CUP: a Framework for Resource-Efficient Review-Based Recommenders

Ghazaleh H. Torbati[1][0009−0005−3183−3595], Anna Tigunova[2][0009−0005−7323−2173], Gerhard Weikum[1][0000−0003−4959−6098], and Andrew Yates[3,4][0000−0002−5970−880X]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
{ghazaleh,weikum}@mpi-inf.mpg.de
[2] Amazon, Germany
tigunova@amazon.com
[3] University of Amsterdam, Amsterdam, Netherlands
[4] HLTCOE, Johns Hopkins University, Baltimore, Maryland, USA
andrew.yates@jhu.edu

Abstract. Recommender systems perform well for popular items and users with ample interactions (likes, ratings etc.). This work addresses the difficult and underexplored case of users who have very sparse interactions but post informative review texts. This setting naturally calls for encoding user-specific text with large language models (LLM). However, feeding the full text of all reviews through an LLM has a weak signal-to-noise ratio and incurs high costs of processed tokens. This paper addresses these two issues. It presents a light-weight framework, called CUP, which first computes concise user profiles and feeds only these into the training of transformer-based recommenders. For user profiles, we devise various techniques to select the most informative cues from noisy reviews. Experiments, with book reviews data, show that fine-tuning a small language model with judiciously constructed profiles achieves the best performance, even in comparison to LLM-generated rankings.

Keywords: recommender system · user profile · language model.

1 Introduction

Motivation: Recommender-system methods fall into two major families or hybrid combinations [30]: i) *interaction-based* recommenders that leverage binary signals (e.g., membership in personal playlists or libraries) or numeric ratings for user-item pairs, and ii) *content-based* recommenders that exploit item features and user-provided content, ranging from metadata attributes (e.g., item categories) all the way to review texts.

In settings where interaction data is sparse, content-based methods are the only option, and this is the focus of this work. The most promising approach for this regime is to harness *review texts* by users (e.g., [3,16,40,42]). In domains where users spend substantial time per item (e.g., books, travel destination),

unlike short-attention-span items (e.g., music, video streams), users tend to leave detailed reviews expressing their interests and tastes, even with few interactions. This paper presents a new framework to tackle review-based recommendation with sparse data, long-tail users and items, and rich review texts, especially when computational resources are limited. We focus on the domain of books as a prime case for low interaction rates (i.e., users with few items) with high interaction efforts (i.e., value in user reviews), in combination with high diversity of user tastes (both across users and also per user). Although this is not in the mainstream business, we advocate that long-tail users should receive better service as well.

State-of-the-Art Limitations: Recent works integrated item descriptions and textual reviews into various kinds of recommender architectures, including some based on large language models (LLMs) (e.g., [4,10,13,27]). Our approach differs fundamentally, by making user profiles explicit and transparent, before feeding them into a recommender. This way, lay users can inspect, edit, extend or customize their profiles in a human-friendly manner, while personalizing the downstream application.

There are established works on cold-start support and long-tail items (e.g., [2,12,15,19,28]). These are driven by similarity-aware architectures such as graph models, matrix factorization, and neural methods. Their key asset is to infer explicit or implicit properties of long-tail items and the resulting user preferences, by learning from similar items with richer data. This approach does not carry over to long-tail *users*, though, when most users have sparse data and high diversity in tastes and interests. In book communities such as Goodreads, we encounter many users with just a few tens of items across widely different genres.

Research Challenges: Our approach constructs *concise user profiles* from rich but noisy reviews, and feeds these into the training of a two-tower transformer-based recommender system. This comes with three main challenges: *1) Long-tail Users:* Unlike for long-tail items, there is no way for transferring knowledge from dense-interaction *users* to users in the long tail. The sparseness and the high diversity of user interests and tastes pose unique challenges. *2) Noisy Reviews:* User reviews express a mix of aspects: personal background, interesting traits of the reviewed item, and sentiment expressions. Figure 1 depicts an example review with a mix of informative and uninformative signals. The challenge lies in identifying the scarce parts of a review that convey information about the *item itself* and *why* the user likes (or dislikes) it. *3) Low-resource Computation:* LLMs can handle large inputs, but the computational and energy cost increases with the number of input tokens. The challenge is to get high mileage while keeping the footprint and computation low.

Approach: We devise a light-weight framework, called **CUP**, for constructing Concise User Profiles and encoding them in a recommender system. We adopt a two-tower transformer-based architecture that supports end-to-end learning of item and user encodings, making use of a (small) language model (LM). The end-to-end learning operates on short, judiciously constructed profiles. Our choice

for the encoder is BERT; alternatives such as T5, GPT or Llama can be easily plugged in.

On top of the transformer, we place feed-forward layers, which provide controllable fine-tuning for the downstream recommendation task. The prediction scores for user-item pairs yield the per-user ranking of items. This architecture is relatively simple, but very versatile in supporting different configurations and being able to incorporate a wide range of user profiling techniques.

Our experiments compare against several baselines, including LLM-based ranking. We focus on the book domain, using slices of datasets from Amazon and Goodreads, where we select users with long review text (see Table 1 for dataset statistics). Code and data are available at `https://personalization.mpi-inf.mpg.de/CUP`.

Contributions: Salient contributions of this work are: i) a new framework, called CUP, for transformer-based recommenders that leverage concise user profiles from reviews; ii) judicious techniques for selecting and encoding informative cues from long and noisy reviews; iii) comprehensive experiments with data-poor but text-rich users with highly diverse preferences.

~~The most influential book I've ever read! I first read this book when I was 13, and sometimes I wish I hadn't. This book took away all the naiveté I needed then. I've read this book twice, but it is a long and engaging read.~~ ~~This book will always be my number one.~~ If you want to read something with depth, wisdom, and tragedy, then this book is for you. This book is mostly about the beginnings of myth, superstition, and religion. It challenges the idea of kings and gods, but it isn't a story of emancipation. The Egyptian is an adventure. You can't read this book, and be the same person. You will always live as though you've lived as the Egyptian. ~~It's just really really beautiful. My favorite book of aaaaall time! Please read it.~~

Fig. 1: User-written review, with uninformative text crossed over. Personal background is in purple, pure sentiment in orange, most informative cues in green.

Table 1: Datasets statistics.

	# items	# users per i	# users	# items per u	avg review len
GR	1.5M	6.54	280K	36.77	178
GR-1K-rich	45K	1.17	1K	53.32	426
AM	2.3M	12.75	3M	9.37	121
AM-1K-rich	16K	1.07	1K	16.79	282

2 Related Work

Exploiting User Reviews: Incorporating user-provided reviews into recommenders has been pursued with deep neural networks [3,16,39,40,43,44] and latent-factor models [9,22,32]. Some works augment collaborative filtering (CF) models with user text, to mitigate data sparseness (e.g., [17,18,33]). [38] proposes to learn the importance of review meta-data (age, length etc.), but textual content is disregarded. Other works like [34] incorporate the similarity of user reviews and item descriptions into graph-based learning. Mostly pre-dating the advent of large language models, these methods have been found to have only limited effects [31].

Recent work by [27,35] takes advantage of large language models to generate short user profiles from reviews and item descriptions. Our framework subsumes

this approach as a special case. The brand-new work of [4] includes reviews in learning a model over a shared latent space. In contrast, our framework makes user profiles explicit before feeding them into the recommender, giving the user a chance to inspect and edit this personal data.

Exploiting Language Models: Pre-trained LMs can be leveraged to i) encode item-user signals into transformer-based embeddings, ii) infer recommended items from rich representations of review texts, or iii) implicitly incorporate the latent "world knowledge" of the LM.

A represenative of the first line is P5 [7], which employs prompt templates for the T5 language model. We include an enhanced variant of the P5 method in our experiments. The recent work of [27] generates user profiles by prompting LLMs like Llama, Mistral. The profiles are used for rating prediction, not for ranking a larger pool of candidate items. The method is designed for short reviews; text-rich book reviews are not considered.

On the second direction, the work of [24,25], uses BERT to create representations for user and item text, with (short chunks of) single reviews as granularity. The method then aggregates these per-review vectors by averaging [24] or k-means clustering [25]. Our experiments include BENEFICT [24] as a baseline.

On the "world knowledge" direction, early works, using BERT, elicit knowledge about movie, music and book genres [23]. Recent works prompt large language models (LLMs), such as GPT or Llama, to generate item rankings for user-specific recommendations [8,37] or predict user ratings [11], in a zero-shot or few-shot fashion. Our experiments include [8] as an LLM-powered baseline.

Supporting the Long Tail: Support for long-tail items and users falls under the theme of cold-start and zero-shot recommendations (e.g., [2,12,15,19,28,41]). State-of-the-art methods are reasonably successful on new items, by embedding the item features into the same space as warm items, thus learning relatedness between warm and cold items. This assumes that cold items come with tags and descriptions. For the user side, this assumption is not practical: users would not likely expose a rich profile when they are new to a community or merely occasional contributors. In this data-poor regime, the only option is to harness textual cues from a small number of reviews.

3 Methodology

3.1 System Architecture

The CUP framework is based on a two-tower architecture for representation learning (one "tower" for users, the other for items, following the prevalent architecture in neural information retrieval with query and document/passage encodings). The two towers are jointly trained, coupled by the shared loss function.

User profile and item metadata are fed into BERT followed by a feed-forward network to learn latent representations. Downstream, the vectors are simply compared by a dot product for scores that indicate whether the user likes an item or not. The *per-item* text usually comprises book titles, tags like categories or genre labels, which can be coarse (e.g., "thriller") or fine-grained (e.g., "Scandinavian

crime noir"), and a short description of the book contents. The *per-user* text can comprise the titles and tags of her training-set books and the entirety of her review texts, which vary widely in length and informativeness, hence the need for smart text selection.

3.2 Training

The input to CUP consists of an item metadata (almost always short, otherwise truncated) and a judiciously selected subset of the user-provided text (which may total to a longer text). For user u, this is a sequence of text tokens $w_1^u \dots w_b^u$, where b is the token budget by which the input is limited (set to 128 in our experiments). The sequence is fed through the user tower, consisting of BERT encoder and a feed-forward network (FFN), to obtain a user-representation vector t^u (by averaging the per-token vectors). The FFN has two layers with ReLU activation, computing the final user representation as $t^u = ReLU(t^u W_1^u + c_1^u)W_2^u + c_2^u$; and analogously for items. The score for a *user-item pair* is calculated by $s^{ui} = \sigma(< t^u, t^i >)$ with dot product $<,>$ and sigmoid function σ.

We use the Adam optimizer to minimize the binary cross-entropy loss between predicted labels and the ground truth with sampled negatives. During training we update the top-most layer of BERT, which allows end-to-end training of all components.

3.3 Inference

Prediction for Ranking. At test time, a prediction is made for user-item pairs. We encode the item description by running it through the trained network, and we compare it to the already learned user vector, which is based on the user's training-time reviews. The scores for different test items, computed by the final dot product, yield the ranking of the candidate items. This is a very efficient computation, following established practice in neural IR [14].

Search-based Recommendation. In a deployed system (as opposed to lab experiments with test samples), a typical usage mode would be search-based re-ranking: a user provides context with a keyphrase query or an example of a specific liked item, which can be thought of as query-by-example. The user's expectation is to see a ranked list of recommended items that are similar to her search intent (as opposed to recommendations from all kinds of categories). The system achieves this by first computing approximate matches to the query (i.e., similarity-search neighbors), and then re-ranking a shortlist of say top-100 candidates. The CUP framework supports this mode, by using a light-weight BM25 retrieval model. To evaluate the model in this mode without ground truth query-user-item triples, we search over all unlabeled items with the category and textual description of the positive test point at hand, and keeping the top-100 highest scoring matches.

3.4 Coping with Long and Noisy Texts

For constructing user profiles from text, the simplest idea would be to concatenate all available reviews into a long token sequence. Two problems arise, though.

User reviews are a noisy mix of descriptive elements (e.g., "the unusual murder weapon"), sentiment expressions (e.g., "it was fun to read") and personal but irrelevant statements (e.g., "I read only on weekends"). Only the first aspect is helpful for content-based profiling (as the sentiment is already captured by user liking the book). Second, the entirety of user-provided text can be too long to be fully digested by the Transformer. Even when it would fit into the token budget, the computational and energy costs are quadratic in the number of input tokens. Therefore, we tightly limit the tokens for each user's text profile to 128, and devise a suite of light-weight techniques for judiciously selecting the most informative pieces.

Our techniques for selecting the most informative parts of user reviews into a concise user profile are as follows:

- **Weighted Phrases:** selected words or 3-grams, ordered by descending tf-idf weights, where tf is the frequency of the phrase in all of the user's reviews, and idf is pre-computed on Google books n-grams to capture informativeness.
- **Weighted Sentences:** selected sentences, ordered by descending idf weights, where a sentence's total weight is the sum of the per-word idf weights normalized by sentence length.
- **Similar Sentences:** selected sentences, ordered by descending similarity scores computed via Sentence-BERT [29] for comparing the user-review sentences against the description of the corresponding item. To ensure that the selected set is not dominated by a single review, the sentences are picked from different reviews in a round-robin manner.
- **ChatGPT-generated Profiles:** feeding all reviews of a user, in large chunks, into ChatGPT and instructing it to characterize the user's book interests with a few keyphrases.
- **T5-generated Keywords:** using a T5 model fine-tuned for keyword generation, to cast each user's review text into a set of keywords, concatenated to create the profile.
- **Llama-generated Profiles:** instructing Llama [5] to generate profiles given the user reviews in two formats: keywords, and first-person narrative, by giving it a few in-context examples.

We provide anecdotal examples of selected user profile constructions in Table 6.

3.5 Coping with Unlabeled Data

A challenge for training in the data-poor regime is handling the extreme skew between positive samples and unlabeled data points for sparse users. The crux in many recommender applications is that there are extremely few, if any, explicitly negative samples, such as books rated with low marks. This holds also for the datasets in this work. Therefore, we introduce and experiment with two different techniques to construct negative training samples from unlabeled data:

Uniform random samples. Under the closed world assumption (CWA), aka Selected Completely At Random, negative training points are sampled uniformly from all unlabeled data. This is a widely used standard technique.

Weighted pos-neg samples. Prior works on PU learning [1], with positive and unlabeled data and without explicitly negative samples, is largely based on treating unlabeled points as pairs of samples, one positive and one negative with fractional weights. The weights can be based on (learned estimates of) class priors, but the extreme skew in our data renders these techniques ineffective. Instead, we leverage the fact that relatedness measures between item pairs can be derived from interaction data. We compute *item-item relatedness* via matrix factorization of the user-item matrix for the entire dataset. The relatedness of two items is set to the scalar product between their latent vectors, re-scaled for normalization between 0 and 1. Each originally unlabeled sample is cloned: one instance positive with weight proportional to its average relatedness to the user's explicitly positive points; the negative clone's weight is set to the complement.

4 Experimental Design

Rationale: As a difficult and less explored application area for recommenders, we investigate the case of book recommendations in online communities. These come with a long-tailed distribution of user activities, highly diverse user interests, and demanding textual cues from reviews and book descriptions.

Unlike in many prior works' experiments, often on movies, restaurants or mainstream products, the data in our experiments is much sparser regarding user-item interactions. We design the evaluation as a *stress-test* experiment, with focus on text-rich users. With the focus on lightweight computation, we limited the input context to 128 tokens, hence the need for smart user profile extraction. Our experiments supports the choice of budget and architecture.

We further enforce the *difficulty of predictions* when items belong to groups with high relatedness within a group, by constraining disjointedness of authors per user in training and evaluation set. Thus, we rule out the near-trivial case of predicting that a user likes a certain book given that another book by the same author has been used for training.

Datasets: We use two book datasets, Goodreads [36] (**GR**) and Amazon books [21] (**AM**), from the UCSD recommender systems repository [20]. Both datasets contain item titles, genre or category tags, item description, and user reviews.

Prior works mostly consider C-core data variants where all users and items have at least C interactions (C=10 or 5). This pre-processing focuses on interaction-based predictions, whereas our intention is to study the case of data-poor users and items. Instead, our data pre-processing is designed to evaluate text-based recsys performance with text-rich users. We select 1K users from each of the two datasets, based on descending order of average review length per book (filtered for English reviews). We view all book-user interactions with a rating of 4 or higher as positive, and disregard the lower ratings as they are rare anyway. We split the data into training, validation, and test sets (60:20:20), filtering out users with less than 3 items to guarantee at least one interaction per user in each set. Table 1 gives statistics for the datasets. Both 1K-rich data variants are extremely sparse in terms of users that share the same items; so the emphasis is on leveraging text.

Baselines: We compare our approach to several state-of-the-art baselines, which cover different methods , ranging from traditional collaborative filtering approaches to text-centric neural models and LLM-based rankers:

- **CF**: collaborative filtering operating on user-item interaction matrix by computing per-user and per-item vectors (dim=200) via matrix factorization [6].
- **LLMRank**: following [8], we use ChatGPT to rank the test items, given the user's reading history. The history is given by the sequence of titles of the 50 most recent books of the user, prefixed by the prompt "I've read the following books in the past in order:". This prompt is completed by a list of titles of test-time candidate items, asking the LLM to rank them.
- **P5-profile**: prompting the T5 language model [26], to provide a recommended item for a user, given their ids. Following [7], we train P5 using the prompts for *direct recommendation* to generate a *"yes"* or *"no"* answer. Pilot experiments show that the original method does not work well on sparse data. Therefore, we extend P5 to leverage review texts and item descriptions. Instead of ids, the prompts include item descriptions and sentences from reviews with the highest idf scores (i.e., one of our own techniques).
- **BENEFICT**[24]: uses a frozen BERT model to create representations for each user review, which are averaged and concatenated to the item vectors. Predictions are made by a feed-forward network on top. Following the original paper, each review is truncated to its first 256 tokens.
- **BENEFICT-profile**: our own variant of BENEFICT where the averaging over all user reviews is replaced by our idf-based selection of most informative sentences, with the total length limited to 128 tokens (for comparability CUP).

Performance Metrics: Following the literature, we report NDCG@5 (Normalized Discounted Cumulative Gain) with binary 0-or-1 gain and P@1 (precision at rank 1). We compute these by micro-averaging over all test items of all users. Macro-averaged results over users were not significantly different, hence they are not reported in the paper. NDCG@5 reflects the observations that users care only about a short list of top-N recommendations; P@1 is suitable for recommendations on mobile devices (with limited UI). We also measured other metrics, like NDCG@k for higher k, MRR and AUC. None of these provides any additional insight, so they are not reported here.

Evaluation Modes: At test time, for each positive test item we sample 100 negative items from all unlabeled data. The system scores and ranks these 101 data points, creating a ranked list of items to be evaluated by the performance metrics introduced above. We evaluate all methods in two different modes with respect to the negative sampling strategy:

- **Standard:** sampling the 100 negative test points uniformly at random.
- **Search-based:** given the positive test item, searching for the top-100 approximate matches to the item's description, using the BM25 scoring model.

Configurations: In the experiments, all CUP variants use an input budget of 128 tokens as a stress test, emphasizing our goal of limiting the computational

and energy costs. The following CUP configurations cover different ways of user profile creation (see Subsection 3.4):

- **CUP**$_{idf}$: review sentences selected by idf scores (weighted sentences).
- **CUP**$_{sbert}$: review sentences selected by similarity to the corresponding item description, using Sentence-BERT (similar sentences).
- **CUP**$_{1gram}$: unigrams selected by tf-idf scores (weighted phrases).
- **CUP**$_{3gram}$: 3-grams selected by tf-idf scores (weighted phrases).
- **CUP**$_{kwT5}$: set of keywords generated by a fine-tuned T5 model.[5]
- **CUP**$_{kwGPT}$: a keyword profile generated by ChatGPT (gpt-3.5-turbo).
- **CUP**$_{kwLlama}$: a keyword profile generated by Llama (Meta-Llama-3.1-8B-Instruct), given hand-crafted few-shot examples.
- **CUP**$_{absLlama}$: an abstractive 1st-person narrative profile generated by Llama, given hand-crafted few-shot examples.

For comparison, we also configure a simpler variant, **CUP***tags*, which uses genre tags from the user's training items as the user text. To observe the effect of item metadata, we further restrict this variant to use only the item title and genre as item text, denoted as CUP*basic*.

We used the following hyperparameters for CUP configuration, obtained though grid search: 4e-5 as learning rate, 256 as batch size, 200 as FFN size. All methods were run on NVIDIA Quadro RTX 8000 GPU with 48 GB memory, and we implemented the models with PyTorch.

5 Experimental Results

Table 2: Standard evaluation.

AM-1K-rich				
	Train Uniform		Train Weighted	
Method	NDCG@5	P@1	NDCG@5	P@1
CF	3.06	1.0	2.88	0.69
LLMRank	4.62	2.27	n/a	n/a
BENEFICT	9.4	3.53	14.4	5.74
BENEFICT$_{prof}$	24.38	12.98	24.66	12.81
P5$_{prof}$	24.9	14.5	n/a	n/a
CUP$_{basic}$	25.42	13.64	26.95*	14.27
CUP$_{tags}$	27.31*	14.99*	28.83*	16.05*
CUP$_{idf}$	**29.21***	**15.82***	**31.09***	**17.71***
GR-1K-rich				
CF	4.44	2.69	3.83	2.26
LLMRank	4.86	2.1	n/a	n/a
BENEFICT	23.76	12.17	25.23	13.26
BENEFICT$_{prof}$	30.26	16.83	31.77	17.74
P5$_{prof}$	28.01	14.46	n/a	n/a
CUP$_{basic}$	26.75	13.28	28.4	14.89
CUP$_{tags}$	30.92	16.3	33.28*	17.83
CUP$_{idf}$	**38.39***	**22.26***	**39.41***	**22.01***

Table 3: Search-based evaluation.

AM-1K-rich				
	Train Uniform		Train Weighted	
Method	NDCG@5	P@1	NDCG@5	P@1
CF	3.02	1.0	3.03	0.83
LLMRank	3.49	1.1	n/a	n/a
BENEFICT	2.45	0.66	3.69	1.29
BENEFICT$_{prof}$	6.49	2.33	8.11	3.53
P5$_{prof}$	8.4*	**3.88***	n/a	n/a
CUP$_{basic}$	7.13	2.76	6.87	2.61
CUP$_{tags}$	7.41	2.93	7.0	2.84
CUP$_{idf}$	**8.93***	3.53*	**9.14**	**4.02**
GR-1K-rich				
CF	3.73	2.02	3.25	1.74
LLMRank	4.57	1.79	n/a	n/a
BENEFICT	6.73	2.38	6.88	2.44
BENEFICT$_{prof}$	8.95	3.4	9.63	3.59
P5$_{prof}$	9.15	3.01	n/a	n/a
CUP$_{basic}$	9.35	3.46	9.84	3.55
CUP$_{tags}$	10.66*	4.3*	11.18*	3.94
CUP$_{idf}$	**14.18***	**5.9***	**13.76***	**5.32***

5.1 Comparison of CUP against Baselines

Table 2 shows the results for the AM-1K-rich and GR-1K-rich data, comparing our default configuration CUP$_{idf}$ against all baselines, for the two different

[5] `https://huggingface.co/ml6team/keyphrase-generation-t5-small-inspec`

ways of sampling negative training points. Results with statistical significance over the BENEFICT$_{prof}$ baseline, by a paired t-test with p-value < 0.05, are marked with an asterisk. Bonferroni correction for multi-hypotheses testing is applied, reducing the test level of each pairwise comparison to 0.005. We make the following key observations:

- The interaction-centric CF fails for this extremely sparse data. BENEFICT, utilizing the entire user review texts, shows poor performance. At the same time, BENEFICT$_{prof}$ and P5$_{prof}$, extended with our text-derived profiling, achieve decent performance.
- LLMRank also performs poorly. Solely relying on the LLM's latent knowledge about books is not sufficient when coping with long-tail items. Popularity and position bias [8] further aggravate this adverse effect. To mitigate position bias, we ran a variant with smaller test sets of only 20 candidate negative items (per positive test item), as in the original setup of [8]. This boosts the NDCG@5 for LLMRank from 4.8% to 23.5%, on GR-1k-rich in standard evaluation, which is still a large margin below CUP$_{idf}$ reaching 38.3% (and similarly big gaps for the other dataset).
- Between the three CUP configurations, we see a clear trend: review-based user profiling (*idf*) outperforms user profiles based on genres or categories (*tags*), and simplifying item-side text to titles and tags alone by removing item description (*basic*) performs worst. Remarkably, even CUP$_{basic}$ is better than all the baselines. CUP$_{idf}$ is ca. 5 percentage points better in NDCG@5 than the baselines.
- In search-based evaluation (Table 3), the absolute results are much lower, emphasizing the difficulty of this realistic mode. Still, the relative comparisons between methods are nearly identical to the results with standard evaluation. Again, CUP$_{idf}$ is the winner, with a clear margin.
- The absolute numbers on GR-1K-rich are generally higher, due to the different data characteristics. The gains by CUP$_{idf}$ over the baselines and over the simpler CUP configurations are even more pronounced (e.g., outperforming BENEFICT$_{prof}$ by 8 percentage points with standard evaluation).

5.2 Efficiency of CUP

Two architectural choices make CUP efficient:

- input length restricted to 128 tokens, and
- fine-tuning only the last layer of BERT and the FFN layers.

For more analysis, we measured the training time and resulting NDCG on the GR-1K-rich data, comparing different input size budgets and choices of tunable parameters. Figure 2 shows the NDCG@5 results on the validation set.

We observe that the 128-token configuration has the lowest training cost: significantly less time per epoch than the other variants and fast convergence (reaching its best NDCG already after 15 epochs in ca. 3000 seconds). The 256- and 512-token models eventually reach higher NDCG, but only by a small margin and after much longer training time.

As for the number of trainable parameters, we observe that the variant with frozen BERT takes much longer to converge and is inferior to the preferred CUP method even after more than 50 epochs. The other extreme, allowing all BERT parameters to be altered, performs best after enough training epochs. However, it takes almost twice as much time per epoch. From the benefit/cost perspective, our design choice hits a sweet spot in the spectrum of model configurations.

Table 4: CUP results, by user/item groups (NDCG@5 with Search-based evaluation).

AM-1K-rich							
Method	ALL	u-s	u-r	u-b	s-s	s-r	s-b
CUP_{idf}	9.14	5.93	7.09	12.0	8.6	11.71	12.78
CUP_{sbert}	9.0	5.19	**7.32**	11.46	9.44	14.53	14.23
CUP_{1gram}	9.08	**6.24**	7.14	11.21	**9.76**	**17.0**	13.25
CUP_{3gram}	8.98	5.54	7.01	11.81	8.2	13.97	10.99
CUP_{kwT5}	**9.5**	6.03	7.15	**12.48**	8.71	12.38	**17.78**
CUP_{kwGPT}	8.93	5.95	6.95	11.59	8.29	11.3	13.85
$CUP_{kwLlama}$	9.01	6.23	6.9	11.82	7.83	11.15	12.45
$CUP_{absLlama}$	8.04	5.14	5.86	10.63	5.23	16.11	11.94
GR-1K-rich							
CUP_{idf}	13.76	7.32	**11.42**	14.96	17.37	17.43	19.56
CUP_{sbert}	13.51	**8.47**	10.41	15.05	16.14	16.85	19.79
CUP_{1gram}	13.47	7.82	10.61	14.64	16.14	17.95	20.32
CUP_{3gram}	13.19	7.3	10.47	15.08	14.06	16.16	17.94
CUP_{kwT5}	13.76	7.51	11.03	14.43	17.43	19.33	22.17
CUP_{kwGPT}	**13.78**	7.97	10.56	14.33	18.32	**20.36**	**23.18**
$CUP_{kwLlama}$	13.54	8.06	10.63	**15.25**	13.88	16.83	19.54
$CUP_{absLlama}$	13.22	6.75	9.87	14.2	**18.76**	19.59	21.37

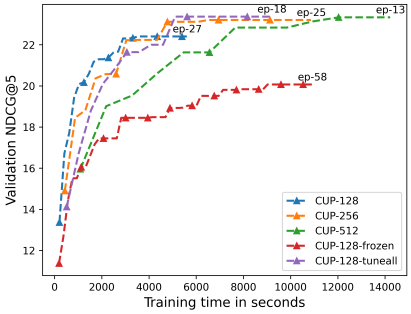

Fig. 2: Training time for different input lengths and trainable parameters (lines are marked every 5th epoch).

5.3 Comparison of CUP Configurations

For insight on specific groups of users and items, we split the 1000 users and their items into the following groups, reporting NDCG@5 for each group separately. Note that this refinement drills down on the test outputs; the training is unaffected.

- Items are split into **unseen (u)** and **seen (s)** items. Unseen test items have not been seen at training time. Seen test items appeared as positive training items for a different user.
- Users are split into groups based on the ♯books-per-user distribution:
  - **Sporadic (s)** users are the lowest 50% with the least numbers of books. For GR-1K-rich, this threshold is 13 books per user; for AM-1K-rich it is 5 (with means 6 and 3, resp.).
  - **Regular (r)** users are those between the 50th percentile and 90th percentile, which is between 13 and 71 books per user for GR-1K-rich, and between 5 and 20 for AM-1K-rich (with means 31 and 9, resp.).
  - **Bibliophilic (b)** users are the highest 10%: above 75 books per user for GR-1K-rich and above 20 for AM-1K-rich (with means 156 and 43, resp.).

Table 4 shows NDCG@5 (for all and per item/user group) with search-based mode, comparing all CUP configurations with weighted training. We offer the following notable observations:

- Across all groups , all CUP configurations are competitive. The overall differences between them are relatively small. The winner, by a small margin, is CUP_{kwT5} for AM and CUP_{kwGPT} for GR datasets, closely followed by the default configuration CUP_{idf} and $\text{CUP}_{kwLlama}$ as well as CUP_{sbert}. None of the methods is able to extract the "perfect" gist from the noisy review texts; but all of them do a decent job. Despite the fact that generated profiles are slightly ahead of the others, the bottom line is that a relatively simple configuration, like idf-selected sentences, is a good choice.
- The CUP_{kwGPT} variant achieves its highest gains for the richer item/user groups: seen items and regular or bibliophilic users (GR dataset). This provides ChatGPT with longer and more informative texts. A similar effect, but to a lesser and noisier extent, can be observed for T5-based CUP_{kwT5}. Conversely, these methods perform substantially worse on the sporadic-unseen group.

	Amazon					Goodreads				
	POS					POS				
	NN	VB	AD	avg. idf	KL-div	NN	VB	AD	avg. idf	KL-div
all reviews	0.25	0.19	0.17	0.009	3.28	0.28	0.19	0.16	0.008	2.93
idf sentences	0.33	0.16	0.18	0.008	1.72	0.39	0.15	0.19	0.011	1.19
Llama abstractive	0.35	0.15	0.18	0.065	1.7	0.36	0.14	0.19	0.084	1.31
Llama keywords	0.67	0.05	0.23	0.42	1.55	0.68	0.04	0.23	0.503	1.27
unigrams	0.47	0.23	0.25	0.537	1.99	0.54	0.18	0.22	0.921	1.22

Table 5: Statistical comparison of different kinds of user profiles.

Desiderata for an ideal profile are i) *high utility:* leading to strong recommender performance, ii) *easy interpretability:* supporting humans in understanding the gist of somebody's interests, and iii) *sound faithfulness:* capturing the user's style in writing reviews. Clearly, there are trade-offs between these dimensions. In terms of utility, our experiments show that several kinds of profiles are roughly on par, with some simple ones being slightly ahead. On the other hand, the most interpretable profiles are the generative ones using an LM. Finally, the extractive profiles, like salient sentences from reviews, appear most faithful. For illustration, Table 6 provides examples of different kinds of profiles.

To obtain more insights, we computed various statistics: distributions of part-of-speech (POS) tags (i.e., word categories) and distributions of idf-weight mass among frequent words. These are derived by concatenating profiles of all users, for each of the most interesting profile types, including the users' original reviews.

The statistics are shown in Table 5. The first column denotes the fractions of the three most frequent POS types. We observe that the LM-generated profiles have a much higher share of nouns, which are more informative than verbs or adjectives/adverbs. The second column shows the average idf weight for the words in the 20-th percentile of the word frequency distribution. We observe that the original reviews carry low idf weight, reflecting their verbose and noisy nature. In constrast, the generative profiles select high-idf words as most informative cues. Finally, the third column shows the Kullback-Leibler divergence (with Laplace smoothing, $\alpha = 0.1$) for the user profiles generating the book descriptions (with removal of stopwords). We observe that the vocabulary of original reviews differs

significantly from the wording in book descriptions, whereas idf-aware profiles and keyword-generated profiles are much closer to the vocabulary that describes book contents.

6 Conclusion

This work presented a transformer-based framework CUP, with novel techniques for constructing concise user profiles by judiciously selecting informative pieces of user text. Our experiments, with both standard evaluation and a search-based mode, show that leveraging user text is beneficial in this data-poor regime, and that CUP methods clearly outperform state-of-the-art baselines like BENEFICT, P5, and LLM-Rec. Among the CUP configurations with different profiles, we observed that most perform similarly. This suggests two main options in practice: choose the lowest-cost variant which uses idf-based n-grams or sentence-level excerpts of reviews, or choose the ones that are generated by an LLM, at higher cost, if use cases prioritize the human-readability and ability to edit profiles.

Table 6: User profiles constructed by various methods (truncated to max 3 lines).

Method	User Profile
genres	africa, nature ecology, americas, history, europe, leaders notable people, relationships, world, science math, historical, biographies memoirs, self - help, genre fiction, literature fiction
idf sentences	Confederate Navy Raider. Irish history - in a nutsell!. Heroes, US Marine Corps Medal of Honor Winners by Marc Cerasini. Women's Options on the American Fronteir. Norwegian Immigration 1850s. Life on the Praire. Nice Quirky Book.
SBERT sentences	It was actually mostly written by Elisabeth Koren, wife of Reverend U. V. Koren. The area of research is the Arkansas Missouri Borderlands. Story is of a woman radical whose life was brought up short by Senator McCarthy during
unigrams	texas, navy, colt, marines, koren, norwegian, quege, ranger, vilhelm, norwegians, nutsell, leponto, markist, rangers, caliber, fronteir, cerasini, korens, book, cavalry, laundress, pistols, xo, agnes, praire, outstanding, army
T5 keywords	norweger immigration 1850s book, texas, texas, funeral. amateur historian, history buffs, acoustic borderlands, ignoble deeds, south. socialist woman, gender issues, birth control, abortion, labor unions, health care, social care
ChatGPT keywords	norwegian immigration, 1800s life, historical commentary, civil war, confederate navy, mexican war, texas cavalry, american frontier, lost states, irish history, texas rangers, mexican war, agnes smedley, norwegian female lives
Llama keywords	Norwegian Immigration, 1850s, historical reference, Civil War, eyewitness accounts, Arkansas Missouri Borderlands, Confederate Navy Raider, Mexican War, Fort Brown, Western Frontier, Simon Kenton, D. Boone, Crusades
Llama 1st-person	I'm a history buff with a passion for non - fiction books, particularly those that delve into the lives of ordinary people during extraordinary times. I enjoy reading about the experiences of women, immigrants, and marginalized groups

References

1. Bekker, J., Davis, J.: Learning from positive and unlabeled data: a survey. Mach. Learn. **109**(4) (2020)
2. Cao, E., Wang, D., Huang, J., Hu, W.: Open knowledge enrichment for long-tail entities. In: WWW '20. ACM / IW3C2 (2020)
3. Chen, C., Zhang, M., Liu, Y., Ma, S.: Neural attentional rating regression with review-level explanations. In: WWW '18. ACM (2018)
4. Dong, H.V., Fang, Y., Lauw, H.W.: A contrastive framework with user, item and review alignment for recommendation. In: WSDM '25. ACM (2025)
5. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., et al.: The Llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
6. Funk, S.: Netflix Update: Try This at Home (2006), `https://sifter.org/~simon/journal/20061211.html,`, accessed on Sep 29, 2022
7. Geng, S., Liu, S., Fu, Z., Ge, Y., Zhang, Y.: Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In: RecSys '22. ACM (2022)
8. Hou, Y., Zhang, J., Lin, Z., Lu, H., Xie, R., McAuley, J.J., Zhao, W.X.: Large language models are zero-shot rankers for recommender systems. In: ECIR '24. Springer (2024)
9. Hu, G., Zhang, Y., Yang, Q.: Transfer meets hybrid: A synthetic approach for cross-domain collaborative filtering with text. In: WWW '19. ACM (2019)
10. Huang, C., Yu, T., Xie, K., Zhang, S., Yao, L., McAuley, J.J.: Foundation models for recommender systems: A survey and new perspectives. arXiv preprint arXiv:2402.11143 (2024)
11. Kang, W.C., Ni, J., Mehta, N., Sathiamoorthy, M., Hong, L., Chi, E., Cheng, D.Z.: Do LLMs understand user preferences? Evaluating LLMs on user rating prediction. arXiv preprint arXiv:2305.06474 (2023)
12. Li, J., Jing, M., Lu, K., Zhu, L., Yang, Y., Huang, Z.: From zero-shot learning to cold-start recommendation. In: AAAI '19 (2019)
13. Lin, J., Chen, B., Wang, H., Xi, Y., Qu, Y., Dai, X., Zhang, K., Tang, R., Yu, Y., Zhang, W.: Clickprompt: CTR models are strong prompt generators for adapting language models to CTR prediction. In: WWW '24. ACM (2024)
14. Lin, J., Nogueira, R.F., Yates, A.: Pretrained Transformers for Text Ranking: BERT and Beyond. Synthesis Lectures on Human Language Technologies, Springer (2022)
15. Liu, B., Bai, B., Xie, W., Guo, Y., Chen, H.: Task-optimized user clustering based on mobile app usage for cold-start recommendations. In: KDD '22. ACM (2022)
16. Liu, D., Li, J., Du, B., Chang, J., Gao, R.: DAML: dual attention mutual learning between ratings and reviews for item recommendation. In: KDD '19. ACM (2019)
17. Liu, H., Wang, Y., Peng, Q., Wu, F., Gan, L., Pan, L., Jiao, P.: Hybrid neural recommendation with joint deep representation learning of ratings and reviews. Neurocomputing (2020)
18. Lu, Y., Dong, R., Smyth, B.: Coevolutionary recommendation model: Mutual learning between ratings and reviews. In: WWW '18. ACM (2018)
19. Luo, S., Ma, C., Xiao, Y., Song, L.: Improving long-tail item recommendation with graph augmentation. In: CIKM '23. ACM (2023)
20. McAuley, J.: Recommender Systems and Personalization Datasets (2022), `https://cseweb.ucsd.edu/~jmcauley/datasets.html,`, accessed on Sep 29, 2022

21. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: EMNLP-IJCNLP '19 (2019)
22. Peña, F.J., O'Reilly-Morgan, D., Tragos, E.Z., Hurley, N., Duriakova, E., Smyth, B., Lawlor, A.: Combining rating and review data by initializing latent factor models with topic models for top-n recommendation. In: RecSys '20. ACM (2020)
23. Penha, G., Hauff, C.: What does BERT know about books, movies and music? Probing BERT for conversational recommendation. In: RecSys '20. ACM (2020)
24. Pugoy, R.A., Kao, H.Y.: BERT-based neural collaborative filtering and fixed-length contiguous tokens explanation. In: ACL '20 (2020)
25. Pugoy, R.A., Kao, H.: Unsupervised extractive summarization-based representations for accurate and explainable collaborative filtering. In: ACL/IJCNLP '21. ACL (2021)
26. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research (2020)
27. Ramos, J., Rahmani, H.A., Wang, X., Fu, X., Lipani, A.: Transparent and scrutable recommendations using natural language user profiles. In: ACL '24 (2024)
28. Raziperchikolaei, R., Liang, G., Chung, Y.: Shared neural item representations for completely cold start problem. In: RecSys '21. ACM (2021)
29. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: EMNLP-IJCNLP '19. ACL (2019)
30. Ricci, F., Rokach, L., Shapira, B. (eds.): Recommender Systems Handbook. Springer US (2022)
31. Sachdeva, N., McAuley, J.J.: How useful are reviews for recommendation? A critical review and potential improvements. In: SIGIR '20. ACM (2020)
32. Shalom, O.S., Uziel, G., Kantor, A.: A generative model for review-based recommendations. In: RecSys '19. ACM (2019)
33. Shalom, O.S., Uziel, G., Karatzoglou, A., Kantor, A.: A word is worth a thousand ratings: Augmenting ratings using reviews for collaborative filtering. In: SIGIR '18 (2018)
34. Shuai, J., Zhang, K., Wu, L., Sun, P., Hong, R., Wang, M., Li, Y.: A review-aware graph contrastive learning framework for recommendation. In: SIGIR '22. ACM (2022)
35. Torbati, G.H., Tigunova, A., Weikum, G.: SIRUP: search-based book recommendation playground. In: WSDM '24. ACM (2024)
36. Wan, M., McAuley, J.: Item recommendation on monotonic behavior chains. In: RecSys '18 (2018)
37. Wang, L., Lim, E.P.: Zero-shot next-item recommendation using large pretrained language models. arXiv preprint arXiv:2304.03153 (2023)
38. Wang, X., Ounis, I., Macdonald, C.: Leveraging review properties for effective recommendation. In: WWW '21. ACM / IW3C2 (2021)
39. Wu, C., Wu, F., Ge, S., Qi, T., Huang, Y., Xie, X.: Neural news recommendation with multi-head self-attention. In: EMNLP-IJCNLP '19 (2019)
40. Wu, L., He, X., Wang, X., Zhang, K., Wang, M.: A survey on neural recommendation: From collaborative filtering to content and context enriched recommendation. arXiv preprint arXiv:2104.13030 (2021)
41. Zang, T., Zhu, Y., Liu, H., Zhang, R., Yu, J.: A survey on cross-domain recommendation: Taxonomies, methods, and future directions. arXiv preprint arXiv:2108.03357 (2021)
42. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Comput. Surv. (2019)

43. Zhang, Y., Ai, Q., Chen, X., Croft, W.B.: Joint representation learning for top-n recommendation with heterogeneous information sources. In: CIKM '17. ACM (2017)
44. Zheng, L., Noroozi, V., Yu, P.S.: Joint deep modeling of users and items using reviews for recommendation. In: WSDM '17. ACM (2017)