estimate the relevance of each given candidate term. Our motivation to use L2R methods are threefold: (1) they can effectively exploit many features in the generation of ranking functions, (2) they can be easily extended to include more features, and (3) there is a strong theoretical background on learning methods, which has been recently extended for ranking problems [27].

We evaluate the proposed tag recommendation functions, comparing them against three state-of-the-art techniques, namely Sum^+, the best function proposed in [22], which exploits co-occurrence of pre-assigned tags along with some document frequency statistics, $LATRE$ [19] a very recent, efficient and effective tag co-occurrence based method, and the winner of the ECML Discovery Challenge 2009 [16], referred to here as *Co-occurrence and Text-based Tag Recommender* (CTTR). CTTR exploits the contents of textual features associated with the target object along with one metric of relevance, but does not consider the tags previously assigned to the target object. Our evaluation is performed with real datasets collected from three popular Web 2.0 applications: the video sharing sites YouTube and YahooVideo, and the online radio station LastFM[2].

Our results indicate that our best heuristic outperforms the best baseline in any scenario by up to 40% in precision and 32% in recall. This heuristic extends the $LATRE$ baseline by incorporating a new metric that tries to capture the descriptive power of each candidate term and by exploiting multiple textual features. Moreover, some further improvements over our heuristics (up to 12% in precision) can also be achieved with our L2R based strategies, although the best of the two techniques depends on the dataset. Despite the somewhat modest improvements over our heuristics, the L2R based strategies provide a flexible framework that can be easily extended to include other metrics of relevance or to address other aspects of the tag recommendation problem.

The rest of this paper is organized as follows. Section 2 discusses related work, whereas Section 3 defines the problem and the main metrics used for tag recommendation. Section 4 introduces the methods we analyze here, while our experimental evaluation is presented in Section 5. Finally, conclusions and future work are offered in Section 6.

2. RELATED WORK

Many tag recommendation strategies exploit co-occurrence patterns computed over a history of tag assignments. In particular, some of them exploit tag co-occurrences to expand an initial set of tags $\mathcal{I}_o$ associated with an object o [22, 9, 11, 26, 15, 19]. For this purpose, Heymann *et al.* [11] use association rules, i.e., implications of the form $X \rightarrow y$, where the antecedent X is a set of tags, and the consequent y is a candidate tag for recommendation, restricting the rules by a confidence threshold. However, the authors do not provide a ranking of the recommended tags. Sigurbjornsson and Zwol [22], on the contrary, exploit simple global metrics of tag co-occurrence (e.g., confidence), applying them over all tags in the initial set to produce a final ranking of candidate tags. Some of the considered metrics, related to tag frequency, try to capture the "relevance" of each candidate. In comparison, we here consider a much richer set of metrics, including new metrics based on multiple textual features, which, as we shall see, are responsible for our largest improvements.

Due to efficiency issues, most of these strategies usually compute co-occurrences between only two tags (i.e., X contains only one tag), thus possibly missing important co-occurrence relationships. To address this problem, Menezes *et al.* [19] propose LATRE - Lazy Associative Tag Recommendation, which computes association rules in an on-demand manner, allowing an efficient generation of more complex and potentially better rules, and producing superior results in comparison with the best method in [22].

A few other efforts do not exploit tags previously assigned to the target object, focusing, rather, on other data sources [16, 25]. Lipczak *et al.* [16] extract terms from other textual features (e.g., title) of the target object, expanding them by association rules, and sorting the extracted terms by their usage as tags in a training set. Wang *et al.* [25] use the traditional $TF \times IDF$ metric to extract and rank the most important terms from the object's textual content.

In sum, these previous methods address the tag recommendation problem by exploiting (i) co-occurrence patterns

[2]http://www.youtube.com, http://www.yahoo.com/video, and http://www.last.fm, respectively

among previously assigned tags, (ii) multiple textual features, and (iii) metrics of relevance. However, to our knowledge, none of them jointly exploits all three dimensions. In contrast, in a very preliminary work [4], we investigated the benefits of combining all three approaches to perform tag recommendation, obtaining results that outperform those of some previous techniques. These preliminary findings motivated the present study, which greatly extends our prior work by proposing new metrics of relevance and several new recommendation strategies (including L2R based strategies), evaluating them on more recently collected and larger datasets, comparing them against more baselines, and reaching significantly better results.

In addition to co-occurrence and text based strategies, other approaches have also been exploited. For example, Wu *et al.* [26] add image content information to rank tags. Siersdorfer *et al.* [21] create a graph of videos based on content similarity, and make recommendations by propagating tags through its edges, whereas Song *et al.* [23] exploit a bipartite graph containing tags, documents and words. Other studies address the problem of *personalized recommendation*, often exploiting the history of per-user tag assignments. Collaborative filtering and *FolkRank* [12], as well as PITF - Pairwise Interactions Tensor Factorization [20] fall into this category. Other graph-based personalized recommendation strategies exploit social network relationships [10, 18]. We do not tackle personalization here, leaving the task of extending our strategies to address it for the future.

Finally, the only prior efforts to apply learning-to-rank techniques to recommend tags were pursued by Cao *et al.* [6], who exploited RankSVM, and by Wu *et al.* [26], who exploited RankBoost. However, the latter considered only metrics related to tag co-occurrences and image content, and did not exploit textual features, whereas the former considered only metrics related to tag frequency, disregarding tag co-occurrence metrics.

3. TAG RECOMMENDATION: PROBLEM STATEMENT AND METRICS

A Web 2.0 *object* refers to an instance of a media (text, audio, video, image) in a given Web 2.0 application. There are various sources of information related to an object, here referred to as its features. In particular, *textual features*, our main source of information, comprise the self-contained textual blocks that are associated with an object, usually with a well defined functionality [7]. The textual features here exploited are the object *tags*, *title* and *description*.

We define the recommendation problem as follows. Given a tag set $\mathcal{I}_o$ previously assigned to an object[3] o, and the set of textual features (other than tags) $\mathcal{F}_o = \{\mathcal{F}_o^1, \mathcal{F}_o^2, ..., \mathcal{F}_o^n\}$, where each element $\mathcal{F}_o^i$ is the set of terms in textual feature i associated with o, generate a candidate set $\mathcal{C}_o$ and recommend the k most relevant terms of this set.

Many tag recommendation strategies, and in particular the ones proposed here, exploit co-occurrence patterns by mining relations among tags assigned to the same object in an object collection. The process of learning such patterns is defined as follows. There is a training set $\mathcal{D} = \{\langle \mathcal{I}_d, \mathcal{F}_d \rangle\}$, where $\mathcal{I}_d$ ($\mathcal{I}_d \neq \emptyset$) contains all tags assigned to object d, and $\mathcal{F}_d$ contains the term sets of the other textual features associated with d. There is also a test set $\mathcal{O}$, which is a collection of objects $\{\langle \mathcal{I}_o, \mathcal{F}_o, \mathcal{Y}_o \rangle\}$, where both $\mathcal{I}_o$ and $\mathcal{Y}_o$ are sets of tags associated with object o. However, while tags in $\mathcal{I}_o$ are known (and given as input to the recommender), tags in $\mathcal{Y}_o$ are assumed to be unknown and are taken as the relevant recommendations to the target object o (i.e., the *expected answer*). Splitting the tags of each test object into these two subsets facilitates an automatic assessment of the recommendations, as performed in [9] and further discussed in Section 5.2. This evaluation simulates the recommendation of new tags ($\mathcal{Y}_o$) to an object that already has been annotated with $\mathcal{I}_o$. Similarly, there might be also a validation set $\mathcal{V}$ used for tuning parameters and "learning" recommendation functions (see Section 5.2). Thus, each object v in $\mathcal{V}$ also has its tag set split into input tags $\mathcal{I}_v$ and expected answer $\mathcal{Y}_v$.

Thus, we define associative and text based tag recommendation methods as algorithms that estimate the relevance of a tag candidate set relying on the elements described above. Next, we present several metrics that can be used to estimate the relevance of a candidate for tag recommendation. Some of them, like *Sum*, *Stability*, TF and *Entropy*, have been previously applied for recommending tags [22, 19, 11]. Others, such as IFF and AFS, were proposed in [7] for assessing the quality of textual features on the Web 2.0, and have not been exploited for tag recommendation yet.

3.1 Tag Co-occurrence

Tag recommendation approaches based on co-occurrence usually exploit association rules, that is, implications of type $X \rightarrow y$, where the antecedent X is a set of tags and the consequent y is a candidate tag for recommendation. The importance of an association rule is estimated based on **support** (σ), which is the number of co-occurrences of X and y in the training set, and **confidence** (θ), which is the conditional probability that y is assigned as a tag to an object $d \in \mathcal{D}$ given that all tags in X are also associated with d.

As the number of rules mined from $\mathcal{D}$ can be very large and some of them may not be useful for recommendation, minimum support and confidence thresholds (σ_{min} and θ_{min}, respectively) are used as lower bounds to select only the most frequent and/or reliable rules. This selection can improve both effectiveness and efficiency of the recommender.

At recommendation time, we select the rules whose antecedents are included in $\mathcal{I}_o$, the available tags in the target object o. For each term c appearing as consequent of any of the selected rules, we estimate its relevance as a tag for o as the sum of the confidences of all rules containing c, i.e.:

$$Sum(c, o, \ell) = \sum_{X \subseteq \mathcal{I}_o} \theta(X \rightarrow c), \quad (X \rightarrow c) \in \mathcal{R}, |X| \leq \ell \qquad (1)$$

where $\mathcal{R}$ is a set of association rules computed offline over the training set $\mathcal{D}$ given thresholds σ_{min} and θ_{min}, and ℓ is the size limit for the association rules' antecedents.

3.2 Discriminative Power

One may argue that recommending more infrequent terms (provided that they are not too rare) may be desirable, since they may better *discriminate* objects into different cate-

[3]We focus on cases in which there are a few available tags in the target object, and we want to recommend new (different) tags to it. However, our methods are also able to recommend relevant tags to an object with no initial tag by exploiting other textual features and metrics of relevance. We leave this for future work.

gories, 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 can be heuristically captured by the Inverse Feature Frequency (IFF) metric [7], an adaptation of the traditional Inverse Document Frequency (IDF) that considers the term frequency in a specific textual feature (in our case, tags). Given the number of objects in the training set $|\mathcal{D}|$, the IFF of a candidate c is defined as:

$$IFF(c) = \log \frac{|\mathcal{D}| + 1}{f_c^{tag} + 1} \quad (2)$$

where f_c^{tag} is the number of objects in $\mathcal{D}$ which contain c *attached as a tag*. Note that c may be extracted from other textual features. The value 1 is added to both numerator and denominator to deal with new terms that do not appear as tags in the training data. We note that this metric may privilege terms from other textual features that do not appear as tags in the training data. Nevertheless, this metric will be combined with the other metrics into a function, using learning-to-rank algorithms. Thus, its relative weight can be adjusted, as we describe in Section 4.

Along the same lines, one may consider that terms that are very common, such as "video" in a YouTube object collection, are too general, whereas very rare terms may be too specific or may represent noise (e.g., misspellings, neologisms and unknown words). In either case, such terms represent poor recommendations as they have very poor *discriminative power*. Sigurbjornsson *et al.* [22] propose the Stability (*Stab*) metric, which gives more importance to terms with intermediate frequency values:

$$Stab(c, k_s) = \frac{k_s}{k_s + |k_s - \log(f_c^{tag})|} \quad (3)$$

where k_s represents the "ideal frequency" of a term and must be adjusted to the data collection. We here also use *Stab* to assess the relevance of a tag candidate, but, unlike [22], we apply it not only to tags but also to terms extracted from all textual features $\mathcal{F}_o$ associated with target object o.

3.3 Descriptive Power

We here exploit four heuristic metrics that try to capture, to some extent, the *descriptive power* of a candidate c. Two of them, Term Spread (TS) and Term Frequency (TF), were previously applied to tag recommendation. However, TS was exploited in a very preliminary study [4], whereas previous studies employing TF [25, 6] have not exploited it jointly with co-occurrence of pre-assigned tags, as we do here. The other metrics, derived from TS and TF, are applied here for the first time to 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, in the present context)[4] of o that contain c [7]:

$$TS(c, o) = \sum_{\mathcal{F}_o^i \in \mathcal{F}_o} j, \text{ where } j = \begin{cases} 1 & \text{if } c \in \mathcal{F}_o^i \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The assumption behind $TS(c, o)$ is that the larger the number of features of o containing c, the more related c is to o's content. For example, if the term "Sting" appears in all features of a video, there is a high chance that the video is related to the famous singer. The maximum TS is given by the number of textual features, other than tags, considered. As we here consider title and description, $TS \leq 2$.

The *Term Frequency* of c in object o, $TF(c, o)$, is:

$$TF(c, o) = \sum_{\mathcal{F}_o^i \in \mathcal{F}_o} tf(c, \mathcal{F}_o^i), \quad (5)$$

where $tf(c, \mathcal{F}_o^i)$ is the number of occurrences of c in textual feature i of object o. Thus, TF considers all textual features of o as a single list of terms, counting all occurrences of c in it. In contrast, TS considers the structure of an object, composed by textual features, which are well-defined blocks of text, counting the number of blocks containing c.

Although both TS and TF try to capture how accurately a term describes an object's content, neither of them considers that different features may present, in general, different descriptive capacities. For example, the title may describe an object's content more accurately than other textual features [7]. Thus, we propose two other metrics, built on TF and TS, that weight a term based on the average descriptive powers of the textual features in which it appears.

The average descriptive power of a textual feature $\mathcal{F}^i$ is assessed by the Average Feature Spread (AFS) heuristic [7]. Let the *Feature Instance Spread* of a feature $\mathcal{F}_o^i$ associated with object o, $FIS(\mathcal{F}_o^i)$, be the average TS over all terms in $\mathcal{F}_o^i$. We define $AFS(\mathcal{F}^i)$ as the average $FIS(\mathcal{F}_o^i)$ over all instances of $\mathcal{F}^i$ associated with objects in the training set $\mathcal{D}$. We then define weighted TS and TF metrics as:

$$wTS(c, o) = \sum_{\mathcal{F}_o^i \in \mathcal{F}_o} j, \text{ where } j = \begin{cases} AFS(\mathcal{F}^i) & \text{if } c \in \mathcal{F}_o^i \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$wTF(c, o) = \sum_{\mathcal{F}_o^i \in \mathcal{F}_o} tf(c, \mathcal{F}_o^i) \times AFS(\mathcal{F}^i) \quad (7)$$

3.4 Term Predictability

Another important aspect for tag recommendation is term predictability. Heymann *et al.* [11] measure this characteristic through the term's *entropy*. The entropy of term c in the tags feature, $H^{tags}(c)$, is defined as:

$$H^{tags}(c) = -\sum_{(c \to i) \in \mathcal{R}} \theta(c \to i) \log \theta(c \to i) \quad (8)$$

If a term occurs consistently with certain tags, it is more predictable, having lower entropy. Terms that occur indiscriminately with many other tags are less predictable, thus having higher entropy. In other words, $H^{tags}(c)$ measures the concentration of confidence values of all association rules whose antecedent is c. If a term is absent in the training set, it receives an arbitrarily high entropy, as, in this case, the result is not a real number. Term entropy can be useful particularly for breaking ties, as it is better to recommend more "consistent" or less "confusing" terms. Whereas term entropy was used in [11] only to evaluate recommendations, we here apply it as an input to the recommendation functions.

4. TAG RECOMMENDATION STRATEGIES

We now turn our attention to the analyzed tag recommendation strategies. We start by describing, in Section 4.1, the three baselines, which, as previously mentioned, exploit at most two of the following: term co-occurrence with

[4]We do not include tags to compute any of the descriptive power metrics, as it does not make sense to use tags previously assigned to the target object as candidates for recommendation.

pre-assigned tags, multiple textual features and metrics of relevance. Next, in Section 4.2, we introduce our new heuristics, which exploit these three dimensions and include novel relevance metrics. Our learning-to-rank based strategies, which also exploit the three dimensions as well as a larger set of relevance metrics, are presented in Section 4.3.

4.1 State-of-the-Art Baselines

Our first baseline is Sum^+, the best function proposed in [22], which exploits both tag co-occurrences and metrics of relevance. It extends the *Sum* metric (Eq. 1) by weighting the confidence values by the *Stability* of the terms in the antecedent and consequent of the corresponding association rules, which are here generated by the *Apriori* algorithm [1]. Given a candidate c for an object o, Sum^+ is defined as:

$$Sum^+(c,o,k_x,k_c,k_r) = \sum_{x\in\mathcal{I}_o} \theta(x\rightarrow c) \times Stab(x,k_x) \times Stab(c,k_c) \times Rank(c,o,k_r) \quad (9)$$

where k_x, k_c and k_r are tuning parameters. $Rank(c,o,k_r)$ is equal to $k_r/(k_r+p(c,o))$, where $p(c,o)$ is the position of c in the ranking of candidates according to the confidence of the corresponding association rule. This factor is employed to make confidence values decay smoother.

Sum^+, as most co-occurrence based strategies [22, 9], restricts the size of the association rules to only one tag in the antecedent (i.e., ℓ=1) due to efficiency issues. In contrast, $LATRE$ - Lazy Associative Tag Recommender [19], our second baseline, is able to efficiently generate larger association rules by doing it *on demand*. This is in contrast to other strategies (e.g., *Apriori*), which compute all rules from the training set beforehand (i.e., offline), possibly including rules that might not be useful when recommending for objects in the test set. $LATRE$ ranks each candidate c by the sum of the confidences of all rules containing c. That is, it uses the *Sum* metric (Eq. 1) with $\ell\geq$1, thus exploiting solely co-occurrence patterns. We note that two variations of $LATRE$, with and without calibration, are presented in [19]. We here consider the $LATRE$ without calibration since it has lower complexity and, according to [19], produces very similar results to those obtained with calibration, with no significant differences in most cases.

Our third baseline is *Co-occurrence and Text based Tag Recommender* (CTTR), which exploits terms extracted from other textual features and a metric of relevance, but does not consider tags previously assigned to the *target object*. CTTR is an adaptation of the winner of the ECML Discovery Challenge 2009 [16], which, in addition to both aforementioned aspects, also takes the user tag assignment history into account to provide personalized recommendations. We here do not include user statistics in CTTR as we aim at providing recommendations for a target object, thus leaving personalization for future work. We note however that, comparing our methods, which also consider terms extracted from other features, against CTTR allows us to assess the benefits of applying our metrics of relevance to such terms and to exploit co-occurrence of previously assigned tags.

In a nutshell, CTTR first extracts term candidates from the title and the description of the target object. For each such term c, it assigns a basic score, which is the fraction of objects containing c in the given textual feature (title or description) that also contain c as a tag. The authors then use a *leading precision rescorer* for weighting the different candidate sources, thus producing new scores for each term c. These new scores are then merged in a probabilistic sum. Terms extracted from the title (first step) as well as from the title-description merge are then expanded through association rules[5]. CTTR distinguishes two types of co-occurrences: (1) between tags, in which the antecedents are tags in the objects of the training set, and (2) between terms in the title of an object and its tags, in which the antecedents are terms in the titles of such objects. At recommendation time, the sets of rules related to the extracted terms are combined using corresponding scores. As a final step, the scores obtained from the association rules and from the title and description of the target objects are rescored once again, and merged, resulting in the final ranking. We omit a detailed description of CTTR due to space constraints.

4.2 Our New Heuristics

Our new heuristics extend the Sum^+ and $LATRE$ baselines to also include one of the four metrics of descriptive power, i.e., TF, TS, wTF or wTS (Section 3.3). We thus propose eight new ranking functions composed by a weighted linear combination of the output of Sum^+ (or of $LATRE$) and one of the four metrics. Let DP be the selected metric, and c be a candidate term for target object o. The proposed heuristics have the following general structures:

$$Sum^+DP(c,o,k_x,k_c,k_r,\alpha) = \alpha Sum^+(c,o,k_x,k_c,k_r) + (1-\alpha)DP(c,o) \quad (10)$$

$$LATRE+DP(c,o,\ell,\alpha) = \alpha Sum(c,o,\ell) + (1-\alpha)DP(c,o) \quad (11)$$

Parameter α ($0 \leq \alpha \leq 1$) is used as a weighting factor. Note that Sum^+ and *Sum* are computed only over candidates generated from the association rules, whereas DP (i.e., TS, TF, wTS or wTF) is computed for terms extracted from other textual features of target object o.

4.3 Learning-to-Rank Based Strategies

We here investigate the potential benefits of applying L2R techniques to the tag recommendation problem. The basic idea is to use such algorithms to "learn" a *good ranking function* based on a list $L_{metrics}$ of metrics of relevance. We here consider both the traditional RankSVM method (Section 4.3.1) and the Genetic Programming (GP) framework (Section 4.3.2). In both cases, $L_{metrics}$ includes *Sum*, IFF, *Stab*, TS, TF, wTS, wTF, H^{tags} and Sum^+, defined in Eqs. 1-9. In particular, we include *Sum* with ℓ=3, as used by the $LATRE$ baseline. We also include in $L_{metrics}$ two other functions, called $Vote$ and $Vote^+$, proposed in [22]. For a given candidate c and target object o, $Vote(c,o)$ is the number of association rules whose antecedent is a term in $\mathcal{I}_o$ and whose consequent is the candidate term c. $Vote^+$, like Sum^+, is built from $Vote$ by weighting each vote by the Stabilities of both antecedent and consequent. That is:

$$Vote(c,o) = \sum_{x\in\mathcal{I}_o} j, \text{ where } j = \begin{cases} 1, & \text{if}(x\rightarrow c)\in\mathcal{R} \\ 0, & otherwise \end{cases} \quad (12)$$

$$Vote^+(c,o,k_x,k_c,k_r) = \sum_{x\in\mathcal{I}_o} j \times Stab(x,k_x) \times Stab(c,k_c) \times Rank(c,x,k_r), \text{ where } j = \begin{cases} 1, & if \quad x\rightarrow c\in\mathcal{R} \\ 0, & otherwise \end{cases} \quad (13)$$

For both proposed strategies, the candidate tags $\mathcal{C}_o$ for each object o include all terms generated by $LATRE$ and

[5]We here apply the *Apriori* method.

all terms extracted from other textual features. For each candidate $c \in \mathcal{C}_o$, for each object o, we compute all metrics in $L_{metrics}$ using the training set $\mathcal{D}$ (e.g., for $Stab$, IFF) and the textual features associated with o. Each candidate c is then represented by a vector of metric values $M_c \in \mathbb{R}^m$, where m is the number of considered metrics. We also assign a binary label Y_c to each candidate c for each object v in validation set $\mathcal{V}$, indicating whether c is a relevant recommendation for v (Y_c=1) or not (Y_c=0), based on the contents of $\mathcal{Y}_v$. Note that we use training set $\mathcal{D}$ only to extract the association rules and to compute the metrics, relying on validation set $\mathcal{V}$ to learn the solutions (see discussion in Section 5.2). Thus, we only assign labels Y_c for objects in $\mathcal{V}$. The learned model, i.e., ranking function $f(M_c)$, is then applied to each candidate c for each object o in the test set. All reported results are based on the performance on the test sets. Next, we describe how we applied RankSVM and GP to the tag recommendation problem.

4.3.1 RankSVM Based Strategy

RankSVM is based on the state-of-the-art Support Vector Machine (SVM) classification method [13]. We use the SVM-rank tool[6] to learn a function $f(M_c)$=$f(W, M_c)$, where $W = <w_1, w_2, ...w_m>$ is a vector of weights associated with the considered metrics (i.e., $W \in \mathbb{R}^m$). W is learned by a maximum-margin optimization method that tries to find a hyperplane, defined by W, that best separates the "closest" candidate tags (represented by their metric vectors in $\mathbb{R}^m$) belonging to two different *levels of relevance* (i.e., relevant and irrelevant) assigned to each object-candidate pair in the training. They are employed to produce ranking statements (i.e., relevant tags must precede irrelevant ones), which in turn are used as input to the RankSVM learning process. At recommendation time, $f(M_c)$ is used to rank all candidates for target object o according to the their relative distances to the separating hyperplane.

RankSVM has two main parameters, namely, the type of kernel function, which indicates the structure of the solution function, and cost j, which controls the penalty given to classification errors in the training process. RankSVM is still a very competitive performer when considering average results across all collections in the LETOR 3.0 L2R benchmark[7], besides being readily available for experimentation, unlike other methods, which are proprietary.

4.3.2 GP Based Strategy

We apply GP to find a good ranking function $f(M_c)$ for tagging recommendation purposes. This is done through the evolution of a population in which each individual represents a different ranking function. This evolution is inspired on the biological mechanisms of natural selection and survival of the "strongest" or most "adapted" individuals. Next, we provide an overview of a GP algorithm, and introduce how we model the tag recommendation problem with it.

Overview of the GP Framework

Genetic Programming (GP) implements a global search mechanism. A GP algorithm evolves a population of tree-represented individuals (i.e., the solutions for the problem at hand), created from a set of terminals and functions related to the target problem. In each generation of the evolutionary process, individuals are evaluated according to a specific quality metric, calculated by a *Fitness* function. That is, the Fitness value is used as a criterion to select the best individuals, which will transmit their characteristics to the future generations through operations like crossover and mutation. We here implement the *tournament selection* method, which randomly chooses, with replacement, k individuals from the population and takes the one with highest Fitness value.

While the number of new individuals is smaller than the desired population size n, two individuals are picked through the adopted selection method, and, with probability p_c, have their "genetic material" exchanged in the *crossover* operation to generate a new individual. That is, we randomly choose one node of each of the two trees (the two individuals) and exchange the subtrees below them. The role of the crossover operation is to combine good solutions towards the most promising solutions in the search space. Moreover, with probability p_m, the *mutation* operation is performed to introduce new individuals (i.e., solutions) in the population, increasing its diversity. This is useful, for example, to avoid that the whole evolutionary process gets trapped in local minima during the search process. Here, the mutation of an individual (i.e., a tree) is performed by first randomly selecting one of its nodes, and then replacing it (and its corresponding subtree) by a new randomly generated subtree, without exceeding a maximum tree depth d.

The whole process is repeated until a target Fitness value f^{target} or a maximum number of generations g is reached. At the end, the individual with best fitness value, which usually belongs to the last generation in the evolutionary process, is chosen as the final solution for the problem.

GP is an effective non-linear method that has been successfully employed when there is a large search space and a goal to be optimized, having produced results close to the optimal in many applications [3]. Indeed, GP has been applied to various Information Retrieval tasks such as classification, search/ranking and image retrieval (e.g., [27]). However, to our knowledge, we are the first to apply it to tag recommendation. Our motivations are twofold. First, in comparison with RankSVM, GP exploits a different learning paradigm, which directly optimizes a target function (e.g., recommendation precision). Second, GP has been demonstrated to be competitive with other learning-to-rank techniques such as RankSVM itself and RankBoost [27, 8][8]. As we shall see, GP outperforms RankSVM in a few scenarios being statistically tied with it in many others.

GP Applied to Tag Recommendation

To apply GP to tag recommendation, we need to define the tree representation of an individual, which, in our case, is a tag ranking function, and a Fitness function. A tree is composed of terminals (leaves) and non-terminals (internal nodes). We choose the sum ($+$), subtraction ($-$), multiplication ($\times$), division ($/$) and natural logarithm (ln) operations as non-terminals. The terminals are composed of variables and constants (uniformly distributed between 0 and 1). The variable set includes the (values of the) metrics in the given $L_{metrics}$ list, which, for a candidate c, is given by vector M_c. To ensure the closure property, we implement protected di-

[6] http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html

[7] http://research.microsoft.com/en-us/um/beijing/projects/letor

[8] In some initial tests with RankBoost and with an associative method for learning-to-rank [24], we found them to be very inefficient to train for the size of our collections.

vision and logarithm, such that these operators return the default value 0 when their inputs are out of their domains. As an example, a tree representing function $Sum+0.7\times TS$ contains operation "+" as root, with variable "Sum" and operation "×" as its children. Internal node "×", in turn, has constant "0.7" and variable "TS" as children.

The Fitness of an individual in this context represents the quality of the recommendations produced by the corresponding ranking function, which is here assessed in terms of the precision of the top-k terms in the ranking of recommended terms[9]. Given a ranking function $f(M_c)$, let $\mathcal{Y}_o$ be the set of relevant tags for an object o (identified by labels Y_c), $\mathcal{R}_o^f$ be the ranked recommendations produced by $f(M_c)$ for o, and $\mathcal{R}_{k,o}^f$ the top k elements of $\mathcal{R}_o^f$. The quality of the recommendations produced by f for o is measured as:

$$P@k(\mathcal{R}_o^f, \mathcal{Y}_o, f) = \frac{|\mathcal{R}_{k,o}^f \cap \mathcal{Y}_o|}{min(k, |\mathcal{Y}_o|)} \quad (14)$$

The Fitness (i.e., quality) of ranking function $f(M_c)$ is then computed as the average $P@k$ over all recommendations produced by $f(M_c)$ in a sample of objects of size s, extracted from validation set $\mathcal{V}$.

5. EXPERIMENTAL EVALUATION

In this section, we first present the data sets used to evaluate the tag recommendation strategies (Section 5.1) as well as our evaluation methodology (Section 5.2). We then describe how we parameterized each strategy (Section 5.3), and discuss a set of representative results (Section 5.4).

5.1 Data Collections

We evaluate the tag recommendation methods on three datasets, each containing the *title*, *tags* and *description* associated with real objects from LastFM, YouTube and YahooVideo. The LastFM and YouTube datasets were collected in August 2009, following a *snowball sampling*. That is, starting from a set of users (the most popular users) selected as seeds, the crawler recursively collects the objects posted by them and follows their social links to other users, collecting the objects posted by them. Our datasets include the textual features associated with 2,758,992 LastFM artists and with more than 9 million YouTube videos. The YahooVideo dataset was also gathered by snowball sampling, but using the most popular objects as seeds and following links of related videos[10]. It was gathered in October 2008, and contains the features of 160,228 objects. The YahooVideo dataset was the same used in [4], whereas the other two were collected more recently and are significantly larger than the collections analyzed in that work[11].

We considered only objects with textual features in English, and used the Porter Stemming algorithm[12] to remove the affixes of each word in each collected feature. Stemming was used to avoid trivial recommendations such simple variations of the same word (e.g., plural). We also removed stopwords and terms that are either too frequent (with more than 100,000 occurrences) or too rare (with fewer than 30 occurrences), as such terms are poor recommendations [22].

Table 1: Object Subsets.

Dataset	Smallest Interval		Medium Interval		Largest Interval	
	#tags/ object	#objs	#tags/ object	#objs	#tags/ object	#objs
LastFM	2-4	63,973	5-6	93,080	7-507	78,439
Yahoo	2-6	43,353	7-11	54,143	12-52	49,372
YouTube	2-5	2.2mi	6-9	1.7mi	10-77	1.6mi

5.2 Evaluation Methodology

Similarly to [9, 20, 19, 10], we adopt a fully-automated evaluation methodology: we use a subset of the object's pre-assigned tags as an *expected answer* for the recommendation, that is, as the relevant tags for that object. As such, these tags are disregarded for the computation of metrics of relevance. This methodology was adopted because the manual evaluation of tag relevance is a very expensive process that may be affected by the subjectivity of human judgments. We note that the results obtained according to the proposed methodology represent lower bounds, as some of the recommended tags, although not in the expected answer, might still be considered relevant for the given object.

Following the proposed methodology, for each object $o = \langle \mathcal{I}_o, \mathcal{F}_o, \mathcal{Y}_o \rangle$ in the test and validation sets, we randomly select half of its tags to be included in $\mathcal{I}_o$. The other half are included in $\mathcal{Y}_o$, the expected answer for o. We also use title and description as textual features in $\mathcal{F}_o$, for each object o.

We use the $P@k$ metric (Eq. 14) with k=5 as the primary metric to evaluate all recommendation strategies. We also measure the *recall* and the *mean average precision (MAP)* of the recommendations. Recall is the fraction of the set of relevant tags for an object that were indeed recommended, whereas MAP considers the order in which tags are recommended, emphasizing ranking relevant tags higher [2].

We evaluate each recommendation method on each dataset in two scenarios. In the former, we divide each dataset into three object subsets based on the number of tags available in the object, so that each subset contains roughly the same number of objects, as shown in Table 1. We refer to the subsets as the *smallest*, *medium* or *largest* interval, referring to the range of number of tags per object in each of them. We evaluate each method separately for each subset, thus assessing its effectiveness under different amounts of training data. In a second, *mixed* scenario, we consider each dataset entirely, thus building a solution for a more heterogeneous object set. Thus, in the first scenario, we compute tag co-occurrence patterns and other metrics of relevance, "learn" solutions and perform parameter tuning for each subset separately, thus building more specialized recommendation functions. In the mixed scenario, a single function is built for all objects, with obviously lower cost.

For the first scenario, we randomly sample 50,000 objects from each subset (40,000 for YahooVideo). For the second scenario, we use a sample consisting of the combination of all three subset samples used in the first scenario. In both cases, each sample is divided into five equal-sized portions, which are used in a five-fold cross validation. That is, three

[9]We also experimented with other Fitness functions (e.g., Mean Average Precision - MAP) obtaining similar results.

[10]We adopted a slightly different sampling strategy for LastFM and YouTube, exploiting users and social links as opposed to videos and related video links, as we intend to use the collected datasets to evaluate personalized recommendation strategies in the future. As YahooVideo does not publish per-user information on tag assignment, we chose not to crawl that application again, thus relying on our previously gathered dataset.

[11]http://vod.dcc.ufmg.br/recc/

[12]http://tartarus.org/ martin/PorterStemmer/

Table 2: Selected Values for Subset of Parameters of Analyzed Recommendation Methods.

Parameter	LastFM				YahooVideo				YouTube			
	Mixed	Smallest	Medium	Largest	Mixed	Smallest	Medium	Largest	Mixed	Smallest	Medium	Largest
σ_{min}	2	2	2	2	2	2	2	2	1	1	1	1
θ_{min}	0.2	0.01	0.1	0.4	0.2	0.01	0.2	0.3	0.1	0.01	0.1	0.2
$\alpha(Sum^+DP)$	0.95	0.95	0.95	0.95	0.8	0.8	0.8	0.7	0.8	0.8	0.8	0.7
$\alpha(LATRE{+}DP)$	0.99	0.95	0.95	0.99	0.9	0.8	0.9	0.95	0.9	0.7	0.8	0.95

portions are treated as training set ($\mathcal{D}$), which is used for extracting association rules and computing all metrics. A fourth portion is used as validation set $\mathcal{V}$, which, in turn, is used to "learn" the solutions (i.e., to compute the Fitness function in the GP evolutionary process, and to learn vector W in RankSVM) and to tune parameters of all recommendation methods (including the RankSVM based strategy, using cross-validation in $\mathcal{V}$). The last portion is used for testing.

Note that, our use of the 5-fold cross-validation process for both L2R based strategies is slightly different from the traditional use: we learn the ranking function and select parameter values in the validation set $\mathcal{V}$. We do so to avoid overfitting, which could occur if the solutions were learned in the same set from which association rules and metrics were extracted (i.e., training set $\mathcal{D}$), as metrics derived from the rules could be over-inflated[13]. We argue that our experimental design is fair because: 1) we do not use any privileged information from the test set in which results of all methods are reported; 2) all parameters are discovered in the same validation set; 3) our collections are large enough for effective learning, even when we conduct cross-validation in $\mathcal{V}$ for parameterization (e.g., results in these preliminaries experiments are basically identical to the ones reported in the test with the discovered parameters); and 4) the very tight confidence intervals reported in our results (Section 5.4) are evidence of low variation and thus learning convergence. Moreover, having a large amount of training data to generate the tag co-occurrence rules can help increasing the coverage of the rules (i.e., more co-occurrences can be potentially found), thus generating more candidates and more precise metric values. This benefits all methods.

5.3 Parameterization

Our evaluation starts with a series of experiments with the validation set $\mathcal{V}$ to find the best parameters of each method. The results of such experiments are omitted due to space constraints. In summary, we found that, for both Sum^+ and Sum^+DP (for DP equal to TS, TF, wTS and wTF), the best parameters are k_r=k_x=k_c=5. Best results for α varied between 0.7 and 1.0. We search for the aforementioned parameters sequentially, varying one of them at each time and fixing the others. For $LATRE$, $LATRE{+}DP$ and L2R based strategies, we set ℓ=3, as in [19]. We also set k_s, parameter of the $Stab$ metric, equal to 5. Parameters σ_{min} and θ_{min} directly impact the number of association rules generated, thus affecting the processing time of the recommender. We look for a good trade off between processing time and recommendation precision. The lower σ_{min} (or θ_{min}), the larger the number of rules, thus, the longer the processing time. In most cases, precision decreases as σ_{min} and θ_{min} increase, particularly when fewer tags are provided, as the number of generated candidates is already small in this case. However, when the number of input tags is larger, the choice of thresholds can be more aggressive with no significant impact on precision. Thus, we chose σ_{min} and θ_{min} so that the precision loss, with respect to results for σ_{min}=θ_{min}=0, is under 3%. Table 2 shows the selected values of some parameters. The chosen values for σ_{min} and θ_{min} are the same for all co-occurrence based methods.

For RankSVM, we tested two kernel functions, Linear and Radial Basis (RBF), choosing the former as the latter did not scale to our data sets. Using cross-validation in $\mathcal{V}$, we also varied cost j between 10^{-3} and 10^3, finding that the best choice was j=100, in most cases. We also tried different strategies to normalize our feature vectors, including L2-norm, z-score and the LETOR normalization procedure [17], with no improvements. Thus, the results reported here refer to non-normalized data. For the GP based strategy, we experimented, using $\mathcal{V}$, with population sizes n equal to 50, 100, and 200, selecting n=200, as the larger population allows a greater coverage of the solution space, leading to better results. For this population size, the algorithm converges (i.e., Fitness values stop improving) before 200 generations, value assigned to g. We fixed k=2, and set d=7, p_c=0.6 and p_m=0.1, as usually done in the literature [3]. Computing the Fitness during the evolutionary process over all validation objects can be infeasible. Thus, we used a sample of s=500 of those objects, as this was enough to learn functions that are more effective than our heuristics.

5.4 Representative Results

We now discuss the most relevant results of our new tag recommendation methods (8 heuristics and 2 L2R based strategies), comparing them against the 3 baselines. Table 3 shows P@5 results for all methods, datasets and scenarios. Recall and MAP results are omitted due to space constraints. We comment on them whenever appropriate.

All reported results are averages over 5 folds (test sets). For the GP-based strategy, which is stochastic, each experiment is repeated 5 times. Thus, results are averages over 25 runs (5 folds, 5 seeds). Table 3 also shows 95% confidence intervals, indicating that, with that confidence, results deviate from the reported means by up to 3%. For each dataset, the table is broken into 3 blocks: baselines, new heuristics and L2R based methods. Best results and statistical ties (according to a *2-sided t-test* with p-value < 0.05), within each block are shown as shaded entries. Best overall results (and statistical ties) are shown in bold.

We start with a general finding. The improvements obtained with our methods over the baselines are more modest in the LastFM dataset. This is possibly due to two factors: (1) there tends to be less overlap between the contents of title, description and tags associated with the same object on LastFM [7], which leads to a greater concentration of TS (and wTS) around small values, making it difficult to distinguish "good" from "bad" terms using these metrics; and (2) the number of tags per object tends to be smaller in our

[13]We did observe this overfitting problem in initial tests, with the learned solutions having lower generalization capabilities.

LastFM dataset (e.g., 48% and 73% of our YahooVideo and YouTube objects have fewer than 10 tags, against 88% of LastFM objects). These factors limit the benefits from using TS and wTS and from exploiting co-occurrence patterns among pre-assigned tags in that dataset.

We now turn our attention to the relative performance of specific methods, starting with the baselines. Consistently with [19], we find that $LATRE$ outperforms Sum^+ (up to 25% in P@5, 33% in recall, and 31% in MAP) in most cases. However, $CTTR$ does appear as a good alternative to $LATRE$ in a few cases, particularly on YouTube (gains in P@5, recall and MAP of up to 111%, 63% and 75%).

Considering the best baseline in each scenario and dataset, we find that our heuristics produce gains in P@5 of as much as 40%, and in recall and MAP of up to 32% and 62%, respectively. Thus, introducing a metric of descriptive power can greatly improve recommendation effectiveness, particularly for objects with smaller number of tags (smallest interval). This is because fewer available tags restrict the benefits from exploiting tag co-occurrence. In such cases, our heuristics greatly benefit from also exploiting descriptive metrics.

Comparing each new heuristic with the original method on which it was based (Sum^+ or $LATRE$), we find that our heuristics provide gains of up to 181% in P@5, 116% in recall, and 186% in MAP (e.g, Sum^+wTS over Sum^+ on the smallest interval of YouTube). As discussed, the gains for LastFM are more modest (e.g., up to 4.6% in P@5). In comparison with $CTTR$, the gains in P@5, recall and MAP reach 98%, 32% and 96%, respectively, indicating great benefits from exploiting co-occurrences with tags previously assigned to the target object as well as descriptive metrics.

Among the new heuristics, the most promising one is $LATRE{+}wTS$, as it yields the best results in most cases. To reach this conclusion, we make two observations. First, for any given descriptive metric DP (i.e., TS, TF, wTS or wTF), $LATRE{+}DP$ outperforms Sum^+DP in most cases (up to 15% in P@5, 20% in recall, and 15% in MAP), confirming the benefits of exploiting more complex association rules. In the few cases where Sum^+DP outperforms $LATRE{+}DP$, the gains in P@5 are under 3%. This happens specifically in LastFM, where, in general, the differences between Sum^+DP and $LATRE{+}DP$ are smaller, possibly due to the smaller number of tags per object which restricts the number of more complex rules that can be mined. Conversely, the improvements of $LATRE{+}DP$ over Sum^+DP tend to be higher for objects with larger number of tags, particularly on YouTube and YahooVideo.

Our second observation is that, comparing all four descriptive metrics, wTS tends to yield the best results (or close to them), often followed by wTF, TS and TF. In particular, the use of wTS in $LATRE{+}wTS$, as opposed to the traditional TF metric, leads to gains of up to 8.3% in P@5. This is mainly because wTS considers that objects are composed of different features which may have different descriptive capacities. TF, in turn, tends to favor very frequent terms, even those appearing in a single feature, which are often less relevant than terms appearing in multiple features.

Turning our attention to the L2R based strategies, both methods provide some further improvements. For instance, the gains in P@5 and MAP obtained with the RankSVM based strategy over $LATRE{+}wTS$ can reach 4.8% and 3.2%, respectively. With the GP based strategy, corresponding gains reach 12% and 10% (e.g., largest interval on LastFM).

Table 3: Average P@5 Results and 95% Confidence Intervals: Best Results Within Each Block (Baselines, Heuristics, L2R based Strategies) in Shaded Entries. Best Overall Results in Bold.

LastFM

Strategy	Mixed	Smallest	Medium	Largest
Sum+	0.411±.001	.454±.003	.433±.004	.391±.001
LATRE	0.405±.001	.465±.004	.457±.005	.384±.004
CTTR	0.260±.001	.285±.003	.267±.002	.217±.002
Sum+TF	0.417±.001	.470±.004	.436±.004	.395±.001
Sum+TS	0.418±.002	.472±.005	.439±.004	.396±.001
Sum+wTF	0.417±.001	.470±.005	.436±.004	.395±.001
Sum+wTS	0.417±.002	.470±.005	.438±.004	.396±.001
LATRE+TF	0.408±.001	.479±.004	.459±.005	.385±.004
LATRE+TS	0.411±.001	.487±.005	.466±.005	.388±.003
LATRE+wTF	0.408±.001	.479±.004	.459±.005	.385±.004
LATRE+wTS	0.411±.001	.486±.005	.465±.005	.388±.003
GP-based	**0.433±.003**	**.500±.005**	**.476±.002**	**.434±.003**
SVM-based	0.411±.002	**.499±.007**	.450±.005	.393±.003

YahooVideo

Strategy	Mixed	Smallest	Medium	Largest
Sum+	0.484±.003	.453±.007	.511±.004	.615±.004
LATRE	0.608±.003	.525±.006	.619±.004	.731±.006
CTTR	0.465±.004	.649±.005	.407±.004	.394±.003
Sum+TF	0.643±.003	.755±.002	.597±.003	.663±.005
Sum+TS	0.674±.003	.764±.003	.631±.003	.714±.004
Sum+wTF	0.666±.002	.784±.003	.612±.003	.673±.004
Sum+wTS	0.707±.002	.795±.004	.660±.003	.721±.004
LATRE+TF	0.698±.002	.781±.002	.673±.004	.745±.007
LATRE+TS	0.716±.003	.785±.004	.710±.003	.778±.007
LATRE+wTF	0.729±.002	.821±.004	.706±.003	.754±.006
LATRE+wTS	0.733±.003	.818±.005	.729±.003	.780±.007
GP-based	**0.743±.007**	**.822±.004**	**.734±.003**	**.789±.006**
SVM-based	**0.752±.002**	**.826±.003**	.725±.003	**.800±.005**

YouTube

Strategy	Mixed	Smallest	Medium	Largest
Sum+	0.245±.002	.211±.006	.212±.003	.282±.004
LATRE	0.285±.004	.219±.005	.242±.005	.326±.006
CTTR	0.376±.002	.463±.005	.337±.004	.269±.002
Sum+TF	0.462±.001	.552±.004	.421±.006	.403±.003
Sum+TS	0.475±.002	.560±.004	.433±.005	.419±.004
Sum+wTF	0.490±.002	.583±.003	.451±.005	.416±.004
Sum+wTS	0.502±.003	.593±.003	.461±.005	.431±.004
LATRE+TF	0.472±.002	.557±.004	.436±.004	.425±.006
LATRE+TS	0.467±.003	.561±.004	.445±.004	.441±.007
LATRE+wTF	0.503±.003	**.596±.004**	.464±.004	.439±.005
LATRE+wTS	0.489±.003	.593±.003	.471±.005	.450±.007
GP-based	0.507±.001	**.595±.003**	.477±.002	**.462±.003**
SVM-based	**0.512±.002**	**.601±.005**	**.484±.005**	**.468±.003**

Thus, improvements are, in general, somewhat modest, although they can be more significant in a few specific cases. In terms of recall, results are the same as the three strategies generate the same candidates for each object.

Such overall modest improvements are due to the fact that both GP and RankSVM, in spite of receiving a list with several metrics, are mostly exploiting those used by $LATRE{+}wTS$. Indeed, we verified that the most frequent metric used by the best function produced by the GP process at its final generation is $Sum(c, o, 3)$ (also used by LATRE). Considering all datasets and scenarios, $Sum(c, o, 3)$ occurs in 95% of the generated functions. The next most frequent metric is wTS, which occurs in 74% of the functions. IFF comes next, occurring in 49% of the functions. All other metrics appear in less than 25% of the functions. Similarly, we also analyzed the weight vector W learned by RankSVM, finding that the largest weights, on average, are indeed associated with $Sum(c, o, 3)$ and wTS.

Although the L2R strategies yield only modest gains over our best heuristic, we note that they provide flexible frame-

works that can be easily extended to incorporate new metrics and to exploit other aspects of the tag recommendation problem (e.g., personalization). We also note that, after the offline training step, the use of the functions generated by either GP or RankSVM, at recommendation time, does not incur in significant extra processing time in relation to the best heuristic. One possible source of delay for all of them is the on-demand generation of association rules by $LATRE$. However, as shown in [19], $LATRE$'s average processing time is well-suited for real-time recommendation.

The best L2R based strategy depends on the dataset. On LastFM, the GP based method leads to somewhat better results (up to 10% in $P@5$ for the largest interval), whereas on YahooVideo and YouTube, there is a statistical tie in most object sets with a slight advantage for either method over the other in a few cases. Comparing both solutions in terms of training time on our datasets, we also find that the most efficient technique depends on the dataset. For instance, the GP based method requires, on average, 63.5, 50.4 and 30.6 minutes to train on the LastFM, YouTube and YahooVideo datasets, respectively, running on an Intel 2.83GHz Quad-Core with 8GB RAM. Corresponding numbers for RankSVM are 100.8, 28.3 and (surprisingly) 0.81 minutes. We also note that, except on the YahooVideo dataset, RankSVM's training time exhibits a much higher variability across different folds than GP. We computed 95% confidence intervals for the training times measured over a fixed number of folds, and observed that GP training times deviate from the means by up to 10%, whereas RankSVM results deviate by as much as 48%. In other words, RankSVM's training time seems much more sensitive to the inherently difficulties of the input data.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed several new tag recommendation strategies that jointly exploit term co-occurrence with pre-assigned tags, multiple textual features and metrics of relevance. We proposed and evaluated eight new heuristics and two learning-to-rank based strategies, comparing them against three state-of-the-art techniques, in various datasets and scenarios. Our results indicate that our best heuristic, $LATRE{+}wTS$ provides gains of up to 40% in terms of precision, 32% in recall and 62% in MAP, over the best baseline in any analyzed scenario. Some further improvement (up to 12% in precision) can also be achieved with the RankSVM and GP based methods, although the best learning-to-rank strategy depends on the specific dataset. These results illustrate the benefits of jointly exploiting the three aforementioned dimensions, particularly metrics of descriptive power. Moreover, the use of learning-to-rank techniques arise as a promising solution, as they yield very competitive results while still being quite flexible, allowing the easy inclusion of new metrics and the extension of the scope of the problem.

Future work includes a manual evaluation of our solutions and extensions to address personalized recommendation.

7. REFERENCES

[1] R. Agrawal and R. Srikant. Fast Algorithms For Mining Association Rules. In *VLDB*, 1994.

[2] R. Baeza-Yate and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming – An Introduction On The Automatic Evolution Of Computer Programs And Its Applications*. Morgan Kaufmann, 1998.

[4] F. Belém, E. Martins, J. Almeida, M. Gonçalves, and G. Pappa. Exploiting Co-occurrence and Information Quality Metrics to Recommend Tags in Web 2.0 Applications. In *CIKM*, 2010.

[5] S. Boll. MultiTube–Where Web 2.0 And Multimedia Could Meet. *IEEE MultiMedia*, 14(1), 2007.

[6] H. Cao, M. Xie, L. Xue, C. Liu, F. Teng, and Y. Huang. Social Tag Prediction Based On Supervised Ranking Model. In *PKDD*, 2009.

[7] F. Figueiredo, F. Belem, H. Pinto, J. Almeida, M. Gonçalves, D. Fernandes, E. Moura, and M. Cristo. Evidence Of Quality Of Textual Features On The Web 2.0. In *CIKM*, 2009.

[8] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An Efficient Boosting Algorithm For Combining Preferences. *Journal Of Machine Learning Research*, 2003.

[9] N. Garg and I. Weber. Personalized, Interactive Tag Recommendation For Flickr. In *RecSys*, 2008.

[10] Z. Guan, J. Bu, Q. Mei, C. Chen, and C. Wang. Personalized Tag Recommendation Using Graph-Based Ranking On Multi-Type Interrelated Objects. In *SIGIR*, 2009.

[11] P. Heymann, D. Ramage, and H. Garcia-Molina. Social Tag Prediction. In *SIGIR*, 2008.

[12] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt - Thieme, and G. Stumme. Tag Recommendations In Folksonomies. In *PKDD*, 2007.

[13] T. Joachims. Training Linear SVMs In Linear Time. In *KDD*, 2006.

[14] G. Koutrika, F. A. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina. Combating Spam In Tagging Systems: An Evaluation. *ACM Trans. Web*, 2(4), 2008.

[15] R. Krestel, P. Fankhauser, and W. Nejdl. Latent Dirichlet Allocation For Tag Recommendation. In *RecSys*, 2009.

[16] M. Lipczak, Y. Hu, Y. Kollet, and E. Milios. Tag Sources For Recommendation In Collaborative Tagging Systems. In *PKDD*, volume 497, 2009.

[17] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR*, 2007.

[18] H. Ma, I. King, and M. Lyu. Learning To Recommend With Social Trust Ensemble. In *SIGIR*, 2009.

[19] G. Menezes, J. Almeida, F. Belém, M. Gonçalves, A. Lacerda, E. Moura, G. Pappa, A. Veloso, and N. Ziviani. Demand-Driven Tag Recommendation. In *PKDD*, 2010.

[20] S. Rendle and S. Lars. Pairwise Interaction Tensor Factorization For Personalized Tag Recommendation. In *WSDM*, 2010.

[21] S. Siersdorfer, J. San Pedro, and M. Sanderson. Automatic Video Tagging Using Content Redundancy. In *SIGIR*, 2009.

[22] B Sigurbjornsson and R. van Zwol. Flickr Tag Recommendation Based On Collective Knowledge. In *WWW*, 2008.

[23] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W. Lee, and C. Giles. Real-Time Automatic Tag Recommendation. In *SIGIR*, 2008.

[24] A. Veloso, H. M. de Almeida, M. A. Gonçalves, and W. Meira Jr. Learning To Rank At Query-Time Using Association Rules. In *SIGIR*, 2008.

[25] J. Wang, L. Hong, and B. D. Davison. Tag Recommendation Using Keywords And Association Rules. In *PKDD*, 2009.

[26] L. Wu, L. Yang, N. Yu, and X. Hua. Learning To Tag. In *WWW*, 2009.

[27] J. Yeh, J. Lin, H. Ke, and W. Yang. Learning To Rank For Information Retrieval Using Genetic Programming. In *SIGIR 2007 Workshop: Learning To Rank For Information Retrieval*, 2007.