Personalised Tag Recommendation

Nikolas Landia
University of Warwick
Coventry CV4 7AL
UK
N.Landia@warwick.ac.uk

Sarabjot Singh Anand
University of Warwick
Coventry CV4 7AL
UK
S.S.Anand@warwick.ac.uk

ABSTRACT

Personalised tag recommenders are becoming increasingly important since they are useful for many document management applications including social bookmarking websites. This paper presents a novel approach to the problem of suggesting personalised tags for a new document to the user. Document similarity in combination with a user similarity measure is used to recommend personalised tags. In case the existing tags in the system do not seem suitable for the user-document pair, new tags are generated from the content of the new document as well as existing documents using document clustering. A first evaluation of the system was carried out on a dataset from the social bookmaking website, Bibsonomy[1]. The results of this initial test indicate that adding personalisation to an unsupervised system through our user similarity measure gives an increase in the precision score of the system.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing Methods*; I.5.3 [**Pattern Recognition**]: Clustering

General Terms

Algorithms

Keywords

tag recommendation, tag extraction, social bookmarking

INTRODUCTION

With the advancement of Web 2.0 and social bookmarking websites, the recommendation of tags for these systems is becoming increasingly important. The major difficulties with tag recommendation for social bookmarking are that tags are related to specific users and the documents, users and tags might be unknown to the recommender system. Suggested tags have to be personalised and the system has to be able to successfully recommend tags for scenarios where the document, user and/or appropriate tag are not in the training set.

Existing approaches to automated tagging include supervised as well as unsupervised tag recommender systems. Classifiers [2] and clustering [5] have been used when a set of predefined tags is known and the task is to assign these tags to new documents. However, these supervised systems do not have a solution to the new tag problem, akin to the new item problem in recommender systems. Clustering can be used to generate tags from the vocabulary of the document set and is thus able to overcome the new tag problem. However, tags from a vocabulary different from that of the document's authors will not be recommended. The supervised and unsupervised techniques on their own ignore information about users that may be available to the system. People differ in their interests (documents/topics), vocabulary (tags) and context. In order to successfully recommend tags to users, vocabulary and tagging habits have to be taken into consideration.

In this paper we suggest a solution to this problem that consists of clustering the existing documents in order to identify sets of similar documents which in turn identifies the set of users whose tags may be propagated to the current target user-document pair. A list of potential tags for the new user-document pair is obtained from the tags of the similar documents. A score is then calculated for each potential tag by taking a weighted combination of the similarity of the document that the tag is assigned to and the similarity of the user who assigned it, averaged over all posts the tag appears in. If the score of a tag is below a set threshold, it means that this tag is unsuitable for the user-document pair. When the number of suitable tags is below a predefined number t, new tags are generated from the content of the document.

DOCUMENT CLUSTERING

There are various techniques that can be used to cluster documents. The clustering approach presented by Song et al. [5] takes into consideration document words and associated tags. Song's approach clusters documents into a predefined number of clusters nc. When finding the cluster for a new document, each of the nc clusters is considered and a cluster membership vector is generated for the new document. The labelling of a document thus has a linear time complexity $O(nc)$. In our unsupervised approach the number of clusters is defined by the data and indirectly by the number of tags w considered in document clustering. When finding the cluster for the new document, the whole tree of clusters is considered, so that a document dealing with a specialised topic is assigned to a leaf node while a document dealing with a more general, broad topic is assigned to a cluster higher up the tree. Nevertheless, the labelling of documents in our system has a time complexity of $O(\log nc)$ as explained later. The main advantage of our

[1]http://www.bibsonomy.org/

approach is that it can find the level of specialisation of a document in a topic area and the potential set of tags for the new document reflect its level of specialisation. Moreover, the system is able to generate new tags.

The algorithm proposed here is a two part unsupervised document clustering method consisting of a divisive and an agglomerative clustering part. The aim of the algorithm is to organise documents into reasonable clusters and define a predefined number of **unpersonalised** tags, *w,* for the documents based on the clusters they belong to. Note that these w tags will be generated from words which appear in the word corpus of the document dataset.

In the first stage of clustering (the divisive part), documents are clustered using bisecting K-Means [6], for which different initialisation techniques were examined. Division of clusters is performed until the documents in a cluster satisfy the condition that they share at least s of their w most important words (defined by their tf-idf-score) within the cluster. The second clustering stage (the agglomerative part) takes the final set of clusters generated by the divisive part and merges them hierarchically to create a tree structure of clusters.

The document representation used for clustering is bag-of-words. Stop-words are removed and the dimensionality of the data set is reduced to n using document frequency, which proved to be an effective and cheap technique for dimensionality reduction in the experiments carried out by Yang and Pedersen [7].

Divisive Part

1. define the number of tags w required to be assigned to each document. This parameter, along with the parameter s below, indirectly determines the number of document clusters (leaf nodes in the document hierarchy)
2. for each document, find w words with the highest tf-idf score, $tfidf_d$, as the tags for that document
3. put all documents in one cluster
4. find w words as the tags on the cluster level for this cluster (see section on finding tags at cluster level)
5. check if the cluster has only documents which share at least s tags with the cluster level, $1<=s<=w$. If yes final cluster for this recursion path was found, stop; if no proceed to split the cluster into two
6. split the cluster by finding two new cluster centroids (as explained in Initialising KMeans below) and assigning each of the documents in the cluster to one of the new centroids using cosine similarity.
7. for each of the two new clusters, perform steps 4-6.

Finding Tags at Cluster Level

Tags at the cluster level can be found by

- taking the t words with highest average tf-idf score across all documents in the cluster
- recalculating the tf-idf on cluster level $tfidf_c$, treating clusters as single documents and the words from all documents within a cluster as members of this single document

The second approach of recalculating the tf-idf values at the cluster level has two advantages. Firstly, it makes sure that the tags found for the given cluster are good for distinguishing it from other clusters. Moreover, it is interesting to observe the change in cluster tags on different levels of the clustering hierarchy. On the lowest level where clusters are smallest, the cluster tags will be very specific to the topic of the documents that are in the cluster. Higher up the tree, when documents of different specialised topics share the same cluster, the words that are common between these documents and at the same time distinguish them from documents in other clusters will be chosen as cluster tags. The tags assigned to clusters on different levels in the tree will thus create a hierarchy of topics from general to specialised.

Initialising K-Means

The standard K-Means algorithm is initialised with random points for centroids [1]. This is the cheapest with regard to cost, however the choice of the location of cluster centroids is not based on any underlying rationale. A better method is to find two points which are far apart from each other as centroids.

Our approach is to select the cluster centroids with the goal of sooner satisfying the end condition of the clustering process. The end condition in our case is that all documents assigned to a cluster share at least s of their top w words with the tags of the cluster. When faced with a cluster that has to be split, we find the mean of the documents in the cluster that do satisfy the end condition and set the centroid of the first cluster to this mean. The candidates for the centroid of the second cluster are all the documents which do not satisfy the end condition. From these, the document that is farthest away from the first centroid by cosine similarity is set as the second cluster centroid. An alternative is to select the document for which the $tfidf_d$ scores for words which are tags of the cluster are lowest.

One problem that arises when clustering is that documents which do not share any attributes with any of the two cluster centroids will have a cosine similarity of zero to both centroids and thus cannot be assigned to one of them. This is overcome by slightly pulling in the cluster centroids towards the mean of all documents (described as "shrinking" in the CURE Algorithm [3]) which eliminates values of zero for attributes.

Agglomerative Part

During the divisive step, documents are partitioned greedily and research has shown that a second pass (using an agglomerative clustering approach) across the resulting clusters (leaf nodes) can help improve the clustering quality by reversing sub-optimal splits occurring in the divisive step [8]. Hence, after the divisive step we use the leaf nodes of the tree and perform agglomerative clustering on them as described below.

1. start with clusters produced by divisive part
2. find the two clusters for which the mean vectors are closest by cosine similarity and merge them
3. repeat step 2 until a desired number of clusters r is reached or until all clusters are merged into one
4. find tags for all clusters by calculating tf-idf on cluster level ($tfidf_c$) based on the documents in the clusters

The result is a hierarchy of cluster merges, with each cluster having w tags which form a topic hierarchy from general to specialised from the root to leaf nodes.

GENERATING NEW TAGS

To generate new tags for a document, the tf-idf score in the document is calculated for each of its words and a user-defined number k of the words with highest scores are proposed as candidate tags. The target document is also assigned to a cluster within the cluster tree generated by the clustering algorithm (see section on finding the document neighbourhood) and the tags associated with the cluster are also added to the candidate list of tags for the target document. Note that some of these words may not appear in the target document and are assigned to the document on the basis of its similarity with other members of the same cluster characterised by these words. The score assigned to a tag is the average between the tf-idf scores on document level ($tfidf_d$) and cluster level ($tfidf_c$).

$$wscore(word) = \frac{tfidf_d(word) + tfidf_c(word)}{2}$$

FINDING PERSONALISED TAGS

The initial set of potential tags for the new document-user pair (u_a, doc_{new}) consists of all tags assigned to documents in the δ-neighbourhood of the new document by any user (see section on finding document neighbourhood below). For each of these tags, a weight is calculated which measures the suitability of the tag based on the similarity scores of the documents that are related to the tag and the similarity scores of the users who assigned that tag. The set of users that we are interested in consist of the users who assigned tags to one or several documents in the neighbourhood of the new document.

The t tags with the highest suitability value are finally recommended to the active user. The formula for calculating the suitability of each tag is as follows.

$$suit(u_a, tag_i) = \frac{1}{|posts_{tag_i}|} \sum_i dsim(doc_{new}, doc_i)^{\beta} \cdot usim(u_a, u_b)^{(1-\beta)}$$

where $|posts_{tag_i}|$ is the number of posts for tag_i, $dsim$ and $usim$ are the document and user similarity measures (described below) and $\beta \leq 1$ is the weight given to document similarity compared to user similarity.

If there are not enough tags in the resulting tag set with a *suitability score* above a user-defined threshold, new tags are generated as described in the section above.

Finding the Document Neighbourhood

As described previously the document clustering algorithm results in a cluster hierarchy where each parent node has two child nodes. To select a set of similar documents for a new document from the cluster hierarchy, the tree is traversed from the top down and the new document is assigned to one of the two clusters at each branch split. Thus a path through the tree is obtained for the new document in *O(log nc)* time. From this path, the cluster with the highest cosine similarity is assigned to the document as its cluster and the documents in the cluster added to the set of similar documents D_{sim}. For each of the documents in D_{sim}, the similarity to the new document $dsim(doc_{new}, doc_i)$ is calculated by cosine similarity. If the document similarity is above a threshold δ for each of the documents in D_{sim}, further documents are added to D_{sim} by travelling up the tree from the cluster and adding all documents in the clusters on the path until a cluster containing a document with $dsim(doc_{new}, doc_i) < \delta$ is encountered.

User Similarity

The similarity between two users can be calculated in three different ways.

The *document set similarity* considers how many document are shared between the two users and is calculated using the Jaccard co-efficient.

$$sim_D(u_a, u_b) = \frac{|D_a \cap D_b|}{|D_a \cup D_b|}$$

where D_a and D_b are the document sets of the two users.

The *vocabulary similarity* measures the overlap in the two users' vocabulary sets and is given by

$$sim_V(u_a, u_b) = \frac{|V_a \cap V_b|}{|V_a \cup V_b|}$$

where V_a and V_b are the sets of tags used by the two users.

The *tagging similarity* considers whether the two users assigned the same tags to the same documents and is calculated as follows.

$$sim_T(u_a, u_b) = \frac{1}{|D_a \cup D_b|} \sum_{i=1}^{|T_{d_i a} \cap T_{d_i b}|} \frac{|T_{d_i a} \cap T_{d_i b}|}{|T_{d_i a} \cup T_{d_i b}|}$$

where $T_{d_i a}$ and $T_{d_i b}$ are the tag sets of the two users for document i. The *tagging similarity* measure incorporates *document set similarity* through the division by $|D_a \cup D_b|$. This ensures that the *tagging similarity* reflects not only the documents for which two users have common tags but also takes into account the number of documents for which the two users do not have common tags.

The *tagging similarity* measure is the most valuable since it takes into consideration both, document and tag sets and the relationship between these, while the other two measures focus on only one of these aspects. However, due to sparsity of the data the two users often do not have any shared document-tag pairs which means the *tagging similarity* is zero. Hence, the *tagging similarity* is combined with *vocabulary similarity* which results in zero less frequently. A *tagging similarity* of zero and a non-zero *vocabulary similarity* indicates that the users share some tags but have assigned them to different documents, thus the tags used by user u_a might be suitable for u_b. The combined user similarity score is given by

$$usim(u_a, u_b) = \alpha sim_T(u_a, u_b) + (1 - \alpha) sim_V(u_a, u_b)$$

where $\alpha \leq 1$ specifies the weight given to the *tagging similarity* compared to the *vocabulary similarity*.

EVALUATION

At the time of writing, the implementation of the suggested system is not fully completed and an extensive evaluation to determine the best thresholds and parameters still has to be done. An initial evaluation was carried out on a dataset from the social bookmarking website Bibsonomy. The dataset consists of a variety of websites and academic papers that were bookmarked on Bibsonomy and the tags that were assigned to these documents by the users. The

post-core at level 5 was used so each user, document and tag appeared at least five times in the data, resulting in 7538 posts (user, document pairs) containing 244 users, 953 documents and 811 tags. Similarly to [4], leave-one-out cross validation was used to evaluate the algorithm. Each document was represented using a vector space model, where the dimensions were the 1000 words with largest document frequency in the corpus. Five tags were recommended for each test post and the performance measure was standard F1. Since the test dataset contained only users and tags already known to the system no new tags were generated.

The most significant parameters of our recommender system are thresholds on document and user similarity. These thresholds allow us to consider only documents and users with a similarity score higher than the set value when calculating the scores for the potential tags. If we set the threshold for document similarity to 1.0, only tags assigned to the new document in other posts (by other users) are considered. This is referred to as the set of *popular tags* in other systems such as the current del.icio.us tag recommender. While in other systems the score for each tag is given simply by the number of posts it appears in, our system also weights each post's importance by considering the user similarity to the active user, hence personalising the tag recommendation. If we set the threshold for user similarity to 1.0, only the tags from the active user's other posts are considered. This is referred to as the set of *personal tags*. For this scenario each tag's score is influenced by the similarity of the new document to the documents in the user's posts.

When tested on their own, the approach of *popular tags* produced better recommendations (F1 of 0.25) than the *personal tags* (F1 of 0.13). However, the best results were achieved through generating two sets of tags, one from each approach and combining them to give a final recommendation set. The two sets were combined by a weighted sum of the two scores for each tag in order to increase the overall score of tags which appeared in both sets. The final tag scores were then normalised. By recommending tags from the combined set of popular and personal tags, the F1 was increased to 0.35. The results of the combined approach when recommending different sizes of tag sets is shown in Figure 1 below. To additionally evaluate our method of generating new tags, we set the threshold on the tag score to 1.0 in another test run so that all recommended tags were generated from the content of the documents. The resulting F1 was 0.12 when generating five tags for each post.

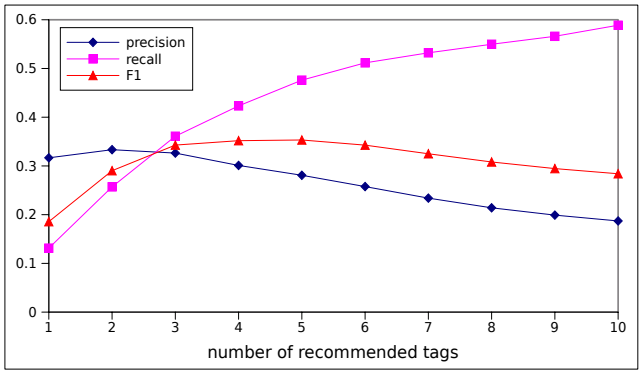

Figure 1: Results on Bibsonomy dataset at p-core 5

CONCLUSION AND FUTURE WORK

In this paper we suggested a novel approach to personalised tag recommendation. By creating a cluster tree of documents we generate a topic hierarchy from general to specialised, and determine the level of specialisation of a new document. The recommended tags reflect the generality of the new document. Through identifying a set of users similar to the active user and measuring their similarity, we are able to suggest personalised tags. If the required number of tags cannot be found in the existing tag set, new tags are generated from the vocabulary of the document corpus.

We plan to perform an extensive evaluation of the suggested system on different datasets in order to find the best values for the many parameters such as the weights given to the different similarity measures and thresholds for tag suitability. We will also evaluate our methods ability to address the new-user/new-document/new-tag instantiation of the new-item problem well known in Recommender literature. To further improve the tag recommender, we plan to implement and evaluate different clustering and cluster representation techniques such as those used by CURE [3].

In the future, we plan to investigate different approaches to tag recommendation, taxonomy generation and dimensionality reduction. Techniques for feature extraction such as finding synonym sets and topic models for documents are very interesting and have the potential increase the performance of any tag recommendation or classification system.

REFERENCES

[1] P. S. Bradley and U. M. Fayyad. Refining initial points for K-Means clustering. In *Proc. 15th International Conf. on Machine Learning*, pages 91–99, 1998.

[2] J. Gemmell, T. Schimoler, M. Ramezani, and B. Mobasher. Adapting k-nearest neighbor for tag recommendation in folksonomies. In *Proc. 7th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems*, 2009.

[3] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.

[4] R. Jaeschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Proc. 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2007.

[5] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *Proc. 31st international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, 2008.

[6] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. Technical Report 00-034, University of Minnesota, 2000.

[7] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. ICML-97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

[8] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD '96: Int. Conf. on Management of Data*, pages 103–114, 1996.