Neurocomputing 413 (2020) 23–30

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Hybrid deep neural networks for recommender systems ☆

Mourad Gridach [a,*]

[a] Department of Computer Science, High Institute of Technology, Agadir, Morocco

ARTICLE INFO

Article history:
Received 23 January 2020
Revised 1 June 2020
Accepted 9 June 2020
Available online 16 June 2020
Communicated by Xin Luo

Keywords:
Recommder systems
Deep Neural Networks
Sparsity
Soft logic
Semi-supervised learning

ABSTRACT

More recently, deep neural networks received much attention from the recommender systems community where CNNs and RNNs were applied in different recommendation contexts and achieved state-of-the-art results on many publicly available benchmarks. While the success of the neural models came from their high expressiveness, the uninterpretability problem still one of their main drawbacks, which can have a negative side effect on the whole learning process. A good candidate to reduce the uninterpretability is using Probabilistic Soft Logic (PSL), which showed a strong performance in dealing with this problem. Moreover, sparsity is another major problem of the most previous recommendation systems. In this paper, we introduce a new recommender system framework based on the generalized distillation principle that combines two modeling approaches: PSL for the knowledge-driven modeling approach and deep neural networks for the data-driven approach. Experimental results on publicly available datasets show that our method significantly outperforms the previous state-of-the-art based on deep neural networks emphasizing the utility of PSL rules in handling inconsistencies, reducing the uninterpretability of the neural models and shows promising results in dealing with sparsity.

1. Introduction

In the last decade, with the explosion of e-commerce, there is an increasing number of products and services provided by commercial companies. On the one hand, users have more flexibility in choosing between the huge numbers of articles produced by different companies. On the other hand, making choice between different articles and products became a difficult task for users. In order to guide users to choose the right products, recommender systems are useful information extraction tools able to help users in their choice between large products and services. Nowadays, recommender systems are widely used by customers in their daily life in selecting the best movie to watch, the right paper or book to read, the song to listen to, etc.

The most dominant approach to develop appropriate recommender systems is the collaborative filtering (CF) technique. To make future predictions, this technique exploits the fact that customers who share the same preferences in the past are likely to choose close or similar products in the future. Using matrix factorization [1] based approach, the goal of a CF technique is to find common features that led a user to rate a specific product. For example, in a music recommender system, these features can be singers, writer of the song or music genre. Despite their success, CF techniques have their own limitations and shortcomings. We can consider sparsity problem as the main challenge for these techniques [2]: it is always difficult to recommend products with few ratings or to give recommendations to the users with few ratings.

Another successful technique that has been successfully studied in recommender systems is the method based on probabilistic soft logic [3]. It has been shown that PSL rules have the power to be interpretable, handle inconsistencies and represent similarities. This method exploits the power of probabilistic programming to derive a set of PSL rules to compute similarities between users and items. They rely on the fact that similar users tend to give the same ratings to similar items and similar items should have similar ratings from the same users.

In addition to the previous techniques, some recent work in recommendation tasks [2] used deep neural networks (DNNs) [4–7] since they have achieved state-of-the-art results in many areas such as natural language processing [8–13], speech recognition [14,15] and computer vision [16–18].

Despite the success of deep neural networks in learning useful representations in an end-to-end manner without relying heavily on hand-engineering features, they still have some limitations. The learning process of the neural models depends heavily on massive datasets that cause them to learn uninterpretable and sometimes counter-intuitive features [19].

To tackle the problem of uninterpretability of the neural models, we propose to use the powerful expressiveness of probabilistic

☆ Fully documented templates are available in the elsarticle package on CTAN.

* Corresponding author.
E-mail address: m.gridach@uiz.ac.ma

https://doi.org/10.1016/j.neucom.2020.06.025
0925-2312/

soft logic [20], which has been proven to be very useful in capturing human intention in natural language processing [21–23], computer vision applications such as Semantic Image Interpretation [24], recommender systems [3] and reasoning systems such as causal discovery [25]. It is considered as one of the useful frameworks for reasoning in relational domains where the logical atoms are used to represent the random variables and first-order logic rules are used to capture dependencies between these random variables. The main difference between PSL and classical logic is allowing soft truth values in the interval $[0, 1]$ instead of just two values 0 or 1.

In this paper, we present NeuralPSL a new promising framework that unifies neural networks with probabilistic soft logic for recommender systems. Unifying neural methods with symbolic logic has been explored in various conditions. Donadeloo et al. [24] constructed a framework called Logic Tensor Networks (LTNs) where they were able to integrate deep neural networks with first-order fuzzy logic. LTNs showed to be very useful for tasks like detecting relevant part-of relations between objects in an image and also the classification of an image's bounding boxes. Aditya et al. [26] developed a system that permits explicit reasoning built on the top of a neural network to do visual question answering task where PSL was adopted to build the reasoning layer.

To develop NeuralPSL, we adapt the generalized distillation framework presented by Lopez-Paz et al. [27], which enables the system to learn simultaneously from labeled examples as well as PSL rules. In our problem, the dataset is given by a triplet $\mathscr{D} = \{(x_i, x_i^*, y_i)\}_{i=1}^{n}$, where we consider x_i^* as an additional information about the example (x_i, y_i) and it is given by the PSL rules. Following the same settings, x_i^* is an intelligent teacher that's trying to support the whole learning process. Our framework can be seen as a knowledge transfer from the PSL rules to the neural networks parameters. Furthermore, based on generalized distillation, NeuralPSL shows strong performance in the semi-supervised learning where the model can be enhanced with unlabeled data used with the labeled examples for better exploitation of the PSL rules. NeuralPSL is a very successful approach in dealing with the sparsity problem which is considered as one of the limitations of many previous recommender systems approaches.

We apply our framework to the recommendation task where we used PSL rules defined by Kouki et al. [3], which we unify with a convolutional neural networks architecture as presented in DeepCoNN model [2]. It should be noted that our framework could be extended to other neural networks architectures such as recurrent neural networks (RNNs) or feed-forward neural networks for the recommendation task. We evaluate our approach on two publicly available benchmarks: *Yelp* and *Beer*. Experimental results showed that we were able to outperform all the state-of-the-art models. The results confirm our intuition about the effectiveness of unifying neural models and PSL in the same framework.

We make the following contributions:

1. To the best of our knowledge, this is the first work that unifies neural networks and PSL framework applied to recommender systems;
2. NeuralPSL gives the opportunity for machine learning methods to mix between structured and unstructured approaches, probabilistic and logical inference and data-driven and knowledge-driven modeling;
3. Experimental results show that we were able to get the state-of-the-art results on two publicly available datasets;
4. Our NeuralPSL model shows promising results in dealing with sparsity, which is a major problem in recommender systems.
5. NeuralPSL shows good results on semi-supervised learning.

2. Our approach

In this section, we present the details about our model called NeuralPSL. We begin with neural networks architecture that we followed in the implementation. Then, we highlight the representation of the PSL rules used in our model together with the algorithm behind NeuralPSL. In order to develop NeuralPSL, we adapt the generalized distillation framework [27], which falls into the machine-teaching-machines paradigm. Our model follows the same paradigm where we consider that the first machine is based on the PSL framework while the second is using neural networks. The model is trained to transfer the knowledge encoded in the PSL rules to the weights of the neural network.

2.1. Neural network architecture

We use DeepCoNN architecture developed by Zheng et al. [2] based on convolutional neural networks. It exploits the reviews written by the user in order to reduce the main problem arises by sparsity. More specifically, DeepCoNN uses two parallel convolutional neural networks to compute the rating r_{ui} given by a user u to an item i.

The first CNN called Net_u takes as input the text reviews written by a user u, which consists of a concatenation of all the reviews that user u has written. Similarly, the second CNN called Net_i takes as an input the text written for an item i, which consists of a concatenation of all the reviews that have been written about that item. Both outputs are passed through a convolutional layer followed by a max-pooling layer and fully-connected layer. Dropout [28] is used on the top of the fully-connected layer to prevent overfitting. A vector $z = (x_u, y_i)$ is obtained by concatenating the output from the first CNN denoted as x_u and the second CNN denoted as y_i. The model then passes the vector z through a regression layer consisting of a Factorization Machine (FM) [29] to compute the second order interactions between the two models using the following objective function:

$$L = w_0 + \sum_{i=1}^{|z|} w_i z_i + \sum_{i=1}^{|z|} \sum_{j=i+1}^{|z|} < v_i, v_j > z_i z_j \tag{1}$$

where $w_0 \in \mathscr{R}$ is the global bias, $w = \{w_1, w_2, \ldots, w_n\} \in \mathscr{R}^n$ is a vector of n parameters, and $< v_i, v_j >= \sum_{f=1}^{|z|} v_{i,f}, v_{j,f}$. We note that $< v_i, v_j >$ models the second order interactions, where v_i and v_j describes the ith and jth variables respectively with $|z|$ factors. Finally, the model is trained using the mean squared error (MSE) loss.

2.2. PSL rules representations

PSL is a first-order logical language with the main difference is allowing continuous truth values from the interval $[0, 1]$. To derive the PSL rules [20], a continuous relaxation of Boolean logical operators is used and reformulated using the Lukasiewicz logic as the following:

$$\begin{aligned} P \Rightarrow Q &= max(P - Q, 0) \\ P \wedge Q &= max(P + Q - 1, 0) \\ P &= 1 - P \end{aligned} \tag{2}$$

In a music recommendation setting, we can derive a PSL rule that illustrates users rate songs belonging to their favorite genres. In the following PSL rule, if a user U likes a genre G and a song S belongs to this genre G then the user U tends to rate this song S:

$$LikesGenre(U,G) \wedge Genre(S,G) \Rightarrow Rates(U,S) \quad (3)$$

In this PSL rule, we combine two operators: the logical conjunction operator $\wedge$ and the logical implication operator $\Rightarrow$. Following the first two expressions in Eq. (2), we obtain:

$$Max(LikesGenre(U,G) + Genre(S,G) - Rates(U,S) - 1, 0) \quad (4)$$

We note that *LikesGenre (U, G)* is a binary predicate, *Genre (S, G)* is a continuous predicate that takes values in the interval $[0, 1]$ and *Rates(U, S)* is a continuous variable taking values between 0 and 1, with values close to 1 indicates a higher ratings. Next, these PSL rules are embedded in special functions called potential functions defined as the following: $\phi_l \in \mathscr{X} \times \mathscr{Y} \rightarrow \mathscr{R}$ where $\mathscr{X}$ is a set of input variables, $\mathscr{Y}$ is a set of output variables and ϕ_l represents the lth PSL rule, which has the form: $\phi_l(x,y) = max\{f_l(x,y), 0\}$ where f_l is a linear function of $x \in \mathscr{X}$ and $y \in \mathscr{Y}$. For each potential function, we associate a weight $w_l \in \mathscr{R}$ indicating the importance of the rule in the model. These weights will be learned from the data. We employ the previous settings to define the PSL rules used in NeuralPSL. Taking advantages from the principles of the neighborhood-based approach, these PSL rules enable our model to capture the relationships between users, items, ratings, content and social information. The first PSL rule captures the idea that similar users tend to rate the same item. It is defined as the following:

$$SimilarUsers_{SM}(A,B) \wedge Rates(A,I) \Rightarrow Rates(B,I) \quad (5)$$

where if users *A* and *B* are similar and A rates item *I*, the user *B* will also rate the item *I*. *SM* represents the similarity measure used to compute the similarity between users. The second PSL rule defines the relationship between two items and a user where if the items are similar, then the user will give them the same rating. Similarly, we define this rule using PSL as followed:

$$SimilarItems_{SM}(I,J) \wedge Rates(A,I) \Rightarrow Rates(A,J) \quad (6)$$

where if items *I* and *J* are similar and user *A* rates item *I*, the same user *A* will also rate the item *J*. In the previous two PSL rules, to compute the similarity between users and items, we use the cosine similarity measure. Furthermore, we can use other similarity measures such as the Pearson correlation.

2.3. NeuralPSL

Using generalized distillation principle, NeuralPSL simultaneously learns from both PSL rules and deep neural networks. For the purpose of this paper, we assume that our dataset is given by the triplet $\mathscr{D} = \{(x_i, x_i^*, y_i)\}_{i=1}^n$, where $x \in \mathscr{X}$ is the input variable, $y \in \mathscr{Y}$ is the target variable which could be a real number for a regression problem or a class label in the case of classification and x_i^* as an additional information about the example (x_i, y_i). In the recommendation task, x_i^* will be represented by a PSL rule concerns the user and item introduced at x_i. These rules will capture the similarity between users and items and used as additional information to the input *x*, which will be the user reviews and item reviews fed to the DeepCoNN model, while the output *y* corresponds to the rating.

In general, a neural network defines a conditional probability $P_\theta(y|x)$ parameterized by θ. Standard neural network training has been to iteratively update θ to produce the correct outputs of training instances. Furthermore, we derive a set of PSL rules following the settings in the previous section. Following the same generalized distillation principle, in each iteration, we want to build a teacher network T from the neural model $P_\theta(y|x)$ by integrating the PSL rules. We can formulate our goal in two steps: (1) find the optimal teacher T that satisfy the PSL rules, and (2) ensure that the model P_θ is close to the teacher T in the prediction results.

In the first step, our goal is to guide the model towards desired behaviour by using the PSL rules, which can be seen as constraints functions. We enforce these PSL rules in terms of expectation and we claimed that each PSL rule $\phi_l(x,y)$ is true, which means that $\mathbb{E}_{T(Y|X)}[\phi_l(x,y))] = 1$. Every PSL rule ϕ_l is associated with a weight $w_l \in \mathscr{R}$ to measure its importance. In the second step, the closeness between the teacher T and P_θ is measured using the Kullback–Leibler (KL) divergence where our goal is to minimize this quantity. We unify the two steps in the following optimization problem:

$$\min_{T \in \mathscr{S}} KL(T(Y|X)||P_\theta(Y|X)) - C\sum_l w_l \mathbb{E}_T[\phi_l(X,Y)] \quad (7)$$

where $\mathscr{S}$ denotes the appropriate distribution space; and C is a regularization parameter. There are two ways to specify the weight w_l that measures the relative importance of the PSL rule in the overall framework: the first solution is to specify the values of the PSL rules by associating higher weights to the important rules while less important ones will be associated with smaller weight. The second solution is to learn the weights from the data itself. In our model, we choose the second solution by following the same method in [3]. It should be noted that the problem (7) is convex and has a closed-form solution given by the following:

$$T^*(Y) \propto P_\theta(Y|X) \exp\{C\sum_l w_l \phi_l(X,Y)\} \quad (8)$$

From Eq. (8), we can see that the teacher T will include the neural network prediction $P_\theta(y|x)$ and the PSL rules weighted by the parameter w_l. More specifically, during training, the teacher T is obtained by projecting the neural model P_θ into the feasible space containing the PSL rules. This projection allows the model to incorporate the information encoded in the PSL rules. We use the distillation objective [27] that balances between imitating soft predictions of the teacher T and the true hard predictions:

$$L_D = \underset{\theta \in \Theta}{argmin} \frac{1}{n}\sum_{i=1}^{n}(1-\lambda)l(y_i, \sigma_\theta(x_i)) + \lambda l(s_i, \sigma_\theta(x_i)) \quad (9)$$

where l denotes the cross entropy loss function; n is the training size; σ_θ is the softmax output of P_θ on x_i, s_i is the soft prediction vector of $T^*(Y)$ on x_i. Following the generalized distillation framework, the first term in Eq. (9) is called the primary loss representing the neural network prediction while the second term is called the imitation loss representing the teacher T prediction including the PSL rules. $\lambda \in [0,1]$ is the imitation parameter that balances the importance between the two losses. In the first stages of the training, the teacher T will have poor predictions, therefore we encourage the second term in Eq. (9) to dominate, which means λ must be high, hence $1-\lambda$ must be small. As training goes on, we decay λ which favors the first term and distill the knowledge. Fig. 1 illustrates the NeuralPSL architecture.

3. Experiments

We evaluate our NeuralPSL model on the wide recommender systems benchmarks to demonstrate its effectiveness compared to extensive other state-of-the-art recommender systems algorithms.

3.1. Datasets and evaluation metrics

We evaluate our model on two popular datasets namely Yelp and Beer:

- *Yelp*: This dataset consists of restaurant reviews taken from the 2017 challenge. It contains 4 M reviews and ratings of businesses by around 1 M users.

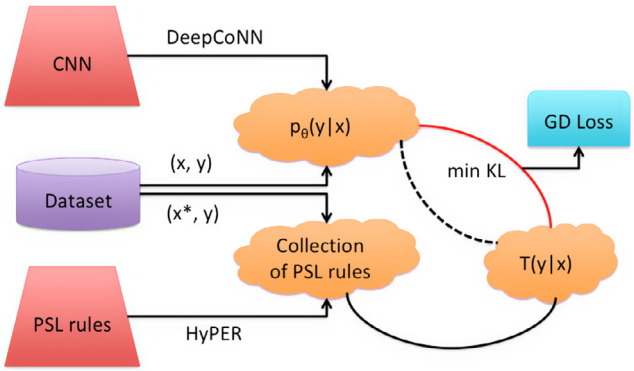

Fig. 1. Our NeuralPSL architecture. The student network $P_\theta(Y|X)$ is obtained from DeepCoNN based model and the teacher network $T(Y|X)$ represents the projection of the student into the feasible space by adapting the collection of PSL rules obtained from HyPER (dashed line). The KL distance is used to measure the closeness between the two networks while GD Loss is the Generalized Distillation loss used to balance between the two predictions (red solid line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

- *Beer*: it is the largest dataset consisting of beer review extracted from *ratebeer.com*. It includes around 3 M reviews gathered from a period of 10 years (until November 2011).

Table 1 presents the details about the two datasets. In our experiments, we use the Mean Square Error (MSE) to evaluate the performance of our model. MSE is the most commonly used metric for many recommender systems. It is giving by the following formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2 \tag{10}$$

where N is the number of training examples, r_i is the ith observed value and $\hat{r}_i$ is the predicted value for the ith example. We note that lower MSE indicates better performance.

3.2. Training details

In deep learning, the choice of the hyperparameters can play an important role in the model performance. We follow the same strategy where the hyperparameters were selected based on performance on validation data. This is a critical step in all deep learning models. As an example, if one choose a high learning rate, the network will overshoot the minimum or by choosing a very small learning rate, the network will take a very long time to find the minimum. In our experiments, we train our models using stochastic gradient descent (SGD) over mini-batches with the Adadelta update rule [30]. Word vectors are initialized using pretrained word embeddings, which are trained on more than 100 billion words from Google News [31] [1]. We have explored the word embeddings trained using GloVe model, but we did not get an improvement over word2vec model. We choose the imitation parameter to be $\lambda = 0.2$ and the regularization parameter to be $C = 300$. All algorithms were trained/tested on NVIDIA GeForce GTX TITAN Xp GPU.

3.3. Baselines

In order to show the important performance of our model compared to the previous systems, we have selected five baselines:

[1] https://code.google.com/archive/p/word2vec/

Table 1
Statistics of the Yelp and Beer datasets.

Dataset	#Users	#Items	#Reviews
Yelp	1,029,432	144,072	4,153,150
Beer	40,213	110,419	2,924,127

Collaborative Deep Learning (CDL), Deep Cooperative Neural Networks (DeepCoNN), Transformational Neural Networks (TransNets), Hybrid Probabilistic Extensible Recommender (HyPER) and Matrix Factorization (MF). The first three models are all based on deep neural networks, the fourth model (HyPER) is based on probabilistic soft logic and the last model is based on matrix factorization techniques. By selecting these five baselines, we will have a mixture of models in terms of using classical techniques, probabilistic soft logic, and deep neural networks.

- *MF*: Matrix Factorization [1] is one of the first techniques used in recommender systems. It is based on Collaborative Filtering method where it uses a rating matrix as input and estimates two low-rank matrices to predict ratings.
- *CDL*: called Collaborative Deep Learning (CDL) is a hierarchical Bayesian model developed by Wang et al. [32] using Stacked De-noising Auto Encoder (SDAE)[33] for learning latent representations of items using their content. Then, it is used as an input to a Collaborative topic regression (CTR) model [34] to predict the ratings. The model was evaluated on three popular datasets namely citeulike-a, citeulike-t, and Netflix where they were able to get the state-of-the-art performance.
- *DeepCoNN*: proposed by Zheng et al. [2] consists of two parallel convolutional neural networks coupled in the last layers. The first network learns user behaviors by exploiting reviews written by the user, and the second network learns item properties from the reviews written for the item. The authors used a shared layer on the top of the CNNs to allow interactions as in factorization machines.
- *TransNets*: Transformational Neural Networks [35] is an extension of DeepCoNN model where the authors introduce an additional latent layer representing the target user-target item pair. At training time, they regularize this latent layer in order to be the same as the latent representation of the current review written by the target user for the target item.
- *HyPER*: called HYbrid Probabilistic Extensible Recommender, developed by Kouki et al. [3]. HyPER reasons over a large range of information sources and it's the most popular model that uses probabilistic soft logic for recommendation systems. They derived a series of PSL rules to capture the relationship between users and items. This relationship includes multiple user-user and item-item similarity measures, content, and social information.

3.4. Experimental results

In order to carry out the experiments, we split the Yelp and Beer datasets into training set representing 80% of the overall dataset, validation set with 10% and the test set representing 10% of the dataset. In the first stage, we evaluate the performance of our model by considering each PSL rule individually with the neural network in order to measure their relative importance for the model. In the second stage, we combine all the PSL rules with the deep neural networks using our framework. Following Kouki et al.[3], the user and item similarities were computed using the PSL rules presented in Section 2.2 where we set the number of neighbors for both users and items to be 50.

Table 2
Performance comparison of our model variants on Yelp and Beer datasets.

Model	DeepCoNN	NeuralPSLItem	NeuralPSLUser	NeuralPSL
Yelp	1.797	1.524	1.597	**1.387**
Beer	0.273	0.253	0.235	**0.222**

Table 2 shows the recommendation performance of our model on the two datasets. We developed two variants of NeuralPSL namely, *NeuralPSLItem* based on Item PSL rules while ignoring similarities in the user side. The second variant of our model called *NeuralPSLUser* based on User PSL rules without exploiting the similarities between items. The results show that with just one PSL rule leading to two variants of our model namely NeuralPSLItem and NeuralPSLUser provide a strong boost on performance over the two datasets compared to the DeepCoNN baseline model. Further comparison between the two variants of our model showed that the model called NeuralPSLItem, which exploits the Item-based PSL rules, performs better on both Yelp and Beer datasets compared to NeuralPSLUser, which exploits the User-based PSL rules. Therefore, combining item and user PSL rules lead to our general model *NeuralPSL*. As we expect, combining all the PSL rules gives the best performance on both datasets.

Moreover, we compare our model with the best previous recommender systems models namely FM [1], DeepCoNN [2], Collaborative Deep Learning (CDL) [32], TransNets [35] and HyPER [3]. Table 3 shows the comparison results on the two datasets. The last column indicates the percentage of improvements gained by NeuralPSL compared to the best baseline in the corresponding dataset.

On Yelp and Beer datasets, TransNets model outperformed all the other baselines namely FM, HyPER, DeepCoNN, and CDL models. Furthermore, a comparison between TransNets and NeuralPSL, our model achieved 15.3% improvements on Yelp dataset and 12.9% improvement on the Beer dataset. We note that the average improvement over the two datasets is 14.1%. From the previous results, we can see that our model outperforms all the models by a good margin and gets state-of-the-art results on the Yelp and Beer datasets.

3.5. Performance evaluation

It should be noted that the performance of NeuralPSL, all its variants, and the baselines are reported in terms of MSE (see Table 3). In Table 3, all models perform better on Beer dataset than on Yelp. These performances are mainly related to the sparsity of Yelp, which confirms all previous results obtained by recommender systems on sparse datasets. We can see that MF performs worse than all the models. Moreover, we note that almost all the models based on deep neural networks (TransNets and DeepCoNN) perform better than the other baselines which confirm the usefulness of these models in the recommendation task.

One important remark from the previous results is the competitiveness of the HyPER model showing the importance of probabilistic soft logic in these settings. We note that HyPER outperformed Collaborative Deep Learning (CDL) on both datasets. With the capability of unifying probabilistic soft logic and deep neural networks, NeuralPSL showing an important improvement over all the baselines, especially the two models that form NeuralPSL namely HyPER and DeepCoNN. We note an average improvement of 34% and 32% over HyPER and DeepCoNN respectively. It validates our hypothesis that unifying the two approaches can significantly improve rating prediction.

3.6. NeuralPSL and semi-supervised learning

Moreover, we investigate the effectiveness of our NeuralPSL framework in exploiting the unlabeled examples to allow semi-supervised learning. Our overall vision is that using unlabeled data will help the system to improve its performance by exploiting the derived PSL rules for both similarities between items and users. It should be noted that allowing semi-supervised learning will have a very important effect to solve previous issues in recommender systems such as sparsity [2] and cold-start [36] problems.

To evaluate the performance of our model in allowing semi-supervised learning, we conduct the following experiments: we select respectively 10%, 15% and 20% of the Yelp training set to be unlabeled data while we keep the test set as the same in the previous experiments. In the next step, we evaluate our model on the three training datasets and compare the results with DeepCoNN baseline. Table 4 shows the performance of both models on the selected training datasets. As shown in the results, NeuralPSL outperformed significantly the DeepCoNN baseline in all the three datasets. From Table 4, our model obtained an improvement of 33%, 31% and 34% over DeepCoNN on the 10%, 15%, and 20% selected datasets respectively leading to an average of 32.67%. These obtained results confirm our intuition about the contribution of the PSL rules in the semi-supervised settings.

Further investigation shows that NeuralPSL obtained higher improvement over DeepCoNN on 15% dataset compared to 10% and 20%. While we do not have a reasonable answer of the NeuralPSL performance on the selected 15% dataset, we believe that 75% of the training data was sufficient for the model to learn almost the useful representations for the recommendation task.

Table 4
The performance of our model (NeuralPSL) in MSE on three selected datasets from Yelp representing respectively 10%, 15% and 20% from the training set with a comparison to the DeepCoNN baseline model.

Dataset	DeepCoNN	NeuralPSL	Improvement
10%	1.792	1.194	33%
15%	1.687	1.156	31%
20%	1.803	1.189	34%
Average	1.76	1.18	32.67%

Table 3
Performance comparison using MSE of NeuralPSL with the best previous models namely Matrix Factorization, CDL, DeepCoNN, HyPER and TransNets on Yelp and Beer datasets. Best results are indicated in bold.

Model	MF	CDL	DeepCoNN	HyPER	TransNets	NeuralPSL	Improvement
Yelp	1.992	1.885	1.797	1.841	1.638	**1.387**	15.3%
Beer	0.612	0.299	0.273	0.275	0.255	**0.222**	12.9%
Average	1.302	1.092	1.035	1.058	0.947	**0.699**	14.1%

3.7. NeuralPSL and sparsity

In this section, we show how useful NeuralPSL is in reducing the sparsity problem, which is considered as one of the main issues in almost all the previous recommender systems. To show the robustness of NeuralPSL on sparse datasets, we create three subsets of the training data. For this reason, we respectively remove all the reviews of 20%, 50% and 80% of users and items randomly to create more sparse datasets compared to the original datasets. Moreover, these new sparse datasets contain a fewer number of users and items, which induce cold-start users and items. Table 5 shows the statistics about the new sparse datasets.

To validate the usefulness of our method on sparse datasets, we compare NeuralPSL with TransNets model on the three sparse datasets derived from both Yelp and Beer. Table 6 shows the results. As shown in the table, NeuralPSL outperforms TransNets on all sparse datasets with a good margin. More importantly, the difference between the MSE of NeuralPSL and TransNets increases as the number of reviews decrease for both Yelp and Beer datasets, which means that the level of sparsity becomes more crucial: while our NeuralPSL model only improves by −0.275 and −0.027 respectively on Yelp and Beer over TransNets on the less sparse dataset namely Sparse20, the improvement is significant on the more sparse dataset (Sparse80) with a gain of −0.463 and −0.401 respectively on Yelp and Beer.

Moreover, to show the interesting success of NeuralPSL, we select Sparse50 of both Yelp and Beer datasets to study the performance of our model over users and items with a different number of user and item reviews and compare the results with TransNets model. Fig. 2 shows plots of MSE of NeuralPSL and TransNets over Yelp dataset while Fig. 3 shows plots on Beer dataset. On both datasets, we notice that when the number of reviews is very large, the two models converge to almost the same MSE values. The most important observation from the two figures is when the number of reviews is small, the performance gain of NeuralPSL over TransNets is very large. These results confirm that our method is very useful when we have fewer reviews (cold-start users and items).

Fig. 2. The MSE values over the different number of reviews on Yelp dataset.

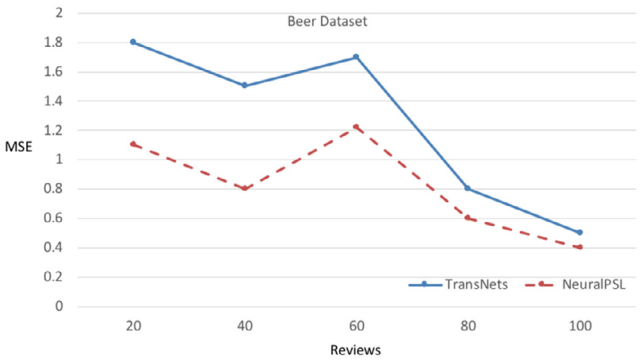

Fig. 3. The MSE values over the different number of reviews on Beer dataset.

4. Related work

In the last decade, there is an explosion of papers focusing in recommender systems regarding the importance of this task in many industry areas. In this section, we give a short overview on two recommender systems approaches: the first approach uses deep neural networks while the second approach is based on knowledge driven approach in order to differentiate our framework with these two approaches. Most of the previous recommendation models were mainly based on collaborative filtering techniques where the recommendation task is considered to be a two-way interaction between users preferences and items features, but more recently, with the success of deep learning (DL) in areas such natural language processing, computer vision and speech recognition, there have been much interests using DL in recommender systems exploiting the powerful expressiveness of DL architectures such as CNNs and RNNs. These architectures are able to learn useful representations and capture complex relationships within the data.

Luo et al. [37] proposes a new model called alternating direction method (ADM)-based nonnegative latent factor (ANLF) for recommendations. To get high convergence rate with lower complexity, the authors implement the ADM-based optimization with regard to each single feature. The performances of the model were demonstarted on various datasets. Seo et al. [38] developed a model based on CNN similar to DeepCoNN where they added

Table 5
Statistics of the two datasets created for sparsity reason.

	Sparse20			Sparse50			Sparse80		
	#Users	#Items	#Reviews	#Users	#Items	#Reviews	#Users	#Items	#Reviews
Yelp	818,398	114,537	3,301,754	514,611	71,987	2,069,897	200,739	28,094	809,864
Beer	31,969	87,783	2,324,680	20,056	55,189	1,461,989	7,841	21,531	570,204

Table 6
The performance of our NeuralPSL model compared to TransNets over the sparse datasets.

	Yelp			Beer		
Models	Sparse20	Sparse50	Sparse80	Sparse20	Sparse50	Sparse80
TransNets	1.681	1.934	2.216	0.253	0.516	1.016
NeuralPSL	1.406	1.631	1.753	0.226	0.306	0.615

attention mechanism to build the latent representations, the inner product of which gave the predicted ratings. AlMahairi et al. [39] developed two models: Language Model regularized Latent Factor model (LMLF) and Bag-of-Words regularized Latent Factor model (BoWLF). Furthermore, they use matrix factorization approach [40–42] in order to learn the latent factors and likelihood of the review text. In a music recommendation task, den Ooord et al. [43] proposed a model based on deep convolutional neural networks able to generate the latent factors of items from the content, audio signals using the Million Song Dataset. The predicted latent factors of the item were used with the latent factors of the user where they were able to produce sensible recommendations.

In addition to deep learning approaches, knowledge driven approach were successfully used in recommender systems where they exploited the powerful interpretability of the logic rules. HyPER is one of the most popular models developed by [3] where they use probabilistic soft logic to derive a set of rules to capture relationship between users and items. Rose and Cohen [35] proposed a model using a general-purpose probabilistic logic system called Programming with Personalized PageRank (ProPPR). Exploiting the information included in the entities together with the link structure of the knowledge graph allowed the system to improve the prediction task. Hoxha and Rettinger [44] used Markov Logic Networks (MLNs) [45,46] to combine content with collaborative filtering approach. Their model benefits from both theoretical foundations as well as empirical results. Compared to the two previous approaches, our NeuralPSL model takes advantages from both the powerful interpretability of the logic rules as well as the strong performances obtained by training deep neural networks to learn useful representations. Moreover, our model has the opportunity to mix between "structured and unstructured approaches", "probabilistic and logical inference", and finally "the data-driven and knowledge driven modeling".

5. Conclusion

In this paper, we presented NeuralPSL – a new promising framework that combines probabilistic soft logic with deep neural networks for recommender systems. Seeking to solve the uninterpretability problem of the neural models and using the powerful expressiveness of logic rules and their interpretability to derive new knowledge from the data, we use generalized distillation method to transfer this knowledge encoded in the PSL rules into the weights of the neural networks. To derive the PSL rules for the recommendation task, we use the similarity principle between users and items following the intuition that similar items should have similar ratings from the same users while similar users give similar ratings to the same items. The advantages of our framework is its ability to combine different machine learning approaches: the data-driven and knowledge-driven approaches, we can also add the opportunity of unifying the structured and unstructured approaches. While our framework gives remarkable results on recommendation task, relying on specific number of PSL rules is one of the main limitations of this method. Deriving these rules is not an easy task because it requires domain specific knowledge and a form of hand-crafted engineering features. In the future work, we are planning to apply this framework to other tasks such as natural language processing and computer vision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (8) (2009) 30–37. .

[2] L. Zheng, V. Noroozi, P. S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, ACM, 2017, pp. 425–434..

[3] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, L. Getoor, Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems, in: Proceedings of the 9th ACM Conference on RecSys, ACM, 2015, pp. 99–106..

[4] P.M. Kebria, A. Khosravi, S.M. Salaken, S. Nahavandi, Deep imitation learning for autonomous vehicles based on convolutional neural networks, IEEE/CAA J. Autom. Sin. 7 (1) (2019) 82–95.

[5] J. Wang, T. Kumbasar, Parameter optimization of interval type-2 fuzzy neural networks based on pso and bbbc methods, IEEE/CAA J. Autom. Sin. 6 (1) (2019) 247–257.

[6] M. Gridach, H. Haddad, Arabic named entity recognition: a bidirectional gru-crf approach, in: International Conference on Computational Linguistics and Intelligent Text Processing, Springer, 2017, pp. 264–275.

[7] M. Gridach, Deep learning approach for arabic named entity recognition, in: International Conference on Intelligent Text Processing and Computational Linguistics, Springer, 2016, pp. 439–451.

[8] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Proceedings of NIPS, 2014, pp. 3104–3112.

[9] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches, arXiv preprint arXiv:1409.1259. .

[10] M. Gridach, Character-level neural network for biomedical named entity recognition, J. Biomed. Inf. 70 (2017) 85–91.

[11] M. Gridach, H. Haddad, H. Mulki, Churn identification in microblogs using convolutional neural networks with structured logical knowledge, in: Proceedings of the 3rd Workshop on Noisy User-generated Text, 2017, pp. 21–30.

[12] M. Gridach, Character-aware neural networks for arabic named entity recognition for social media, in: Proceedings of the 6th Workshop on South and Southeast Asian Natural Language Processing (WSSANLP2016), 2016, pp. 23–32. .

[13] M. Gridach, H. Haddad, H. Mulki, Empirical evaluation of word representations on arabic sentiment analysis, in: International Conference on Arabic Language Processing, Springer, 2017, pp. 147–158.

[14] G. Hinton, L. Deng, D. Yu, G.E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.

[15] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, IEEE Trans. Audio Speech Lang. Process. 20 (1) (2012) 30–42.

[16] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of CVPR, 2014, pp. 580–587. .

[17] A. Karpathy, L. Fei-Fei, Deep visual-semantic alignments for generating image descriptions, in: Proceedings of CVPR, 2015, pp. 3128–3137. .

[18] M. Gridach, I. Voiculescu, OXENDONET: a dilated convolutional neural networks for endoscopic artefact segmentation, Vol. 2595 of CEUR Workshop Proceedings, CEUR-WS.org, 2020, pp. 26–29. .

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199. .

[20] S.H. Bach, M. Broecheler, B. Huang, L. Getoor, Hinge-loss markov random fields and probabilistic soft logic, J. Mach. Learn. Res. 18 (109) (2017) 1–67.

[21] T. Rocktäschel, S. Singh, S. Riedel, Injecting logical background knowledge into embeddings for relation extraction, in: Proceedings of NAACL, 2015, pp. 1119–1129.

[22] Z. Hu, X. Ma, Z. Liu, E. Hovy, E. Xing, Harnessing deep neural networks with logic rules, arXiv preprint arXiv:1603.06318. .

[23] T. Demeester, T. Rocktäschel, S. Riedel, Lifted rule injection for relation embeddings, in: Proceedings of EMNLP, 2016.

[24] I. Donadello, L. Serafini, A. d. Garcez, Logic tensor networks for semantic image interpretation, arXiv preprint arXiv:1705.08968. .

[25] D. Sridhar, J. Pujara, L. Getoor, Using noisy extractions to discover causal knowledge, arXiv preprint arXiv:1711.05900. .

[26] S. Aditya, Y. Yang, C. Baral, Explicit reasoning over end-to-end neural architectures for visual question answering, in: AAAI, 2018. .

[27] D. Lopez-Paz, L. Bottou, B. Schölkopf, V. Vapnik, Unifying distillation and privileged information, arXiv preprint arXiv:1511.03643. .

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[29] S. Rendle, Factorization machines with libfm, ACM Trans. Intell. Syst. Technol. (TIST) 3 (3) (2012) 57. .

[30] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701. .

[31] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of NIPS, 2013, pp. 3111–3119.

[32] H. Wang, N. Wang, D.-Y. Yeung, Collaborative deep learning for recommender systems, in: Proceedings of the 21th ACM SIGKDD, ACM, 2015, pp. 1235–1244.
[33] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (2010) 3371–3408.
[34] C. Wang, D. M. Blei, Collaborative topic modeling for recommending scientific articles, in: Proceedings of the 17th ACM SIGKDD, ACM, 2011, pp. 448–456.
[35] R. Catherine, W. Cohen, Transnets: Learning to transform for recommendation, arXiv preprint arXiv:1704.02298. .
[36] M. Volkovs, G. Yu, T. Poutanen, Dropoutnet: addressing cold start in recommender systems, in: Proceedings of NIPS, 2017, pp. 4964–4973.
[37] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, Q. Zhu, A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method, IEEE Trans. Neural Networks Learn. Syst. 27 (3) (2015) 579–592.
[38] S. Seo, J. Huang, H. Yang, Y. Liu, Representation learning of users and items for review rating prediction using attention-based convolutional neural network, in: 3rd International Workshop on Machine Learning Methods for Recommender Systems (MLRec) (SDM-17), 2017. .
[39] A. Almahairi, K. Kastner, K. Cho, A. Courville, Learning distributed representations from reviews for collaborative filtering, in: Proceedings of the 9th ACM Conference on Recommender Systems, ACM, 2015, pp. 147–154.
[40] X. Luo, M. Zhou, S. Li, M. Shang, An inherently nonnegative latent factor model for high-dimensional and sparse matrices from industrial applications, IEEE Trans. Ind. Inf. 14 (5) (2017) 2011–2022.
[41] X. Luo, J. Sun, Z. Wang, S. Li, M. Shang, Symmetric and nonnegative latent factor models for undirected, high-dimensional, and sparse networks in industrial applications, IEEE Trans. Ind. Inf. 13 (6) (2017) 3098–3107.
[42] X. Luo, M. Zhou, S. Li, Y. Xia, Z.-H. You, Q. Zhu, H. Leung, Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing qos data, IEEE Trans. Cybern. 48 (4) (2017) 1216–1228.
[43] A. Van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, Adv. Neural Inf. Process. Syst. (2013) 2643–2651.
[44] J. Hoxha, A. Rettinger, First-order probabilistic model for hybrid recommendations, in: Machine Learning and Applications (ICMLA), 2013 12th International Conference on, vol. 2, IEEE, 2013, pp. 133–139. .
[45] M. Richardson, P. Domingos, Markov logic networks, Mach. Learn. 62 (1) (2006) 107–136.
[46] A.K. Sangaiah, D.V. Medhane, T. Han, M.S. Hossain, G. Muhammad, Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics, IEEE Trans. Ind. Inf. 15 (7) (2019) 4189–4196.

Mourad Gridach is a Professor of Computer Science and Artificial Intelligence at the High Institute of Technology, Ibn Zohr University, Agadir, Morocco. Previously, he was a Postdoctoral Researcher at the University of Colorado Anschutz Medical Campus working on biomedical NLP. Also, he was a visiting fellow at the University of Oxford working with Doctor Irina Voiculescu on biomedical image segmentation. His main area of research is Natural Language Processing focusing on Named Entity Recognition (NER) for classical texts as well as biomedical texts, Sentiment Analysis (SA), etc. He is interested in Computer Vision as well working especially in semantic segmentation and biomedical image segmentation.