This paper was accepted to **RecSys 2024**. Please cite it instead once published.

beeFormer: Bridging the Gap Between Semantic and Interaction Similarity in Recommender Systems

Vojtěch Vančura
vancurv@fit.cvut.cz
Faculty of Information Technology, Czech Technical University in Prague
Prague, Czech Republic
Recombee
Prague, Czech Republic

Pavel Kordík
pavel.kordik@fit.cvut.cz
Faculty of Information Technology, Czech Technical University in Prague
Prague, Czech Republic
Recombee
Prague, Czech Republic

Milan Straka
straka@ufal.mff.cuni.cz
Faculty of Mathematics and Physics, Charles University
Prague, Czech Republic

ABSTRACT

Recommender systems often use text-side information to improve their predictions, especially in cold-start or zero-shot recommendation scenarios, where traditional collaborative filtering approaches cannot be used. Many approaches to text-mining side information for recommender systems have been proposed over recent years, with sentence Transformers being the most prominent one. However, these models are trained to predict semantic similarity without utilizing interaction data with hidden patterns specific to recommender systems. In this paper, we propose beeFormer, a framework for training sentence Transformer models with interaction data. We demonstrate that our models trained with beeFormer can transfer knowledge between datasets while outperforming not only semantic similarity sentence Transformers but also traditional collaborative filtering methods. We also show that training on multiple datasets from different domains accumulates knowledge in a single model, unlocking the possibility of training universal, domain-agnostic sentence Transformer models to mine text representations for recommender systems. We release the source code, trained models, and additional details allowing replication of our experiments at https://github.com/recombee/beeformer.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender systems, Text mining, Sentence embeddings, Cold-start recommendation, Zero-shot recommendation

ACM Reference Format:
Vojtěch Vančura, Pavel Kordík, and Milan Straka. 2024. beeFormer: Bridging the Gap Between Semantic and Interaction Similarity in Recommender Systems. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3640457.3691707

RecSys '24, October 14–18, 2024, Bari, Italy

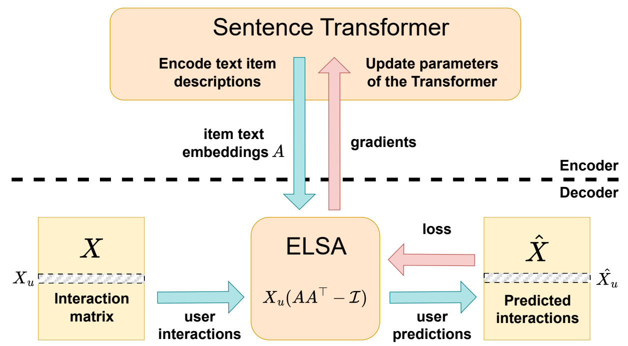

Figure 1: Training with the beeFormer framework: A sentence Transformer model (up) act as a an encoder to generate item embeddings represented by the matrix A. Then ELSA act as a decoder (down) in a training step on interactions to obtain gradients used to optimize the Transformer model.

1 INTRODUCTION

Recommender systems (RS) aim to help users find what they are looking for in various domains. Many approaches to building algorithms for RS have been proposed over the past years, with Collaborative Filtering (CF) [18] being the most popular choice. CF methods predict (filter) user preferences by analyzing past interactions. Popular CF techniques include neighborhood-based methods [23, 35], matrix-factorization (MF) [24, 46], deep neural networks (DNN) [8, 27], or shallow linear autoencoders (SLAs) [41, 42, 47].

SLAs became popular recently, mainly because the EASE [42] model has a closed-form solution while yielding high performance comparable to deep models. Since EASE cannot scale to datasets with a high number of items, scalable variants of EASE have been proposed: SANSA [41] and ELSA [47]. SANSA keeps the original structure of EASE but uses (sparse) incomplete Cholesky factorization approximating the inverse of the (potentially) large sparse matrix X^TX to construct the asymmetric approximation of the item-to-item weight matrix. ELSA, on the other hand, approximates learned item-to-item weight matrix W with low-rank approximation $W = AA^T$, with $\text{diag}(W) = 0$ to prevent trivial solution such as the identity solution. The matrix A is optimized using backpropagation.

Despite the popularity and state-of-the-art performance of CF methods in recommendation tasks, they cannot provide any predictions when there are no interactions. In such cases, also known as cold-start [36] and zero-shot [13] recommendation, one can use content-based filtering (CBF) [3, 4] using side information (attributes, images, text) directly to produce recommendations or learn a transformation function to transform side information to CF representations [15, 51].

Using texts (item descriptions, user reviews, etc.) as side information has become extremely popular after the invention of the Transformer [48] neural architecture. Transformer models can be used to encode text into a vector representation, and sentence Transformers [34] were explicitly developed to mine text latent representations from the whole blocks of text (sentences, paragraphs), which can then be used for various tasks, with the recommendation being one of them.

However, sentence Transformers trained to predict semantic similarity often fail to capture patterns and user behaviors hidden in the interaction data. In many cases, users may look for a specific item (for example, batteries when buying a kid's toy, or cables when buying a new printer) with very low semantic similarity compared to other items in the catalog.

To bridge this gap between the semantic and the interaction similarity, we employ the following idea: We use the training procedure from the ELSA model, but instead of optimizing the matrix A, we generate matrix A with a sentence Transformer model and optimize parameters of the sentence Transformer instead of optimizing A directly, as illustrated in Figure 1. However, this approach faces one critical problem: In every training step, we need to generate and optimize embeddings for all items in the catalog, which leads to very high effective batch size for Transformer training, e.g., possibly over a million for some datasets. We propose to overcome this problem by employing the following three techniques: gradient checkpointing [19], gradient accumulation [7], and negative sampling [14]. Combining these techniques with sentence Transformer and ELSA training procedure, we present beeFormer (short for the Recom**bee** Trans**former** in camel case), a sentence Transformer training framework that uses text-side information and interactions directly to update the parameters of a Transformer model.

The main contributions of this paper are listed as follows:

- We propose beeFormer, a framework for training sentence Transformers on interaction data with text-side information.
- Our experiments show that sentence Transformer models trained with beeFormer outperform all baselines in cold-start, zero-shot and time-split recommendation scenarios.
- We demonstrate the beeFormer's ability to transfer knowledge between datasets.
- We show that training models on combined datasets from various domains further increase performance in the domain-agnostic recommendation.
- We create and publish LLM-generated item descriptions for all used datasets for reproducibility of our experiments.
- Models trained with beeFormer are easily deployable into production systems using the sentence Transformers library.

We believe the above improvements open a path towards a potentially universal, domain-agnostic, and multi-modal Transformer models for recommender systems.

2 RELATED WORK

SLAs [41, 42, 47] have recently gained much attention, mainly because of their simplicity while retaining performance comparable to deep models, with EASE [42] being the most promising. ELSA [47] solved the scalability issue of EASE and enabled its use on large datasets by low-rank approximation of the item-to-item weight matrix of EASE. We use the idea of training the ELSA's embeddings using backpropagation to obtain the gradients for training the sentence Transformer model.

The first attempt to use Transformers in RS was in sequential recommendation. Bert4rec [43] used item IDs as tokens and treated the sequential recommendation problem in the same manner as NLP. Several improvements to this approach have been proposed since then [10, 16, 37]. Promising approach is to create artificial text sequences combining descriptions of interacted items and train a Transformer-based model on them [21, 25].

Another direction is to use Large Language Models (LLMs), such as the Chat-GPT [12, 28]. LLMs can be used for various tasks, e.g., as conversational recommendation [44, 50], to generate standardized item text descriptions [1], recommendation explanation [38], or to produce recommendations through its text output [26] directly.

Sentence Transformers [34] use a pooling function on top of the Transformer architecture and provide a robust, easy-to-use framework for tokenization, embedding generation, and training sentence Transformer models. While using sentence Transformers is very popular in the dense-retrieval domain [5, 17], using sentence Transformers in recommender systems is limited to generating side information for cold-start methods [15, 51] or using neural networks [22] or graph neural networks [40] on top of sentence Transformer models.

Improving sentence Transformers with training on interaction data is crucial for boosting the performance of all methods using sentence embeddings as side information mentioned above. Sadly, to our best knowledge, there is no prior work on training sentence Transformers directly with interaction data in the RS domain.

3 TRAINING PROCEDURE

We follow notation from [47]: assume a set of users $\mathbb{U} = \{u_1, u_2, \ldots, u_U\}$, a set of items $\mathbb{I} = \{i_1, i_2, \ldots, i_I\}$, and a set of token sequences representing corresponding items $\mathbb{T} = \{t_1, t_2, \ldots, t_I\}$. Let $X \in \{0,1\}^{|\mathbb{U}|\times|\mathbb{I}|}$ be a user-item interaction matrix: $X_{a,b} = 1$ if the user u_a interacted with item i_b, and $X_{a,b} = 0$ otherwise. Assume that M_a is a *column* vector corresponding to the a-th row of matrix M. Let $\text{norm}(M)$ be a function that L^2-normalizes each row of a matrix, so $\text{norm}(M)_a = M_a/\|M_a\|$. Finally, let $g(\bullet, \theta_g) : \mathbb{T} \to \mathbb{R}^{I\times d}$ be a Transformer-based neural network with parameters θ_g.

The training procedure starts with generating latent representations of items – the matrix A:

$$A = g(\mathbb{T}, \theta_g). \tag{1}$$

Then, we can compute our loss [47] as:

$$L = \left\| \text{norm}\left(X_u\right) - \text{norm}\left(X_u(AA^\top - \mathcal{I})\right)\right\|_F^2. \tag{2}$$

Finally, we compute the gradients of L with respect to A, and then the gradients for θ_g are computed using the chain rule. However, when using common deep learning frameworks, there is a practical problem regarding the memory needed to track gradients for θ_g because the size of matrix A depends on the number of *all* items.

We employ the following procedure (described in Algorithm 1) to address this memory problem: First, we compute the matrix A in batches without tracking the gradients for θ_g (lines 1-5 of the algorithm). Then, we compute predictions (line 8), loss (lines 10-13) for a batch of users X_u, and *gradient checkpoint* [19] for the matrix A (lines 14,16). Finally, we compute the matrix A in batches again and use the gradient checkpoint to compute the gradients for θ_g. We *accumulate gradients* [7] during the loop (lines 19-21). Finally, we update θ_g with a PyTorch optimizer (line 22).

Algorithm 1 beeFormer training step procedure in Python using PyTorch

Input:
- batch of interactions X
- Transformer model `transformer`
- sequences of tokens `tokenized_texts`
- PyTorch optimizer `optimizer` optimizing the parameters of the Transformer model `transformer`

Output:
- Transformer model with updated weights `transformer`
- predictions X_pred
- computed loss `loss`

```
1  with torch.no_grad():
2      A_list = []
3      for t in tokenized_texts:
4          A_list.append(transformer(t))
5      A = torch.vstack(A_list)
6
7  A.requires_grad = True
8  X_pred = X @ A @ A.T - X
9
10 loss = torch.nn.MSELoss(
11     torch.nn.functional.normalize(X),
12     torch.nn.functional.normalize(X_pred),
13 )
14 loss.backward()
15
16 checkpoint = A.grad
17
18 optimizer.zero_grad()
19 for i, t in enumerate(tokenized_texts):
20     A_i = transformer(t)
21     A_i.backward(gradient=checkpoint[i])
22 optimizer.step()
```

Using the algorithm above, we effectively enable training of a Transformer model using gradients computed with the ELSA algorithm on top of it, from the memory point of view. However, computing A for all items in every training step quickly becomes time-consuming for datasets with large number of items. At this point, we would like to note one property specific to recommender systems: interaction matrix X is typically (very) sparse. This property means that when we sample a random batch of user interaction vectors from X, we obtain interactions only with a limited number of items, meaning that it is not necessary to encode all items to matrix A in every training step. More formally, let $\mathbb{I}_b$ be a subset of $\mathbb{I}$ derived from the interactions presented in a random batch b sampled from X. Recent work shows that significant improvement in performance during training CF models can be achieved by negative sampling [14]. We implement negative sampling by adding random items to $\mathbb{I}_b$ and fixing the total number of items observed in each step during training to m. The m becomes an additional hyperparameter fulfilling $|\mathbb{I}_b| \leq m \ll |\mathbb{I}|$ for any possible $\mathbb{I}_b$, to control the size of the matrix A. Finally, in a situation when $|\mathbb{I}_b| < m$, we select uniformly at random $m - |\mathbb{I}_b|$ items from the catalog and add them to the sampled batch from X with zeros in corresponding columns. A notable advantage of our approach is that the asymptotic complexity of beeFormer does not depends on the overall number of items $|\mathcal{I}|$ but only on the hyperparameter m.

Table 1: Detailed statistics of datasets used for evaluation.

	GB10k	ML20M	AB
# of items in X	9 975	16 902	63 305
# of users in X	53 365	136 589	634 964
# of interactions in X	4.20 M	9.69 M	8.29 M
density of X	0.77 %	0.42 %	0.02 %
density of $X^T X$	41.22 %	26.93 %	7.59 %

4 EXPERIMENTAL EVALUATION

We evaluate our models on several popular datasets for evaluating recommender systems: **MovieLens20M** (ML20M) [20], **Goodbooks-10k** (GB10k) [49], and **Amazon Books** (AB) [31]. Since these datasets contain explicit ratings, we transform them into implicit feedback datasets by considering a rating of four or higher as an interaction between the user and the item, and we keep only users with at least five interactions. Then, we collect movie plots from the IMDB Movies Analysis dataset [29] to obtain the descriptions for the items of the MovieLens20M dataset, and we collect book descriptions from the Goodreads books - 31 features dataset [33] and Goodreads 100K books dataset [11] to get the descriptions for the Goodbooks-10k dataset; most items in the Amazon Books dataset already contains descriptions. We then remove the items without descriptions. Finally, we use the Meta-Llama-3.1-8B-Instruct[1] [2] to generate standardized item descriptions from the existing ones to train our models.[2] Since the LLM refuse to generate descriptions for some items (for example, because it refuses to generate explicit content), we remove such items from the dataset. Summary of the resulting dataset's properties is available in Table 1.

We use two setups for our experiments. First, to test the ability of beeFormer to generalize toward new items, we split ML20M and

[1]LLAMA 3.1 license allows the use of generated output to train new language models. We add the prefix "Llama" to the names of our models to comply with license terms.
[2]Using Llama to generate the item descriptions allows us to publish them. More details about the item description generation are available on our GitHub page.

GB10k datasets item-wise (item-split): we randomly choose 2000 items as the test set, and we use the rest to cross-validate and train our models. Next, we simulate a real-world scenario with the AB dataset: we sorted all interactions by timestamp and used the last 20% of interactions as a test set (time-split). Again, the remaining interactions were used as validation and training sets.

Item-split Setup. In the item-split setup, we use the following scenarios: Firs, a zero-shot scenario, where the models were not trained on the evaluated dataset – they need to transfer knowledge from other datasets. We use CBF in this scenario – we generate item embeddings with a sentence Transformer, and then we use cosine similarities between item embeddings to provide recommendations. Second, in a cold-start scenario, the models use the text side information to generalize towards new, previously unseen items. We use the Heater model [51] mapping interaction data to side information, to benchmark the baseline models. Again, for our models, we use CBF.

We choose to compare our beeFormer-trained models to three best-performing sentence Transformer models:

- **all-mpnet-base-v2**, the best performing model from the sentence Transformers [34] library. It is based on MPNET [39], a model pre-trained with combined masked and permuted language modeling and then finetuned on various datasets.
- **BAAI/bge-m3** [6], which uses three-fold versatility (dense, sparse/lexical, and multi-vector) retrieval during training. It is trained and finetuned on various datasets and synthetic data.
- **nomic-ai/nomic-embed-text-v1.5** [32], which uses Flash attention [9] to handle longer context (up to 8192 tokens for predictions.) It was trained using a Contrastor framework.[3]

Time-split Setup. Similarly, for the time-split setup, we use a zero-shot scenario and supervised models trained with interaction data. This setup allows the use of classical CF; we chose KNN [35], ALS matrix factorization [45], ELSA [47], and SANSA [41] as baselines.

We train four models by finetuning the initial *all-mpnet-base-v2* model: one for each dataset and one model combining the data from the ML20M and GB10k datasets (denoted *goodlens*). The resulting models are available on our Huggingface page.[4]

We use Recall@20 (R@20), Recall@50 (R@50), and NDCG@100 (N@100) metrics computed using the RecPack [30] framework to compare the models. We also calculate the standard error for each experiment with bootstrap resampling.

We publicly share further details about the datasets, description of the LLM generative procedure, resulting data for reproducibility, hyperparameters, and other technical details on our GitHub page , along with all the source code.[5]

4.1 Results

Item-split setup. In the zero-shot scenario, we observe that the beeFormer models trained on a different domain (books vs. movies) significantly outperform all baselines. Detailed results are in Table 2.

For the cold-start scenario, we observe that beeFormer-trained models outperform all baselines when the Heater model approach is used for mapping semantic embeddings to interactions. The model demonstrate interesting behavior when trained on multiple datasets: *goodlens* model outperforms the models trained solely on the evaluated dataset both for ML20M and GB10k. This indicates the possibility to train one (universal) recommender model on multiple datasets from multiple domains. Detailed results are in Table 3.

Comparing results from Tables 2 and 3, models trained on different datasets within the same domain yield similar performance to models trained on the evaluated dataset. This demonstrates the critical capability of beeFormer to transfer knowledge from one dataset to another.

Time-split setup. We utilize the time-split setup to compare the beeFormer-trained models with the CF models. The beeFormer models outperform all CF baselines for both training on the evaluated dataset and in the zero-shot scenario within the same domain. The model trained on multiple datasets performs slightly worse in cold-start scenario than pure in-domain knowledge transfer, but the results are still comparable. Detailed results are in Table 4.

Table 2: Results for zero-shot scenario in item-split setup. Names of our models trained with beeFormer are in italics, the best-performing models are represented in bold, and the best baseline for each scenario is underlined. The standard error of all values is below 0.0001 (evaluated via bootstrap resampling).

Dataset	Sentence Transformer	R@20	R@50	N@100
GB10K	all-mpnet-base-v2	0.1017	0.1886	0.1739
	nomic-embed-text-v1.5	<u>0.1146</u>	<u>0.2069</u>	<u>0.1896</u>
	bge-m3	0.1134	0.1953	0.1838
	Llama-movielens-mpnet	0.1782	0.2837	0.2719
	Llama-amazbooks-mpnet	**0.2649**	**0.3957**	**0.3787**
ML20M	all-mpnet-base-v2	0.0788	0.1550	0.1042
	nomic-embed-text-v1.5	0.1113	<u>0.2143</u>	0.1511
	bge-m3	<u>0.1409</u>	0.2125	<u>0.1578</u>
	Llama-goodbooks-mpnet	**0.1589**	**0.2647**	**0.2066**

5 CONCLUSIONS

We introduce beeFormer, a novel training procedure that enhances neural representations of items by training sentence encoders on interactions. Our approach is scalable, utilizing ELSA linear autoencoder as the decoder during the training process, enabling it to handle datasets with a large number of items. BeeFormer-trained models are easily deployable into existing production systems since they are compatible with the widely adopted sentence Transformers library and can produce recommendations via embedding tables with cosine similarity criterion.

We observe performance improvements over various state-of-the-art baselines in all evaluated scenarios. Notably, in the time-split setup on the Amazon Books dataset, our models achieves significantly better results over CF methods in both supervised and zero-shot scenarios, demonstrating both superior performance and the ability to transfer knowledge from one dataset to another.

[3] https://github.com/nomic-ai/contrastors
[4] https://huggingface.co/beeformer
[5] https://github.com/recombee/beeformer

Table 3: Results for cold-start scenario in item-split setup. Names of our models trained with beeFormer are in italics, the best-performing models are represented in bold, and the best baseline for each scenario is underlined. The standard error of all values is below 0.0001 (evaluated via bootstrap resampling).

Dataset Method	Sentence Transformer	R@20	R@50	N@100
GB10K CBF	*Llama-goodbooks-mpnet*	0.2505	0.3839	0.3747
	Llama-goodlens-mpnet	**0.2710**	**0.4218**	**0.4066**
GB10K Heater	all-mpnet-base-v2	0.2078	0.3221	0.3195
	nomic-embed-text-v1.5	0.2154	0.3317	0.3193
	bge-m3	0.2052	0.3113	0.3099
	Llama-movielens-mpnet	0.2060	0.3161	0.3196
ML20M CBF	*Llama-movielens-mpnet*	0.4291	0.6108	0.4054
	Llama-goodlens-mpnet	**0.4630**	**0.6152**	**0.4066**
ML20M Heater	all-mpnet-base-v2	0.3114	0.4331	0.3407
	nomic-embed-text-v1.5	0.3049	0.4285	0.3270
	bge-m3	0.2847	0.3932	0.3161
	Llama-goodbooks-mpnet	0.3204	0.4669	0.3381

Table 4: Results for time-split setup on the Amazon Books dataset. Names of our models trained with beeFormer are in italics, the best-performing models are represented in bold, and the best baseline for each scenario is underlined. The standard error of all values is below 0.00005 (evaluated via bootstrap resampling).

Scenario Method	Model	R@20	R@50	N@100
zero-shot CBF	all-mpnet-base-v2	0.0218	0.0336	0.0193
	nomic-embed-text-v1.5	0.0387	0.0560	0.0320
	bge-m3	0.0398	0.0546	0.0313
	Llama-goodbooks-mpnet	**0.0649**	**0.0931**	**0.0515**
	Llama-goodlens-mpnet	0.0617	0.0891	0.0492
supervised CF	KNN	0.0370	0.0562	0.0303
	ALS MF	0.0344	0.0580	0.0313
	ELSA	0.0367	0.0628	0.0346
	SANSA	0.0421	0.0678	0.0362
CBF	*Llama-amazbooks-mpnet*	**0.0706**	**0.1045**	**0.0571**

We also demonstrate that training the *Llama-goodlens-mpnet* model on two datasets (GB10K and ML20M) from different domains further increases performance when evaluating on individual datasets. This ability to accumulate knowledge from multiple datasets marks an important step towards training universal, domain-agnostic, content-based models for RS.

In our future work, we plan to build one (big) dataset from several RS domains and train a universal sentence Transformer model on it. We want to also explore the possibility of using beeFormer with computer vision models. Building multi-modal encoders with beeFormer could be especially useful in domains such as fashion recommendation.

ACKNOWLEDGMENTS

We want to thank anonymous reviewers for their suggestions, many of which helped us improve this paper. Our research has been supported by the Grant Agency of Czech Technical University (SGS23/210/OHK3/3T/18) and by the Grant Agency of the Czech Republic under the EXPRO program as project "LUSyD" (project No. GX20-16819X).

REFERENCES

[1] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. LLM Based Generation of Item-Description for Recommendation System. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 1204–1207. https://doi.org/10.1145/3604915.3610647

[2] AI@Meta. 2024. The Llama 3 Herd of Models. (2024). https://ai.meta.com/research/publications/the-llama-3-herd-of-models

[3] Fahad Anwaar, Naima Iltaf, Hammad Afzal, and Raheel Nawaz. 2018. HRS-CE: A hybrid framework to integrate content embeddings in recommender systems for cold start items. *Journal of computational science* 29 (2018), 9–18.

[4] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *Proceedings of the eighth ACM international conference on web search and data mining*. 285–294.

[5] Jan A. Botha, Zifei Shan, and Daniel Gillick. 2020. Entity Linking in 100 Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 7833–7845. https://doi.org/10.18653/v1/2020.emnlp-main.630

[6] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv:2402.03216* (2024).

[7] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174* (2016).

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems* 35 (2022), 16344–16359.

[10] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential / Session-Based Recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems* (Amsterdam, Netherlands) *(RecSys '21)*. Association for Computing Machinery, New York, NY, USA, 143–153. https://doi.org/10.1145/3460231.3474255

[11] Manav Dhamani. 2021. Goodreads 100K books. https://www.kaggle.com/datasets/mdhamani/goodreads-books-100k

[12] Dario Di Palma, Giovanni Maria Biancofiore, Vito Walter Anelli, Fedelucio Narducci, Tommaso Di Noia, and Eugenio Di Sciascio. 2023. Evaluating chatgpt as a recommender system: A rigorous approach. *arXiv:2309.03613* (2023).

[13] HAO DING, Anoop Deoras, Bernie Wang, and Hao Wang. 2022. Zero-Shot Recommender Systems. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*.

[14] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and robustify negative sampling for implicit collaborative filtering. *Advances in Neural Information Processing Systems* 33 (2020), 1094–1105.

[15] Xiaoyu Du, Xiang Wang, Xiangnan He, Zechao Li, Jinhui Tang, and Tat-Seng Chua. 2020. How to learn item representation for cold-start multimedia recommendation?. In *Proceedings of the 28th ACM International Conference on Multimedia*. 3469–3477.

[16] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems* (<conf-loc>, <city>Seattle</city>, <state>WA</state>, <country>USA</country>, </conf-loc>) *(RecSys '22)*. Association for Computing Machinery, New York, NY, USA, 299–315. https://doi.org/10.1145/3523227.3546767

[17] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning Dense Representations for Entity Retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, Mohit Bansal and Aline Villavicencio (Eds.). Association for Computational Linguistics, Hong Kong, China, 528–537. https://doi.org/10.18653/v1/K19-1049

[18] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Commun. ACM* (1992),

61–70. https://doi.org/10.1145/138859.138867

[19] Andreas Griewank and Andrea Walther. 2000. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Softw.* 26, 1 (mar 2000), 19–45. https://doi.org/10.1145/347837.347846

[20] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[21] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 585–593. https://doi.org/10.1145/3534678.3539381

[22] Budi Juarto and Abba Suganda Girsang. 2021. Neural collaborative with sentence BERT for news recommender system. *JOIV: International Journal on Informatics Visualization* 5, 4 (2021), 448–455.

[23] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. 1997. GroupLens. *Commun. ACM* 40, 3 (1997), 77–87. https://doi.org/10.1145/245108.245126

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* (2009). https://doi.org/10.1109/MC.2009.263

[25] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (<conf-loc>, <city>Long Beach</city>, <state>CA</state>, <country>USA</country>, </conf-loc>) *(KDD '23)*. Association for Computing Machinery, New York, NY, USA, 1258–1267. https://doi.org/10.1145/3580305.3599519

[26] Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. 2023. Exploring the Upper Limits of Text-Based Collaborative Filtering Using Large Language Models: Discoveries and Insights. arXiv:2305.11700 [cs.IR]

[27] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*. https://doi.org/10.1145/3178876.3186150 arXiv:1802.05814

[28] Ahtsham Manzoor, Samuel C. Ziegler, Klaus Maria. Pirker Garcia, and Dietmar Jannach. 2024. ChatGPT as a Conversational Recommender System: A User-Centric Analysis. In *Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization* (Cagliari, Italy) *(UMAP '24)*. Association for Computing Machinery, New York, NY, USA, 267–272. https://doi.org/10.1145/3627043.3659574

[29] Samruddhi Mhatre. 2020. IMDB movies analysis. https://www.kaggle.com/datasets/samruddhim/imdb-movies-analysis

[30] Lien Michiels, Robin Verachtert, and Bart Goethals. 2022. RecPack: An(Other) Experimentation Toolkit for Top-N Recommendation Using Implicit Feedback Data. In *Proceedings of the 16th ACM Conference on Recommender Systems* (Seattle, WA, USA). Association for Computing Machinery, New York, NY, USA, 648–651. https://doi.org/10.1145/3523227.3551472

[31] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

[32] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. 2024. Nomic Embed: Training a Reproducible Long Context Text Embedder. *arXiv preprint arXiv:2402.01613* (2024).

[33] Austin Reese. 2020. Goodreads books - 31 features. https://www.kaggle.com/datasets/austinreese/goodreads-books

[34] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[36] Rachna Sethi and Monica Mehrotra. 2021. Cold start in recommender systems—A survey from domain perspective. In *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020*. Springer, 223–232.

[37] Walid Shalaby, Sejoon Oh, Amir Afsharinejad, Srijan Kumar, and Xiquan Cui. 2022. M2TRec: Metadata-aware Multi-task Transformer for Large-scale and Cold-start free Session-based Recommendations. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 573–578.

[38] Ítallo Silva, Leandro Marinho, Alan Said, and Martijn C. Willemsen. 2024. Leveraging ChatGPT for Automated Human-centered Explanations in Recommender Systems. In *Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI '24)*. Association for Computing Machinery, New York, NY, USA, 597–608. https://doi.org/10.1145/3640543.3645171

[39] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems* 33 (2020), 16857–16867.

[40] Giuseppe Spillo, Cataldo Musto, Marco Polignano, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2023. Combining Graph Neural Networks and Sentence Encoders for Knowledge-aware Recommendations. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization* (Limassol, Cyprus) *(UMAP '23)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3565472.3592965

[41] Martin Spišák, Radek Bartyzal, Antonín Hoskovec, Ladislav Peska, and Miroslav Tůma. 2023. Scalable Approximate NonSymmetric Autoencoder for Collaborative Filtering. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 763–770. https://doi.org/10.1145/3604915.3608827

[42] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*. https://doi.org/10.1145/3308558.3313710 arXiv:1905.03375

[43] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1441–1450. https://doi.org/10.1145/3357384.3357895

[44] Ruixuan Sun, Xinyi Li, Avinash Akella, and Joseph A. Konstan. 2024. Large Language Models as Conversational Movie Recommenders: A User Study. arXiv:2404.19093 [cs.IR]

[45] Gábor Takács, István Pilászy, and Domonkos Tikk. 2011. Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. Association for Computing Machinery, New York, NY, USA, 297–300. https://doi.org/10.1145/2043932.2043987

[46] Gábor Takács and Domonkos Tikk. 2012. Alternating Least Squares for Personalized Ranking. In *Proceedings of the Sixth ACM Conference on Recommender Systems* (Dublin, Ireland) *(RecSys '12)*. Association for Computing Machinery, New York, NY, USA, 83–90. https://doi.org/10.1145/2365952.2365972

[47] Vojtěch Vančura, Rodrigo Alves, Petr Kasalický, and Pavel Kordík. 2022. Scalable Linear Shallow Autoencoder for Collaborative Filtering. In *Proceedings of the 16th ACM Conference on Recommender Systems* (Seattle, WA, USA) *(RecSys '22)*. Association for Computing Machinery, New York, NY, USA, 604–609. https://doi.org/10.1145/3523227.3551482

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[49] Zygmunt Zajac. 2017. Goodbooks-10k: a new dataset for book recommendations. http://fastml.com/goodbooks-10k. *FastML* (2017).

[50] Gangyi Zhang. 2023. User-Centric Conversational Recommendation: Adapting the Need of User with Large Language Models. In *Proceedings of the 17th ACM Conference on Recommender Systems* (Singapore, Singapore) *(RecSys '23)*. Association for Computing Machinery, New York, NY, USA, 1349–1354. https://doi.org/10.1145/3604915.3608885

[51] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 1121–1130.