Extended Recommendation Framework: Generating the Text of a User Review as a Personalized Summary

Mickaël Poussevin
Sorbonne-Universités UPMC
LIP6 UMR 7606 CNRS
4 Place Jussieu, Paris, France
mickael.poussevin@lip6.fr

Vincent Guigue
Sorbonne-Universités UPMC
LIP6 UMR 7606 CNRS
4 Place Jussieu, Paris, France
vincent.guigue@lip6.fr

Patrick Gallinari
Sorbonne-Universités UPMC
LIP6 UMR 7606 CNRS
4 Place Jussieu, Paris, France
patrick.gallinari@lip6.fr

ABSTRACT

We propose to augment rating based recommender systems by providing the user with additional information which might help him in his choice or in the understanding of the recommendation. We consider here as a new task, the generation of personalized reviews associated to items. We use an extractive summary formulation for generating these reviews. We also show that the two information sources, ratings and items could be used both for estimating ratings and for generating summaries, leading to improved performance for each system compared to the use of a single source. Besides these two contributions, we show how a personalized polarity classifier can integrate the rating and textual aspects. Overall, the proposed system offers the user three personalized hints for a recommendation: rating, text and polarity. We evaluate these three components on two datasets using appropriate measures for each task.

1. INTRODUCTION

The emergence of the participative web has enabled users to easily give their sentiments on many different topics. This opinionated data flow thus grows rapidly and offers opportunities for several applications like e-reputation management or recommendation. Today many e-commerce websites present each item available on their platform with a description of its characteristics, average appreciation, ratings together with individual user reviews explaining their ratings.

Our focus here is on user - item recommendation. This is a multifaceted task where different information sources about users and items could be considered and different recommendation information could be provided to the user. Despite this diversity, the academic literature on recommender systems has focused only on a few specific tasks. The most popular one is certainly the prediction of user preferences given their past rating profile. These systems typically rely on collaborative filtering [9] to predict missing values in a *user/item/rating* matrix. In this perspective of rating prediction, some authors have make use of additional information sources available on typical e-commerce sites. [5] proposed to extract topics from consumer reviews in order to improve ratings predictions. Recently, [11] proposed to learn a latent space common to both textual reviews and product ratings, they showed that rating prediction was improved by such hybrid recommender systems. Concerning the information provided to the user, some models exploit review texts for ranking comments that users may like [1] or for answering specific user queries [17].

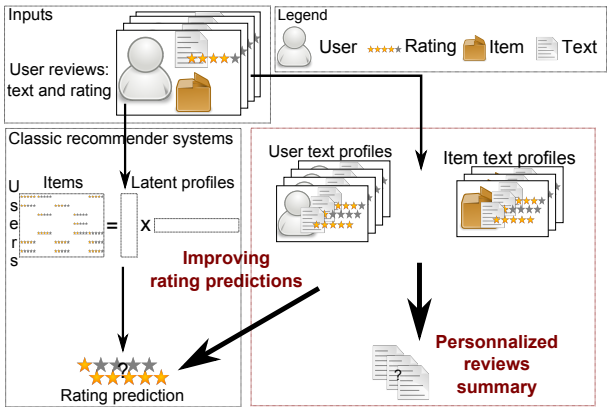


Figure 1: Our contribution is twofold: (1) improving rating predictions using textual information, (2) generating personalized reviews summaries to push recommender systems beyond rating predictions

We start here from the perspective of predicting user preference and argue that the exploitation of the information present in many e-commerce sites, allows us to go beyond simple rating prediction for presenting users with complementary information that may help him making his choice. We consider as an example the generation of a personalized review accompanying each item recommendation. Such a review is a source of complementary evidence for the user appreciation of a suggestion. Similarly as it is done for the ratings, we exploit past information and user similarity in order to generate these reviews. Since pure text generation is a very challenging task [2], we adopt an extractive summary perspective: the generated text accompanying each rating will be extracted from the reviews of selected users who share similar tastes and appreciations with the target user. Ratings and reviews being correlated, this aspect could also be exploited to improve the predictions. Our rating pre-

CBRecSys 2015, September 20, 2015, Vienna, Austria.

dictor will make use of user textual profiles extracted from their reviews and summary extraction in turn will use predicted ratings. Thus both types of information, predicted ratings and generated text reviews, are offered to the user and each prediction, rating and generated text, takes into account the two sources of information. Additional information could also be provided to the user. We show here as an example, that predicted ratings and review texts can be used to train a robust sentiment classifier which provides the user with a personalized polarity indication about the item. The modules of our system are evaluated on the two main tasks, rating prediction and summary extraction, and on the secondary task of sentiment prediction. For this, experiments are conducted on real datasets collected from *amazon.com* and *ratebeer.com* and models are compared to classical baselines.

The recommender system is compared to a classic collaborative filtering model using the mean squared error metric. We show that using both ratings and user textual profiles allows us to improve the performance of a baseline recommender. Gains are motivated from a more precise understanding of the key aspects and opinions included in the item and user textual profiles. For evaluating summary text generation associated to a couple (user, item), we have at our disposal a gold standard, the very review text written by this user on the item. Note that this is a rare situation in summary evaluation. However contrarily to collaborative filtering, there is no consensual baseline. We then compare our results to a random model and to oracle optimizing the ROUGE-n metric. They respectively provide a lower and an upper bound of the attainable performance. The sentiment classifier is classically evaluated using classification accuracy.

This article is organized as follows. The hybrid formulation, the review generator and the sentiment classifier are presented in section 2. Then, section 3 gives an extensive experimental evaluation of the framework. The overall gains associated to hybrid models are discussed in section 4. A review of related work is provided in section 5.

2. MODELS

In this section, after introducing the notations used throughout the paper, we will describe successively the three modules of our system. We start by considering the prediction of ratings [11]. Rating predictors answer the following question: *what rating will this user give to this item?* We present a simple and efficient way to introduce text profiles representing the writing style and taste of the user in a hybrid formulation. We then show how to exploit reviews and ratings in a new challenging task: *what text will this user write about this item?* We propose an extractive summary formulation of this task. We then proceed to describe how both ratings and text could be used together in a personalized sentiment classifier.

2.1 Notations

We use u (respectively i) to refer to everything related to a user (respectively to an item) and the rating given by user u to the item i is denoted r_{ui}. U and I refer to anything relative to all users and all items, such as the rating matrix R_{UI}. Similarly, lower case letters are used for scalars or vectors and upper case letters for matrices. d_{ui} is the actual review text written by user u for item i. It is composed of κ_{ui} sentences: $d_{ui} = \{s_{uik}, 1 \leq k \leq \kappa_{ui}\}$. In this work, we consider documents as bags of sentences. To simplify notations, s_{uik} is replaced by s_{ui} when there is no ambiguity. Thus, user appreciations are quadruplets (u, i, r_{ui}, d_{ui}). Recommender systems use past information to compute a rating prediction $\hat{r}_{ui}$, the corresponding prediction function is denoted $f(u, i)$.

For the experiments, ratings and text reviews are split into training, validation and test sets respectively denoted S_{train}, S_{val} and S_{test} and containing m_{train}, m_{val} and m_{test} user appreciations (text and rating). We denote $S^{(u)}_{train}$, the subset of all reviews S_{train} that were written by user u and $m^{(u)}_{train}$ the number of such reviews. Similarly, $S^{(i)}_{train}$ and $m^{(i)}_{train}$ are used for the reviews on item i.

2.2 Hybrid recommender system with text profiles

Recommender systems classically use rating history to predict the rating $\hat{r}_{ui}$ that user u will give to item i. The hybrid system described here makes use of both *collaborative filtering* through matrix factorization and textual information to produce a rating as described in (1):

$$f(u,i) \quad = \quad \mu + \mu_u + \mu_i + \gamma_u.\gamma_i + g(u,i) \qquad (1)$$

The first three predictors in equation (1) are biases (overall bias, user bias and item bias). The fourth predictor is a classical matrix factorization term. The novelty of our model comes from the fifth term (1) that takes into account text profiles to refine the prediction f. Our aim for the rating prediction is to minimize the following empirical loss function:

$$\underset{\mu,\mu_u,\mu_i,\gamma_u,\gamma_i,g}{\operatorname{argmin}} \quad L \quad = \quad \frac{1}{m_{train}} \sum_{S_{train}} \left(r_{ui} - f(u,i)\right)^2 \qquad (2)$$

To simplify the learning procedure, we first optimize the parameters of the different components independently as described in the following subsections. Then we fine tune the combination of these components by learning weighting coefficients so as to maximize the performance criterion (2) on the validation set.

2.2.1 Matrix factorization

We first compute the different bias from eq. (1) as the averaged ratings over their respective domains (overall, user and item). For the matrix factorization term, we approximate the rating matrix R_{UI} using two latent factors: $R_{UI} \approx \Gamma_U \Gamma_I^T$. Both Γ_U and Γ_I are two matrices representing collections of latent profiles, with one profile per row. We denote γ_u (resp. γ_i) the row of Γ_U (resp. Γ_I) corresponding to the latent profile of user u (resp. item i).

The profiles are learned by minimizing, on the training set, the mean squared error between known ratings in matrix R_{UI} and the approximation provided by the factorization $\Gamma_U \Gamma_I^T$. This minimization problem described in equation (3), with an additional L2 constraint (4) on the factors is solved here using non-negative matrix factorization.

$$\Gamma_U^*, \Gamma_I^* = \underset{\Gamma_U,\Gamma_I}{\operatorname{argmin}} \qquad \|M_{train} \odot (R_{UI} - \Gamma_U \Gamma_I)\|_F^2 \qquad (3)$$

$$+\lambda_U \|\Gamma_U\|_F^2 + \lambda_I \|\Gamma_I\|_F^2 \qquad (4)$$

In this equation M_{train} is a binary mask that has the same dimensions as matrix R_{UI}, an entry is 1 only if the corresponding review is in the training set, $\odot$ is the element-wise product and $\|\cdot\|_F$ denotes the Frobenius norm.

2.2.2 Text profiles exploitation

Let us denote π_u the text profile of user u and $\sigma_t(\pi_{u'}, \pi_u)$ a similarity operator between user profiles. The last component of the predictor f in (1) is a weighted average of user ratings for item i, where weight $\sigma_t(\pi_{u'}, \pi_u)$ is the similarity between the text profiles $\pi_{u'}$ and π_u of users u' and u, the latter being the target user. This term takes into account the fact that two users with similar styles or using similar expressions in their appreciation of an item, should share close ratings on this item. The prediction term for the user/item couple (u, i) is then expressed as a weighted mean:

$$g(u,i) = \frac{1}{m^{(i)}_{train}} \sum_{S^{(i)}_{train}} r_{u'i}\sigma_t(\pi'_u, \pi_u) \tag{5}$$

Two different representations for the text profiles π_u of the users are investigated in this article: one is based on a latent representation of the texts obtained by a neural network autoencoder, the other relies on a robust bag of words coding. Each one is associated to a dedicated metric σ_t.

This leads to two formulations of g, and thus, to two rating prediction models. We denote the former f_A (autoencoder) and the latter f_T (bag of words). Details are provided below.

Bag of words.

A preprocessing step removes all words appearing in less than 10 documents. Then, the 100 000 most frequent words are kept. Although the number of features is large, the representation is sparse and scales well. π_u is simply the binary bag of words of all texts of user u. In this high dimensional space, the proximity in style between two users is well described by a cosine function, a high value indicates similar usage of words:

$$\sigma_t(\pi_{u'}, \pi_u) = \pi_{u'}\pi_u/(\|\pi_{u'}\|\|\pi_u\|) \tag{6}$$

Autoencoder.

The neural network autoencoder has two components: a coding operator and a decoding operator denoted respectively *cod* and *dec*. The two vectorial operators are learned so as to enable the reconstruction of the original text after a projection in the latent space. Namely, given a sentence s_{uik} represented as a binary bag of words vector, we obtain a latent profile $\pi_{s_{uik}} = cod(s_{uik})$ and then, we reconstruct an approximation of the sentence using $\hat{s}_{uik} = dec(\pi_{s_{uik}})$.

The autoencoder is optimized so as to minimize the reconstruction error over the training set:

$$cod^*, dec^* = \underset{cod,dec}{\operatorname{argmin}} \sum_{S_{train}} \frac{1}{\kappa_{ui}} \sum_{k=1}^{\kappa_{ui}} \|s_{uik} - dec(cod(s_{uik}))\|^2 \tag{7}$$

We use the settings proposed in [6]: our dictionary is obtained after stopwords removal and selecting the most frequent 5000 words. we did not use a larger dictionary such as the one used for the bag of word representation since it does not lead to improved performance and simply increases the computational load. All sentences are represented as binary bag of words using this dictionary. The coding dimension has been set to 1000 after a few evaluation trials. Note that the precise value of this latent space is not important and the performance is similar on a large range of dimension values. Both *cod* and *dec* use sigmoid units $\text{sig}(t) = \frac{1}{1+\exp(-t)}$:

$$\begin{aligned} \text{cod}(s_{uik}) &= \pi_{uik} = \text{sig}(W s_{uik} + b) \\ \text{dec}(\pi_{uik}) &= \text{sig}(W^T \pi_{uik} + b') \end{aligned} \tag{8}$$

Here, π_{uik} is a vector, W is a 5000x1000 weight matrix and sig() is a pointwise sigmoid operator operating on the vector $Ws_{uik} + b$.

As motivated in [11, 5], such a latent representation helps exploiting term co-occurrences and thus introduces some semantic. It provides a robust text representation. The hidden activity of this neural network produces a continuous representation for each sentence accounting for the presence or absence of groups of words.

π_u is obtained by coding the vector corresponding to all text written by the user u in the past. It lies in a latent word space where a low Euclidean distance between users means a similar usage of words. Thus, for the similarity σ_t, we use an inverse Euclidean distance in the latent space:

$$\sigma_t(\pi_{u'}, \pi_u) = 1/(\alpha + \|\pi_{u'} - \pi_u\|) \tag{9}$$

2.2.3 Global training criterion for ratings prediction

In order to connect all the elementary components described above with respect to our recommendation task, we introduce (positive) weighting parameters β in (1). Thus, the initial optimization problem (2) becomes:

$$\begin{aligned} &\beta^* = \underset{\beta}{\operatorname{argmin}} \tfrac{1}{m_{train}} \textstyle\sum_{S_{train}} \\ &\Big(r_{ui} - \big(\beta_1\mu^* + \beta_2\mu^*_u + \beta_3\mu^*_i + \beta_4\gamma^*_u.\gamma^*_i + \beta_5 g(u,i)\big)\Big)^2 \end{aligned} \tag{10}$$

The linear combination is optimized using a validation set: this step guaranties that all components are combined in an optimal manner.

2.3 Text generation model

The goal here is to generate a review text for each (u,i) recommendation. During the recommendation process, this text is an additional information for users to consider. It should catch their interest and in principle be close to the one that user u could have written himself on item i. Each text is generated as an extractive summary, where the extracted sentences $s_{u'i}$ come from the reviews written by other users ($u' \neq u$) about item i. Sentence selection is performed according to a criterion which combines a similarity between the sentence and the textual user profile and a similarity between the actual rating $r_{u'i}$ and the prediction made for (u,i), $\hat{r}_{ui}$ computed as described in section 2.2. The former measure could take into account several dimensions like vocabulary, sentiment expression and even style, here it is mainly the vocabulary which is exploited. The latter measures the proximity between user tastes. For the text measure, we make use of the σ_t similarity introduced in section 2.2. As before, we will consider two representations for texts (latent coding and raw bag of words). For the ratings similarity, we use $\sigma_r(r_{u'i}, r_{ui}) = 1/(1 + |r_{u'i} - r_{ui}|)$.

Suppose one wants to select a single sentence for the extracted summary. The sentence selection criterion will then be a simple average of the two similarities:

$$h(s_{u'i}, r_{u'i}, u', u, i) = \frac{\sigma_t(s_{u'i}, \pi_u) + \sigma_r(r_{u'i}, \hat{r}_{ui})}{2} \quad (11)$$

Note that this function may score any piece of text. In the following, we then consider three possibilities for generating text reviews: The first one simply consists in selecting the best sentence $s_{u'i}$ among all the training sentences for item i with respect to h. We call it 1S for single sentence. The second one selects a whole review $d_{u'i}$ among all the reviews for i. The document is here considered as one long sentence. This is denoted CT for complete text. The third one is a greedy procedure that selects multiple sentences, it is denoted XS. It is initialized with 1S, and then sentences are selected under two criteria: relevance with respect to h and diversity with respect to the sentences already selected. Selection is stopped when the length of the text is greater than the average length of the texts of the target user. Algorithm 1 sums up the XS procedure for generating the text $\hat{d}_{ui}$ for the couple user u, item i.

Data: u, i, $S = \{(s_{u'i}, r_{u'i}\ u'\}$
Result: $\hat{d}_{ui}$
$s^*_{u'i} \leftarrow \underset{s_{u'i} \in S}{\operatorname{argmax}} \big(h(s_{u'i}, r_{u'i}, u', u, i)\big)$;
$\hat{d}_{ui} \leftarrow s^*_{u'i}$;
Remove $s^*_{u'i}$ from S;
while length $\hat{d}_{ui} <$ averagelength(u) **do**
  $s^*_{u'i} \leftarrow \underset{s_{u'i} \in S}{\operatorname{argmax}} \big(h(s_{u'i}, r_{u'i}, u', u, i) - \cos(s_{u'i}, \hat{d}_{ui})\big)$;
  $\hat{d}_{ui} \leftarrow s^*_{u'i}$;
  Remove $s^*_{u'i}$ from S;
end

Algorithm 1: XS greedy procedure: selection of successive sentences to maximize both relevance and diversity. $\hat{d}_{ui}$ is the text that is generated, sentence after sentence.

2.4 Sentiment prediction model

We show here how polarity information about an item can be estimated by exploiting both the user predicted ratings and his textual profile. Exploiting both information sources improves the sentiment prediction performance compared with a usual text based sentiment classifier.

Polarity classification is the task of predicting whether a text d_{ui} (here of a review) is positive or negative. We use as ground truth the ratings r_{ui} and follow a standard thresholding procedure [15]: reviews rated 1 or 2 are considered as negative, while items rated 4 or 5 are positive. All texts that are rated 3 are ignored as it is unclear whether that are positive or negative: it strongly depends on the rating habits of the user.

For evaluation purpose, we consider two baselines. A first one only uses the rating prediction of our recommender system $f(u, i)$ as a label prediction, this value is then thresholded as indicated above. A second one is a classical text sentiment classifier. Denoting by d_{ui} the binary bag of word representation of a document and c_{ui} the binary label associated to the rating r_{ui}, one uses a linear SVM $s(d_{ui}) = d_{ui}.w$. Note that this is usually a strong baseline for the polarity classification task. Our final classifier will combine $f(u, i)$ and $s(d_{ui})$ in order to solve the following optimization problem:

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{S_{train}, r_{ui} \neq 3} \Big(1 - \big(d_{ui}.w + f(u,i)\big)c_{ui}\Big)_+ + \lambda \|w\|^2 \quad (12)$$

with $(x)_+ = x$ when x positive and $(x)_+ = 0$ elsewhere. In the experimental section, we will also compare the results obtained with the two versions of our rating predictor: f_T and f_A (cf section 2.2.2).

Source	Subset names	#Users	#Items	#Reviews		
				#Training	#Validation	#Test
Ratebeer	RB_U50_I200	52	200	7200	900	906
	RB_U500_I2k	520	2000	388200	48525	48533
	RB_U5k_I20k	5200	20000	1887608	235951	235960
Amazon	A_U200_I120	213	122	984	123	130
	A_U2k_I1k	2135	1225	31528	3941	3946
	A_U20k_I12k	21353	12253	334256	41782	41791
	A_U210k_I120k	213536	122538	1580576	197572	197574

Table 1: Users, items & reviews counts for every datasets.

Subsets	μ	μ_u	μ_i	$\gamma_u.\gamma_i$	f_A	f_T
RB_U50_I200	0.7476	0.7291	0.3096	0.2832	**0.2772**	**0.2773**
RB_U500_I2k	0.6536	0.6074	0.3359	0.3168	**0.3051**	**0.3051**
RB_U5k_I20k	0.7559	0.6640	0.3912	0.3555	**0.3451**	**0.3451**
A_U200_I120	1.5348	2.0523	1.6563	1.7081	**1.4665**	1.4745
A_U2k_I1k	1.5316	1.4391	1.3116	1.0927	**1.0483**	**1.0485**
A_U20k_I12k	1.4711	1.4241	1.2849	1.0797	**1.0426**	**1.0426**
A_U210k_I120k	1.5072	2.1154	1.5318	1.2915	**1.1671**	**1.1678**

Table 2: Test performance (mean squared error) for recommendation. μ, μ_u, μ_i are the overall bias, user bias and item bias baselines. $\gamma_u.\gamma_i$ is the plain matrix factorization baseline. f_A, f_T are our hybrid recommender systems relying respectively on latent and raw text representations. The different datasets are described in table 1.

3. EXPERIMENTS

All three modules, ratings, text, sentiments, are evaluated independently since there is no global evaluation framework. These individual performances should however provide together a quantitative appreciation of the whole system.

We use two real world datasets of user reviews, collected from *amazon.com* [8] and *ratebeer.com* [11]. Their characteristics are presented in table 1.

Below, one presents first how datasets are preprocessed in 3.1. The benefits of incorporating the text in the ratings prediction for the recommender system are then discussed in section 3.2. The quality of the generated reviews is evaluated and analyzed in section 3.3 Finally, the performance of the sentiment classifier combining text and ratings is described in 3.4.

3.1 Data preprocessing

Reviews from different websites have different formats (rating scales, multiple ratings, ...). We focus on the global rating and scaled it to a 1 to 5 integer range. For titled reviews, the title is considered as the first sentence of the text of the review. Each dataset is randomly split into three parts: training, validation and test containing respectively 80%, 10% and 10% of the reviews.

As described in 2.2, two representations of the text are considered each with a different dictionary:

- for the autoencoder, we have selected the 5000 most frequent words, with a stopwords removal step; The

autoencoder input vector is then a binary vector of dimension 5000.

- for the raw representation, we have selected the 100000 most frequent words appearing in more than 10 documents (including stopwords) and used a binary vector representation.

For the experiments, we consider several subsets of the databases with different numbers of users and items. Each dataset is built by extracting, for a given number of users and items, the most active users and the most commented items. Dataset characteristics are given in table 1.

Subsets	LL	μ_i	$\gamma_u.\gamma_i$	f_A	f_T	LL + f_A	LL + f_T
RB_U50_I200	5.35	5.12	6.01	5.57	5.57	**3.79**	**3.79**
RB_U500_I2k	7.18	10.67	9.73	8.55	8.55	**6.52**	6.92
RB_U5k_I20k	8.44	11.80	10.04	9.17	9.17	**8.33**	**8.35**
A_U200_I120	**10.00**	15.83	22.50	20.00	20.83	**10.00**	**10.00**
A_U2k_I1k	7.89	15.25	12.85	12.62	12.62	**7.54**	**7.54**
A_U20k_I12k	**6.34**	13.99	12.79	12.38	12.37	**6.29**	**6.29**
A_U210k_I120k	**6.25**	14.04	14.40	13.32	13.31	**6.22**	**6.22**

Table 3: Test performance (classification error) as polarity classifiers. LL stands for LibLinear (SVM), μ_i, $\gamma_u.\gamma_i$, f_A, f_T are the recommender systems as in table 2. LL + f_A and LL + f_T are two hybrid opinion classification models combining the SVM classifier and f_A and f_T recommender systems.

3.2 Recommender system evaluation

Let us first consider the evaluation of the rating prediction. The metric used here is the mean squared error (MSE) between rating predictions $\hat{r}_{ui}$ and actual ratings r_{ui}. The lower the MSE is, the better the model is able to estimate the correspondence between user tastes and items. Results are presented in table 2.

The models are referenced using the notations introduced in section 2.2. The first column corresponds to a trivial system which predicts μ the overall bias, the second predicts the user bias μ_u. Both give poor performance as expected.

The third column corresponds to the item bias μ_i baseline. It assumes that user taste is not relevant and that each item has its own intrinsic quality. The improvement with respect to μ and μ_u is important since MSE is halved. The fourth column corresponds to a nonnegative matrix factorization baseline, denoted $\gamma_u.\gamma_i$. It jointly computes latent representations for user tastes and items characteristics. Unsurprisingly, it is our best baseline.

It could be noted that performance tends to degrade when the subset size increases. This is a side effect associated to the review selection process used for building the different datasets. Smaller datasets contain the most active users and the most commented items. The estimation of their profiles benefits from the high number of reviews per user (and item) in this context.

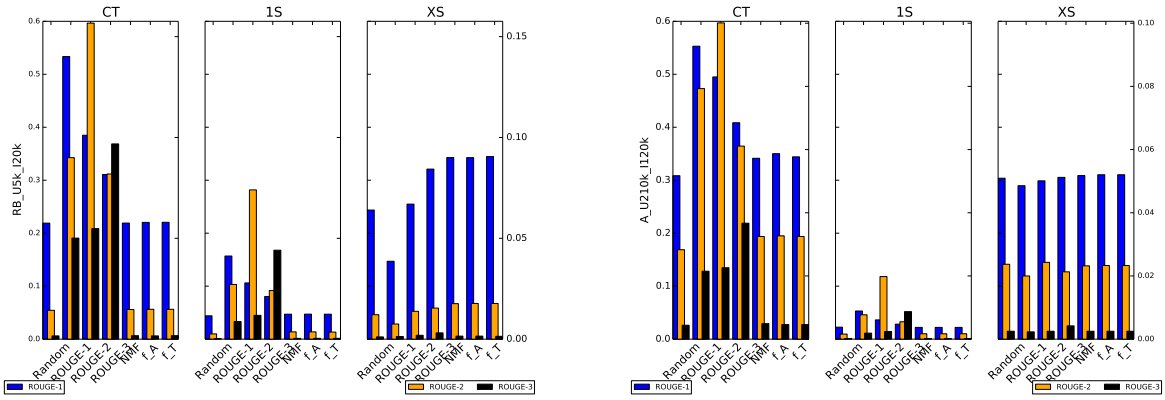
The last two columns refer to our hybrid recommender systems, using the two text representations introduced in section 2.2. Both f_A (autoencoder) and f_T (raw text) perform better than a baseline collaborative filtering system and both have similar approximation errors. The main difference between the systems comes from the complexity of the approach: during the learning step, f_T is much faster than f_A given the fact that no autoencoder optimization is required. On top of that, f_T remains faster in the inference step: the inherent sparsity of the bag of word representation enables f_T to provide faster computations than f_A. The autoencoder works in a smaller dimensional space but it is not sparse.

3.3 Text generation evaluation

We move on now to the evaluation of the personalized review text generation module. Since we are using an extractive summary procedure, we make use of a classical loss used for summarization systems: we use a recall-oriented ROUGE-n metrics, by comparing the generated text against the actual text of the review produced by the user. As far as we know, generating candidate reviews has never been dealt with in this context and this is a novel task. The ROUGE-n metric is the proportion of n-grams of the actual text found in the predicted (candidate) text, we use $n = \{1, 2, 3\}$. The higher ROUGE-n is, the better the quality of the candidate text is. A good ROUGE-1 means that topics or vocabulary are correctly caught while ROUGE-2 and ROUGE-3 are more representative of the user's style.

A first baseline is given by using a random scoring function h (instead of the formulation given in (11)). It provides a lower bound of the performance. Three oracles are then used to provide an upper bound on the performance. They directly optimize the metrics ROUGE-n from the data on the test set.A matrix factorization baseline is also used. It is a special case of our model where no text information is used. This model computes a similar score for all the sentences of a given user and relative to an item. When one sentence only is selected, it is taken at random among the sentences of this user for the item. With greedy selection, the first sentence is chosen at random and then the cosine diversity term (algorithm 1) allows a ranking of the next candidate sentences. Our proposed method is evaluated with the two different user profile π_u representation (auto-encoder and raw text). The performance of these seven models on the the two biggest datasets with respect to the three metrics are aggregated in figure 2.

An histogram corresponds to a text selection entity (whole review text, best single sentence, greedy sentence selection. Groups in the histograms (respectively row block of the tables) are composed of three cells corresponding respectively to the ROUGE-1, -2, -3 metrics. Not surprisingly, the results for the single sentence selection procedure (1S) are always worse than for the other two (CT: complete review and XS: multiple sentences). This is simply because a sentence contains fewer words than a full review and it can hardly share more n-grams than the full text with the reference text. For the *ratebeer.com* datasets, selecting a set of sentences clearly offers a better performance than selecting a whole review in all cases. Texts written to describe beers also describe the tasting experience. Was it in a bar or at home ? Was it a bottle or on tap ? Texts of the community share the same structure and vocabulary to describe both the tasting and the flavors of the beer. Most users write short and precise sentences. This is an appropriate context for our sentence scoring model, where the habits of users are caught by our recommender systems. The performance slightly decreases when the size of the dataset is increased. As before, this is in accordance with the selection procedure of these datasets which focuses first on the most active users and commented items. For Amazon, the conclusion is not so clear and depending on the conditions, either whole reviews or selected



(a) RateBeer experiments

(b) Amazon experiments

Figure 2: Histograms of the performances of the summarizer on the two biggest datasets. The scores of the ROUGE-1 metrics are represented in blue while the scores of the ROUGE-2 and ROUGE-3 metrics are in yellow and black. 7 models are compared: random, 3 oracles, NMF based model, f_A and f_T based models. 3 frameworks are investigated: CT (review extraction), 1S (One sentence extraction), XS (Multiple sentence extraction).

sentences get the best score. It is linked to the higher variety in the community of users on the website: well structured sentences like those present in RateBeer are here mixed here with different levels of English and troll reviews.

The different models, overall, are following a clear hierarchy. First, stating the obvious, the random model has the worst performance. Then, using a recommender system to select relevant sentences helps in terms of ROUGE-n performance. Using the text information brings most of the time only a small score improvement. Overall our models only offer small improvements here with respect to random or NMF text selection (i.e. based on rating similarity only). After analyzing this behavior, we believe that this is due to the shortness of the text reviews, to their relatively standardized form (arguments are very similar from one review to another), to the peaked vocabulary distribution of the reviews, and to the nature of ROUGE. The latter is a classical recall oriented summarization evaluation measure, but does not distinguishes well between text candidates in this context. This also shows that there is room for improvement on this aspect.

Concerning the oracle several conclusions can be drawn. For both single sentence and complete text selection, the gap between the ROUGE measures and the proposed selection method is important suggesting that there is still room for improvements here too. For the greedy sentence selection, the gap between the oracles and the hybrid recommender systems is moderate suggesting that the procedure is here fully efficient. However this conclusion should be moderated. It can be observed that whereas, ROUGE is effectively an upper bound for single sentence or whole review selection, this is no more the case for multiple sentences selection. Because of the complexity of selecting the best subset of sentences according to a loss criterion (which amounts at a combinatorial selection problem) we have been using a sub-optimal forward selection procedure: we first select the best ROUGE sentence, then the second best, etc. In this case the ROUGE procedure is no more optimal.

Concerning the measures, the performance decreases rapidly when we move from ROUGE-1 to ROUGE-2, 3. Given the problem formulation and the context of short product reviews, ROUGE-2,3 are clearly too constraining and the corresponding scores are not significant.

3.4 Sentiment classification evaluation

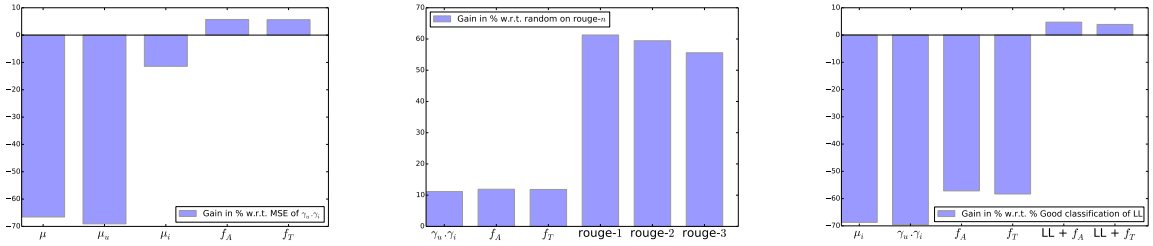
The performance of the different models, using the sentiment classification error as an evaluation metric, are presented in table 3. Because they give very poor performance, the bias recommendation models (μ and μ_u) are not presented here. The item bias μ_i, second column, gives a baseline, which is improved by the matrix factorization $\gamma_u.\gamma_i$, third column. Our hybrid models f_A, fourth column, and f_T, fifth column, have lower classification errors than all the other recommender systems. The first column, LL is the linear support vector machine (SVM) baseline. It has been learnt on the training set texts, and the regularization hyperparameter has been selected using the validation set. Our implementation relies on liblinear (LL) [4].

Its performance is better than the recommender systems but it should be noted that it makes use of the actual text d_{ui} of the review, whereas the recommender systems only use past information regarding user u and item i. Note that even in this context, the recommender performance on RateBeer is very close to the SVM baseline.

It is then possible to combine the two models, according to the formulation proposed in section 2.4. The resulting hybrid approaches, denoted LL + f_A and LL + f_T, exploit both text based decision (SVM) and user profile (f_A and f_T). This combined model shows good classification performance and overcomes the LL baseline in 4 out of 7 experiments in table 3, while performing similarly to LL in the other 3 experiments.

4. OVERALL GAINS

In order to get a global vision of the overall gain provided by the proposed approach, we summarize here the results obtained on the different tasks. For each task, the gain with respect to the (task dependent) baseline is computed and averaged (per task) over all datasets. The metric depends on the task. Results are presented in figure 3.



(a) Recommender systems. Baseline=matrix factorization

(b) Summarizers. Baseline=random selection procedure

(c) Opinion classifiers. Baseline=SVM

Figure 3: Aggregated gains on the 3 tasks w.r.t. classic baselines: our hybrid recommender systems are better overall.

For the mean squared error metric (figure 3a) the matrix factorization is used as baseline. The user bias μ_u heavily fails to generalize on two datasets. The item bias is closer to the baseline (-11.43%). Our hybrid models, which uses texts to refine user and item profiles bring a gain of 5.71% for f_A, 5.63% for f_T. This demonstrates the interest of including textual information in the recommender system. Autoencoder and raw text approaches offer similar gains, the latter approach being overall faster.

For the text generation, we take the random model as baseline and results are presented in figure 3b. The gain is computed for the three investigated framework (CT: review selection, 1S: one sentence selection, XS: multiple sentence selection) and per measure (ROUGE-1, 2, 3) and then averaged to one overall gain. ROUGE-n oracles clearly outperform other models, which seems intuitive. The different recommender systems have very close behaviors with respective gains of 11.15% (matrix factorization), 11.89% (auto-encoder), 11.83% (raw text). Here textual information helps but does not clearly dominate ratings and provide only a small improvement. Remember that although performance improvement with respect to baselines is desirable, the main novelty of the approach here is to propose a personalized summary generation together with the usual rating prediction.

For the opinion classifier, presented in figure 3c, the baseline consists in a linear SVM. Basic recommender systems perform poorly with respect to the baseline (LL). Surprisingly, the item bias μ_i (-68.71%) performs slightly better than matrix factorization $\gamma_u.\gamma_i$ (-69.54%) in the context of sentiment classification (no neutral reviews and binary ratings). Using textual information increases the performance. The autoencoder based model f_A (-57.17%) and raw text approach f_T (-58.31%) perform similarly. As discussed in 3.4, the linear SVM uses the text of the current reviews when the recommender systems does not. As a consequence, it is worth combining both predictions in order to exploit text and past profiles: the resulting models give respective gains of 4.72% (autoencoder) and 3.89% (raw text) w.r.t the SVM.

5. RELATED WORK

Since the paper covers the topics of rating prediction, summarization and sentiment classification, we briefly present each of them.

5.1 Recommender systems

Three main families of recommendation algorithms have been developed [3]: content-based knowledge-based, and collaborative filtering. Given the focus of this work on consumer reviews, we considered collaborative filtering. For merchant websites the goal is to encourage users to buy new products and the problem is usually considered either as the prediction of a ranked list of relevant items for each user [13] or as the completion of missing ratings [9]. We have focused here on the latter approach for evaluation concerns: since we use data collected from third party sources.

5.2 Text summarization for consumer reviews

Early reference work [7] on consumer reviews has focused on global summarization of user reviews for each item. The motivation of this work was to extract the sentiments associated to a list of features from all the item review texts. The summarization took the form of a rating or of an appreciation of each feature. Here, contrarily to this line of work, the focus is on personalized item summaries for a target user. Given the difficulty of producing a comprehensive synthetic summary, we have turned this problem into a sentence or text selection process.

Evaluation of summaries is challenging: how to assess the quality of a summary when the ground truth is subjective? In our context, the review texts are available and we used them as the ground truth. We have used classical ROUGE-n summary evaluation measures [10].

5.3 Sentiment classification

Different text latent representations have been proposed in this scope: [14] proposed a generative model to represent jointly topic and sentiments and recently, several works have considered matrix factorization or neural network, in an attempts to develop robust sentiment recognition systems [6]. [16] go further and propose to learn two types of representation: a vectorial model is learned for word representation together with a latent transformation model, which allows the representation of negation and quantifiers associated to an expression.

We have investigated two kinds of representation for the texts: bag of words and a latent representation through the use of autoencoders as in [6]. [11] also uses a latent representation for representing reviews, although in a probabilistic setting instead in a deterministic one like we are doing here.

5.4 Hybrid approaches

In the field of recommendation, a first hybrid model was proposed by [5]: it is based on hand labeling of review sentences (topic and polarity) to identify relevant characteristics of the items. [11] pushes further the exploitation of

texts, by using a joint latent representation for ratings and textual content with the objective of improving the rating accuracy. These two works are focused on rating prediction and do not consider delivering additional information to the user. Very recently, [19] has considered adding an explanation component to a recommender system. For that, they propose to extract some keywords from the review texts, which are supposed to explain why a user likes or dislikes an item. This is probably the work whose spirit is closest to ours but they do not provide a quantitative evaluation.

[7] combined opinion mining and text summarization on product reviews with the goal of extracting the qualities and defaults. [17] proposed a system for delivering personalized answers to user queries on specific products. They built the user profiles relying on topic modeling without any sentiment dimension. [1] proposed a personalized news recommendation algorithm evaluated on the Yahoo portal using user feedback, but it does not investigate ratings or summarization issues. Overall, we propose in this article to go beyond a generic summary of item characteristics by generating for each user a personalized summaries that is close to what they would have written about the item themselves.

For a long time, sentiment classification has ignored the user dimension and has focused for example on the conception of *"universal"* sentiment classifiers able to deal with a large variety of topics [15]. Considering the user has become an issue only very recently. [18] for example exploited explicit relations in social graphs for improving opinion classifiers, but their work is only focused on this aspect. [12] proposed to distinguish different rating behaviors and show that modeling the review authors in a scale ranging from connoisseur to expert offers a significant gain for an opinion prediction task.

In our work, we have experimented the benefits of considering the text of user reviews in recommender system for their performance as sentiment classifier. We have additionally proposed, as a secondary contribution, an original model mixing recommender systems and linear classification.

6. CONCLUSION

This article proposes an extended framework to the recommendation task. The general goal is to enrich classical recommender systems with several dimensions. As an example we show how to generate personalized reviews for each recommendation using extracted summaries. This is our main contribution. We also show how rating and text could be used to produce efficient personalized sentiment classifiers for each recommendation. Depending on the application, other additional information could be brought to the user. Besides producing additional information for the user, the different information sources can take benefit one from the other. We thus show how to effectively make use of text review and rating informations for building improved rating predictors and review summaries. As already mentioned, the sentiment classifiers also benefits from the two information sources. This part of the work demonstrates that multiple information sources could be useful for improving recommendation systems. This is particularly interesting since several sources are effectively available now at many online sites. Several new applications could be developed along the lines suggested here. From a modeling point of view, more sophisticated approaches can be developed. We are currently working on a multitask framework where the representations used in the different components are more closely correlated than in the present model.

Acknoledgements The authors would like to thank the AMMICO project (F1302017 Q - FUI AAP 13) for funding our research.

7. REFERENCES

[1] D Agarwal, BC Chen, and B Pang. Personalized recommendation of user comments via factor models. *EMNLP'11*, 2011.

[2] M Amini and N Usunier. A contextual query expansion approach by term clustering for robust text summarization. *DUC'07*, 2007.

[3] R Burke. Hybrid recommender systems: Survey and experiments. *UMUAI'02*, 2002.

[4] R-E Fan, K-W Chang, C-J Hsieh, X-R Wang, and C-J Lin. Liblinear: A library for large linear classification. *JMLR'08*, 2008.

[5] G Ganu, N Elhadad, and A Marian. Beyond the Stars: Improving Rating Predictions using Review Text Content. *WebDB'09*, 2009.

[6] X Glorot, A Bordes, and Y Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML'11*, 2011.

[7] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *KDD '04*, page 168, 2004.

[8] N Jindal and B Liu. Opinion spam and analysis. In *WSDM*, pages 219–230. ACM, 2008.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, pages 42–49, 2009.

[10] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *ACL Workshop: Text Summarization Branches Out*, 2004.

[11] J McAuley and J Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. *RecSys'13*, 2013.

[12] JJ McAuley and J Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. *WWW'13*, 2013.

[13] Matthew R McLaughlin and Jonathan L Herlocker. A Collaborative Filtering Algorithm and Evaluation Metric That Accurately Model the User Experience. In *SIGIR'04*, 2004.

[14] Q Mei, X Ling, M Wondra, H Su, and CX Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *WWW*. ACM, 2007.

[15] B Pang and L Lee. Opinion mining and sentiment analysis. *Information Retrieval*, 2008.

[16] R Socher, B Huval, CD Manning, and A Ng. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP'12*. ACL, 2012.

[17] C Tan, E Gabrilovich, and B Pang. To each his own: personalized content selection based on text comprehensibility. In *ICWDM'12*. ACM, 2012.

[18] C Tan, L Lee, J Tang, L Jiang, M Zhou, and P Li. User-level sentiment analysis incorporating social networks. In *KDD'11*. ACM, 2011.

[19] Y Zhang, G Lai, M Zhang, Y Zhang, Y Liu, and S Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. 2014.