Cross-domain recommendations using attention and multitask learning

Gerasimos Papachronopoulos, Alexandros Gkillas[1 [0000−0001−5339−2018]], and Dimitrios Kosmopoulos[1 [0000−0003−3325−1247]]

University of Patras, Computer Engineering and Informatics Department, 26504 Patras, Greece {st1059629,st1003586,dkosmo}@ceid.upatras.gr

Abstract. In this study, we propose a novel deep learning method for cross-domain recommendations that effectively combines attention mechanisms, autoencoders, and multitask learning. Our approach leverages multiple datasets from diverse domains and incorporates domain-specific encoders, a shared self-attention mechanism, and a multilayer perceptron (MLP) to capture both intra-domain and inter-domain relationships. By jointly modeling these interactions, we improve recommendation accuracy across domains. Experimental results using the MovieLens dataset demonstrate that our proposed cross-domain recommendation system outperforms traditional approaches including matrix factorization, standard MLPs, and self-attention-based baselines.

Keywords: cross-domain recommendations · attention · autoencoders · multitask learning

1 Introduction to cross-domain recommendation systems

The growing volume of online content has intensified the need for effective recommendation systems, particularly in scenarios involving cross-domain recommendations. While traditional recommender systems typically operate within a single domain (e.g., recommending books, movies, or music separately), users often exhibit preferences that may span multiple domains. Addressing that requires cross-domain recommendation techniques capable of transferring knowledge and making accurate predictions across distinct content areas.

Recommender systems typically use collaborative filtering and content-based methods to predict user-item interactions. However, these methods suffer from challenges such as data sparsity and cold-start problems [3, 8, 18]. Recent research has focused on leveraging deep learning approaches, including autoencoders, attention mechanisms, and multitask learning, to overcome these limitations [7, 6, 10]. In particular, attention mechanisms help capture complex dependencies between users and items, while autoencoders provide a compact latent representation of user behavior. Multitask learning, on the other hand, allows shared knowledge across domains, leading to more robust recommendations.

Cross-domain collaborative filtering [9] transfers data from source domains to target domains for more accurate predictions. For example, a user's movie

genre preferences might be inferred from their book preferences. It can also create cross-category recommendations, like suggesting music similar to a movie's soundtrack. While many current recommender systems rely on matrix factorization, this method only uncovers shallow, linear attributes. Several deep learning approaches have been published [22], but they often use one-hot vectors as input, preventing capture of genuine collaborative interactions between users and items, while most existing deep learning recommendation techniques are designed exclusively for single-domain recommendations [19].

In this paper, we present a deep learning framework for cross-domain recommendations that integrates domain-specific autoencoders, a shared attention mechanism, and a multitask learning strategy to effectively transfer knowledge across domains. We validate our approach using the MovieLens dataset and show that it significantly improves performance.

2 Background on Cross-Domain Recommendations

Cross-domain recommendation systems have been widely applied in various domains, such as e-commerce, social media, and digital entertainment. In e-commerce cross-domain recommendation systems can be used to recommend products to customers based on their browsing history and purchase history across different categories [14]. In social media, cross-domain recommendation systems can be used to recommend content to users based on their activity across different platforms [20]. In the entertainment industry, cross-domain recommendation systems can be used to recommend movies or TV shows to users based on their listening and watching history [2].

The concept of domain can be defined at four levels based on the attributes and types of recommended items [3]:

- At the *attribute level*, items with different attribute values such as different movie genres are considered as belonging to distinct domains.
- At the *type level*, items with different attribute subsets such as movies and TV shows are considered as belonging to different domains.
- At the *item level*, items that differ in most or all of their attributes such as movies and books belong to different domains.
- At the *system level*, recommended items from different recommendation systems such as MovieLens and Netflix are considered as different domains.

This classification is used to increase the diversity of recommendations.

One of the main challenges in cross-domain recommendation systems is dealing with the heterogeneity of data across different domains. This can include differences in the data format, feature representation, and user feedback [4]. To address this challenge, researchers have proposed various methods for aligning the data across domains, such as feature mapping and domain adaptation [16]. Other methods focus on learning a shared representation across domains, such as multi-task learning and transfer learning [4].

Recent research has focused on developing methods for transfer learning in cross-domain recommendation systems [4]. This includes techniques such as domain adaptation and multi-task learning, which aim to improve the performance of the system by leveraging knowledge from related domains.

Cross-domain recommendation is a challenging task that aims to make personalized recommendations to users in a new domain, where little or no information about them is available. Various methods have been proposed to tackle this problem, among which deep learning based approaches have received significant attention due to their ability to learn complex representations of users and items. In this context, autoencoder-based methods have been shown to be effective in learning shared representations across multiple domains. In [21], the authors present a method for cross-domain recommendation using deep autoencoders to learn shared representations of users and items across multiple domains. They evaluate their method on a dataset of movie ratings and show that it outperforms traditional matrix factorization-based methods.

Meanwhile [21] proposes a transfer learning approach for cross-domain recommendation using autoencoders. The authors pre-train the autoencoder on a source domain and fine-tune it on a target domain, showing that this approach can improve recommendation performance compared to training the autoencoder from scratch on the target domain. [11] presents a deep cross-domain recommendation method that uses a combination of deep autoencoders and neural networks to learn shared representations of users and items across multiple domains. The authors evaluate their method on a dataset of music ratings and show that it outperforms traditional matrix factorization-based methods as well as a baseline autoencoder-based method. [5] proposes a new architecture that couples the autoencoders of the source and target domains and show that this approach can improve recommendation performance compared to other cross-domain recommendation methods using autoencoders.

It is possible to utilize both domain-invariant and domain-specific information to make recommendations [12]. The authors show that such a method outperforms traditional matrix factorization-based methods, as well as other autoencoder-based methods. [22] proposes an autoencoder framework with an attention mechanism (AAM) for cross-domain recommendation. The method involves extracting user and item features from original rating matrices of different domains using an autoencoder, which are then fed into a multilayer perceptron (MLP) to learn user and item-latent factor vectors. These vectors are then fused into one final user-latent factor vector using an attention mechanism, and the predicted rating is obtained from the user and item-latent factor vectors. This is the method that we build upon.

Building on this body of work, our proposed method integrates autoencoders, attention layers, and a multitask learning framework to jointly model domain-specific and cross-domain user preferences. This architecture allows the system to better generalize across domains while preserving domain-unique characteristics. More specifically our contributions include:

- the introduction of a novel attention-based for cross-domain recommendation that transfers knowledge among different domains;
- the experimental comparison to some popular cross-domain recommendation algorithms.

The notations that we use are summarized in Table 1.

Table 1: Mathematical Notations for the source domain. Similar notations apply to the target domain, by replacing the superscript "s" with "t".

Symbol	Description
n	The number of users of the source and target domain
m	The number of movies of the source and target domain
	Source
$i \in \mathbb{N}$	Number of source domains
s_i	Source domain i
$\boldsymbol{R}^s, \in \mathbb{R}^{n \times m}$	The rating matrix of the source domain
$\hat{\boldsymbol{R}}^s \in \mathbb{R}^{n \times m}$	The predicted rating matrix of the source
$\boldsymbol{U}^s \in \mathbb{R}^{n \times m}$	The user rating matrix of the source domain
$\boldsymbol{M}^s \in \mathbb{R}^{m \times n}$	The movie rating matrix of the source domain
$\boldsymbol{X}_e^s \in \mathbb{R}^{n \times k}$	The output of the encoder of $\boldsymbol{U}^s$
$\hat{\boldsymbol{U}}^s \in \mathbb{R}^{n \times m}$	The output of the decoder of $\boldsymbol{X}_e^s$
$\boldsymbol{Y}_d^s \in \mathbb{R}^{m \times k}$	The output of the encoder of $\boldsymbol{M}^s$
$\hat{\boldsymbol{M}}^s \in \mathbb{R}^{m \times n}$	The output of the decoder of $\boldsymbol{Y}_e^s$

3 Related Methodologies on Cross-Domain Recommendations

3.1 Baseline: cross-domain Matrix Factorization

The users in the source domain are typically represented by the rating matrix $\boldsymbol{R}^s$. A user's rating information is captured in each row of the rating matrix. For movies representation in the source domain we use the movies rating matrix $\boldsymbol{M}^s$ (transpose of $\boldsymbol{R}^s$). We can obtain latent representations for users and movies by using autoencoders in the source and target domains.

For cross-domain recommendation, we can use matrix factorization, e.g., [14, 8] to factorize the output of the user encoder in one domain and the output of the movie encoder in the other domain, where the latent space k represents a shared space between the two domains. By using this shared space, we can recommend movies to users in one domain based on the ratings of users in the other domain. For multiple source domains, e.g., for three source domains as in our case study, we initially concatenate all three source domains using shared

users for this method. We train the movie encoder of the target domain and the user encoder of the combined source domain after generating the merged domain. The latent factor vectors of the encoders are then multiplied to predict the target domain rating matrix $\hat{\boldsymbol{R}}^t$:

$$\hat{\boldsymbol{R}}^t \approx \boldsymbol{X}_e^s \cdot \boldsymbol{Y}_e^t \tag{1}$$

where $\boldsymbol{X}_e^s$ is the output of the users encoder of source domain, $\boldsymbol{Y}_e^t$ is the output of the movies encoder of target domain. For the combined Source Domain the user rating matrix $\boldsymbol{R}^s \in \mathbb{R}^{n \times \hat{m}}$, where $\hat{m} = m_{s1} + m_{s2} + m_{s3}$ and m_{s_i} is the number of movies in source domain i. The architecture is illustrated in figure 1.

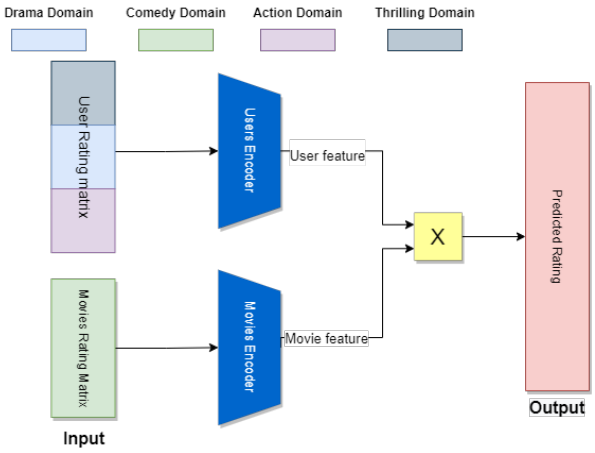

Fig. 1: Matrix Factorization for the three source domains and one target domain.

3.2 Shared Self-Attention for user representation across domains

In order to capture the correlation between the latent features of users across all source domains, we propose to create a fused latent factor matrix as presented in [22]. However, in our case, we need to extend this mechanism to cover all the source domains at the same time. Specifically, we apply the self-attention mechanism across all source domains to generate a new representation that captures the interactions between users in each source domain, and then fuse these representations to obtain a joint representation of all users across all domains.

The self-attention mechanism computes the attention weights $\boldsymbol{a}_u^t$, for each user u across several domains in relation with t-domain, as in (2), followed by a

normalization step, such as the Softmax function:

$$\begin{aligned} \boldsymbol{W}_u(p,q) &= \boldsymbol{X}^p_{u,:} \cdot (\boldsymbol{X}^q_{u,:})^T \\ \boldsymbol{a}^p_u &= \sum_{q=1}^{S} \frac{\exp(\boldsymbol{W}_u(p,q))}{\sum_{j=1}^{S} \exp(\boldsymbol{W}_u(p,j))} \boldsymbol{X}^q_u \end{aligned} \tag{2}$$

where the $\boldsymbol{a}^p_u \in \mathbb{R}^{1\times k}$ is the latent representation of the user u in the p domain. Given that, the user latent representation in all domains can be represented by the sum:

$$\boldsymbol{u_i} = \sum_{s=1}^{S} \boldsymbol{a}^s_i \tag{3}$$

We are using the output in matrix factorization technique, which has proven to be an effective technique for personalized recommendation systems [14]. The following equation applies:

$$\hat{\boldsymbol{R}}^t \approx \boldsymbol{X}^s_e (\boldsymbol{Y}^t_e)^T \tag{4}$$

where $\boldsymbol{X}^s_e$ is the output of the self-attention mechanism across all domains, $\boldsymbol{Y}^t_e$ is the output of the movies encoder of target domain and $\hat{\boldsymbol{R}}^{\boldsymbol{t}}$ the approximation of target domain's rating matrix. This architecture system is in accordance to [22] for single head attention and is illustrated in figure 2.

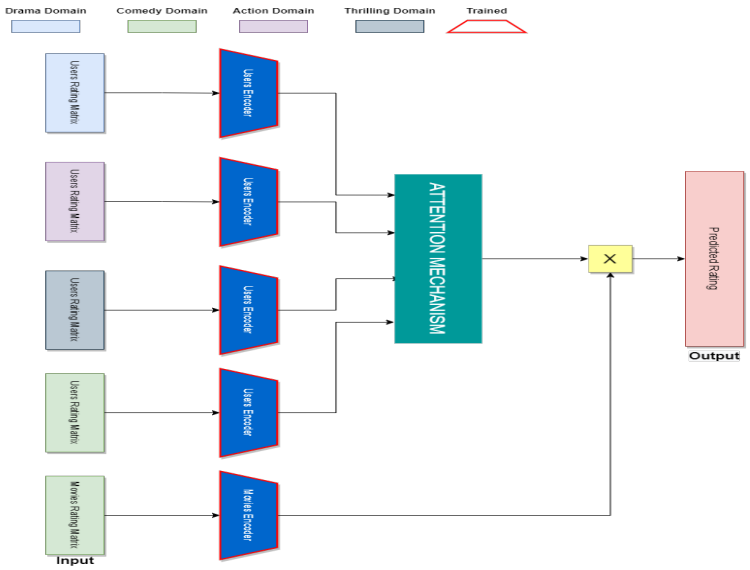

Fig. 2: Self-Attention Mechanism for three source domains and one target domain.

4 Deep Learning Framework for Cross-Domain Recommendations

4.1 Self-Attention Mechanism and MLP

In this study, we try another approach for cross-domain recommendation using deep learning. We utilize self-attention mechanisms from the output of trained encoders as input for a multi-layer perceptron (MLP). This allows for better representation and weighting of the encoded features, resulting in more accurate and personalized recommendations. For example, let's consider a user represented by a vector $\boldsymbol{U_i}$ that contains information about the user's preferences, and a movie represented by a vector $\boldsymbol{M_i}$ that contains information about the movie's ratings from all the users for the domain $\boldsymbol{s_i}$. The self-attention mechanism would then compute the attention weights $\boldsymbol{a}$ as the dot product between the input representations and a parameterized weight matrix $\boldsymbol{W}$, followed by a normalization step, such as a Softmax function (Fig. 2).

The final self-attention output is then computed as a sum of the representations of each user, as follows: $\boldsymbol{A} = [\boldsymbol{u}_1^T \boldsymbol{u}_2^T \dots \boldsymbol{u}_n^T]^T$, where n is the number of users and $\boldsymbol{A} \in \mathbb{R}^{n \times k}$.

The user representation $\boldsymbol{A}$ is then passed as input to an MLP, which consists of several layers of neurons, each layer applies a linear transformation to the input, followed by a non-linear activation function. For example, a simple MLP architecture with L hidden layers can be represented as:

$$\begin{aligned} \boldsymbol{X}_{e1}^t &= \sigma(\boldsymbol{W}_1 \cdot \boldsymbol{A} + \boldsymbol{B}_1) \\ &\dots \\ \hat{\boldsymbol{X}}_e^t &= \sigma(\boldsymbol{W}_L \cdot \boldsymbol{X}_{e_{L-1}}^s + \boldsymbol{B}_L) \end{aligned} \tag{5}$$

where $\boldsymbol{W}_1, ..., \boldsymbol{W}_L$, and $\boldsymbol{B}_1, ..., \boldsymbol{B}_L$ are the encoder's weight and bias matrices, L is the number of hidden layers, $\hat{\boldsymbol{X}}_e^t$ is the target domain's calculated intrinsic representation, and σ is the activation function (e.g., $ReLU$). The architecture is presented in figure 3.

4.2 Self-Attention Mechanism and MLP with Coupled Training

In this architecture, we use a multi-task learning framework that consists of domain-specific encoders, a shared self-attention mechanism, and an MLP. The domain-specific encoders are responsible for extracting domain-specific features from the input data, while the shared self-attention mechanism is used to capture the relationships between the features within each domain. The outputs of the self-attention mechanism are then merged and passed through an MLP for final prediction.

The difference with the previous one is that it allows the model to learn both the domain-specific and cross-domain relationships, while also capturing the unique characteristics and preferences of each domain. By using a multi-task learning framework, the model can leverage the knowledge learned from different

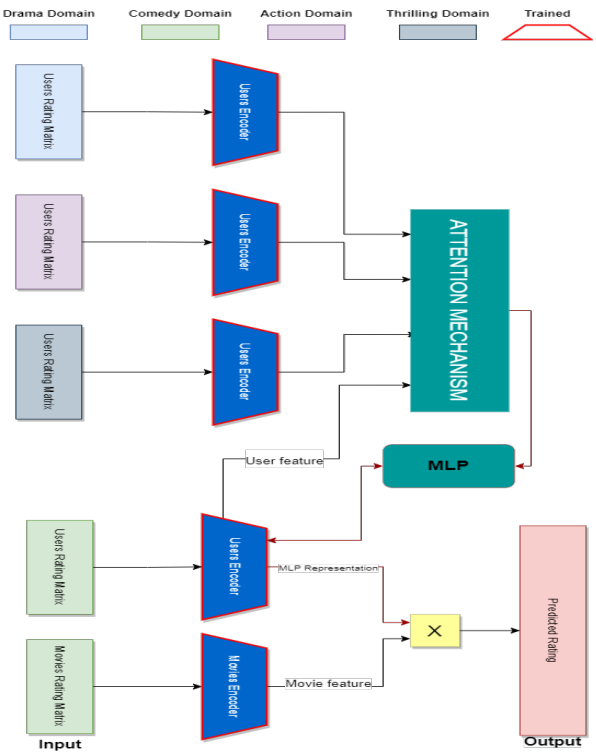

Fig. 3: Self-Attention Mechanism and MLP for three source domains and one target domain.

domains to improve the performance of each domain. The multiple tasks refer to predicting the ratings of movies for each source domain, as well as predicting the ratings for the target domain. Each domain is considered a separate task, and the model must simultaneously learn to perform well on all tasks in order to provide accurate recommendations across all domains. This approach allows the model to jointly optimize the encoders and the MLP, resulting in better performance in terms of MAE. In order to achieve this we are using a coupled training optimizer that updates the parameters of the encoders and MLP jointly, we can define the loss function ($\boldsymbol{\mathcal{L}}$) as follows:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \lambda_1 \sum_{d=1}^{D} \|\theta_d\|_2^2 + \lambda_2 \|\theta_{\text{out}}\|_2^2 \tag{6}$$

where N is the total number of user-movie pairs across all domains, $\hat{\boldsymbol{y}}_i$ is the predicted rating for the i-th user-movie pair, $\boldsymbol{y}_i$ is the true rating for the i-th user-movie pair, λ_1 and λ_2 are regularization hyperparameters, $\boldsymbol{\theta}_d$ are the parameters of the encoder for domain d, and $\boldsymbol{\theta}_{\text{out}}$ are the parameters of the output MLP.

To train the model using a coupled optimizer, we can use stochastic gradient descent (SGD) or a variant such as Adam. We can compute the gradient of the loss function with respect to the parameters of the encoders and the output MLP jointly using backpropagation. The update rules for the optimizer are given in figure 4.

$$\begin{aligned}
\boldsymbol{m}_d^{(t+1)} &= \beta_1 \boldsymbol{m}_d^{(t)} + (1-\beta_1)\nabla_{\boldsymbol{\theta}_d}\mathcal{L} \\
\boldsymbol{v}_d^{(t+1)} &= \beta_2 \boldsymbol{v}_d^{(t)} + (1-\beta_2)(\nabla_{\boldsymbol{\theta}_d}\mathcal{L})^2 \\
\hat{\boldsymbol{m}}_d &= \frac{\boldsymbol{m}_d^{(t+1)}}{1-\beta_1^{t+1}} \\
\hat{\boldsymbol{v}}_d &= \frac{\boldsymbol{v}_d^{(t+1)}}{1-\beta_2^{t+1}} \\
\boldsymbol{\theta}_d^{(t+1)} &= \boldsymbol{\theta}_d^{(t)} - \frac{\eta}{\sqrt{||\hat{\boldsymbol{v}}_d||}+\epsilon}\hat{\boldsymbol{m}}_d
\end{aligned}
\qquad
\begin{aligned}
\boldsymbol{m}_{\text{out}}^{(t+1)} &= \beta_1 \boldsymbol{m}_{\text{out}}^{(t)} + (1-\beta_1)\nabla_{\boldsymbol{\theta}_{\text{out}}}\mathcal{L} \\
\boldsymbol{v}_{\text{out}}^{(t+1)} &= \beta_2 \boldsymbol{v}_{\text{out}}^{(t)} + (1-\beta_2)(\nabla_{\boldsymbol{\theta}_{\text{out}}}\mathcal{L})^2 \\
\hat{\boldsymbol{m}}_{\text{out}} &= \frac{\boldsymbol{m}_{\text{out}}^{(t+1)}}{1-\beta_1^{t+1}} \\
\hat{\boldsymbol{v}}_{\text{out}} &= \frac{\boldsymbol{v}_{\text{out}}^{(t+1)}}{1-\beta_2^{t+1}} \\
\boldsymbol{\theta}_{\text{out}}^{(t+1)} &= \boldsymbol{\theta}_{\text{out}}^{(t)} - \frac{\eta}{\sqrt{||\hat{\boldsymbol{v}}_{\text{out}}||}+\epsilon}\hat{\boldsymbol{m}}_{\text{out}}
\end{aligned}$$

Fig. 4: Update rule for the encoder parameters (left column) and for the MLP (right column).

In these equations, $\boldsymbol{m}_d^{(t)}$ and $\boldsymbol{v}_d^{(t)}$ are the first and second moment estimates for the gradient of the loss with respect to the encoder parameters $\boldsymbol{\theta}_d$ at iteration t, and $\boldsymbol{m}_{\text{out}}^{(t)}$ and $\boldsymbol{v}_{\text{out}}^{(t)}$ are the first and second moment estimates for the gradient of the loss with respect to the output MLP parameters $\boldsymbol{\theta}_{\text{out}}$ at iteration t. The hyperparameters β_1 and β_2 control the decay rates for the first and second moments, respectively, and ϵ is a small constant to prevent division by zero. The

hat notation denotes bias-corrected estimates of the moments, which are used to adjust the step size for the updates.

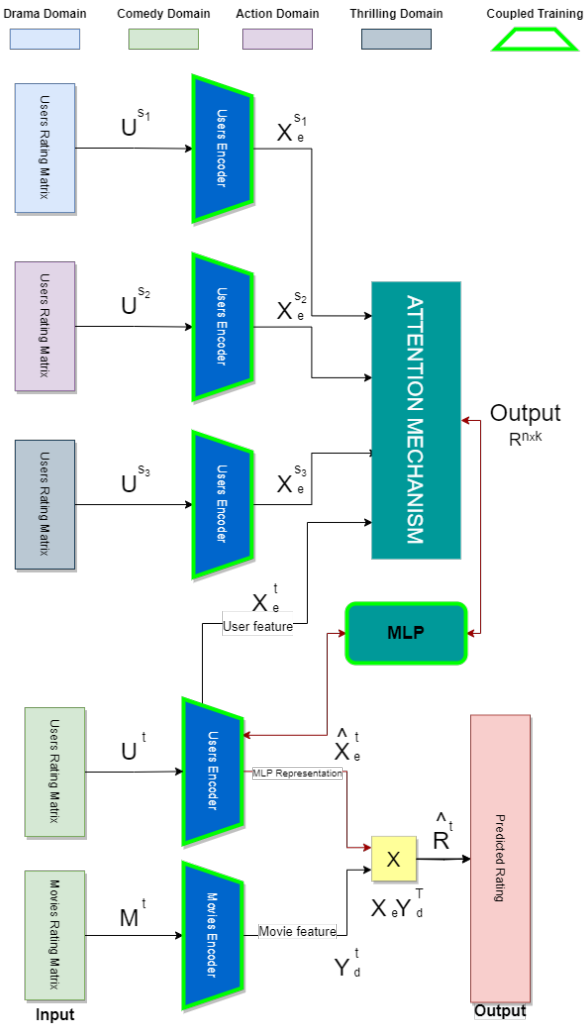

Fig. 5: Self-Attention Mechanism-MLP Coupled Training with three source domains and one target domain.

5 Experimental Results on MovieLens Dataset

We use the MovieLens 20M dataset, which is a collection of movie ratings provided by users on the MovieLens website. It is generated by GroupLens Research,

containing 138493 users, 27278 movies, and 20000263 ratings from MovieLens. We have selected this dataset because (a) it is used as a benchmark dataset for various recommendation system approaches in papers e.g., [13, 14, 22], which may facilitate comparisons, (b) it contains a large amount of movie ratings and diverse user-item interactions, providing a realistic representation of the challenges faced by real-world recommendation systems, (c) it contains different movie genres that can be used for cross-domain recommendations and (d) is sparse and thus challenging. We kept the users with a user Id up to 20,000. This resulted in the domains being of a sufficient size while maintaining a sparsity level of approximately 99%.

Domain	Training Set Sparsity	Test Set Sparsity	Splitting
Comedy	99.06%	99.07%	60/40
Drama	99.30%	99.30%	60/40
Action	97.99%	98.01%	60/40
Thrilling	97.98%	98.08%	60/40

Table 2: Sparsity and splitting table for the domains.

We divided the dataset into four domains based on movie genre: action, drama, thriller, and comedy comprising 1215, 1215, 7246 and 5981 movies respectively (see Table 2). These domains possess a common user base of 9,778 individuals. We sought to predict the user ratings for the comedy domain while using the other three domains.

Nr	Method	RMSE	MAE
1	Cross-Domain Matrix Factorization (Fig. 1)	0.8317	0.7968
2	Cross-Domain Self-Attention [22] (Fig. 2)	0.5499	0.2609
3	Cross-Domain Self-Attention with MLP (Fig. 3)	0.5629	0.1557
4	Cross-Domain Self-Attention-MLP with Coupled Training (Fig. 5)	0.4985	0.1092

Table 3: Experimental comparison of the presented methods in terms of RMSE and MAE errors.

We have used the pytorch library for our implementation. The encoder-decoder architecture used employ two linear layers of 128 nodes each, leading to a latent vector of size 64. Leaky ReLU was used as activation function. The same two-layer architecture was used for the MLP.

The performance of the four presented methods was evaluated using two metrics, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). The results of the evaluation for RMSE and MAE are presented in the table below (Fig. 3). The Mean Absolute Error (MAE) is considered to be an important evaluation metric in movie recommendation systems as it provides a more ro-

bust evaluation of the recommendation performance. Unlike Root Mean Square Error (RMSE), MAE measures the absolute differences between the predicted ratings and the ground truth, making it more insensitive to outliers [17]. This is especially important in movie recommendation systems, as the ratings are usually on a scale of 1 to 5 and a small absolute error may have a large relative impact on the overall prediction performance [15]. Additionally, MAE is more interpretable and easier to understand for users compared to RMSE, making it a preferred choice for evaluating the recommendation performance in real-world applications [1].

Method 2 clearly outperforms the method 1, which underlines the superiority of attention-based methods in comparison to matrix factorization. The latter is able to capture linear dependencies and probably that is the reason for the differences that we noticed in performance. Method 3 has similar RMSE to Method 2 with a value of 0.5629, and significantly better MAE. The added component of the MLP that maps user representations from the source to the target appears to offer added value. The method 5 clearly demonstrates significantly lower RMSE and MAE errors (0.4985, 0.1092) in comparison to methods 2,3. This is a clear indication of the benefits of multitask learning. Indeed, the addition of the coupled learning between domains offers serious reduction of the prediction error.

6 Conclusion and Future Directions in Cross-Domain Recommendation Research

We have proposed a method that integrates domain-specific encoders, a shared self-attention mechanism, and an MLP to capture both domain-specific and cross-domain relationships, improving recommendation accuracy. We presented our method and showcase that it outperforms methods based on matrix factorization, MLP, and self-attention in the cross-domain scenario. The proposed method was validated on the MovieLens dataset and the experimental results indicate that it outperforms purely cross-domain factorization and self-attention methods.

In the near future we are going to extend the method with multi-head attention. Extending the current single-head attention to a multi-head variant could help the model attend to different types of domain-specific relationships and improve representation learning across diverse content types. Future research could also integrate item and user metadata (e.g., demographics, tags, textual reviews) to enhance autoencoder inputs, improving performance especially in sparse or cold-start scenarios. Using domain-adversarial training could ensure domain-invariant feature extraction while preserving discriminative properties, potentially improving generalization in low-data target domains. Expanding the evaluation beyond MovieLens would validate the model's effectiveness across diverse recommendation tasks. Furthermore, we aim to apply the proposed transfer learning methods to other domains such as tasks combining vision and text (e.g., medicine, precision agriculture etc.).

Acknowledgement: This research was funded by the Greek Ministry of Agricultural Development and Food, under Sub-Measure 16.1 - 16.2 "Establishment and Operation of Business Groups of the European Innovation Partnership for Productivity and the Sustainability of Agriculture", Action 2, project number M16SYN2-00229.

References

1. Bobadilla, J., Hernando, A., Ortega, F., Bernal, J.: A framework for collaborative filtering recommender systems. Exp. Syst. with Applic. **38**, 14609–14623 (2011)
2. Bobadilla, J., Ortega, F., Hernando, A., GutiÃĪrrez, A.: Recommender systems survey. Knowledge-Based Systems **46**, 109–132 (2013)
3. Cantador, I., Cremonesi, P.: Tutorial on cross-domain recommender systems. ACM Conference on Recommender Systems pp. 401–402 (2014)
4. Chen, X., Li, L., Pan, W., Ming, Z.: A survey on heterogeneous one-class collaborative filtering. ACM Trans. Inf. Syst. **38**(4) (2020)
5. Gkillas, A., Kosmopoulos, D.: A cross-domain recommender system using deep coupled autoencoders (2022), `https://arxiv.org/abs/2112.07617`
6. Guo, Y., Liu, Y., Zhang, Z.: Cross-domain recommendation with adversarial autoencoders. In: World Wide Web Conference. pp. 1267–1276 (2018)
7. Hu, G., Zhang, Y., Yang, Q.: Conet: Collaborative cross networks for cross-domain recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. pp. 667–676 (2018)
8. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: IEEE Int. Conf. on Data Mining. pp. 263–272 (2008)
9. Kanagawa, H., Kobayashi, H., Shimizu, N., Tagami, Y., Suzuki, T.: Cross-domain recommendation via deep domain adaptation. arXiv (2018)
10. Kang, S., Hwang, J., Lee, D., Yu, H.: Semi-supervised learning for cross-domain recommendation to cold-start users. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 1563–1572 (2019)
11. Kang, S., Hwang, J., Lee, D., Yu, H.: Semi-supervised learning for cross-domain recommendation to cold-start users. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 1563–1572 (2019)
12. Khan, M.M., Ibrahim, R., Ghani, I.: Cross domain recommender systems: A systematic literature review. ACM Comput. Surv. **50**(3) (2017)
13. Koren, Y.: Factorization meets the neighborhood: A multifaceted collaborative filtering model. In: ACM SIGKDD. pp. 426–434 (2008)
14. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
15. Lemire, D., Maclachlan, A.: Slope one predictors for online rating-based collaborative filtering. arXiv (2007)
16. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering **22**(10), 1345–1359 (2010)
17. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Int. Conf. on World Wide Web. pp. 285–295 (2001)
18. Wang, C., Blei, D.M.: Collaborative topic modeling for recommending scientific articles. In: ACM SIGKDD. pp. 448–456 (2011)
19. Xi, W.D., Huang, L., Wang, C.D., Zheng, Y.Y., Lai, J.: Bpam: Recommendation based on bp neural network with attention mechanism. In: Int. Joint Conf. on Artificial Intelligence, IJCAI-19. pp. 3905–3911 (2019)

20. Zang, T., Zhu, Y., Liu, H., Zhang, R., Yu, J.: A survey on cross-domain recommendation: Taxonomies, methods, and future directions. ACM Trans. Inf. Syst. **41**(2) (2022)
21. Zhang, Y., Wang, Y., Metzler, D.: Cross-domain recommendation with deep autoencoders. In: ACM SIGIR conference on Research and Development in Information Retrieval. pp. 1043–1046 (2016), https://arxiv.org/pdf/1910.05189.pdf
22. Zhong, S.T., Huang, L., Wang, C.D., Lai, J.H., Yu, P.S.: An autoencoder framework with attention mechanism for cross-domain recommendation. IEEE Transactions on Cybernetics **52**(6), 5229–5241 (2022)