RecoNet: An Interpretable Neural Architecture for Recommender Systems

Francesco Fusco[1*], **Michalis Vlachos**[1,2*], **Vasileios Vasileiadis**[3], **Kathrin Wardatzky**[1] and **Johannes Schneider**[5]
[1]IBM Research AI
[2]University of Lausanne
[3]ipQuants AG
[4]University of Liechtenstein
{ffu,hri}@zurich.ibm.com, michalis.vlachos@unil.ch, vv@ipquants.com, johannes.schneider@uni.li

Abstract

Neural systems offer high predictive accuracy but are plagued by long training times and low interpretability. We present a simple neural architecture for recommender systems that lifts several of these shortcomings. Firstly, the approach has a high predictive power that is comparable to state-of-the-art recommender approaches. Secondly, owing to its simplicity, the trained model can be interpreted easily because it provides the individual contribution of each input feature to the decision. Our method is three orders of magnitude faster than general-purpose explanatory approaches, such as LIME. Finally, thanks to its design, our architecture addresses cold-start issues, and therefore the model does not require retraining in the presence of new users.

See a demo video here.

1 Introduction

Recommender systems are a branch of machine learning with the goal of predicting preferred future actions for users, based on their historical preferences and individual traits (e.g. age, gender, location). E-commerce is one area that has been transformed by recommender systems, which provide a solution to the information deluge problem: users are presented only with a list of likely future actions (items to purchase, videos to view, etc.), ranked by estimated interest to the user. Today, recommender systems and machine learning are penetrating and transforming many fields, including business [Heckel *et al.*, 2017], finance [Yu *et al.*, 2008], medicine/biology [Neves and Leser, 2015], and law [Surden, 2014]. In many of these disciplines, it is essential not only to have accurate predictions but also to provide a rationale behind the decision that can be made transparent to the end-user [Lipton, 2016]. Therefore, it is becoming increasingly important to deliver systems that provide both accuracy and interpretability.

Recently, deep-learning architectures have been adapted to the context of recommender systems. These architectures exhibit improvements in predictive accuracy compared with existing state-of-the-art methodologies based on matrix factorization, particularly in the case of cold-start problems. However, deep learning still falls short in terms of *interpretability*, and it is quite challenging to rationalize *why* the recommendation process elected a particular action.

We will address this limitation and propose a neural architecture tailored specifically to recommender systems that can retrace the contribution of the original features leading to a given decision. We achieve this by building on notions from relevance propagation methodologies, which have been successful in the domain of image data [Bach *et al.*, 2015]. Contributions of this work include:

1. We propose a simple neural-network architecture to encode user-item interactions and user or item features. The architecture departs from the traditional inner product calculation, given the learned user and item embeddings. To accommodate for the inherent data sparsity in recommendation settings and to allow the network to generalize better under cold-start conditions [Vartak *et al.*, 2017], we introduce a perturbation mechanism in the training process.

2. Owing to the simple topology, it is also possible to understand fully the individual contributions of the original input attributes for the network's decisions. We achieve this by adapting on ideas of relevance propagation, which propagate backwards the relevance of the network function output through the network. To the best of our knowledge, this is the first effort of its kind in the field of recommender systems.

3. Our experiments show that: (a) The accuracy of the technique is comparable to existing state-of-the-art approaches. (b) Cold-start situations are well addressed. (c) We evaluate the pertinence of the features returned by our methodology by means of significance testing and show their high relevance to the decision made. (d) RecoNet offers equivalent or better interpretability than surrogate methodology approaches such as LIME [Ribeiro *et al.*, 2016], but at a *fraction of the cost*.

2 Our Recommender Setting

Let us assume a set of n users $U = \{u_1, \ldots, u_n\}$ and a set of m items $V = \{v_1, \ldots, v_m\}$. For each user u, we have

*Corresponding authors; work conducted while at IBM Research. From June 2019, M. Vlachos is affiliated with the University of Lausanne, Department of Business and Economics.

collected her preference $\mathbf{R_{uv}}$ for an item v. In our setting, we assume that the preference is inferred from an implicit interaction or feedback (e.g. product purchase, view of a movie, like of an item page), which makes the $n \times m$ matrix $\mathbf{R}$ binary. The problem of generating recommendations based on implicit feedback is known as one-class collaborative filtering (OCCF). We denote the user history by $V(u) = \{v \in V | R_{uv} \neq 0\}$, i.e. the subset of items with which user u interacted.

Let us also assume that content information is available to users and items. User features such as gender, age, location, etc. are usually known, whereas for items, one might have textual information such as reviews, or categorical information such as genres, numerical characteristics or images. We assume that, although we might have users or items with zero interactions (cold-start problem), we have a complete user/item profile. We denote by $\mathbf{\Phi}^U$ and $\mathbf{\Phi}^V$ the sets of user and item features, respectively. The features of a specific user $u \in U$ are denoted by the set $\mathbf{\Phi}^U(u) = \{\phi_{u,1}, \ldots, \phi_{u,k}\}$, where $\phi_{u,i} \in \mathbf{\Phi}^U,\ i = 1, \ldots, k$. We define the set of item features $\mathbf{\Phi}^V(v)$ for a specific item $v \in V$ in a similar way. We assume that a user u can be adequately represented by all the information we have collected about her. Thus, the user's features $\mathbf{\Phi}^U(u)$, interactions $V(u)$, and the features of the items with which the user has interacted $\mathbf{\Phi}^V(V(u)) = \bigcup_{v \in V(u)} \mathbf{\Phi}^V(v)$ will be the input for our model. There is no overlap between items and user or their respective features, ie., the feature sets $\mathbf{\Phi}^V$ and $\mathbf{\Phi}^U$ as well as the item set V are pairwise disjoint.

3 Related Work

Much of the work on recommender systems in the past decade has focused on matrix factorization (MF) approaches [Koren *et al.*, 2009]. MF techniques operate by describing users and items in a latent dimensionality and modelling the interactions between users and items through an inner-product computation. MF can predict ratings or preferences well, but the latent feature space makes it difficult to explain recommendations. Recently, deep neural network (DNN) architectures have also been applied in the area of recommender systems. Several variants of DNNs have been presented previously [He and Chua, 2017; He *et al.*, 2017; Cheng *et al.*, 2016; Shan *et al.*, 2016; X. Wang and Chua, 2017; Zhang *et al.*, 2016]. In these works, we can distinguish two main paths of modeling the structure of the DNN. One path follows ideas built on embeddings, combining the resulting features with deeper pooling or stacking of the resulting neural layers, and the classification is typically accomplished using some variant of a logistic regression or nearest-neighbor search. The second path follows a core idea very similar to matrix factorization and tries to model the interaction between users and items using an inner-product calculation. The difference to MF approaches is that the neural network learns the new latent space of users and items. However, none of the works in recommender systems based on neural networks explicitly address the topic of interpretability. In contrast, our work describes a neural recommender system designed from the ground up to provide interpretable results.

Our network architecture shares commonalities with word-embedding approaches such as word2vec [Mikolov *et al.*, 2013a] and other shallow networks for classification tasks (FastText [Bojanowski *et al.*, 2017] and StarSpace [Wu *et al.*, 2017]). A significant difference is that our work not only predicts a class given a set of features, but also highlights and—more importantly—*ranks* the set of features (i.e. items and attributes) that contributed most to the classification output.

Relevance propagation (or attribution) methods have been proposed and, so far, applied predominantly in the context of images to understand the contribution of each input feature in a deep neural network to the classification output. Attribution methods include approaches such as salience maps [Simonyan *et al.*, 2013; Baehrens *et al.*, 2010], layerwise relevance propagation (LRP) [Bach *et al.*, 2015] and DeepLIFT [Shrikumar *et al.*, 2017]. We review how such approaches operate. Let us consider a network with input $x = [x_1, \ldots, x_n]$, which produces an output of $f(x) = [f_1(x), \ldots, f_C(x)]$, where C is the number of output neurons. Given a target neuron $c \in C$, the purpose is to determine the relevance $R = [R_1, \ldots, R_n]$ of each input feature x_k to the output f_c. Salience maps decompose the gradient-squared norm of the output function as a sum of relevances of the input features, i.e. $\sum_i R_i = ||\nabla f(x)||$, helping to measure how much the changes in each pixel contribute to the prediction. LRP decomposes the classification decision into pixel-wise relevances, i.e. $\sum_i R_i = f(x)$, thus indicating the contributions of each pixel to the overall classification score. LRP operates using a layer-wise conservation principle, forcing the total sum of relevances to be preserved between neurons of two consecutive layers. Finally, DeepLIFT operates in a backward fashion similar to LRP, decomposing the output prediction of a neural network for a particular input by back-propagating the contributions of all neurons in the network to each input feature. An effort to provide a unified view of the proposed relevance propagation methods has appeared in [Ancona *et al.*, 2017].

Unlike attribution methods, surrogate model approaches can be applied to any pre-existing model, i.e., the original model is treated as a *black-box*. We compare our approach with LIME [Ribeiro *et al.*, 2016], a well-established and widely used surrogate technique, in Section 5.5.

4 Our Approach

The main objective of our research is to engineer a recommender system that not only delivers high-quality recommendations, but is also capable of identifying the individual items or additional attributes that contributed to the recommendation. Therefore, our goal is to move beyond existing MF-based approaches, which are not able to rank individual items and features by importance, and beyond current deep-learning recommenders, which are difficult to explain and require black-box explanatory tools to generate the explanations.

One of the main challenges to explainability that affects existing deep-learning recommenders comes from the fact that users, items and attributes are not treated homogeneously. For example, embedding-based approaches project each user and attribute into a different space. In our work, users are identified by a set of items and attributes (i.e. each user can be identified by a bitmap), and the user representation is computed by

averaging embedding vectors corresponding to each one of the items and attributes associated to that user.

Items and attributes are projected in the same embedding space. Specifically, each user is identified by a subset of dense vectors belonging to a space (an embedding layer) of size $d \times (|V| + |\mathbf{\Phi}^U| + |\mathbf{\Phi}^V|)$, where d is a hyper-parameter of the model. Given a sparse input for a specific user consisting of a set of items, item attributes and user attributes, the recommendation is implemented as a classification task associated to a user. So each user is represented as a d-dimensional element-wise *average* of its corresponding vectors (user features, items purchased and item features). This vector is then mapped to the $|V|$ items in which the user might be interested. It is worth noting that this approach eliminates one of the main shortcomings of traditional collaborative filtering approaches, which do not generalize to new users and may require extensions to consider a blending of collaborative and content-based methodologies. In our topology, this task is addressed seamlessly.

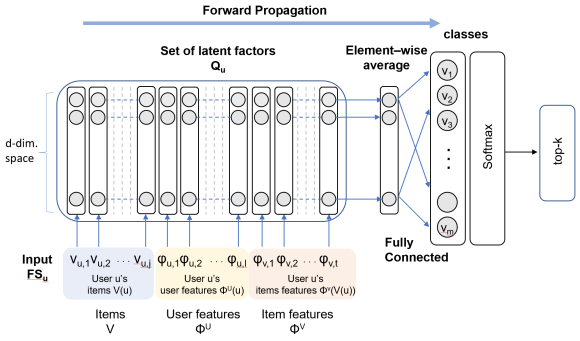

Figure 1: Overview of the RecoNet architecture.

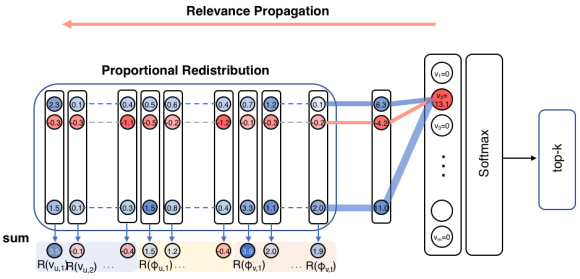

Figure 2: Understanding the importance of each input feature.

4.1 The RecoNet Model Architecture

Our model, called RecoNet, comprises a simple network topology that attempts to balance predictive power with interpretability. We use simple network blocks so that it is feasible to traverse the network backwards and identify the features that contributed to the decision.

For a given user $u \in U$, the feature set $FS_u = V(u) \cup \mathbf{\Phi}^U(u) \cup \mathbf{\Phi}^V(V(u))$ denotes the user's profile, which consists of the union of the user's interactions with the user and item attributes. Let $\mathbf{I}_u \in \{0,1\}^m$ be the corresponding binary vector of the FS_u, where $m = |V| + |\mathbf{\Phi}^U| + |\mathbf{\Phi}^V|$. If $Q \in \mathbb{R}^{d \times m}$ is a latent factor matrix with rows $\mathbf{q}_i$, $i \in [1, \ldots, d]$, then each user can be represented by the average vector of all latent vectors in her feature set, that is

$$\mathbf{x_u} := \frac{Q \cdot \mathbf{I}_u^T}{|FS_u|}.$$

The latent vectors for each feature in FS_u, used for averaging, are illustrated in Figure 1. The representation x_u of a user forms the input of the NN. The NN input is then passed to a fully connected network with $|V|$ units, which produces the final recommendation probabilities with the help of a softmax function:

$$\hat{\mathbf{p}} = \text{softmax}(\hat{\mathbf{y}}) = \text{softmax}(\mathbf{W}\mathbf{x}^T),$$

where $\mathbf{W}$ is a $|V| \times d$ weight matrix. We use the categorical cross-entropy loss. The weights are initialized using the Xavier initialization [Glorot and Bengio, 2010]. Our architecture is inspired by the CBOW architecture for word embeddings [Mikolov *et al.*, 2013b].

An obvious generalization of the model described here would be to add one or more hidden layers between the input layer and the final classes. In that case, $\hat{\mathbf{y}} = f(\mathbf{x})$ where f represents the new model for the classification decision. For the datasets evaluated in our experiment, such a RecoNet model did not produce better results (see Table 1).

4.2 Training

In our topology, items and features are projected in an unified embedding layer, whose parameters are learned during the training process. Learning high-quality representations for items and features is key to providing high-quality recommendations. Our approach uses the complete purchase/preference history of users to *generate* training examples that capture the interactions of users with items. The method we follow to achieve this objective is to concatenate the entire purchase/preference history of the user and then randomly "remove" one item and place it as the label of the training example, keeping the rest of the user's history intact.

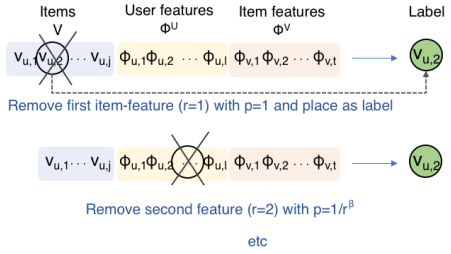

Figure 3: Generating training examples.

To introduce further variability, we also remove more than one item. However, such variations are generated with a progressively lower chance to avoid the risk of deviating too far from the distribution of the training data. Given that r items have already been removed, an additional item is removed with probability $\frac{1}{(r+1)^\beta}$, where the rank is $r = 0$ for the first item

removed, $r = 1$ for the second item removed, etc. A proper value for β for our experiments was determined via cross-validation. Therefore, one item is always removed and placed as the label to be predicted, and additional items are removed with diminishing probability. This process is not confined to the users' preference history, but is also applied to the user and item attributes, allowing the model to generalize better. The above process can be considered a form of dropout, but is applied only to the input layer of the DNN. We illustrate the sampling process in Figure 3. The *intuition* of our sampling approach is that it generates variability, but the distributions for the training and test data are not that different. Only a few training samples with many items removed will ever appear during the training phase, because the probability of an item of rank r being removed is $\frac{1}{(\prod_{i=0}^{r}(i+1))^{\beta}} = \frac{1}{((r+1)!)^{\beta}} \approx \frac{1}{r^{r\beta}}$ (using Stirling's approximation), which declines rapidly with r. The more commonly used uniform sampling, which removes r out of n items with a probability of r/n, is more likely to lead to training samples that promote associations between unrelated items.

4.3 Predictions and Interpretations

We are interested not only in recommending items, but also in ranking the input features that yield a particular recommendation. Our approach therefore requires two steps: (i) a forward step on the network to perform the actual inference, and, (ii) a backward step, which, given the outcome of the network (i.e., the index corresponding to the maximum activation), assigns a ranking score to each individual input feature to measure the feature's relevance to the target prediction.

The simplicity of the model guarantees efficient inferences. In fact, the inference process consists of averaging the embedding vector(s) corresponding to each input feature to a d-dimensional vector x, followed by a matrix-vector multiplication between the weight matrix $\mathbf{W}$ of size $|V| \times d$ of the neural network and the input vector x to the neural network.

To understand which input features contributed to the classification outcome, we build upon the concept of layerwise relevance propagation (LRP) [Bach *et al.*, 2015]. The idea is to distribute the network output activity fully and layer-by-layer from the output layer onto the input features, losing neither positive nor negative evidence. A positive relevance of an input feature means that this feature supports the classification decision, whereas negative relevance values signify that the particular feature value inhibits (and possibly contradicts) the prediction. One of the main benefits of the LRP methodology is that the attribution process happens only at inference time and does not incur computation at training time.

Assume R_k is the relevance of a neuron k for the prediction $f(x)$, and $R_{k\leftarrow j}$ is the contribution of a neuron j of the previous layer to R_k. For our network architecture, back-propagating the relevance involves three propagation steps.

1. *Relevance from classification output to pooling*: Although there are many ways of implementing LRP (known as propagation rules), we use ϵ-LRP when back-distributing the prediction scores in the fully connected part of network. Initially, we zero out all the prediction scores for the non-target classes. This means that, if j is the target class, then $R_i = \sum_{k=1}^{|V|} R_{i\leftarrow k} = R_{i\leftarrow j}$ for the relevances of the pooling layer. The advantage of this approach is that it yields a unique propagation and, therefore, a recommendation explanation for each of the targeted items. In the simple case of a single, linear, and fully connected layer as described in Fig. 1, the attribution of the input x_i to this class is $R_i = R_{i\leftarrow j} = w_{ji}x_i$.

Of course, as described in Section 4.1, one might want to add more layers between the pooling layer and the fully connected layer. In that case,

$$R_i = \sum_j \frac{z_{ji}}{\sum_{i'}(z_{ji'} + b_j) + \epsilon \cdot \text{sign}(\sum_{i'}(z_{ji'} + b_j)} R_j$$

where $z_{ji} = w_{ji}^{(l+1,l)} x_i^{(l)}$ is the weighted pre-activation of a neuron i of layer l to the neuron j of the next layer $l+1$, and b_j is the additive bias of unit j.

2. *Relevance from pooling to latent factors*: For this part of the network, we use a proportional redistribution of the d-relevances $R_1, \ldots, R_d$ from the pooling layer to the latent factors of the feature set FS_u (horizontal distribution of Figure 2). This means that each element $q_{i,k}, i = 1, \ldots, d$ of a latent vector of each original feature ϕ_k, will have a relevance of

$$Z_{i,k} = \frac{q_{i,k}}{\sum_{p=1}^{m} q_{i,p}\mathbf{I}_u(p)} R_i$$

3. Finally, to propagate the *relevance from the latent factors to the original features*, one has to compute the relevance of each original feature ϕ_k, Z_{ϕ_k}, by adding the relevances of all the dimensions of the corresponding latent factor (vertical summation of Figure 2), i.e.,

$$Z_{\phi_k} = \sum_{i=1}^{d} Z_{i,k}$$

5 Experiments

We evaluate two aspects of RecoNet: (a) its predictive power as a recommender system and (b) the validity and quality of the explanations given. We focus our experiments on the implicit recommendation setting because it is the most prevalent (and most difficult) setting. Implicit or one-class problems come with binary ratings, where a value of 1 denotes that the user bought/liked/viewed an item, and 0 denotes an unknown. We use two datasets for our experiments: a proprietary implicit dataset from our institution containing 550,000 purchases (130,000 users and 500 items), called `B2B`. The publicly available `MovieLens` dataset with 1 million ratings (approx. 6,000 users and 4,000 movies). MovieLens contains ratings from 1 to 5. We convert it to a one-class collaborative filtering problem by converting ratings above or equal to 3 into 1 (positive interaction) and ratings lower than 3 into 0 (unknown interaction so far).

5.1 Predictive Power

We split the data using a 60–20–20 split for training, developing and testing, respectively. To be fair to each technique, we performed an extensive grid search to determine good hyper-parameters, and we report the best results achieved by each technique. We allocated at least two days of computational

	Interpretable	MovieLens (w/o attr)	MovieLens (w attr)	B2B (w/o attr)	B2B (w attr)
FM	no	0.3130	0.3177	0.8188	0.8267
OCuLaR	partially	0.3763	**0.3771**	0.8349	**0.8592**
item-2-item	partially	0.2960	-	0.7408	-
user-2-user	partially	0.2657	-	0.8071	-
CML	no	0.3281	0.3060	0.7489	0.8269
RecoNet	yes	0.3294	**0.3508**	0.8317	**0.8589**
RecoNet + Layer	yes	0.3248	0.3186	0.8143	0.8393

Table 1: Comparison of recall@50 across various techniques. The two highest-performing algorithms are highlighted in boldface

time per technique for hyper-parameter searching. To evaluate performance, we adopted the approach used in previous works [He *et al.*, 2017; Hsieh *et al.*, 2017] by using recall-at-k items as the evaluation metric. For our experiments we used a $k = 50$ to evaluate the quality of the best recommendations. For the cold-start experiment we report the recall@k for every value of k. We compare the following techniques:

- *User/item similarity*: Traditional neighborhood-based collaborative filtering techniques compute the similarity between users or items and work well in practice. In addition, they offer partial interpretability by pointing to similar users (or items). We used the *ItemSimilarityRecommender* of the Turicreate package and obtained the best results using cosine similarity.
- *Matrix factorization (MF)*: We use the Overlapping co-Cluster Recommendation (OCuLaR) algorithm [Heckel *et al.*, 2017] as an instance of MF. It is a state-of-the-art technique and has been shown to outperform both wALS [Pan *et al.*, 2008] and BPR [Rendle *et al.*, 2009]. OCuLaR is a non-negative matrix factorization approach that yields partially interpretable recommendations because its explicitly computed factors are able to model user and item participation in the co-clusters, and are interpretable in this manner. However, OCuLaR *does not rank the importance* of features, which RecoNet does.
- *Factorization machines*: Similar to support vector machines (SVM), factorization machines (FM) are a general predictor for real-valued feature vectors. Unlike SVMs, FMs are also able to estimate reliable parameters under very high sparsity as is usually the case in recommender systems. They model all nested variable interactions, but use factorized instead of fully parameterized interactions [Rendle, 2010]. We used the pyFM, which applies stochastic gradient descent with adaptive regularization.
- *Collaborative metric learning (CML)*: CML is a deep-learning technique that uses a joint metric space to encode user–user and item–item similarities [Hsieh *et al.*, 2017].
- *RecoNet*: For RecoNet we also experimented with a variant having an additional layer with ReLu after the pooling layer, which did not yield better results than the simpler RecoNet. Our interpretation of this is that there are no strong nonlinear relationships in the particular setting that would benefit from additional hidden layers.

The summary of results is shown in Table 1 for using and not using the user/items attributes (i.e. using only the users' preference history). RecoNet shows a performance competitive with the best techniques for the datasets used, without sacrificing interpretability—a feature that we highlight next.

5.2 Cold Start

We tested the performance of several techniques under cold-start conditions, that is, when the test set contains users who were not present in the training set. For the B2B dataset from our institution using the time of purchase, we had a natural presence of 35% of users in the test data without a user history in the training data. For the MovieLens dataset, we had to create cold-start situations artificially: 10% of users in the test set were randomly denoted cold-start users, and their history was removed from the training data. The results in Figure 4 show the recall@k (x-axis depicts the k from 1 to 50) and RecoNet exhibits a recall that is comparable with OCuLaR and substantially better than CML and FM.

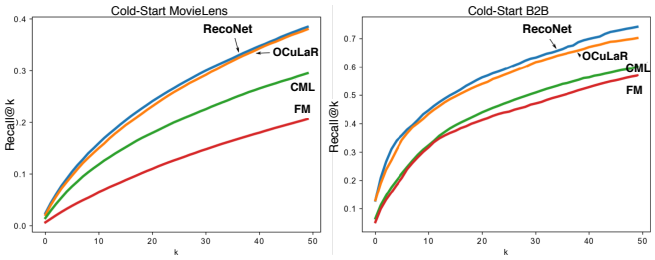

Figure 4: Cold-start performance of RecoNet and other techniques.

5.3 Visual Explanations

We created a demo of RecoNet here for the MovieLens data. The demo also uses speech recognition ("Why do you recommend this movie?") and speech synthesis (enunciating the explanation) to simplify the interaction with the user. We adapt a Sankey diagram to show the importance of the input features for a recommendation. The right-hand side of the diagram shows the item recommended. The left-hand side depicts "flows" that capture the most relevant positive features. The thickness of the flow corresponds to the relevance value attributed to that feature. Figure 5 shows an example of this visualization. It lists the top six positive influences for a user that was recommended to view the movie "Men in Black". Positive endorsements for the recommendation are provided based on *gender* and *age*, as well as previously watched movies in the action and sci-fi genres.

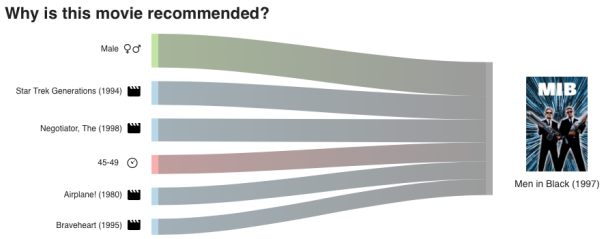

Figure 5: Example of a movie recommendation. The top relevant original features that contributed to this decision are highlighted.

5.4 Evaluating the Prediction

To evaluate the *pertinence* of the features highlighted by RecoNet, we perturb the input by removing the top k most relevant features and measure the corresponding distortion in the classification output. We measure for each k the effect of the perturbation on the target activation, i.e. $\Delta\hat{\mathbf{y}}_c = f_c(x|x_k = 0) - f_c(x)$, where the former part indicates that the top k-th features have been removed from input x. We compare the distribution of $\Delta\hat{\mathbf{y}}_c$ when the top k removed features are selected at random or using the ranking provided by RecoNet. Figure 6 shows the estimated mean difference of the activation of the recommended item after perturbing the top 50 most relevant attributes for both datasets. The gray band represents the 95% confidence interval of the estimated mean. It is obvious that, for both datasets, removing the most relevant features significantly decreases the activation value of the targeted output.

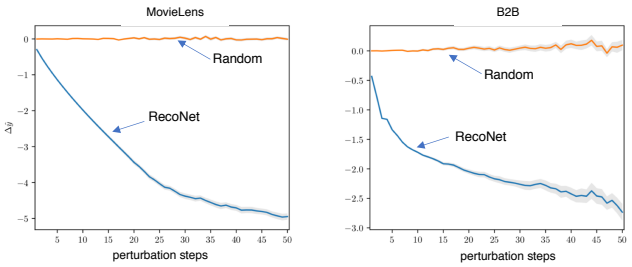

Figure 6: The large drop suggests that removing the features deemed important by RecoNet greatly affects the classification output.

Previous work followed equivalent methodologies [Ancona *et al.*, 2017; Bach *et al.*, 2015] to quantify the validity of the selected features. We provide a stronger result through a formal test of statistical significance. Figure 7 compares the distribution of $\Delta\hat{\mathbf{y}}_c$ when perturbing a random input feature (labeled "random 1") and compares it with the distribution for the case where we perturb the feature that was highlighted as the most important (labeled "top 1"). The shape of the distributions is different: the "top 1'' is negatively skewed, whereas the "random 1" is more symmetric. We test the hypothesis whether the samples come from the same distributions using the Mann–Whitney U test. The p-value is less than 10^{-4} for both datasets; the hypothesis cannot be accepted.

5.5 RecoNet vs LIME

LIME [Ribeiro *et al.*, 2016] is a well-established surrogate technique for interpreting the output of classifiers. LIME

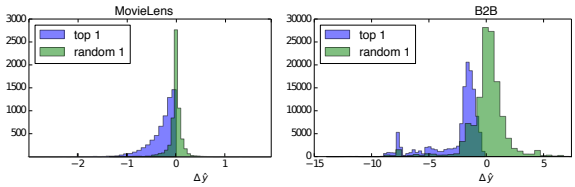

Figure 7: Comparison of the distributions of $\Delta\hat{\mathbf{y}}$ for the top recommendation in two cases: (a) when the most relevant feature ("top 1") is perturbed and (b) when a random feature ("random 1") is modified.

explains a classification instance by building a surrogate explainable model that *locally approximates* the particular classification output. To have a strong baseline for our approach, we implemented an alternate explanation engine built on LIME (and RecoNet as its black-box model). In Figure 8 we repeat the experiment shown in Figure 6, focusing only on RecoNet and LIME. This figure highlights the very good agreement for both techniques among the first four to five features ranked most important. For the remaining features, RecoNet exhibits a larger decrease than LIME in the classifier output when these features are removed, suggesting that the features ranked highest by RecoNet were indeed more important than those discovered by LIME.

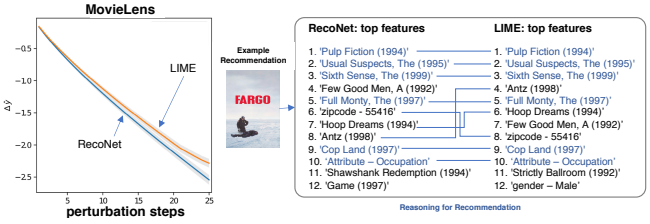

Figure 8: Left: The greater decrease for RecoNet shows that the features selected are more descriptive than those indicated by LIME. Right: A good agreement of feature importance for LIME and RecoNet for a recommendation instance for the movie "Fargo".

RecoNet has been designed from the ground up to provide *interpretable* recommendations without substantially increasing the inference latency. LIME, on the other hand, incurs a large per-inference cost, and it can be impractical for use in industrial and multi-user applications. In fact, for each explanation, LIME probes several *thousands* of inferences on the black-box model. For example, to generate the explanations for each recommendation instance in our setup, LIME required an average of 10–12 seconds. RecoNet required on average 10 milliseconds to perform the inference and generate the explanation; thus, it is more than 1000 times faster.

6 Conclusion

RecoNet is a neural recommender system that (a) exhibits predictive power that is competitive with state-of-the-art recommendation approaches, (b) is explainable, (c) can address cold-start situations. Our experiments suggest that the original features highlighted as relevant are indeed pertinent and statistically significant.

References

[Ancona *et al.*, 2017] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks. *NIPS workshop on Interpreting, Explaining and Visualizing Deep Learning*, 2017.

[Bach *et al.*, 2015] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[Baehrens *et al.*, 2010] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11:1803–1831, 2010.

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 249–256, 2010.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proc. SIGIR*, pages 355–364, 2017.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proc. WWW*, pages 173–182, 2017.

[Heckel *et al.*, 2017] Reinhard Heckel, Michail Vlachos, Thomas P. Parnell, and Celestine Dünner. Scalable and interpretable product recommendations via overlapping co-clustering. In *Proc. IEEE ICDE*, pages 1033–1044, 2017.

[Hsieh *et al.*, 2017] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. Collaborative metric learning. In *Proc. WWW*, pages 193–201, 2017.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[Lipton, 2016] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Neves and Leser, 2015] Mariana Neves and Ulf Leser. Question answering for biology. *Methods*, 74:36–46, 2015.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, N.N. Liu, R. Lukose, M. Scholz, and Qiang Yang. One-class collaborative filtering. In *Proc. ICDM*, pages 502–511, 2008.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *Proc. UAI*, pages 452–461, 2009.

[Rendle, 2010] Steffen Rendle. Factorization machines. In *Proc. ICDM*, pages 995–1000, 2010.

[Ribeiro *et al.*, 2016] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proc. ACM SIGKDD*, pages 1135–1144, 2016.

[Shan *et al.*, 2016] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proc. ACM SIGKDD*, pages 255–262, 2016.

[Shrikumar *et al.*, 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

[Simonyan *et al.*, 2013] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

[Surden, 2014] Harry Surden. Machine learning and law. *Washington Law Review*, 89(1), 2014.

[Vartak *et al.*, 2017] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. A meta-learning perspective on cold-start recommendations for items. In *NIPS*, pages 6904–6914. 2017.

[Wu *et al.*, 2017] Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. Starspace: Embed all the things! *CoRR*, abs/1709.03856, 2017.

[X. Wang and Chua, 2017] L. Nie X. Wang, X. He and T.-S. Chua. Item silk road: Recommending items from information domains to social users. In *Proc. of SIGIR*, pages 185–194, 2017.

[Yu *et al.*, 2008] Lean Yu, Shouyang Wang, and Kin Keung Lai. Credit risk assessment with a multistage neural network ensemble learning approach. *Expert systems with applications*, 34(2):1434–1444, 2008.

[Zhang *et al.*, 2016] Weinan Zhang, Tianming Du, and Jun Wang. Deep learning over multi-field categorical data. In *Advances in Information Retrieval*, pages 45–57, 2016.