Semi-Supervised Tag Extraction in a Web Recommender System

Vasily Leksin Sergey Nikolenko

Surfingbird LLC, Moscow

National Research University Higher School of Economics, St. Petersburg

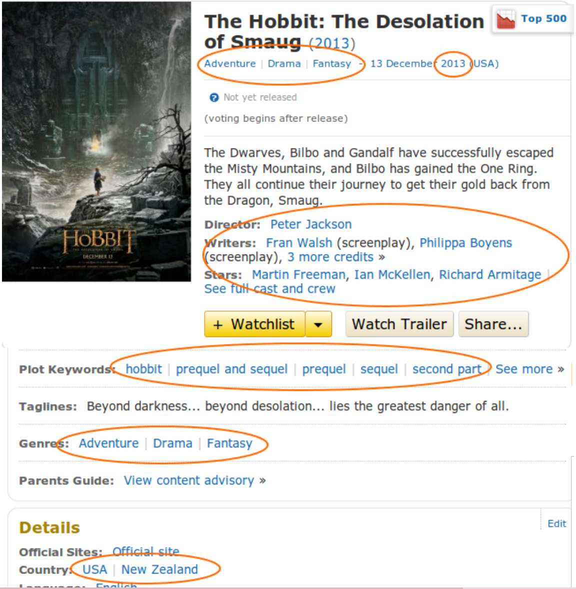
October 3, 2013

Outline

1. Tags in recommender systems
   - Motivation
   - Our approach in general

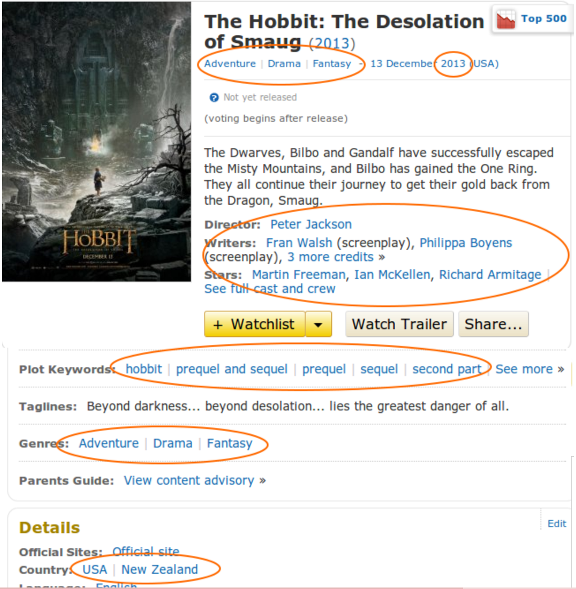
2. Our method
   - The method in detail
   - Beyond the paper

Tags in recommender systems



- In recommender systems, content can often be characterized by tags.
- E.g., movies have lots of tags: genre, director, actors etc.
- Tags can help.

Tags in recommender systems



- There are two common problems:
  - improving recommender algorithms with tags that are already in place;
  - helping users tag items by providing suggestions for tags (tag recommendation).

Tags in recommender systems

- Tags have been used successfully in "classical" recommender systems (based on user-user or item-item similarity):
  - [Sen, Vig, Riedl, 2009]: "Tagommenders", variations of classical recommender systems with tags; a comparison of different models for rating tagged movies;
  - [Zhou et al., 2010]: UserRec, a system that does community detection on a graph of tags, identifying specific topics characterized by tags, and then recommends based on a user's affinity to various topics;
  - [Guy et al., 2010]: personalized recommendations in social media based on tags (basically a feed filter).

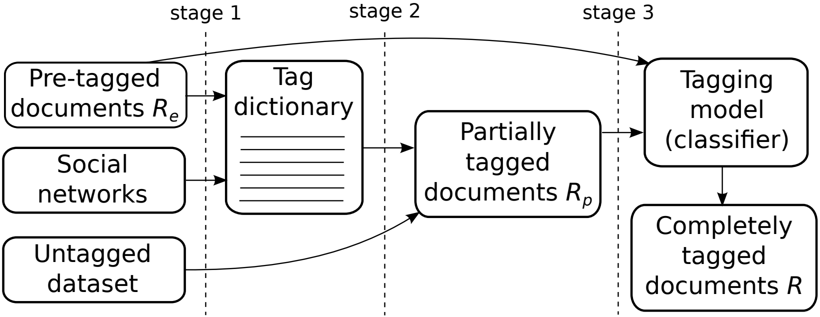
Tags in recommender systems

- Extensive literature exists on tag recommendation, and collaborative filtering is commonly used for this problem.
- In matrix factorization algorithms, tags can serve as an additional dimension, both for item recommendation and tag recommendation:
  - [Symeonidis et al., 2009]: user-item-tag tensor that one can spin either way;
  - [Rendle, Schmidt-Thieme, 2010]: another tensor factorization model for personalized tag recommendation.
- So tags seem a good fit for a system that recommends interesting web pages to users (Surfingbird, StumbleUpon).
- But...

Tags in web recommender systems

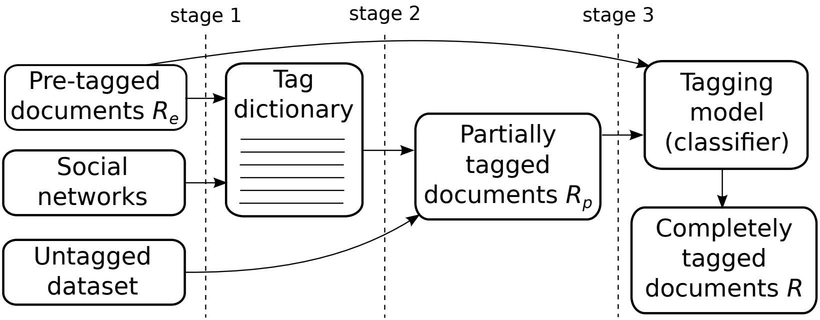
- All these systems assume that users actively tag items, and even in the worst case we only need to help them, provide suggestions for users based on tags that are already in place.
- In a web recommender system like Surfingbird or StumbleUpon:
  - the user is basically just surfing the web, with a generally more passive approach;
  - there are about as many items as users;
  - most items are viewed for a very short time before the user browses on.
- Hence, we cannot expect users to tag items, and we also cannot expect moderators to do it by hand.

Our approach



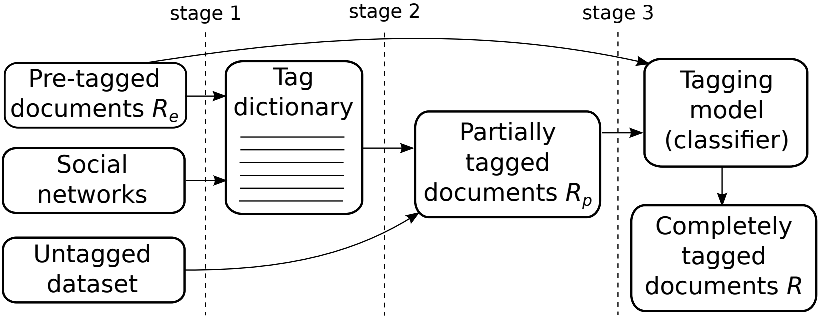
- The basic plan is as follows: for a dataset $R = R_e \cup R_u$ with exactly tagged resources R_e and untagged resources R_u,
  1. extract tags from the pre-tagged part of the dataset R_e and social networks;

Our approach



- The basic plan is as follows: for a dataset $R = R_e \cup R_u$ with exactly tagged resources R_e and untagged resources R_u,
  2. perform partial tag labeling for the untagged part R_u based on key phrase occurrence, getting a partially tagged dataset R_p;

Our approach



- The basic plan is as follows: for a dataset $R = R_e \cup R_u$ with exactly tagged resources R_e and untagged resources R_u,
  3. learn a tagging model (classifier) from $R_e \cup R_p$ and apply it to R_p, getting a completely tagged dataset as well as a model ready to tag new resources (web pages).

Outline

1. Tags in recommender systems
   - Motivation
   - Our approach in general

2. Our method
   - The method in detail
   - Beyond the paper

Extracting tags

- Where do tags come from in a web recommender system?
- First, some web pages come pre-tagged (e.g., tags can be provided by trusted publishers in RSS streams). We assume those to be correct and take them into the tag dictionary directly.
- But that is a small fraction of web pages (5-10%), and we cannot expect to find all interesting tags in this way.

Extracting tags

- So we turn to social networks, mining tags from user profiles.
- Both *facebook* and *vkontakte* may provide lists of:
  - favourite movies,
  - favourite books,
  - favourite music,
  - groups (that also often correspond to interests),
  - ...
- About half of the users register through social networks, so this gives lots of results.
- Then we prune uninformative tags (too rare or too popular).

Extracting tags

A sample of our results (mostly translated from Russian).

Gadgets	Games	Books	Music	Movies
android	assassin creed	short stories	bahh tee	the matrix
hardware	video games	albert camus	britney spears	pearl harbor
google	rally	o. henry	whitney houston	sherlock holmes
software	development	ryunosuke akutagawa	george watsky	apocalypse now
iphone	reviews	audiobook	rap	titanic
samsung	call of duty	steve jobs	slipknot	ocean's thirteen
apple	star wars	arkady gaidar	emma hewitt	comedy
ios	half-life	pierre gamarra	james blunt	south park
tablet pc	releases	biography	ellie white	avatar
smartphones	angry birds	guy endore	izzy johnson	the green mile

Preliminary tagging

- To do pre-tagging, we search for occurrences of tags in the content of untagged web pages:
    - extract textual content from each web page,
    - transform the tag phrase into a search query which is a conjunction of all words,
    - use text search to find the corresponding web pages,
    - filter search results: find tag phrases with inexact string matching, set a threshold for the number of occurrences.
- The search can be efficiently implemented on the database level (e.g., with the PostgreSQL full text search feature); we need inexact matching only to filter search results.

Tag recommendation

- Finally, we get $R = R_e \cup R_p$ with exactly tagged R_e and partially tagged R_p.
- But we still want to augment R_p with tags that may never or rarely occur on the page:
  - e.g., an article about "The Hobbit" movie may never mention "movies";
- Thus, we need to add new tags to R_p based on the content of these web pages.

Tag recommendation

- We pose this as a classification problem:
  - consider a bag of words for each $r \in R$;
  - solve a binary classification problem: does a given tag t match a given resource r defined by its words as features?
- We compare two different sets of resource features: word counts r_w and tf-idf weights

$$\text{tf-idf}(w, r, R) = \mathrm{tf}(w, r)\mathrm{idf}(w, R) = \frac{r_w}{\sum_{w \in W} r_w} \log \frac{|R|}{|\{r \in R \mid w \in r\}|}.$$

Tag recommendation

- For classification, we compare four different approaches – two baseline:
  - *Regularized Least Squares Classification* (RLSC) with linear kernel: solve a minimization problem with the square loss function, i.e., find the weights $\boldsymbol{w}$ that solve the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^{\boldsymbol{d}}} \frac{1}{2}\sum_{i=1}^{n}\|y_i - \boldsymbol{w}^\top \boldsymbol{x}_i\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2.$$

  - *Support Vector Machine* (SVM) with l_2-SVM loss trained with the modified Newton method, i.e., find the weights $\boldsymbol{w}$ that that solve the following optimization problem:

$$\min_{\boldsymbol{w}\in\mathbb{R}^{\boldsymbol{d}}} \frac{1}{2}\sum_{d\in D} l_2\left(y_i \boldsymbol{w}^\top \boldsymbol{x}_i\right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \quad l_2(z) = \max\left(0, 1 - z^2\right);$$

Tag recommendation

- For classification, we compare four different approaches – and two semi-supervised:
  - *Multi-switch Transductive SVM* (MTSVM), a semi-supervised version of SVM with l_2 loss function:

$$\min_{\boldsymbol{w}, y'} \left[\frac{\lambda}{2} \|\boldsymbol{w}\|^2 + \frac{1}{2|D|} \sum_{i \in D} l_2\left(y_i \boldsymbol{w}^\top \boldsymbol{x}_i\right) + \frac{\lambda'}{2|U|} \sum_{j \in U} l_2\left(y_j' \boldsymbol{w}^\top \boldsymbol{x}_j'\right) \right],$$

$$l_2(z) = \max(0, 1 - z^2), \quad \text{subject to } \frac{1}{|U|} \sum_{j \in U} \max\left(0, \operatorname{sign}\left(\boldsymbol{w}^\top \boldsymbol{x}_j'\right)\right) = r,$$

where U is the unlabeled part of the data, y' are labels for unlabeled data that are also being optimized, and r is a predefined fraction of unlabeled data expected to have positive labels (estimated from the labeled part).

Tag recommendation

- For classification, we compare four different approaches – and two semi-supervised:
  - *Deterministic Annealing Semi-supervised SVM* (DASVM), a relaxation of the MTSVM problem with a loss function close to squared loss for large values of temperature (i.e., in the beginning of the annealing process) and converging to the l_2 loss function used in TSVM as temperature drops to zero.
- Semi-supervised SVMs come from [Sindhwani, Keerthi, 2006].
- For evaluation, we count the micro-averaged F-measure (micro-F), the harmonic mean of
  - micro-averaged precision $(\sum \text{true positives})/(\sum \text{true positives} + \sum \text{false positives})$,
  - micro-averaged recall $(\sum \text{true positives})/(\sum \text{true positives} + \sum \text{false negatives})$.

Evaluation results

Table: A comparison of different classification algorithms: Micro-F scores.

Algorithm	RLSC	SVM	MTSVM	DASVM
Term occurrence count features	0.21	0.25	**0.31**	0.30
Tf-idf features	0.24	0.27	**0.35**	0.33
Counts, tags not occurring in the text	0.13	0.15	0.20	**0.21**
Tf-idf, tags not occurring in the text	0.14	0.17	0.21	**0.23**

- Results are similar to those reported for other tag recommendation algorithms, but in our case:
  - the set of tags was generated semi-automatically,
  - the training data for classification was also partially inferred rather than labeled by hand,
- so our problem is much noisier than usual.

Topic modeling

- Now for some further work.
- Topic modeling is an important idea for tag mining; a slew of models based on LDA (latent Dirichlet allocation). In the simplest approach, one can:
  - use LDA to produce a set of topics many of which can be tagged (by hand) with high confidence;
  - tag documents with a large share in the corresponding topic.
- LDA results can also be used directly for recommendation:
  - [Jin et al., 2005]: regular LDA used as additional content features;
  - [Agarwal, Chen, 2010]: fLDA, a recommender variation of LDA that unifies ratings and content in a single model.

Topic modeling

- LDA is very good for general tags like "Japan", "movies", or "fitness" but almost useless for specific tags like "Bruce Willis" or "sashimi" (such specific entities will not form separate topics in a general dataset).
- [Si, Sun, 2008] Tag-LDA, a version of LDA designed to recommend tags.

Thank you!

Thank you for your attention!