On the Role of Numerical Encoding in Foundation Model of Sequential Recommendation with Sequential Indexing

Refaldi Intri Dwi Putra, Toyotaro Suzumura

Graduate School of Information Science and Technology
The University of Tokyo
{refaldiputra,suzumura}@g.ecc.u-tokyo.ac.jp

Abstract

We study a foundation model for recommender systems named P5 on a sequential recommendation task with sequential indexing. The P5 needs to handle numerical values due to the usage of user-item IDs in the prompt. However, it is unclear how the prior numerical encoding affects the task. We think this requires special attention since it may pose a challenge to the performance and future development of recommender systems. To do so, we run experiments where the prior numerical encoding is set from an addition task. We find that by doing so, it can improve the performance compared with the vanilla P5. We also find that this performance is affected by the structure of the priors. However, the models no longer retain their addition ability. This gives insight into the role of numerical encoding in the foundation models for recommender systems.

1 Introduction

Foundation models (FMs) have emerged as a new paradigm in the deep learning field (Bommasani et al. 2021). FMs are trained on broad data sets and can be used for various downstream tasks. Recently, we have seen the power of FMs in the language domain through GPT4, Llamas, etc. (OpenAI 2023; Touvron et al. 2023). They can perform outstanding various language tasks such as translation and generation. Moreover, this exciting new paradigm is prevalent in machine learning applications.

An example is to build an FM for the recommender systems (RS). Such a model may play an important role in a 'healthy' recommender ecosystem (*recosystem*) (Boutilier, Mladenov, and Tennenholtz 2023). In this study, we are interested in an FM for RS named P5 (Geng et al. 2022). The P5 is based on language modeling and was considered the first FM for RS. It is trained by using a unified framework in natural language processing. Its model is based on T5 (Raffel et al. 2020), an encoder-decoder Transformer model (Vaswani et al. 2017). Some common RS tasks such as rating prediction and sequential recommendation are done in a single training. This approach gives P5 the advantage of performing various RS tasks in a single training.

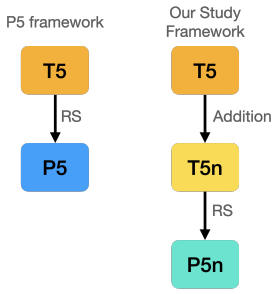
In more detail, the P5 is trained on special prompt templates for each respective task. The prompts would be filled in by user and item IDs. For example, in the sequential recommendation task, a prompt template was "What would ⟨user_id⟩ be likely to purchase next after buying items {⟨item_id⟩} ?". In which, ⟨user_id⟩ and ⟨item_id⟩ were a mixture of text and numerical values such as user_23 and item_7391. Because of this, *P5 needs to handle numerical values*. However, when it comes to numbers, language models seemed to have difficulty handling it. For example, in the GPT-3 technical paper (Brown et al. 2020) (see section 3.9 in Arxiv's version) the models were tested for arithmetic tasks where only the models with billion parameters could produce the correct answers. Even for GPT-4, the accuracy was still 59% as stated in (Dziri et al. 2023). Therefore, it makes sense to ask whether similar problems will happen in FMs for RS, especially related to the numerical encoding.

Recently, there was also an extended work of open-sourced P5, named OpenP5 (Xu, Hua, and Zhang 2023), which we use as our framework. Here, the terms P5 and OpenP5 are used interchangeably. In OpenP5, they re-initialized the numerical encoding of pre-trained T5 with a random distribution, see section 5.2 in (Xu, Hua, and Zhang 2023). This is because the numbers serve as identifiers of the user items and may not follow the natural structure of numbers (e.g. 5>3). However, it is unclear whether this prior setting is suitable for the FMs that *also* need to grasp the concept of the number's structure.

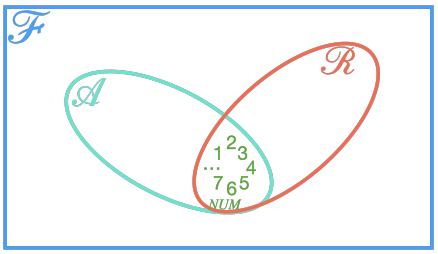
We think this requires special attention, considering that future RS is going to exploit various side information of the users (He et al. 2023). For example, if a timestamp or price tag is used, the models may need to understand the number's structure. This may also pose a problem for recosystem, where the numbers embedding needs to find a balance between the two concepts. Which, the formation of these concepts should be affected by prior distribution of numerical encoding.

Objective of this study. Since it is still unclear whether prior numerical encoding with specific structure affects the performance of RS. We think assessing this problem is *important to get insights on the role of numerical encoding in FMs for RS, as well as the way to improve its performance*. So, this study sets out to investigate this specific problem.

To do so, we first train T5 with an addition task to become T5n. We chose the addition task because it is a number-



(a) The illustration of our framework compared with existing work.



(b) The illustration of our assumptions on concepts in P5n.

Figure 1: The illustrations of our study. (a) Framework of P5n that we study here is fine-tuned with an addition task first, (b) Assumed concepts where the number's concept is shared together with arithmetic ($\mathcal{A}$) and RS ($\mathcal{R}$) tasks.

related task and due to its simplicity. Moreover, T5 had relatively good performance for this task as reported in (Nogueira, Jiang, and Lin 2021) that may induce a specific structure to the prior. After that, we train T5n on a sequential recommendation task to become *P5n*. This framework may induce a shared concept of numbers between RS ($\mathcal{R}$) and the addition task ($\mathcal{A}$). We illustrate the P5n training framework compared with the vanilla P5 and the assumed shared number concept in Figure 1.

Furthermore, indexing is fundamental in P5 to capture collaborative and personalized feature representations. There are several indexing approaches that have been explored by (Hua et al. 2023). One of them is called sequential indexing which we will explain briefly in a later section and the others in the appendix. In that regard, *we want to focus on the sequential indexing first*. An important point is that sequential indexing assigns the user-items's ID in an incremental way, except when the items are repeated. This incremental characteristic may share the same concept of numbers with the addition task. So, the effect of the concepts may be mutual.

With previous considerations, we have a **hypothesis** as follows:

Learning addition task will give better performance for language-based RS model with sequential indexing.

We then pose following **research questions**:

- *Is P5n better than vanilla P5 in performance?*
- *What aspects of T5n affect the performance of P5n?*
- *Can P5n retain its addition ability?*

Contributions. To the extent of our knowledge, this paper is the first work that assesses the effect of the prior distribution of numerical encoding in FMs for RS. Our findings are followings:

- Learning the addition task can improve the P5 sequential recommendation performance with sequential indexing. But, it depends on the structure of the prior numerical encoding, in which there may be an optimum structure. The training can also fail when using randomized T5n or a smaller batch size leading to zero performance of P5n, suggesting instability of training.
- The performance of P5n models on addition tasks has been nullified, suggesting they may use different concepts of numbers (*Figure 1b is wrong*).

Limitation. Our study in this paper used "T5-small" with 60M number of parameters and 5 rather small data sets. But, we will continue to extend our work in the future.

2 Problem Formulation

General notation. Let $\boldsymbol{x}_{1:T}$ be a sequence of words with length T. Let denote words $\langle$user_id$\rangle$,$\langle$item_id$\rangle$ as $x^{(u)}, x^{(v)}$, respectively. In a OpenP5's prompt template $x^{(u)}, x^{(v)} \in \boldsymbol{x}_{1:T}$. Let us refer to the user's ID as u and its history items' IDs as a sequence $\boldsymbol{v}_{1:V}$ with length V. Let denote $\mathcal{D}$ as a set, such that $\mathcal{D} = \{u, \boldsymbol{v}_{1:V}\}$. Let also $x^{(n)}$ denote an integer as word and $\mathcal{D}^{(n)}$ as its set. In the addition task's prompt template $\mathcal{D}^{(n)} \in \boldsymbol{x}_{1:T}$. Let $\mathcal{F}, \mathcal{A}, \mathcal{R}$ represent algorithms for language model, arithmetic, and recommender tasks, respectively. Finally, let $\mathcal{T}$ be a tokenizer splits a word x into subwords (tokens) such that $\mathcal{T}(\boldsymbol{x}_{1:T}) : \boldsymbol{x}_{1:T} \mapsto \boldsymbol{q}_{1:Q}, Q \geq T$.

OpenP5 and addition task formulation. Let f is a function parameterized with θ that takes input $\boldsymbol{q}_{1:Q}$. We train f for next-token prediction following this objective function

$$\min_{\theta} -\Sigma_t^Q \log\left(\boldsymbol{q}_t | \boldsymbol{q}_{1:t-1}, \theta\right) \tag{1}$$

This makes f as a language model and we train f to have an algorithm $\mathcal{F}$. When the OpenP5's prompt template is used, we also train f for $\mathcal{R}$ on $\mathcal{D}$. On the other hand, when we use the addition prompt's template, we also train f for $\mathcal{A}$ on $\mathcal{D}^{(n)}$. Hence, we can have a model for sequential recommender systems and addition tasks, respectively.

Prior's effect on P5n. Since we train P5n by using T5n as the base model, we have a prior of numerical encoding based on $\mathcal{A}$, instead of a random distribution $\mathcal{N}(\mu, \sigma)$. This way, we indirectly change the prior of numerical encoding because $\mathcal{T}(\mathcal{D}) \cap \mathcal{T}(\mathcal{D}^{(n)}) \neq \emptyset$. Hence, we can assess how the prior of numerical encoding affects the performance of sequential recommendation.

Sequential Indexing. P5 uses SentencePiece tokenization and this is cleverly exploited in the sequential indexing. Under SentencePiece, numbers are tokenized into separated numbers such as "1001" which becomes "100" and "1". First, the IDs are assigned by sorting the user and then starting indexing the items from 1001. After that, by iteration from the first item to the last item in a user, the index is added when an item is not repeated. By doing so, there is a co-occurrence of items such as "1001" and "1002", where "100" shares the embedding. Therefore, it can capture the clusters of co-occurrence items in a collaborative way.

3 Methodologies

3.1 Prompts and Base Models

Prompt templates. Since we use OpenP5 as our framework, we follow their prompt templates for a sequential recommendation task. There are 10 different templates for finetuning training and a template for zero-shot learning. The former is denoted as 'seen' while the latter is denoted as 'unseen'. We show these templates in the appendix. On the other hand, the addition task has a single template which is "What is $x_1^{(n)}$ plus $x_2^{(n)}$?" as the input and its answer "$x_3^{(n)}$" as the output. The number of digits for $x^{(n)}$ ranged from 1 to 9 digits.

Details on base models. We use several versions of base models that we denote as T5v, T5n-e, and T5n-r. In more detail, T5v is the vanilla version of the base model available from the HuggingFace repository (Wolf et al. 2020). T5n-e is a base model that we train on the *addition task first* for e epochs, such as T5n-10 and T5n-50. We prepare models with fewer epochs and larger epochs to see the effect of the quality of T5n on the performance of P5n. T5r is a base model that we train on the addition task *but the answers are random*, we train them for 10 epochs. For all the training, we re-initialize the numbers embedding to $\mathcal{N}(0, 1)$. It is also noteworthy that T5n-r can be used to assess whether the model needs to "understand" addition operation or just needs to see numbers in examples. For convenience, we use the same prefix of base models for P5, e.g. P5v and P5n-10.

3.2 Data Sets, Training, Metrics, and Analysis

Data sets. Since we utilize OpenP5, we use the provided data sets. We train the models on 5 rather small data sets. These are Beauty, Clothing, Taobao, ML100K, and LastFM. The details of these data sets can be seen in Table 3. We also follow the train and test data sets splitting provided by OpenP5. For the addition task, we generate 1 million examples as the training data set and 10000 as the test data set.

Training hyperparameters. For every run, we train each model into T5n, P5 or P5n with the same training hyperparameters. Except for T5-e, we train a model for 10 epochs. We use batch optimization using the AdamW algorithm, with a batch size of 128 and 32. We set the peak learning rate as 10^{-3} and use a learning scheduler with warmup on 100 first steps. To train all the models we use 2 GPUs (NVIDIA RTX 3090). We use W&B for training management (Biewald 2020).

Metrics. Following OpenP5, for the sequential recommendation task we use top-k Hit Ratio (hit@k) and Normalized Discounted Cumulative Gain (ndcg@k), we then set $k = \{5, 10\}$. For the addition task, we use the ratio between the exact match number in ground truth and prediction. We give this metric in percentage and denote it as (% Acc). For all the metrics, the *higher means better*.

Representational similarity. To analyze the representations induced by two models in the embedding space, we use the linear centered kernel alignment (CKA), described in (Kornblith et al. 2019), as a similarity score $s \in (0, 1)$. We explain the method in more detail in the appendix. We measure the pairwise of T5n models and since the T5v distribution comes from the random distribution we can also compare with it.

4 Experimental Results

Sequential recommendation. We present the results of the performance of P5 and P5n according to their base models in Table 1 for a batch size of 128. We show the results for a batch size of 32 in Table 5 that we put in the appendix due to space. Accordingly, on both batch sizes 128 and 32, P5n-5 and P5n-10 have the most occurrences of the best and the second-best scores. We found that P5n-50 and P5n-75 have better performance on less sparse data sets (ML100K and LastFM). Meanwhile, P5n-r is generally the worst for all the data sets. We also found that in a batch size of 32, the P5n-5 and P5n-r sometimes fail to learn (zero performance).

Addition task. We show the results of the evaluation of models on the addition task in Table 2 and Table 4. In particular, T5n-75 is the best followed by T5n-50, T5n-10, and T5n-5, while T5n-r and T5v have zero performance. On the other hand, the evaluation of P5n models trained shows that the efficacy of the addition task has been nullified.

Representation of numerical encoding. We show s between the pairwise number embeddings of T5n models for each batch size in Figure 2 as a heatmap. We found that the representation of numerical encoding of T5n-10 and T5n-5 are more similar with T5n-50 and T5n-75 than $\mathcal{N}(0, 1)$.

5 Discussion and Related Work

Based on the results, we found that the prior numerical encoding affects the performance of the RS tasks. In particular, when the training has not failed, *P5n-5 and P5n-10 have better overall performances compared with the vanilla one*. Meanwhile, P5n-50 and P5n-75 are better on less sparse data sets. If we compare their base model's performance on the addition task, actually T5n-50 and T5n-75 are the best. Because of that, we assume that their representation is more structured. But, it turns out that T5n-10 and T5n-5 give better performance. This highlights there may be an optimum prior that gives the balance between the continuous structure of indexing and the sparsity of the data sets. Moreover, this aligns with representation similarity analysis where the lower epochs T5n are in between $\mathcal{N}(0, 1)$ and upper epochs T5n. On the other hand, the performance of P5n-r is generally the worst. Its base model has seen numbers during

Table 1: Performance of P5 and P5n with different base models on datasets using a batch size of 128. Boldface for the best score and underline the second-best score over a metric and a data set.

Model →	**P5v** (Baseline)				**P5n-50**			
Dataset ↓ / Metric →	hit@5	hit@10	ndcg@5	ndcg@10	hit@5	hit@10	ndcg@5	ndcg@10
Beauty (seen)	0.0030	**0.0039**	0.0020	**0.0023**	0.0016	0.0038	0.0012	0.0019
Taobao (seen)	0.1837	**0.2192**	0.1406	**0.1521**	0.1725	0.2077	0.1315	0.1429
Clothing (seen)	0.0013	**0.0020**	0.0008	0.0010	0.0010	0.0015	0.0007	0.0009
ML100K (seen)	0.0700	0.1220	0.0430	0.0601	0.0806	0.1230	**0.0526**	**0.0664**
LastFM (seen)	0.0174	0.0339	0.0104	0.0156	**0.0248**	0.0330	**0.0158**	**0.0184**
Beauty (unseen)	0.0028	0.0039	0.0019	0.0022	0.0017	0.0036	0.0012	0.0019
Taobao (unseen)	**0.1843**	**0.2199**	**0.1412**	**0.1527**	0.1712	0.2089	0.1304	0.1426
Clothing (unseen)	0.0012	0.0019	0.0008	0.0010	0.0010	0.0015	0.0007	0.0008
ML100K (unseen)	0.0657	0.1262	0.0422	0.0620	**0.0848**	0.1262	**0.0540**	**0.0675**
LastFM (unseen)	0.0211	0.0339	0.0120	0.0160	**0.0229**	**0.0330**	**0.0146**	**0.0179**

Model →	**P5n-10**				**P5n-r**			
Dataset ↓ / Metric →	hit@5	hit@10	ndcg@5	ndcg@10	hit@5	hit@10	ndcg@5	ndcg@10
Beauty (seen)	**0.0031**	**0.0039**	**0.0021**	**0.0023**	0.0026	0.0031	0.0016	0.0017
Taobao (seen)	**0.1845**	0.2164	**0.1409**	0.1512	0.1540	0.1791	0.1206	0.1287
Clothing (seen)	**0.0014**	0.0019	**0.0009**	**0.0011**	0.0000	0.0000	0.0000	0.0000
ML100K (seen)	**0.0827**	**0.1315**	0.0497	0.0652	0.0138	0.0276	0.0104	0.0147
LastFM (seen)	0.0239	**0.0349**	0.0134	0.0169	0.0128	0.0193	0.0072	0.0092
Beauty (unseen)	**0.0032**	**0.0041**	**0.0022**	**0.0024**	0.0026	0.0031	0.0016	0.0017
Taobao (unseen)	0.1833	0.2154	0.1400	0.1505	0.1553	0.1796	0.1222	0.1301
Clothing (unseen)	**0.0013**	**0.0020**	**0.0009**	**0.0011**	0.0000	0.0000	0.0000	0.0000
ML100K (unseen)	0.0817	**0.1304**	0.0497	0.0652	0.0138	0.0308	0.0104	0.0156
LastFM (unseen)	0.0220	0.0312	0.0133	0.0163	0.0128	0.0202	0.0067	0.0091

Table 2: Performance of models on addition task. The P5n model results are from the average across data sets and with a batch size of 128.

Model	T5n-50	T5n-10	T5n-r	T5v
% Acc	72.4	43.8	0.0	0.0
Model	P5n-50	P5n-10	P5n-r	P5v
% Acc	$\sim 10^{-3}$	$\sim 10^{-3}$	0.0	0.0

training, but the answer is random. Hence, *it suggests that what matters here is the structure of the priors of numerical encoding*, but it needs to be precise. As also pointed out, the training can fail and this may be related to the training instability problem that is common in the Transformer-based model, as in (Liu et al. 2020). Finally, we also want to point out the poor results of P5, and P5n models on the addition task. This suggests that the assumed shared concept in Figure 1b *is wrong* and they may use different concepts of numbers. If it is so, then building an FM for RS with side information that contains numbers needs to consider this.

This study is also related to work from (Kocmi and Bojar 2017) that explored several initializations of word embeddings and found it could improve models' performance. We also think that the problem we study can be considered as a problem of "Elephant in the Room", described in (Boutilier, Mladenov, and Tennenholtz 2023). Suppose we have only a single FM for our RS in the ecosystem, then the embedding of $x^{(u)}$ and $x^{(v)}$ need to accommodate sparse and continuous representation. Moreover, the embedding of $x^{(n)}$ also needs to accommodate in case of number-related information is used. This may be accessible via a sophisticated loss function specialized for numerical encoding. Consequently, it incites us to delve further into this avenue in the future, as well as consider multiple agents with their own concept of numbers.

6 Concluding Remark and Future Study

In conclusion, we trained the base models of a foundation model for the recommender systems with an additional task first. We found that by doing so, the performances were affected where the prior of the number encoding is important. This gives an insight into the importance of numerical encoding in the foundation model of recommender systems as well as its relationship to other number-related tasks.

Lastly, we acknowledge that our study was still very limited. Hence, it is still not clear the effect of other aspects such as different indexing, other arithmetic tasks, other embedding strategies e.g. whole-word embedding, and other base models. Therefore, more works need to be done to study the role of numerical encoding in the FM for RS.

Acknowledgments

This work was supported by JST SPRING Grant Number JPMJSP2108 and JSPS KAKENHI Grant Numbers JP23H03408.

References

Biewald, L. 2020. Experiment Tracking with Weights and Biases. Software available from wandb.com.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Boutilier, C.; Mladenov, M.; and Tennenholtz, G. 2023. Modeling Recommender Ecosystems: Research Challenges at the Intersection of Mechanism Design, Reinforcement Learning and Generative Models. *arXiv preprint arXiv:2309.06375*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*.

Dziri, N.; Lu, X.; Sclar, M.; Li, X. L.; Jiang, L.; Lin, B. Y.; Welleck, S.; West, P.; Bhagavatula, C.; Bras, R. L.; Hwang, J. D.; Sanyal, S.; Ren, X.; Ettinger, A.; Harchaoui, Z.; and Choi, Y. 2023. Faith and Fate: Limits of Transformers on Compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; and Zhang, Y. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, 299–315.

He, Z.; Liu, W.; Guo, W.; Qin, J.; Zhang, Y.; Hu, Y.; and Tang, R. 2023. A Survey on User Behavior Modeling in Recommender Systems. In Elkind, E., ed., *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 6656–6664. International Joint Conferences on Artificial Intelligence Organization. Survey Track.

Hua, W.; Xu, S.; Ge, Y.; and Zhang, Y. 2023. How to Index Item IDs for Recommendation Foundation Models. *arXiv preprint arXiv:2305.06569*.

Kocmi, T.; and Bojar, O. 2017. An Exploration of Word Embedding Initialization in Deep-Learning Tasks. In *Proceedings of the 14th International Conference on Natural Language Processing (ICON-2017)*, 56–64.

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, 3519–3529. PMLR.

Liu, L.; Liu, X.; Gao, J.; Chen, W.; and Han, J. 2020. Understanding the difficulty of training transformers. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, 5747–5763. Association for Computational Linguistics (ACL).

Nogueira, R.; Jiang, Z.; and Lin, J. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1): 5485–5551.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 38–45.

Xu, S.; Hua, W.; and Zhang, Y. 2023. OpenP5: Benchmarking Foundation Models for Recommendation. *arXiv preprint arXiv:2306.11134*.

A Appendix

A.1 Indexing in P5

There are several indexing approaches for the foundation model of recommender systems, which we will briefly explain here based on (Hua et al. 2023). The important thing here is to balance between capturing co-occurrence and not creating any spurious relationship among user items.

- Random indexing: In random indexing, the user-item IDs are randomly selected from the number random generator. Since it is random, it is really hard to capture co-occurrence as well as can induce spurious relationships.
- Title indexing: It uses the title of the items such as in the movie. The same problem may happen here where as in the random indexing where similar titles might induce spurious relationships.
- Independent indexing: It creates independent out-of-vocabulary tokens for each of the items. This can solve the previously existing problems, but it is hard to capture collaborative representation. This is because each item is independent now.
- Collaborative indexing: In collaborative indexing, user items are clustered based on a graph or tree method. An example is to build a graph where the node is the item and its edge is based on the strength of its co-occurrence.
- Semantic indexing: This indexing utilizes the metadata of the items to assign the IDs. It also uses a tree-based method so it can capture the co-occurrence.

A.2 Prompt Templates

The prompt templates that are used for fine tuning (seen) and zero-shot learning (unseen) for sequential recommendation task are listed below which follows from OpenP5.

- (seen) Considering S $x^{(u)}$ has interacted with S items $\{x^{(v)}\}$. What is the next recommendation for the user ?; S $x^{(v)}$
- (seen) Here is the purchase history of S $x^{(u)}$: S item $\{x^{(v)}\}$. I wonder what is the next recommended item for the user .; S $x^{(v)}$
- (seen) S $x^{(u)}$ has purchased S items $\{x^{(v)}\}$, predict next possible item to be bought by the user ?; S $x^{(v)}$
- (seen) I find the purchase list of S $x^{(u)}$: S items $\{x^{(v)}\}$, I wonder what other items does the user need . Can you help me decide ?; S $x^{(v)}$
- (seen) According to what items S $x^{(u)}$ has purchased : S items $\{x^{(v)}\}$, Can you recommend another item to the user ?; S $x^{(v)}$
- (seen) What would S $x^{(u)}$ be likely to purchase next after buying S items $\{x^{(v)}\}$?; S $x^{(v)}$
- (seen) By analyzing the S $x^{(u)}$'s purchase of S items $\{x^{(v)}\}$, what is the next item expected to be bought ?; S $x^{(v)}$
- (seen) Can you recommend the next item for S $x^{(u)}$, given the user 's purchase of S items $\{x^{(v)}\}$?; S $x^{(v)}$
- (seen) After buying S items $\{x^{(v)}\}$, what is the next item that could be recommended for S $x^{(u)}$?; S $x^{(v)}$
- (seen) The S $x^{(u)}$ has bought items : S items $\{x^{(v)}\}$, What else do you think is necessary for the user ?; S $x^{(v)}$
- (unseen) What is the top recommended item for S $x^{(u)}$ who interacted with S item $\{x^{(v)}\}$?; S $x^{(v)}$

where $S = \{$Beauty, Clothing, Taobao, ML100K, LastFM$\}$ is the name of data set and $\{x^v\}$ denote the user's history as a sequence of x^v for $\boldsymbol{v}_{1:V}$ given u.

A.3 Statistics of Data Sets

We show the statistics of the data sets that we use in this study in Table 3 based on OpenP5.

Table 3: Statistics of the data sets from (Xu, Hua, and Zhang 2023)

Data set	Beauty	Clothing	Taobao
# Users	22363	39387	6104
# Items	12101	23033	4192
# Interactions	198502	278677	46337
Sparsity (%)	99.93	99.97	99.82
Data set	ML100K	LastFM	
# Users	943	1090	
# Items	1349	3646	
# Interactions	99287	52551	
Sparsity (%)	92.20	98.68	

A.4 Performance on a batch size of 32

The evaluation of base models on addition tasks trained with a batch size of 32 can be seen in Table 4. Meanwhile, the results for P5 and P5n for the batch size of 32 are shown in Table 5.

Table 4: Performance of models on addition task. The P5 model results are from the average across data sets and with a batch size of 32.

Model	T5n-75	T5n-5	T5n-r	T5v
% Acc	92.4	9.7	0.0	0.0
Model	P5n-50	P5n-10	P5n-r	P5v
% Acc	$\sim 10^{-3}$	$\sim 10^{-3}$	0.0	0.0

A.5 Representation Similarity Analysis

Consider an embedding of an input as $\boldsymbol{z} \in \mathbb{R}^d$ where d is the dimension of the embedding space. Then, for N samples we stack them row-wise to get the embedding matrix $\boldsymbol{Z} \in \mathbb{R}^{N \times d}$. From this, we can create a kernel matrix that measures the similarity pairwise samples as $\boldsymbol{K} \in \mathbb{R}^{N \times N}$. We then choose $\boldsymbol{K} = \boldsymbol{Z}\boldsymbol{Z}^T$ which is called a linear kernel matrix. Let $s(.,.)$ measure the similarity between $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$, two different kernel matrices. Then, the linear centered kernel alignment (CKA) as s can be written as

$$s(\boldsymbol{K}_1, \boldsymbol{K}_2) = \frac{\|\tilde{\boldsymbol{Z}}_1^T \tilde{\boldsymbol{Z}}_2\|_F^2}{\|\tilde{\boldsymbol{Z}}_1^T \tilde{\boldsymbol{Z}}_1\|_F \|\tilde{\boldsymbol{Z}}_2^T \tilde{\boldsymbol{Z}}_2\|_F}, \tag{2}$$

where $\|.\|_F$ refer to Frobenius matrix norm, and the tilde symbol means it has been centered by applying centering matrix $\boldsymbol{I} - \frac{1}{N}\boldsymbol{1}\boldsymbol{1}^T$. $s = 1$ means that the representations are the same.

We present the heatmap of the pairwise between T5n models and T5v from each batch size in Figure 2. For each plot, the representation similarity is between the T5n-50 and T5n-75 and $\mathcal{N}(0, 1)$. In which, the T5n-10 and T5n-5 are more similar to the upper epochs T5n, suggesting that their structure of representations is away from random.

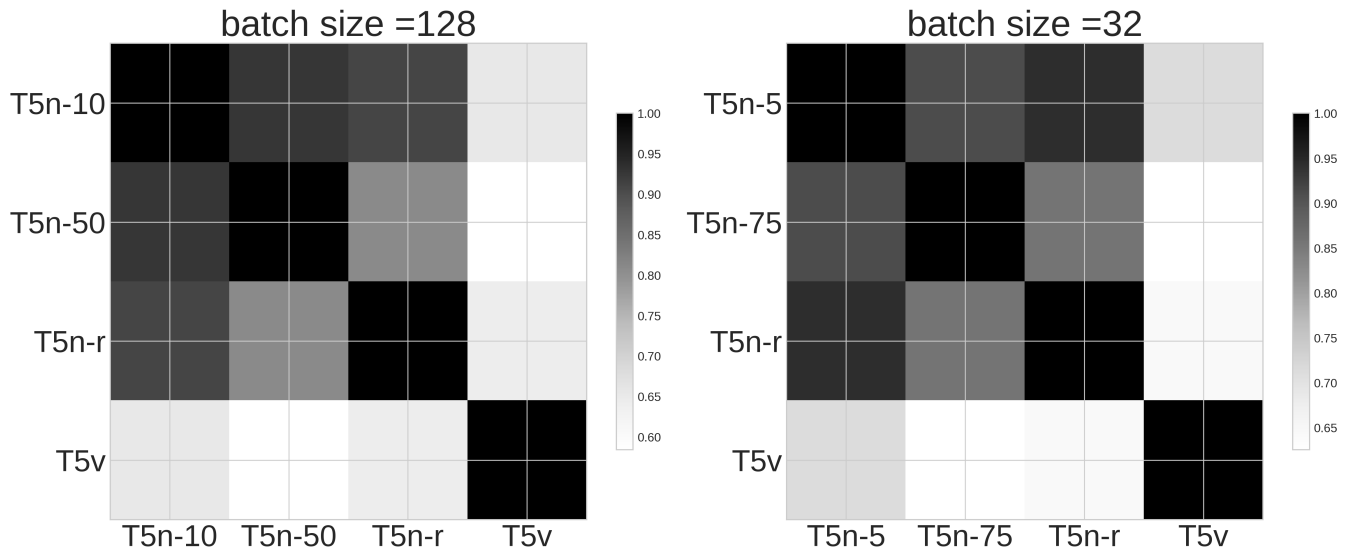


Figure 2: The heatmap plots of the linear CKA of pairwise T5n models, and T5v which its distribution comes from $\mathcal{N}(0, 1)$. The batch size of the training is shown on top of each plot. Compared fewer epochs T5n with larger epochs T5n and $\mathcal{N}(0, 1)$ (first column, the second and fourth row of each plot), the fewer epochs T5n is more similar to the larger epochs T5n than to $\mathcal{N}(0, 1)$ indicates by darker color on second row.

Table 5: Performance of P5 and P5n with different base models on datasets using a batch size of 32. Boldface for the best score and underline the second-best score over a metric and a data set.

Model →	**P5v** (Baseline)				**P5n-75**			
Dataset ↓ / Metric →	hit@5	hit@10	ndcg@5	ndcg@10	hit@5	hit@10	ndcg@5	ndcg@10
Beauty (seen)	0.0033	0.0043	0.0023	0.0027	0.0033	0.0044	0.0026	0.0030
Taobao (seen)	0.1669	0.2048	0.1288	0.1410	0.1635	0.1984	0.1293	0.1407
Clothing (seen)	0.0012	0.0016	0.0007	0.0008	0.0009	0.0014	0.0006	0.0008
ML100K (seen)	0.0530	0.0965	0.0337	0.0477	**0.0583**	**0.1060**	**0.0361**	**0.0513**
LastFM (seen)	0.0128	0.0266	0.0091	0.0135	0.0202	0.0339	**0.0139**	**0.0182**
Beauty (unseen)	0.0034	0.0043	0.0024	0.0027	0.0032	0.0044	0.0027	0.0031
Taobao (unseen)	0.1669	0.2056	0.1290	0.1415	0.1638	0.1979	0.1292	0.1403
Clothing (unseen)	0.0011	0.0015	0.0007	0.0008	0.0008	0.0015	0.0006	0.0008
ML100K (unseen)	0.0498	0.1007	0.0332	0.0495	**0.0573**	**0.1103**	**0.0352**	**0.0522**
LastFM (unseen)	0.0156	0.0248	0.0100	0.0130	**0.0239**	**0.0358**	**0.0160**	**0.0199**

Model →	**P5n-5**				**P5n-r**			
Dataset ↓ / Metric →	hit@5	hit@10	ndcg@5	ndcg@10	hit@5	hit@10	ndcg@5	ndcg@10
Beauty (seen)	**0.0050**	**0.0075**	**0.0032**	**0.0040**	0.0029	0.0039	0.0019	0.0022
Taobao (seen)	0.1709	0.2056	**0.1343**	**0.1455**	**0.1714**	**0.2069**	0.1319	0.1434
Clothing (seen)	**0.0013**	**0.0018**	**0.0009**	**0.0010**	0.0001	0.0002	0.0001	0.0001
ML100K (seen)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
LastFM (seen)	**0.0220**	**0.0349**	0.0138	0.0179	0.0193	0.0321	0.0121	0.0164
Beauty (unseen)	**0.0047**	**0.0070**	**0.0030**	**0.0038**	0.0030	0.0037	0.0020	0.0022
Taobao (unseen)	0.1694	0.2040	**0.1341**	**0.1454**	**0.1712**	**0.2054**	0.1313	0.1423
Clothing (unseen)	**0.0013**	**0.0018**	**0.0009**	**0.0010**	0.0001	0.0002	0.0001	0.0001
ML100K (unseen)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
LastFM (unseen)	0.0193	0.0284	0.0115	0.0145	0.0202	0.0330	0.0135	0.0177