Recommendations in Sparse-Data Low-Resource Settings by Constructing Concise User Profiles from Review Text

Ghazaleh H. Torbati
ghazaleh@mpi-inf.mpg.de
Max Planck Institute for Informatics
Saarbrücken, Germany

Anna Tigunova*
tigunova@amazon.com
Amazon
Berlin, Germany

Gerhard Weikum
weikum@mpi-inf.mpg.de
Max Planck Institute for Informatics
Saarbrücken, Germany

Andrew Yates
a.c.yates@uva.nl
University of Amsterdam
Amsterdam, Netherlands

Abstract

Recommender systems are successful for popular items and users with ample interactions (likes, ratings etc.). This work addresses the difficult case of users who have very sparse interactions but post informative review texts, with the additional desideratum of low-cost computation. We specifically address book communities, as they exhibit sparseness (most users reviewed only tens of books) and a long tail of user interests (including books that are reviewed only by few other users). We design a light-weight framework with transformer-based representation learning, covering user-item interactions, item content, and user-provided reviews. A key contribution is a suite of novel techniques for selecting the most informative cues from user-written reviews to construct concise user profiles. Comprehensive experiments, with datasets from Amazon and Goodreads, show that judicious selection of text snippets achieves the best performance, even in comparison to LLM-generated rankings and to using LLMs to generate user profiles.

CCS Concepts

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

Keywords

recommender, sparse data, user text, language model, transformer

ACM Reference Format:
Ghazaleh H. Torbati, Anna Tigunova, Gerhard Weikum, and Andrew Yates. 2024. Recommendations in Sparse-Data Low-Resource Settings by Constructing Concise User Profiles from Review Text . In *Proceedings of the 3rd International Workshop on Industrial Recommendation Systems (at CIKM 2024) (IRS '24)*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Work done prior to joining Amazon.

IRS '24, October 25, 2024, Boise, Idaho

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

1.1 Motivation

Recommender-system methods fall into two major families or hybrid combinations [22]: 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. In particular, long-tail users and long-tail items pose major challenges. Some works advocate countering data sparseness by clustering long-tail items (e.g., [14]). However, this is not sufficient to gear up for users with few interactions but diverse tastes. Approaches for cold-start and cross-domain recommendations and zero-shot learning (e.g., [8, 10, 20, 31]), mostly focus on the item side only: transferring (latent) knowledge from existing items to new ones. User-side sparseness, where a user likes only a few items, is much less explored, and particularly challenging when these few items are in the long tail of the popularity distribution.

The most promising approach for this data-poor regime is to leverage *review texts* by users (e.g., [2, 11, 29, 32]). In this work, we focus on settings where users spend substantial time per item. This holds for the vertical domains of books and travel destinations, as opposed to items with short attention spans, like music, restaurants and video streams. In book communities, even users with few interactions often leave rich texts that reflect their interests and tastes. The book domain thus provides a challenging stress-test, especially when computational resources are limited and with focus on long-tail items and users. This paper presents a new framework to tackle both data sparseness and diversity of tastes with low computational resources, by constructing concise user profiles from review texts.

1.2 Research Questions

Our approach constructs *concise user profiles* from reviews, focusing on the book domain. Unlike movies or restaurants, books have a much longer tail of popularity and exhibit huge diversity of user tastes. Also, negative reviews are very rare (most have 4 or 5 stars), so that we deal with binary data and have no negative samples. This setting raises several research questions (RQs):

RQ1: **Learning with Language Models for Data-Poor Users.** As most users have very sparse interactions with items, we need to leverage cues in the users' reviews. This naturally calls for making use of language models. How can we best incorporate models like BERT, T5, GPT or Llama for encoding the gist of a user's reviews, enhanced with the LM's world knowledge?

RQ2: **Low-resource Computation.** Although modern language models can handle fairly large text inputs, the computational cost substantially increases with the number of input tokens. This involves the monetary costs of API calls, and also the compute cycles and energy consumption at both training and inference time. Therefore, the issue is how to make best use of model inputs that impose a tight limit on their number of tokens?

RQ3: **Review Aspects.** User reviews express a mix of aspects: personal background (e.g., "I'm a retired teacher"), sentiment expressions (e.g., "what a great story"), general emotions (e.g., "brings back lovely memories"), and comments about the book contents. Figure 1 depicts a challenging example of a mixture of informative and uninformative content found in user-written reviews from real data. As sentiments do not add value for a book already known to be liked, only the content comments yield cues towards new recommendations. How can we identify the truly informative pieces for concise user profiles?

~~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.~~

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

1.3 Approach

We devise a light-weight framework, called **CUP**, for constructing Concise User Profiles and encoding them in a recommender system. We pursue a two-tower transformer-based architecture that supports end-to-end learning of item and user encodings, making use of a language model (LM). The end-to-end learning operates on sufficiently short, judiciously constructed profiles. Our choice for the LM is BERT; alternatives such as T5, GPT or Llama can be easily plugged in. However, the latter incur substantially higher computational cost. We want to focus on low-resource settings, hence the conservative choice of BERT. Our experiments will show that this configuration yields results on par with or better than methods that employ T5 or GPT.

On top of the transformer, we place feed-forward layers, which provide more 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. It can take as input a spectrum of categorical and textual cues about users and items.

The rationale for this design is twofold: i) ensuring that inference-time computations are efficient, leveraging training-time-computed vectors for users and items and solely computing a scalar product at run-time, and ii) being able to express a wide range of user profiling techniques within the same architecture. Alternative architectures, such as CNN-based, or LLM-based with smart prompts, are baselines in our experiments.

In experiments, we limit the number of tokens per user profile, as a stress-test and to keep the computational cost and and energy consumption as light-weight as possible.

The experimental data, from two book communities, exhibits a very long tail of less popular items and users with diverse interests. Our experiments systematically explore the system behavior of coping with user-side sparseness. As an additional stress-test, we study both per-user recommendation and search-based recommendation, where the latter is initiated by user query (asking for books that are both relevant to the query and suitable for the user profile).

1.4 Contributions

Salient contributions of this work are:

- a new framework, called CUP, for Transformer-based recommenders that leverage language models and concise user profiles from reviews;
- judicious techniques for selecting and encoding informative cues from the long and noisy text of user reviews, outperforming methods based on prompting large language models (LLMs);
- comprehensive experiments with data-poor but text-rich users with highly diverse preferences.

Code and data are available on our project page https://personalization.mpi-inf.mpg.de/CUP. An interactive demo for exploring a suite of configurations is running at https://sirup.mpi-inf.mpg.de.

2 Related Work

Content-based recommenders incorporate item tags, item-item similarity, and user-side features. Item-item similarity typically computes distances between item embeddings. This can be combined with interaction-based methods that employ latent-space techniques (e.g., [3, 18, 23, 30, 32]).

The most important user features are reviews of items, posted with likes or ratings. State-of-the-art methods employ deep neural networks with attention mechanisms [2, 11, 28, 29, 33, 34]. However, pre-dating the advent of LLMs, these methods rely on static word-level encodings such as word2vec, and are inherently limited. As a salient representative, we include DeepCoNN [34] in baselines of our experiments.

Recent works leverage pre-trained LMs (mostly BERT) for recommenders, 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.

An early representative of the first line is BERT4Rec [24], which uses BERT to learn item representations for sequential predictions based on item titles and user-item interaction histories, but does not incorporate any text. The P5 method of [5] employs a suite of prompt templates for the T5 language model, in a multi-task learning framework covering direct as well as sequential recommendations along with generating textual explanations. We include an enhanced variant of the P5 method in our experiments.

On the "world knowledge" direction, early works, using BERT, elicit knowledge about movie, music and book genres [15]. Recent works prompt large language models (LLMs), such as GPT or PaLM, to generate item rankings for user-specific recommendations [6, 26] or predict user ratings [7], in a zero-shot or few-shot fashion. Our experiments include [6] as an LLM-powered baseline.

Closest to our approach are the methods of [16, 17], using BERT to create representations for user and item text, aggregated by averaging [17] or k-means clustering [16]. The resulting vectors are used for predicting item scores. A major limitation is that the text encodings are for individual sentences only, losing signals from user reviews where cues span multiple sentences. Also, BERT itself is fixed, and the vectors for users and items are pre-computed without awareness of the prediction task. Our experiments include the BENEFICT method of [17] as a baseline.

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 and ground truth. Figure 2 shows a pictorial overview.

User reviews and items descriptions 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. Importantly, unlike prior works, we allow the top-most layer of BERT to be fine-tuned as part of the end-to-end training process.

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.

In the following, we present the CUP training procedure in Subsection 3.2, the test-time inference in Subsection 3.3, the judicious selection of input text snippets in Subsection 3.4, and the selection of negative training samples in Subsection 3.5.

3.2 Training

The input to CUP consists of an item description (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 $w_1^u \ldots w_b^u$, where b is the token budget

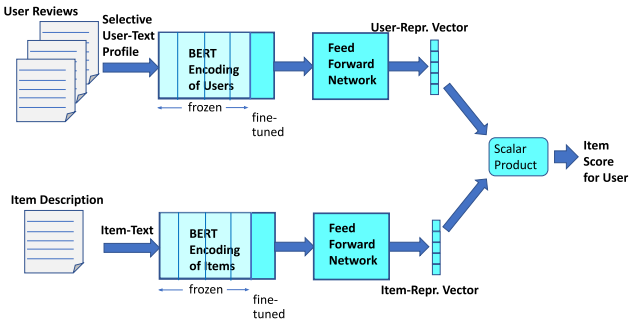

Figure 2: CUP architecture

by which the input is limited (set to 128 in our experiments). The sequences are fed through each of the two towers, consisting of BERT 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

$t^u = ReLU(t^u W_1^u + c_1^u) W_2^u + c_2^u$

for user representation, and analogously for items.

A *user-item pair* is classified with score

$s^{ui} = \sigma(< t^u, t^i >)$

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. This is an important difference to earlier works with similar architectures, which use frozen BERT to encode user and item representations.

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 (none for the given test item). 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 [9].

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 tag-based 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 positive sample (as opposed to recommendations from all kinds of categories). The system achieves this by first computing approximate matches to the query item (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, querying all unlabeled items with the category and textual description of the positive point at hand, and keeping the top-100 highest scoring matches.

Table 1: Statistics for datasets (only ratings ≥ 4, #books per user ≥ 3, train-test disjoint authors per user)

	#books	avg ± stdv #users per book	#users	avg ± stdv #books per user	avg ± stdv review len
GR	1,573,290	6.54 ± 55.95	279,969	36.77 ± 93.25	178 ± 259
GR-10K-dense	385,660	3.74 ± 18.96	10,000	144.16 ± 232.49	187 ± 259
GR-10K-sparse	158,554	1.66 ± 3.09	10,000	26.25 ± 56.46	173 ± 248
GR-1K-rich	45,412	1.17 ± 0.65	1000	53.32 ± 91.04	426 ± 384
AM	2,281,866	12.75 ± 87.25	3,105,696	9.37 ± 26.55	121 ± 183
AM-10K-dense	203,930	2.51 ± 7.3	10,000	51.19 ± 177.37	239 ± 281
AM-10K-sparse	58,443	1.36 ± 1.6	10,000	7.93 ± 11.19	106 ± 169
AM-1K-rich	15,753	1.07 ± 0.36	1000	16.79 ± 23.53	282 ± 265

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 cost is 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.

Variants of our methodology are to create concise profiles by generative language models upfront (e.g., using T5 or ChatGPT), or to digest multiple chunks of text with downstream aggregation by max-pooling or averaging. We include these techniques in our experimental comparisons, but emphasize that both of these alternatives have much higher computational cost and climate footprint.

Our techniques for selecting the most informative parts of user reviews 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 [21] for comparing the user-review sentences against the sentences of the item description. To ensure that the selected set is not dominated by a single item, the sentences are picked from different items 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 short keyphrases. The output size is limited per chunk with a total budget of 128 tokens, to enforce concise profiles.
- **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.

We provide anecdotal examples of selected user profile constructions in Table 7 in the Appendix.

3.5 Coping with Unlabeled Data

A challenge for training in the data-poor regime is how to handle 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. Then, each originally unlabeled sample is cloned, with 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

4.1 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 but otherwise data-poor users: who liked relatively few items but wrote informative, yet noisy reviews. 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.

4.2 Datasets

We use two book datasets from the UCSD recommender systems repository [12], filtered for english reviews:

- **GR [25]:** a Goodreads sample with item-user interactions for 1.5M books and 280K users, incl. titles, genre tags, item descriptions, ratings and textual reviews.
- **AM [13]:** an Amazon crawl for the books domain with 2.3M books and 3.1M users, incl. category tags, ratings and reviews.

While the GR data hardly appears in the literature, the AM-books data has been used for experiments in prior works (e.g., [27]), mostly in the 10-core variant where are all users and items having less than 10 interactions are eliminated. This pre-processing clearly focuses on interaction-based predictions, whereas our intention is to study the underexplored case of sparse interactions with informative user 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 further pre-process the datasets by removing all users with less than 3 books, as we cannot split their interactions into training, validation, and test sets (with ratio 60:20:20).

Our data pre-processing is designed to evaluate text-based recsys performance with low interaction density and text-rich users. We select 1K users from each of the two datasets, based on descending order of average review length per book. In Table 1, rows GR-1K-rich and AM-1K-rich show the characteristics of these data slices. Both GR-1K-rich and AM-1K-rich are extremely sparse in terms of users that share the same items ; so the emphasis is on leveraging text.

4.3 Baselines

We compare our approach to several state-of-the-art baselines, which cover different methods for recommendation, ranging from traditional collaborative filtering approaches to text-centric neural models. We compare the following methods:

- **CF**: collaborative filtering operating on the user-item interaction matrix by pre-computing per-user and per-item vectors via matrix factorization [4] (with 200 latent dimensions).
- **DeepCoNN** [34] is a salient representative of using convolutional neural networks (CNN) over text inputs.
- **LLMRank**: following [6], 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** [5]: prompting the T5 language model [19], to provide a recommended item for a user, given their ids. Following [5], 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** [17] uses BERT 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 reviews of a user is replaced by our idf-based selection of most informative sentences, with the total length limited to 128 tokens (for comparability to the CUP methods).

4.4 Performance Metrics

At test time, we present the trained system with each user's withheld positive items (20% of the user's books, with authors disjoint from those of the user's training items), along with negative items, sampled from all non-positive items, such that the ratio of positive to negative test points is 1:100. The system scores and ranks these data points. We evaluate all methods in two different modes:

- **Standard prediction:** sampling the 100 negative test points uniformly at random from all unlabeled data, and ranking the 100+1 test instances by the methods under test.
- **Search-based:** given the positive test item, searching for the top-100 approximate matches to the item's description, using the BM25 scoring model; then ranking the 100 + 1 candidates by our methods.

Following the literature, our evaluation metrics are 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. We also experimented with macro-averaging over users; as the results were not significantly different, we report only micro-average numbers 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.

4.5 Configurations

The CUP framework supports a variety of methods by specific configurations. All variants use an input budget of 128 tokens, to construct a stress-test and to emphasize that computational and environmental footprint is a major concern as well. In the experiments, we focus on the following text-centric options (see Subsection 3.4):

- **CUP**$_{idf}$: review sentences selected by idf scores.
- **CUP**$_{sbert}$: review sentences selected by Sentence-BERT similarity to a corresponding item description.
- **CUP**$_{1gram}$: unigrams selected by tf-idf scores.
- **CUP**$_{3gram}$: 3-grams selected by tf-idf scores.
- **CUP**$_{keywords}$: set of keywords generated by a fine-tuned T5 model.[1]
- **CUP**$_{GPT}$: a concise set of keyphrases generated by ChatGPT from all reviews of a user.

When comparing against prior baselines, we employ CUP$_{idf}$ as default configuration. For comparison, we also configure a more restricted variant, CUP$_{basic}$, that uses only genre tags as user text and title, and genre as item text. We enhance this by additionally using description for item text, denoted as CUP$_{exp}$.

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.

Table 2: Standard evaluation on both datasets.

Method	Train Uniform NDCG@5	Train Uniform P@1	Train Weighted NDCG@5	Train Weighted P@1
AM-1K-rich				
CF	3.06	1.0	2.88	0.69
DeepCoNN	3.0	0.8	3.01	0.83
LLMRank	4.62	2.27	n/a	n/a
P5$_{text}$	24.9	14.5*	n/a	n/a
BENEFICT	9.4	3.53	14.4	5.74
BENEFICT$_{text}$	24.38	12.98	24.66	12.81
CUP$_{basic}$	25.42	13.64	26.95*	14.27*
CUP$_{exp}$	27.31*	14.99*	28.83*	16.05*
CUP$_{text}$	**29.21***	**15.82***	**31.09***	**17.71***
GR-1K-rich				
CF	4.44	2.69	3.83	2.26
DeepCoNN	10.45	4.02	5.4	1.86
LLMRank	4.86	2.1	n/a	n/a
P5$_{text}$	28.01	14.46	n/a	n/a
BENEFICT	23.76	12.17	25.23	13.26
BENEFICT$_{text}$	30.26	16.83	31.77	17.74
CUP$_{basic}$	26.75	13.28	28.4	14.89
CUP$_{exp}$	30.92	16.3	33.28*	17.83
CUP$_{text}$	**38.39***	**22.26***	**39.41***	**22.01***

5 Experimental Results

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$_{text}$ (i.e., CUP$_{idf}$ with idf-selected sentences) against all baselines, for the two different ways of sampling negative training points. Results with statistical significance over the BENEFICT$_{text}$ baseline, by a paired t-test with p-value < 0.05, are marked with an asterisk. We make the following key observations:

- The interaction-centric CF fails completely for this extremely sparse data. The text-based baseline DeepCoNN also performs very poorly, and the BENEFICT method is only slightly better. Both DeepCoNN and BENEFICT utilize the entire user review texts without any judicious filtering of the noisy text. At the same time, P5$_{text}$ and BENEFICT$_{text}$, extended with our text-derived profiling, achieve decent performance. This emphasizes the need to address RQ3 (see Section 1.2).
- LLMRank also performs poorly. Solely relying on the LLM's latent knowledge about books is not sufficient when coping with long-tail items that appear sparsely in the LLM's training data. Popularity and position bias [6] further aggravate this adverse effect. Thus, to address RQ1 (see Section 1.2), merely and fully resorting to LLMs alone is not a viable solution.
- Between the three CUP configurations, we see a clear trend: titles and tags alone (*basic*) are outperformed by adding item descriptions (*exp*), and the profiling over user reviews (*text*) performs best. This confirms our intuition about RQ2 (see Section 1.2): judicious selection of textual cues is important. Remarkably, even CUP$_{basic}$ is better than all the baselines. CUP$_{text}$ 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$_{text}$ 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$_{text}$ over the baselines and over the simpler CUP configurations are even more pronounced (e.g., outperforming BENEFICT$_{text}$ by 8 percentage points with standard evaluation).

To obtain insight on the root cause for the unsatisfactory performance of LLMRank, we also ran a variant with smaller test sets of only 20 candidate items (per positive test item), as in the original setup of [6]. This boosted 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$_{text}$ reaching 38.3% (and similarly big gaps for the other dataset). Obviously, even in this simplified setting the LLM misses out on "knowledge" about the many long-tail items, the key challenge in our setting.

Another observation is that the weighted training almost always improves performance. Therefore, the following subsections focus on results with weighted training samples.

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

Table 3: Search-based evaluation on both datasets.

Method	Train Uniform NDCG@5	Train Uniform P@1	Train Weighted NDCG@5	Train Weighted P@1
AM-1K-rich				
CF	3.02	1.0	3.03	0.83
DeepCoNN	0.05	0.0	0.05	0.0
LLMRank	3.49	1.1	n/a	n/a
$P5_{text}$	8.4*	**3.88***	n/a	n/a
BENEFICT	2.45	0.66	3.69	1.29
$BENEFICT_{text}$	6.49	2.33	8.11	3.53
CUP_{basic}	7.13	2.76	6.87	2.61
CUP_{exp}	7.41*	2.93	7.0	2.84
CUP_{text}	**8.93***	3.53*	**9.14***	**4.02**
GR-1K-rich				
CF	3.73	2.02	3.25	1.74
DeepCoNN	1.84	0.55	1.35	0.35
LLMRank	4.57	1.79	n/a	n/a
$P5_{text}$	9.15	3.01	n/a	n/a
BENEFICT	6.73	2.38	6.88	2.44
$BENEFICT_{text}$	8.95	3.4	9.63	3.59
CUP_{basic}	9.35	3.46	9.84	3.55
CUP_{exp}	10.66*	4.3*	11.18*	3.94
CUP_{text}	**14.18***	**5.9***	**13.76***	**5.32***

Additionally, we ran a variant of the best performing model (CUP_{text}) using increased input size of 256 tokens. We observed only slight improvements at best. More on this cost/benefit ratio of larger token budgets is in Subsection 5.2.

Finally, to study CUP beyond its primary regime of sparse data, we experimented with data slices that have higher density of interactions. Subsection 5.4 discusses this study.

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 3 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. This confirms that concise user profiles, with a small token budget, are the best approach in terms of benefit/cost ratio.

As for configurations with more or less tunable 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 of BERT parameters to be altered , performs best after enough training epochs, better than the CUP configuration that tunes only the last BERT layer. However, it takes almost twice as much time per epoch. So again, from the benefit/cost perspective, our design choice hits a sweet spot in the spectrum of model configurations.

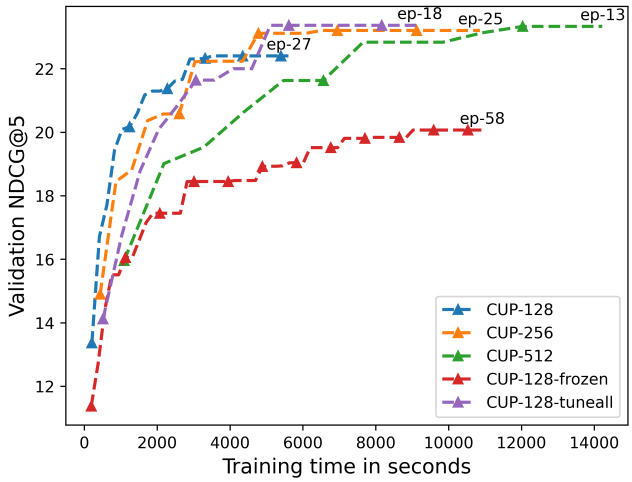

Figure 3: Training time for different input lengths and trainable parameters (lines are marked every 5th epoch).

5.3 Comparison of CUP Configurations

To obtain refined insights into the performance for specific kinds 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. The former consist of all positive test-time items that have not been seen at training time. The latter are those items that appear as positive samples at test-time and are also among the positive training items (for a different user).
- Users are split into three 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.).

We now turn to comparing all CUP configurations, with drilling down on user and item groups. Table 4 shows the NDCG@5 results for the AM-1K-rich and GR-1K-rich data. From here on, all

Table 4: CUP results, by user/item groups, on both datasets (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_{kw}	**9.5**	6.03	7.15	**12.48**	8.71	12.38	**17.78**
CUP_{GPT}	8.93	5.95	6.95	11.59	8.29	11.3	13.85
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_{kw}	13.76	7.51	11.03	14.43	17.43	19.33*	22.17*
CUP_{GPT}	**13.78**	7.97	10.56	14.33	**18.32**	**20.36***	**23.18***

presented results are obtained with weighted training (as this outperforms uniform training) and with search-based evaluation. We focus on NDCG@5, as the P@1 metric strongly correlates. The asterisk for statistical significance is by a paired t-test with $p < 0.05$ against CUP_{idf}. We offer the following notable observations:

- Across all users, all CUP configurations are competitive. The overall differences between them are relatively small. The winner, by a small margin, is CUP_{kw}, closely followed by the default configuration CUP_{idf} and CUP_{sbert} as well as CUP_{1gram}. 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 CUP_{kw} is slightly ahead of the others, the bottom line is that a relatively simple configuration, like idf-selected sentences, is a very good choice. Notably, even the seemingly most promising CUP_{GPT} performs on par with the simpler configurations.
- The CUP_{GPT} variant achieves its highest gains for the richer item/user groups: seen items and regular or bibliophilic users. 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_{kw}. Conversely, these methods perform substantially worse on the sporadic-unseen group.
- On the GR-1K-rich data, the overall performance is higher than for AM across all configurations. This can be attributed to the fact that GR has longer reviews, and these texts tend to be more informative than the ones in the AM data (which sometimes refer to packaging, shipment and other non-content aspects).

5.4 Influence of Interaction Density

The slices AM-1K-rich and GR-1K-rich are constructed with a text-centric stress-test in mind. Both are extremely sparse in terms of user-item interactions. To study the influence of sparseness in isolation, we created two larger samples of both datasets, one still sparse by design and the other denser in terms of users sharing items. We refer to these as AM-10K-sparse and AM-10K-dense, and analogously for GR. All these samples cover 10K users: 10x more users than our previous text-rich slices, to allow more potential for learning from interactions. Table 1 shows the statistics.

The datasets are constructed as follows. To ensure connectivity in the interaction graphs, we first select 500 users and sample 2000 books connected to these users, both uniformly at random. This is the seed for constructing two variants of data, based on the cumulative item degrees of users, that is, the sum of the #users per book for all books that the user has in her collection:

- **10K-dense:** We randomly select 10K (minus the initial 500) users from all users in proportion to the users' cumulative item degrees. This favors users with many or popular books.
- **10K-sparse:** We select users inversely proportional to their cumulative item degrees, thus favoring sparse users.

Table 5: NDCG@5 for AM-10K Sparse vs. Dense (Search-based evaluation, by user/item groups)

Method	ALL	u-s	u-r	u-b	s-s	s-r	s-b
AM-10K-Sparse							
CF	5.06	0.0	0.01	0.01	18.79	17.16	12.39
CUP_{text}	**9.78**	**3.4**	**3.9**	**6.64**	**22.43**	**21.65**	**18.29**
CUP_{text}+CF	5.79	1.65	0.43	0.13	19.45	18.77	13.0
AM-10K-Dense							
CF	19.02	0.13	0.03	0.14	**51.76**	34.75	21.46
CUP_{text}	**20.24**	**1.23**	**2.44**	**5.81**	49.98	34.4	21.12
CUP_{text}+CF	19.12	0.38	0.04	0.27	51.7	**35.04**	**21.49**

Table 6: NDCG@5 for GR-10K Sparse vs. Dense (Search-based evaluation, by user/item groups)

Method	ALL	u-s	u-r	u-b	s-s	s-r	s-b
GR-10K-Sparse							
CF	15.71	0.0	0.0	0.01	**46.44**	36.62	28.0
CUP_{text}	**17.97**	**3.9**	**5.79**	**8.89**	37.29	32.64	26.18
CUP_{text}+CF	16.4	2.25	0.65	0.07	42.91	**39.37**	**29.3**
GR-10K-Dense							
CF	39.73	0.02	0.0	0.01	64.81	55.22	40.38
CUP_{text}	36.42	**4.66**	**5.18**	**7.38**	56.91	48.24	36.34
CUP_{text}+CF	**41.22**	0.29	0.04	0.04	**65.48**	**56.76**	**43.03**

In this sensitivity study, we focus on comparing CF (based on matrix factorization) against our default CUP_{text} configuration (with idf-selected sentences) and the hybrid combination CUP_{text} + CF. In the hybrid setting, we enhance text-based representations with collaborative filtering (CF) signals, by concatenating the learned latent per-user and per-item vectors. These are precomputed by factorizing the training-set user-item interaction matrix, using the method of [4], with gradient descent for minimizing a cross-entropy loss.

Tables 5 and 6 show the results for the AM-10K data and GR-10K data, comparing the sparse vs. the dense variants with search-based

evaluation. Results with the standard evaluation have the same trends, but with a higher absolute values.

Key insights are:

- As the 10K-sparse data is already much denser than the stress-test 1K-rich slices, CF can achieve decent results on the sparse data. Especially in search-based mode, CF is almost on par with CUP. Note that these gains come from the seen items alone, as CF is bound to fail on unseen items.
- CUP$_{text}$ is still the clear winner on the sparse variant. The hybrid configuration, with text and CF vectors combined, is inferior to learning from text alone.
- For the 10K-dense data, CF alone performs well, as this is the traditional regime for which CF has been invented. An interesting point arises for the search-based mode and with dense data. Here, the negative test points are closer to the positive sample (after the BM25 search), and pose higher difficulty for text cues alone to discriminate them. Thus, CF becomes more competitive. On GR, hybrid CUP+CF performs best.

6 Conclusion

This work addressed text-centric recommender systems in data-poor situations, specifically for the demanding case of book-reviewing communities where many users have only a few items and exhibit diverse tastes, but post text-rich comments. We presented a transformer-based framework CUP, with novel techniques for constructing concise user profiles by selecting informative pieces of user text. To mitigate the absence of explicitly negative training points, we used a technique of picking weighted negative samples from unlabeled data. 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 DeepCoNN, BENEFICT, P5, and LLMRank.

One limitation of our study is the size of our curated dataset, which we used for our offline experiments. Running large scale online evaluation is beyond the scope of this work, as such data is available only on propriety platforms of big enterprises.

Finally, we identify the following directions for future work. First, we plan to extend our experiments to further recommendation domains (e.g., movies or video games). Although the book domain stands out, due to its rich textual content, our proposed framework can be effective in further applications. Second, our work is centered around concise user profile construction, leaving the selection of the downstream ranking model for further research.

References

[1] Jessa Bekker and Jesse Davis. 2020. Learning from positive and unlabeled data: a survey. *Mach. Learn.* 109, 4 (2020).
[2] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *WWW '18*. ACM.
[3] Ghazal Fazelnia, Eric Simon, Ian Anderson, Benjamin A. Carterette, and Mounia Lalmas. 2022. Variational User Modeling with Slow and Fast Features. In *WSDM '22*. ACM.
[4] Simon Funk. 2006. *Netflix Update: Try This at Home.* https://sifter.org/~simon/journal/20061211.html, accessed on Sep 29, 2022.
[5] 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 *RecSys '22*. ACM.
[6] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2024. Large language models are zero-shot rankers for recommender systems. In *ECIR '24*. Springer.
[7] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. 2023. Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction. (2023). arXiv:2305.06474
[8] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From Zero-Shot Learning to Cold-Start Recommendation. In *AAAI '19*.
[9] Jimmy Lin, Rodrigo Frassetto Nogueira, and Andrew Yates. 2021. *Pretrained Transformers for Text Ranking: BERT and Beyond.* Morgan & Claypool Publishers.
[10] Bulou Liu, Bing Bai, Weibang Xie, Yiwen Guo, and Hao Chen. 2022. Task-optimized User Clustering based on Mobile App Usage for Cold-start Recommendations. In *KDD '22*. ACM.
[11] Donghua Liu, Jing Li, Bo Du, Jun Chang, and Rong Gao. 2019. DAML: Dual Attention Mutual Learning between Ratings and Reviews for Item Recommendation. In *KDD '19*. ACM.
[12] Julian McAuley. 2022. *Recommender Systems and Personalization Datasets.* https://cseweb.ucsd.edu/~jmcauley/datasets.html, accessed on Sep 29, 2022.
[13] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP-IJCNLP '19*.
[14] Yoon-Joo Park. 2013. The Adaptive Clustering Method for the Long Tail Problem of Recommender Systems. *IEEE Trans. Knowl. Data Eng.* 25, 8 (2013).
[15] Gustavo Penha and Claudia Hauff. 2020. What does BERT know about books, movies and music? Probing BERT for Conversational Recommendation. In *RecSys '20*. ACM.
[16] Reinald Adrian Pugoy and Hung-Yu Kao. 2021. Unsupervised Extractive Summarization-Based Representations for Accurate and Explainable Collaborative Filtering. In *ACL/IJCNLP '21*. ACL.
[17] Reinald Adrian Pugoy and Hung-Yu Kao. 2020. BERT-Based Neural Collaborative Filtering and Fixed-Length Contiguous Tokens Explanation. In *ACL '20*.
[18] Tao Qi, Fangzhao Wu, Chuhan Wu, and Yongfeng Huang. 2021. Personalized News Recommendation with Knowledge-aware Interactive Matching. In *SIGIR '21*. ACM.
[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* (2020).
[20] Ramin Raziperchikolaei, Guannan Liang, and Young-joo Chung. 2021. Shared Neural Item Representations for Completely Cold Start Problem. In *RecSys '21*. ACM.
[21] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP '19*. ACL.
[22] Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2022. *Recommender Systems Handbook.* Springer US.
[23] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Mag.* 42, 3 (2021).
[24] 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 *CIKM '19*. ACM.
[25] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *RecSys '18*.
[26] Lei Wang and Ee-Peng Lim. 2023. Zero-Shot Next-Item Recommendation using Large Pretrained Language Models. (2023). arXiv:2304.03153
[27] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR '19*.
[28] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural news recommendation with multi-head self-attention. In *EMNLP-IJCNLP '19*.
[29] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A Survey on Neural Recommendation: From Collaborative Filtering to Content and Context Enriched Recommendation. (2021). arXiv:2104.13030
[30] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. (2020). arXiv:2011.02260
[31] Tianzi Zang, Yanmin Zhu, Haobing Liu, Ruohan Zhang, and Jiadi Yu. 2021. A Survey on Cross-domain Recommendation: Taxonomies, Methods, and Future Directions. (2021). arXiv:2108.03357
[32] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* (2019).
[33] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources. In *CIKM '17*. ACM.
[34] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *WSDM '17*. ACM.

A User Profile Examples

Table 7 shows anecdotal examples for the profiles generated for an Amazon user and a Goodreads user, all cropped to the 128-token limit (or less if the construction method does not fully utilize its budget).

	method	user profile
Amazon user	**genres**	photography video, worship devotion, arts photography, religion spirituality
	ChatGPT	amateur photography, everyday language, easy explanation, thorough explanations, covers many topics, beautiful photos, settings, down to earth, basic photographer, wealth of information, photo examples, exposure settings, inspirational, graduation gift,
	idf-sentence	This was so inspirational. I love the way Bryan Peterson writes his books. It has sections for shutter speed, aperture, light as well as special techniques and filters. The photos are beautiful. He has also included so much information without cluttering the writing. I decided to purchase this book after seeing some info in a magazine. It even has a chapter on film vs. digital. Great gift idea. Great gift idea. I have given several away as graduation gifts. I have now been sharing it with my granddaughter as she is picking up the camera now. I say if you are an amateur photographer or helping someone else,
	SBERT	Great book on exposure. I have now been sharing it with my granddaughter as she is picking up the camera now. I have given several away as graduation gifts. So many times it is the exposure settings we can not get right and this book will definitely do the job. I have had this book for a while now and as a amateur photographer I refer back to it often. This was so inspirational. The way that " Understanding Exposure " is written, it does not do that. I say if you are an amateur photographer or helping someone else, it is well worth getting this book. The response I received from the graduate was also positive.
	T5 keywords	exposure, british photographer, beginner, professional, information overload, book, explanations. perfect book. reporter and amateur photographer, it is written in everyday language, family and friends, setting, reading. graduation gifts. great gift idea
Goodreads user	**genres**	mystery thriller crime, fiction, romance, history historical fiction biography, fantasy paranormal, non - fiction, children, young - adult, comics graphic
	ChatGPT	clara quinn, finn's harbor, maine, occult bookstore, murder investigation, psychic abilities, small - town setting, friendship, suspects, psychic gift, murder trap, maine setting, cozy mystery, 3 1 / 2 - 4 stars, tv production company, murder, suspect, determined widow, 1960s vegas, rat pack, frank sinatra, judy garland, high stakes poker, las vegas, mystery, arapaho, father john o'malley, vicky holden, short stories, novella, essays,
	idf-sentence	Fans of Erika Chase or Kylie Logan will enjoy the latest in Laura DiSilverio's Readaholics series. Welcome back, Hercules Poirot! Readers who enjoy Elaine Viets will enjoy a trip to Fernglen Galleria. I received this book from NetGalley, through the courtesy of Byliner. Fans of Rick Mofina or Robin Burcell would enjoy this well - written novel.. She is now a "mall cop" at Virginia's Fernglen Galleria. A newcomer named Donald Webster has arrived in Taviscombe. I really enjoyed this suspenseful thriller featuring Dr. Samantha Owens.
	SBERT	She has a rare condition called "Cotard's Syndrome" which is both a help and a danger to Fiona as she goes undercover to help solve her latest case. Interesting mystery and some very touching moments with Peabody, who is my favorite character in this series! Mrs. Jeffries is the housekeeper for Inspector Witherspoon of Scotland Yard. Boston Homocide Detective Jane Rizzoli and Medical Examiner Maura Isles team up to solve a series of horrible murders that could be tied to past events. One of the best books I have ever read. I love this new series and look forward to reading more by author susannah hardy.
	T5 keywords	Sehr geehrte Damen und Herren! cozy mystery authors, american, gardener, gardener, cottages, wrought iron irony, wrought. legal thrillers, national guard, family history, spiritual convictions, narrative. tv series, mystery series, foxfox, foxfox, foxfox, mystery writer, black cat bookshop mysteries, bookshop cat, online game, word game, detective, friedman. amelia thistle, a newcomer, is determined to protect her identity, her deceased family friend, anthon. Jeffries series, french tutor, crime

Table 7: Profiles of selected users constructed by various methods