Personalized Attention for Textual Profiling and Recommendation

Charles-Emmanuel Dias
Sorbonne Université, CNRS,
Laboratoire d'Informatique de Paris 6,
LIP6, F-75005 Paris, France
charles-emmanuel.dias@lip6.fr

Vincent Guigue
Sorbonne Université, CNRS,
Laboratoire d'Informatique de Paris 6,
LIP6, F-75005 Paris, France
vincent.guigue@lip6.fr

Patrick Gallinari
Sorbonne Université, CNRS,
Laboratoire d'Informatique de Paris 6,
LIP6, F-75005 Paris, France
patrick.gallinari@lip6.fr

ABSTRACT

Classically in recommendation, the goal is to extract information from large numerical feedback matrices. Textual data which often accompany such feedback has proven useful to improve those systems. However text is generally considered as a side feature, appended to matrix factorization algorithms. In this article, we attempt to make a better use of review texts and seek to build a recommender system that does not fully rely on matrix factorization. Our goal is to learn relevant and understandable user and item profiles from both textual data and rating. To do so, We propose to bind two closely related tasks –sentiment analysis and recommendation– by using a double network architecture.
(1) We demonstrate the interest of linking two neural networks to perform both text classification & profile learning.
(2) We introduce a personalized attention mechanism to link them; It enables us to provide built-in explanations by exploiting the underlying learnt textual latent space. This latter point is a noteworthy way to overcome the classical black-box phenomenon in collaborative filtering approaches.

CCS CONCEPTS

• **Information systems → Collaborative filtering**; **Personalization**.

KEYWORDS

collaborative filtering, neural networks, sentiment analysis

ACM Reference Format:
Charles-Emmanuel Dias, Vincent Guigue, and Patrick Gallinari. 2018. Personalized Attention for Textual Profiling and Recommendation. In *Proceedings of SIGIR 2019 Workshop on ExplainAble Recommendation and Search (EARS'19).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In collaborative filtering, one usual objective is to predict the missing values within a rating matrix. Since the Netflix challenge, matrix factorization algorithms are mainly used for such task [18]. Thanks to the use of a growing number of factors such as time [17, 22], social links [12] or text [20], those algorithms became more and more accurate. However, despite their growing accuracy, recent collaborative filtering methods are still widely considered as black-boxes [3]. In fact, many commercial recommender systems are just plain lists of items with little or no explanations (i.e youtube, netflix). This stems from the fact that factorization algorithms mainly extract information from numerical matrices and only treat text as a contextual information [1, 9, 19, 20].

In this paper, we argue that text should be as important as ratings when modeling user and items because it enables built-in explanation. We aim at building a recommender system which is able to, in addition to accurately predict one's rating, provide relevant information to the user on why this specific item should please him. One common way to do so is to append a text generation objective to predict the full review text in addition to the rating [23]. Such generated text is supposed to give a glimpse on what a user could like or dislike about one product. However, generating text has a major issue: it's a computationally intensive hard task. A lot of resources are needed to produce a text with relatively low guarantees on its quality. In fact, items can have very specific attributes which are hard to capture for language models.

Here, alternatively, we propose an extractive mechanism. Our model, instead of directly generating text, should be able to find relevant bits of text within existing reviews. To do so, we draw inspiration from recent advances in sentiment analysis. Multiple papers show that attention mechanisms are able to focus on relevant parts of text in order to accurately predict sentiment [7, 27]. We propose to take advantage of this property of attentional models to directly find important user preferences and item features from existing review texts.

Thus, to learn specific features or preferences into user and item profiles, we build a personalized attention mechanism. It is designed to decouple the focus into a two part attention: One generic, non-conditional, part and a personalized one, conditioned on the (user,item) pair. The goal is to encode specific words and sentences such as attributes or brands within user and item profiles. We use this mechanism to bind two tasks: sentiment analysis and recommendation.

Conveniently, both tasks have the same objective: rating prediction. Binding them with our personalized attention module forces the recommender module to be grounded in textual data and provides regularization. It also gives us a built-in way to sort all existing reviews, sentences and words with respect to a target users preferences which enables us to provide extractive explanations.

EARS'19, July 25, 2019, Paris, France

ACM ISBN 978-1-4503-9999-9/18/06.
https://doi.org/10.1145/1122445.1122456

To sum up, we introduce two original ideas with respect to the literature:

(1) We observe that sentiment analysis and recommendation are two closely related tasks and we tackle them using a bi-net architecture –as it can be done with Siamese networks [5] or GANs[11]– to predict ratings from both texts and user/item profiles. Doing this, we preserve the strengths of both architectures.

(2) We propose a personalized attention mechanism as a link between those two neural networks. Doing this, we naturally bring built-in explanations to the recommender system by selecting the words/sentences highlighted by the attention parameters.

In this paper, we propose an elegant way to improve performance on both sentiment classification & recommendation tasks while providing built-in personalized explanations associated to our suggestions.

This article is organized as follows: As preliminaries (section 2), We first review some key concepts about collaborative filtering and how is our work in line with respect to related work. We also explicit how neural networks with attention are used for sentiment analysis. Then (section 3), we describe our bi-model we use to build profiles (section 3). Finally (section 4), we evaluate both quantitatively and qualitatively learnt profiles.

2 PRELIMINARIES

Here, to model each user and product in an interpretable fashion, we propose to bind a collaborative filtering feed-forward net with a hierarchical recurrent net with attention. In the following section, we first review some key concepts about collaborative filtering. Then, we summarize how recurrent neural networks are used for sentiment analysis and how attention improves this process.

2.1 Collaborative Filtering

Theoretically, Collaborative Filtering (CF) is a way to harness the "wisdom" of the crowd by assuming that people expressing common opinions on a subset of items are likely to have common interests. Since the Netflix challenge [4], the most known CF algorithm is matrix factorization [18]. It frames recommendation as a matrix completion problem which is solved by factorizing this matrix. This decomposition produces users $P = \{\mathbf{p}_u\}_{u=1,\dots,N_u} \in \mathbb{R}^{N_u \times Z}$ and items $Q = \{\mathbf{q}_i\}_{i=1,\dots,N_i} \in \mathbb{R}^{N_i \times Z}$ profiles whose dot product should accurately predict the missing values $\hat{r_{ui}}$.

$$\hat{r_{ui}} \approx \mathbf{q}_i^T \mathbf{p}_u, \qquad \mathbf{q}_i, \mathbf{p}_u \in \mathbb{R}^{Z \times Z} \tag{1}$$

As commonly observed with factorization algorithms, each latent dimension corresponds to an abstract attribute; a matching of local attributes results in a higher rating. Also, in recommendation, ratings are highly biased. Some people tend to always over/under rate and some products can be under/over rated. Classically, those rating bias are modeled by decomposing the full rating into the sum of a global mean μ, a product mean b_i, a user mean b_u and the modeled interaction $q_i^T p_u$. Formally a rating is predicted as such:

$$\hat{r_{ui}} = \mu + b_i + b_u + \mathbf{q}_i^T \mathbf{p}_u \tag{2}$$

With such modeling, latent profiles learn deviations from the mean ratings. Multiple variants exists, taking into account multiple biases or factors such as time [17, 22], social links [12] or text [20]. Constraining learnt profiles with additional information is a common way to improve their regularization and thus, the accuracy of their predictions.

Along with recent NLP progress, the focus of profiling has shifted from rating matrix to review text. Text has the advantage of containing more information while being readily understandable. Pure text-similarity based recommender systems show promising results [6, 9]. Our work is in line with those references, yet, it is original in two ways. First, mixing text analysis and profile learning enables us to consider recommendation as a byproduct of sentiment analysis. Second, instead of trying to generate text to explain recommendation, we propose to use attention over existing reviews.

2.2 Recurrent Neural Networks and Sentiment Analysis

Because of it's sequential nature Recurrent Neural Networks (RNN) are often used to model text [10]. Unlike classical Feed-Forward networks, RNN outputs y_t are conditioned both on the input $\mathbf{x}_t$ at time t and on its own hidden state $\mathbf{h}_{t-1}$ at time $t-1$:

$$\begin{cases} \mathbf{h}_t = \sigma_h(W_h \mathbf{x}_t + U_h \mathbf{h}_{t-1} + b_h) \\ y_t = \sigma_y(W_y \mathbf{h}_t + b_y) \end{cases} \tag{3}$$

where σ refers to a sigmoid activation function. $\mathbf{h}_t$, the hidden state at time t, holds the "memory" of the sequence; It can be seen as a latent representation of the past. However that classical version of the recurrent neural network suffers from a well known vanishing gradient problem. Due to the non-linearities and the number of multiplications, the gradient becomes smaller and smaller when back-propagating through time. This makes it hard to learn long-term dependencies and thus to model long sequences. To overcome this particular problem, gated recurrent units were created such as the Long Short Term Memory (LSTM) [13] or the Gated Recurrent Unit (GRU) [8].

More recently, stemming from machine translation, the concept of neural attention [2] brought major improvement to NLP [25, 27]. It is the cornerstone of the *embed, encode, attend, respond* paradigm [14]. In this approach, the embedding of a whole sequence –into a single representation $\mathbf{e}_s$–, is the weighted sum of every RNN hidden state $\mathbf{h}_t$. For instance, in [27], given a sequence of hidden states $S = \{\mathbf{h}_1, \dots, h_i, \dots, \mathbf{h}_n\}$, the global embedding $\mathbf{e}_s$ is:

$$\mathbf{e}_s = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i, \ \mathbf{t}_i = \tanh(W^{ht} \mathbf{h}_i + b_u), \ \alpha_i = \frac{\exp(\mathbf{a}^\intercal \mathbf{t}_i)}{\sum_i \exp(\mathbf{a}^\intercal \mathbf{t}_i)} \tag{4}$$

Normalized attention weights are computed using the dot product between each RNN hidden state attention space projection $\mathbf{t}_i$ and a learnt attention vector $\mathbf{a}$. This give the opportunity to the algorithm to learn which parts of the sequence are significant.

This architecture constitute state of the art on the sentiment classification task in term of accuracy; it is able to highlight the words or phrases playing a major role in the decision. A first attempt to introduce personalization in this process was proposed by [7], but

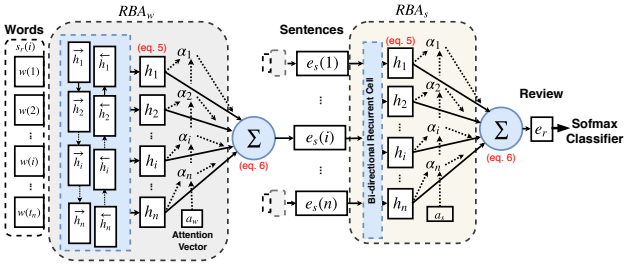


Figure 1: Detailed view of a hierarchical sentiment analysis network as in [27] - It is composed of two RBAs. One –RBA_w– to encode words into sentences and one –RBA_s– to encode sentences into reviews.

they introduced personalization as attributes within the attention projection. On the contrary, we directly personalize the attention vector so we can also exploit it as a collaborative filtering text profile.

3 PARALLEL NETS FOR RECOMMENDATION

Binding text and recommendation has already been done multiple times successfully [6, 9, 19, 22, 23]. In those previous work, three different techniques are opposed:

- Linking factor models to text or topic modeling [1, 22]
- Capturing and using similarities from text reviews [6, 9]
- Appending a text generation objective [23]

Here, we follow the observation that sentiment analysis, when taking into account user and item biases, is in fact implicitly doing recommendation. Theoretically, one can see our approach as a mix between Transnet [6] –Which looks for similarities within writing patterns– and HFT [20] which constrains matrix factorization profiles using latent topics. On one side, we constrain a MLP, on the other we try to learn discriminative features from writing patterns. Hierarchical attentional models for sentiment analysis[7, 27] –personalized or not– can automatically extract discriminant features, which is why we used them as starting point. However, personalization in [7] is minimal: only the projections of words and sentences to attention space is modified by personalization, not the attention vector itself. Here, we do the opposite and personalize the attention vector itself. This way, our personalization can be seen as an embedding and it gives us freedom to link it with other tasks as well.

Our model (fig. 2), which we call Parallel Nets for Recommendation (PNR), is composed of two linked networks:

- $\mathbf{MLP_{rec}}$: A classical multi-layer perceptron (MLP); it takes as input a couple of user and item embeddings to predict a rating, just like matrix factorization in collaborative filtering.
- $\mathbf{RNN_{pol}}$: A hierarchical recurrent neural network composed of two recurrent bi-directional and attentive RNNs (called **RBA** (fig. 1 and detailed in the following); Its role is to encode the reviews by words (level 1) and sentences (level 2) to predict its polarity.

In the following, we first describe how is an **RBA** submodule constructed. Then, we detail how our full system is built; In particular, we will focus on the personalized attention mechanism and the link between the two networks.

3.1 Recurrent Bi-directional network with Attention (RBA)

This module is the main block of our sentiment analysis network. It takes as input a sequence and an attention vector, computes how to weight all elements of the sequence and returns an embedding of it (Fig. 1–RBA_*). Here, how we compute attention is original with respect to the literature as we want it to be a byproduct of the user and item profiles, which should encode preferences and attributes.

Formally, given a sequence $S = \{\mathbf{w}_1, \ldots, \mathbf{w}_i, \ldots, \mathbf{w}_n\}$ of n elements. Its embedding $\mathbf{e}_s$ is computed as follows. First, the sequence is fed through a bi-directional recurrent neural network $RF = \{\overrightarrow{RF}, \overleftarrow{RF}\}$ which encodes the intra-sequence content. Each element representations $\mathbf{h}_i$ (eq. 5) are obtained by concatenating per time-step both RNN outputs. Here, we use LSTM cells [13].

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i}], \qquad \overrightarrow{\mathbf{h}_i} = \overrightarrow{RF}(\mathbf{s}_i), \qquad \overleftarrow{\mathbf{h}_i} = \overleftarrow{RF}(\mathbf{s}_i), \tag{5}$$

Each element $\mathbf{h}_i$ is then projected in a non-linear way into attention space in order to compute its affinity α_i with an attention vector $\mathbf{a}$ in the following way:

$$\mathbf{e}_s = \sum_{i=1}^{n} \alpha_i \mathbf{h}_i, \mathbf{t}_i = \tanh(W^{tu}\mathbf{h}_i + b_u), \alpha_i = \frac{\exp(\mathbf{a}^\intercal \mathbf{t}_i)}{\sum_i \exp(\mathbf{a}^\intercal \mathbf{t}_i)} \tag{6}$$

Finally, these affinities α are normalized using a *softmax* function so that they sum to one, the embedding e_s is the sum of each element, weighted by their affinity to the attention vector a. This attention vector $\mathbf{a}$ – which can be considered as an average representation of all the important information – learns automatically what is discriminant with respect to the task. In order to personalize such attention, we define the vector $\mathbf{a}_g$ as a combination of a global attention vector and a personalized one stemming from the neural network dedicated to rating prediction. Formally, vector $\mathbf{a}$ is defined as such:

$$\mathbf{a} = \tanh(\mathbf{a}_{u,i} + \mathbf{a}_g), \qquad \mathbf{a}_{u,i} = W^{\ell}\ell \tag{7}$$

Where ℓ is an hidden state of the feed-forward rating prediction net. The mapping from one net to another is done through a simple linear transformation W^{ℓ}. Thus, the personalized part attention vector $\mathbf{a}_{u,i}$ stems directly from an interaction between the user and item profiles ℓ (as explained in the following).

The global attention vector aims at encoding general discriminant words for sentiment analysis such as adjectives (i.e *bad, great, awesome*) while its personalized counterpart $\mathbf{a}_{u,i}$ focuses on more specific words, related to one product or one person's interests (i.e *brands, attributes, features...*). Some examples of words selected by this attention mechanism are shown fig 3.

3.2 General Network Architecture

As shown by figure 2, our architecture is composed of two distinctive neural networks. A deep feed forward net for recommendation $\mathbf{MLP_{rec}}$ – which takes as input a user and an item to predict a rating – and a hierarchical recurrent neural network for sentiment analysis $\mathbf{RNN_{pol}}$ – predicting the polarity of review text –. In the following, we first describe the deep recommendation network before describing the recurrent sentiment analysis one .

3.2.1 Multi-Layer Perceptron for Recommendation . The first part of the network –$\mathbf{MLP_{rec}}$–is rather classical as it is a multilayer perceptron. It takes as input a couple of $(user, item)$ embeddings which are concatenated and sequentially transformed into two hidden representations ℓ_w and ℓ_s. Formally, given $\ell = [u; i]$ the concatenation of user and item vectors, the predicted rating $\hat{r}_{ui}$ is obtained by successive transforms, ℓ in ℓ_w, ℓ_w in ℓ_s and finally, ℓ_s in $\hat{r}_{ui}$. Each of these transforms is in itself a parameter. But, for simplicity, we consider in the following a simple two layer deep perceptron with tanh activations (eq. 8).
The objective function of this first network is to minimize the mean squared error of rating prediction.

$$\begin{aligned} \ell_w = \tanh(W^{\ell w}\ell + b_\ell), \ell_s = \tanh(W^{ws}\ell_w + b_w) \\ \hat{r}_{ui} = W^{sr}\ell_s + b_s \end{aligned} \tag{8}$$

3.2.2 Linking the networks with attention. A deep feed-forward neural network produces multiple different hidden vectors during forward propagation. Theoretically, the earliest –resulting of only one (or a few) non-linear transformations– are considered as low-level feature vectors. On the contrary, the latest are viewed as high feature vectors. Therefore, ℓ_w which we want to encode the personalized attention on words from the couple (u, i) should be linked to one of the earliest layer, whereas ℓ_s which encodes the attention over sentences should be linked to more refined features; the latest hidden layers.
To link both networks we propose a simple linear transformation as detailed in eq. 7. The goal is to constrain $\mathbf{MLP_{rec}}$ to produce hidden feature vectors which are related to $\mathbf{RNN_{pol}}$ attentional needs: finding discriminative words and sentences within review text.

3.2.3 Hierarchical Recurrent Net for Sentiment Analysis. The second net –$\mathbf{RNN_{pol}}$– only takes review text as input and tries to predict its associated polarity. It is made of two consecutive RBA, followed by a softmax classification layer. Therefore, It firsts encode hierarchically the text – Word by word then sentence by sentence– in an embedding vector $\mathbf{e}_r$, before being classified. Formally, Each sentences are encoded in $\mathbf{e}_s$ word by word w. Then, each of these sentences are encoded in $\mathbf{e}_r$

$$RBA_w : (\{w\}, \ell_w) \mapsto \mathbf{e}_s \qquad RBA_s : (\{\mathbf{e}_s\}, \ell_s) \mapsto \mathbf{e}_r \tag{9}$$

Finally, the review embedding goes through a final softmax classification layer W^{pred}:

$$p_{pol} = \text{softmax}((W^{pol} e_r) + b_p) \tag{10}$$

The objective function of this net is to minimize negative log likelyhood.

3.2.4 Final layer alignment as regularizer. Our –two layer deep– MLP is aligned with words and sentences embeddings within each RBA's attentional space and directly tries to predict a rating. Thus, its last layer is supposed to be both encoding sentences and the full review. Therefore, we propose to add a third layer to $\mathbf{MLP_{rec}}$. Thus, equation 8 becomes the following:

$$\begin{aligned} \ell_w = \tanh(W^{\ell w}\ell + b_\ell), \quad \ell_s = \tanh(W^{ws}\ell_w + b_w), \\ \ell_s = \tanh(W^{sr}\ell_r + b_r), \quad \hat{r}_{ui} = W^{rec}\ell_s + b_s \end{aligned} \tag{11}$$

Also, Following [6], having two identical embeddings outputs may be beneficial as they both encode the same review. During training, regardless of the network, to enforce such constraint, we minimize the $\mathcal{L}_2$ distance between those two layers. We hope that such constraint will help the MLP learn meaningful transformations.

3.2.5 Hyper-Parameters, Regularization, Objectives and Training. Our models are implemented using *Pytorch*[1]. The two networks have distinct objectives. The first one –$\mathbf{MLP_{rec}}$– predicts the rating by minimizing MSE while the second one –$\mathbf{RNN_{pol}}$– minimizes NLL over ratings. Both objectives – rating regression and polarity prediction – are minimized jointly by mini-batched gradient descent following the adam optimization scheme [16] with and without the last layer regularization as previously detailed.

Hyper-parameters were fixed through our experiments: hidden layers were the same size as word and sentences embeddings: 200 for RBA's and words. 40 for the MLP. (20 for user/20 for item)

Dataset	#Reviews	#Users	#Items	p(1)	p(2)	p(3)	p(4)	p(5)
Instant Video	37 126	5130	1685	4,6	5,1	11,3	22,7	56,3
Digital Music	64 706	5541	3568	4,3	4,7	10,5	25,6	55,0
Video Games	231 780	24 303	10 672	6,4	5,9	12,2	23,6	51,9
Clothes S.J.	278 677	39 387	23 033	4,0	5,5	10,9	20,9	58,6
Movies	1 697 533	123 960	50 052	6,1	6,0	11,9	22,6	53,4

Table 1: Rating stats (in %) of all datasets.

4 EXPERIMENTS

In this paper, our goal is to model users and items using review texts and their associated rating. We aim at creating meaningful profile embeddings that capture preference information. Our model is trained on two separate task –sentiment analysis and recommendation– that we both evaluate in the following. We present beforehand the different datasets used to evaluate our model. Our first set of experiments are of quantitative order, to evaluate whether or not our model is competitive in both learnt task. Then, we propose different qualitative experiments where we tinker with our personalized attention module to visualize what have been learnt by the model. Finally we show how to use what has been learnt by our model to enrich recommendation with existing review data.

4.1 Data and preprocessing

For our experiments we use customer reviews extracted from Amazon [21]. We randomly picked five distinct review databases of different themes and sizes. Dataset statistics are detailed in table 1. Here, we do not evaluate cold-start and use datasets where each user or item have at least 5 reviews.

To tokenize review text and extract sentences we use the NLP library *spacy*[2]. Word embeddings are learnt during training. Only

[1] http://pytorch.org/ – The full code will be released on github
[2] https://spacy.io/

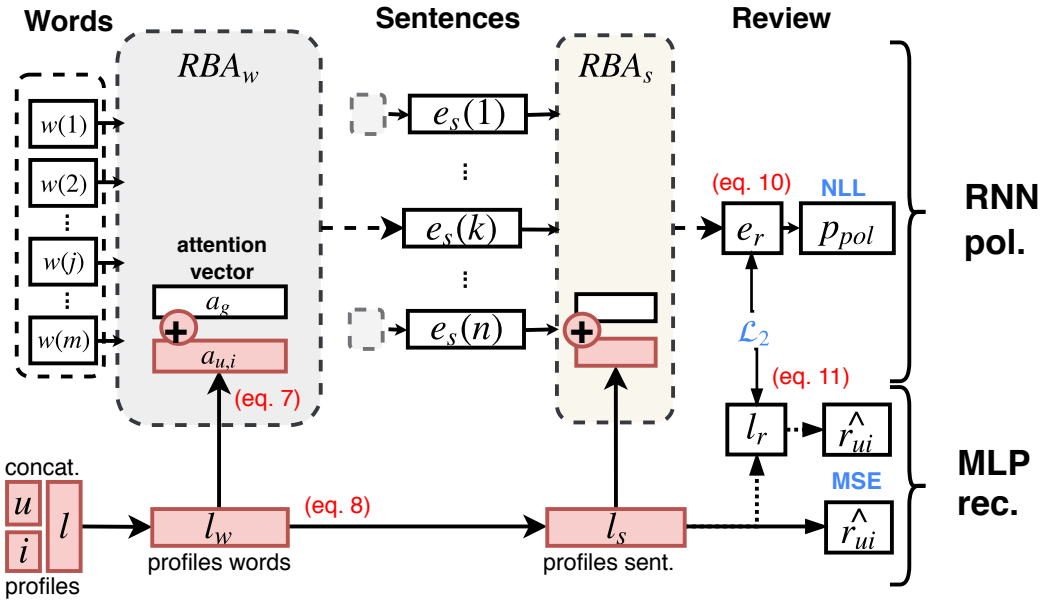


Figure 2: Detailed view of the model for an input of n sentences of m words: (Top) Sentiment RNN, composed of two RBA, one for words (RBA_w) and one for sentences (RBA_s). – (Bottom) Deep Feed-Forward neural net for recommendation.

the 10.000 most used words are considered and embedded, the other ones are replaced by a special token and embedding. Finally, we split datasets in five equal parts to cross-validate our results. Each experiments uses four splits as training (80%), one half split for validation (10%) and the last half for evaluation (10%). This 80/10/10 split is a common way to evaluate recommendation models [1, 20].

4.2 Evaluating Sentiment Analysis:

Our first task is sentiment analysis. The goal is to predict the polarity of a text. We use three distinct baselines which also use text embeddings

- **FastText** [15]. Stemming from the *word2vec* concept, the idea is to learn specific word embeddings for classification. It is largely considered as a strong baseline in text classification.
- **HAN** *Hierarchical Attention Networks for Document Classification* [27]: This model is similar to our recurrent neural model, without personalized attention.
- **NSUPA** *Neural Sentiment Classification with User & Product Attention* [7]: This model – which can be considered as SOTA in sentiment analysis with respect to user reviews– is an evolution of **HAN** to take into account the user and item bias – as we do –. To do so, text is projected in an user-item parametrized attention space. this is equivalent to replacing t_i from equation 6 by $\mathbf{t}_i = \tanh(W^{tu}[h_i; i; u] + b_u)$.
- **SVM** *Support Vector Machines*. SVM are a tough to beat baseline in binary sentiment classification [24]. Also, related work [7, 27] reported bad performance on fine-grained classification task. After verification, SVM performance was indeed poor. Therefore, SVM results are not reported below.

We use accuracy as measure to evaluate sentiment analysis. Results are reported in the table 2. Our first baseline –FastText– is largely beaten by hierarchical models. Taking into account the structure of reviews and sentences seems to yield a non-negligible advantage over a simple aggregation of multiple word embeddings.

When comparing the three hierarchical models, it's clear that adding user and item bias is important. In fact, both personalized model seem to perform as well. However our end goal is more to transfer this learnt personalization to the recommendation task rather than to be competitive on sentiment analysis.

	FastText	HAN†	NSUPA	PNR
Instant Video	62.60	64.50	65.88	**66.60**
Digital Music	63.58	68.03	**70.08**	68.80
Video Games	62.51	67.67	68.60	**69.11**
CSJ	67.83	71.96	**71.99**	71.49
Movies	64.56	68.95	71.20	**71.62**

Table 2: Classification accuracy on the sentiment analysis task - presented values are the mean over 5 splits (en %) – †equivalent to our sentiment net without personalization

4.3 Evaluating recommendation :

The second objective of our model is rating prediction, a standard task in collaborative filtering.

As reference, we use two standard algorithms: Matrix Factorization (MF) as described in eq. 1 and a two-layer deep MLP learned without personalization. They only infer ratings using the learnt user and item profiles on the rating matrix. These methods do not take the text into account. On the contrary, our second reference,

Dataset (#reviews)	Mean (μ)	w/offset	MF	MLP	TransNet	PNR	+ $\mathcal{L}_2$reg
Instant Video (37.126)	1.25	1.137	1.024	0.977	1.174	0.937	**0.936**
Digital Music (64,706)	1.19	0.965	0.903	0.885	1.522	0.876	**0.832**
Video Games (231,780)	1.45	1.281	1.267	1.271	1.313	1.135	**1.117**
CSJ (278,677)	1.215	1.323	1.126	1.172	1.285	1.169	**1.096**
Movie (1,697,533)	1.436	1.148	1.118	1.114	1.359	1.058	**1.030**

Table 3: Mean Squared Error on rating prediction. Baselines are: The global dataset mean μ, the global user-item bias (eq.2), A classic matrix factorization algorithm, a 2-layer MLP (without personalization) and the Transnet model. Presented values are means over five splits.

TransNet[3] [6], only takes the text into account: Rating prediction is viewed as a matching task between all existing user and item text. The rating $\hat{r}_{ui}$ is directly predicted from data without any prior factorization task. Also, as usual, we add trivial mean baselines: One model which always predicts the mean of the train dataset (μ) and one which predicts a user/item corrected mean (*w/offset*). Such baselines represent an easy but surprisingly tough to beat baseline for rating prediction. Results on the rating prediction task are reported table 3. We use the Mean Squared Error as evaluation measure.

With respect to rating prediction our model beats every baselines with or without last layer $\mathcal{L}_2$ regularization. Appending a constrained third layer to the MLP is beneficial. Transnet is always off by a large margin. We believe that the amount of text used for our experiments was not enough to obtain competitive results. Also, our model beats matrix factorization and MLP by a large margin. We argue that it shows the capacity of out model to extract information from text and regularize latent profiles.

4.4 Visualizing attention behavior

Besides accuracy, one major advantages of our model is to use attention which can be exploited to find discriminative data. More precisely, we can extract important words and sentences from the dataset with respect to users preferences.

4.4.1 Word Attention. To extract discriminative words we proceed as follows: For every test sentences, we extract the word which has the maximum attention value. We then obtain a set of all of the most discriminative test words. Those are mainly adjectives such as *great*, *good* or *love*. More interestingly we can analyze these discriminative words with respect to both attention terms: $\mathbf{a}_g$ –general attention– and $\mathbf{a}_{u,i}$ –personalized attention– (cf eq. 7).

For the general attention term, discriminative words are mainly polarized words such as adjectives *good, worst, love*(fig.3 - left). This makes sense as those are the word everyone uses to write polarized reviews. On the contrary, figure 3 (right) shows which words are considered discriminative only with the personalization. We can see that adding a custom factor makes the model focus on more specific words rather than adjectives as with general attention. Here, these words are obviously video game related. We see that some specific entities such as *Wolfenstein, Lara (Croft), Zelda, Sony,*

[3]We used our own implementation on our splits and cross-validated results as we did for our model

Figure 3: Word clouds: (left) Discriminative words for the general part of attention (eq. 7) – (right) Discriminative words for the personalized attention only (eq. 7)

Blizzard, Capcom are extracted. This type of information is exactly what we wanted our profile to learn.

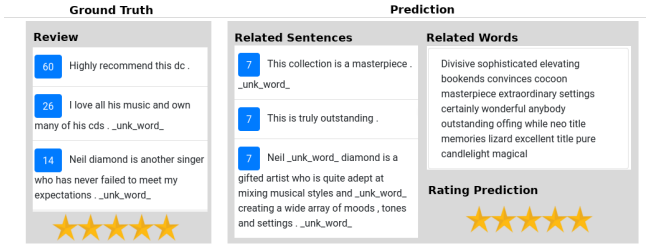
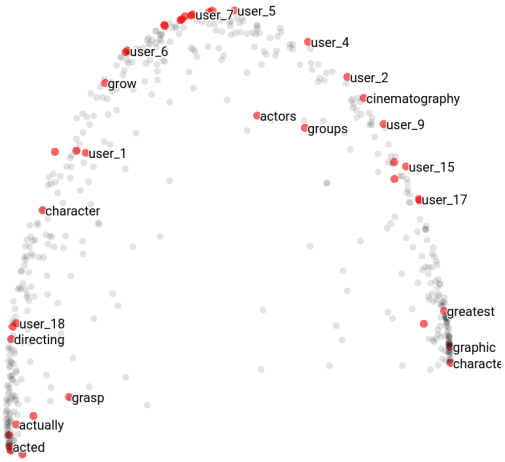


Figure 4: Example –from the test set– of explanations drawn from attention modeling. (Left) a user review and its rating. (Right) Different outputs of our model: the predicted rating, useful sentences –drawn from the target item– and keywords for the user.

4.4.2 Sentence Attention. We can do the same with sentences. For one item, we build a set of all its sentences from all of its existing **train** reviews. Then, we simply use our model in inference with each **test** users by replacing their –unknown– review text by all those sentences. We then obtain an attention score for each of those **train** sentences and are easily able to sort them. An example of extracted sentences is shown (fig.5 - bottom). We can see that while having the same predicted ratings, extracted sentences can be different. Typically, user#4 seem to care about the cast of the show while user#18 seems to care more about the directing and writing. This illustrates how personalized attention works on sentences. User have more affinity to sentences which are similar to what they have previously written. Its worth noting that when visualizing using PCA (fig.5 - top) the related word attention space, the user #18 is indeed closer to directing than the user #4 who is closer to the word "actors".

4.5 Attention as a way to explicit suggestions

Beyond accuracy, being able to explain suggestions is generally considered as important [26]. Commercial recommender system are starting to add some sort of explanation mechanism to ease the black box feeling. Although, those explanations are most of the time item-centric (i.e *this is similar to that*) which mainly due to the abstract nature of learnt profiles.




```
ITEM #487==============================
USER #1 - Rating: 1
--------------------------------------
There was simply nothing
redeeming for this show.
I consider it a waste of my time.
I won't knock it for that.
======================================
USER #18 - Rating: 5
--------------------------------------
Great start !
Great acting, writing and directing.
Really refreshing show,
very unusual writing and comedy.
======================================
USER #4 - Rating: 5
--------------------------------------
Great start !
The cast is great.
humor is deadpan,
satirical spinon media is fantastic.
Great acting, writing and directing.
======================================
```

Figure 5: (top) - PCA view of the word attention space on item #487, attention vectors are labeled "user_#" (bottom) - Most discriminant sentences within all existing train reviews on a single item (#487), for three different test users on the Amazon Instant Video dataset. Ratings are those predicted by MLP_{rec}.

Here, our network being text-based, we are able to propose built-in user-centric explanations, by using our sentiment analysis model as a filter. As shown previously, our personalized attention focuses on specific discriminative words and sentences, with respect to a user tastes and item attributes. By leveraging such capability, we propose two types of explanations: words and sentences of interest extraction. An example using a test (user,item) pair is shown in Figure 4. We argue that extracting personalized discriminative data from existing reviews is a way to mimic a common user behavior, scrolling through reviews.

5 CONCLUSION

Our goal was to model users and items using online review datasets. Mainly, we aimed at adding explanation to the classical recommendation task. Following some recent work on sentiment analysis, we propose a novel personalized attention mechanism which acts as a link between two neural networks. We showed that this attention mechanism, used in an extractive scheme, can bring built-in explanations.

REFERENCES

[1] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 147–154.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Shay Ben-Elazar and Noam Koenigstein. 2014. A Hybrid Explanations Framework for Collaborative Filtering Recommender Systems.. In *RecSys Posters*. Citeseer.

[4] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, NY, USA, 35.

[5] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*. 737–744.

[6] Rose Catherine and William Cohen. 2017. TransNets: Learning to Transform for Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 288–296. https://doi.org/10.1145/3109859.3109878

[7] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 1650–1659.

[8] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[9] Charles-Emmanuel Dias, Vincent Guigue, and Patrick Gallinari. 2017. Text-based collaborative filtering for cold-start soothing and recommendation enrichment. In *AISR2017*.

[10] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[12] Ido Guy. 2015. Social recommender systems. In *Recommender Systems Handbook*. Springer, 511–543.

[13] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[14] Matthew Honnibal. 2016. Embed, encode, attend, predict: The new deep learning formula for state-of-the-art NLP models. https://explosion.ai/blog/deep-learning-formula-nlp.

[15] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Vol. 2. 427–431.

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[19] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 105–112.

[20] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[21] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[22] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 897–908.

[23] Jianmo Ni, Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2017. Estimating Reactions and Recommending Products with Generative Models of Reviews. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 783–791.

[24] Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval* 2, 1–2 (2008), 1–135.

[25] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933* (2016).

[26] Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*. IEEE, 801–810.

[27] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.