NEURAL RE-RANKING FOR PERSONALIZED HOME SEARCH

Giacomo Bugli
Luigi Noto
Guilherme Albertini
Center for Data Science
New York University
{gb2572,ln2205,ga947}@nyu.edu

Abstract

Search Systems generate search results via multiple stages of incremental modeling complexity, where the set of candidate items gets progressively narrowed down and ranked based on some notion of relevance. The last step of this process, the re-ranking layer, is then aimed at refining the ranking produced in the previous steps by modeling the cross-item interactions as well as users' preference behaviors. In this project we partnered with Zillow Group to improve their search feature by implementing context-aware personalized re-ranking for home search. We adopt a self-attention based neural architecture to directly model the mutual influences between all items in the previously ranked list. The experimental results on the dataset provided by Zillow demonstrate significant performance gains of the proposed model with respect to the considered baselines. Finally, we prove that personalization has a positive impact on the re-ranking problem by conducting a feature ablation study.

1 Introduction

Search Systems aim at providing an ordered list of several search results from an exhaustive set of documents that are most pertinent to a user. To overcome the computational limitations that arise when considering billions of items and users, search systems are usually composed of multiple stages where, at every step, the number of relevant items is progressively narrowed down to maintain a system's low response latency. Following a search query from the user, a candidate set of items is generated from a large set of documents (L_0 & L_1 Layers), and a ranking model (L_2 Layer) is applied to display the most relevant documents to the user. In practice, users often compare multiple items on a result page before generating a click action; thus, information from other items in the same ranked list could affect a user's decision on the current item of interest. This usually results in a sub-optimal rank obtained in L_2 due to an inability to model contextual factors, such as users' intents or mutual influences between items in the list. Accordingly, an additional component, the re-ranking layer (L_3 Layer), is needed to capture these relationships. The re-ranking model should take the ranking list produced in the previous stage as input and output a new, re-ordered list that considers cross-item interactions as well as users' preferences: personalization becomes a desideratum for re-ranking. For instance, when a user is sensitive to price one would expect that similar items with different prices should be more aggregated in the list, whereas for users with no obvious purchasing intention items in the re-ranked list should be more diverse.

Through our partnership with Zillow, our project demonstrated the re-ranking of an initially ranked list by utilizing the user's preference behavior and cross-item interactions. To do so, we adopt a slight adaptation of the PRM (Personalized Re-ranking Model) proposed by Pei et al. [2019]. The model is based on the Transformer self-attention mechanism [Vaswani et al., 2017], and is equipped with a

personalization module to represent user's preference and intent on item interactions. Moreover, the self-attention mechanism is used to capture user-specific mutual influences between any two items without degradation over the encoding distance.

We conducted experiments on the dataset provided by Zillow in different training settings, namely binary and multi-level relevance as target labels. We show that PRM significantly outperforms the proposed baselines, including the traditional Learning-To-Rank (LTR) model LambdaMART [Burges, 2010], thereby improving on the initial ranking generated by Zillow. Moreover, the experimental results obtained in the feature ablation study demonstrate the effectiveness of introducing user personalization for the problem of re-ranking.

2 Related Work

Re-ranking constructs a scoring function by encoding item cross-interactions into feature space in an attempt to refine an initially ranked list generated by the base-ranker. Accordingly, many state-of-the-art methods for re-ranking directly consider an initial list as input and model the complex dependencies between items in distinct ways. For RNN-based approaches, such as DLCM [Ai et al., 2018] and Global Rerank [Zhuang et al., 2018], the initial list is fed into the RNN-based structure sequentially, outputting the encoded vector at each time step. In the case of DLCM, a unidirectional GRU is used to encode the information of the whole list followed by a single decoding step with attention, while for GlobalRerank an LSTM is used to not only encode information into item representation but also generate the ranked list by a decoder. The problem with these approaches hinges on the limited ability to model the interactions between items in the list, since the feature information of the previous encoded item degrades along with the encoding distance. Following the encoder-decoder structure, Bello et al. [2018] proposed Seq2Slate, a slate optimization framework that uses pointer networks [Vinyals et al., 2015] to directly predict the ranking of a list of documents by jointly considering their features together. The use of a sequential decoder ingesting the future state of the item selected in the current state inhibits the use of parallelization, a construction undesirable to any online ranking system. To facilitate parallelization, Ai et al. [2019] propose Groupwise Scoring Functions (GSFs): a multivariate scoring framework based on deep neural networks in which the relevance score of a document is determined jointly by multiple documents in the list. Unfortunately, high computational costs are incurred from considering all the unobserved counterfactual permutations that have not actually been displayed to the user when modeling the list-wise context under different permutations. With these factors in mind, an approach that is most similar to the one we selected is SetRank by Pang et al. [2019], which applies a variant of self-attention structure without positional encoding and dropout to obtain a permutation invariant property. However, we consider permutation invariance to be an undesirable property in re-ranking problems. In the L_3 layer, we want to make use of the ranking information produced in the previous layer; moreover, we seek to capture the fact that users' comparison behavior changes based on the items placed next to the current item of interest.

3 Problem Definition and Architecture

3.1 Task

The goal of re-ranking is to construct a multivariate scoring function, taking as input a list of items from the initial ranking, to model the cross-item interactions. For a specific user search query $q \in \mathcal{Q}$, given the initial list of n items $\mathcal{S}_q = [i_1, \ldots, i_n]$ produced in the L_2 layer, and the corresponding relevance labels $\mathbf{y} \in \mathbb{R}^n$, the re-ranking problem is to find the optimal ranking function ϕ^* assigning to each item in the input list a re-ranking score as

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \sum_{q \in \mathcal{Q}} \mathcal{L}(\mathbf{y}, \phi(\mathbf{X}, \mathcal{S}_q))$$

where $\mathcal{L}$ is the loss function, $\mathbf{X}$ is the feature matrix of all items in the list, and $\mathcal{S}_q$ is the initially ranked list. In most re-ranking approaches, only the item features are used to learn the scoring function; however, they don't allow modeling for user preferences. For our project, given the focus on personalization, we consider a personalized matrix $\mathbf{PM}$ in conjunction with the item features, where, for each user, the user-item interactions are encoded by utilizing the user's search history. In

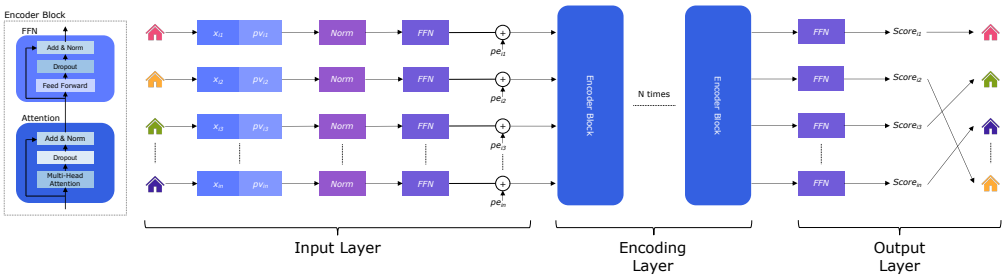

Figure 1: Detailed structure of PRM model and its sub-modules. *Norm* stands for Normalization layer, *FFN* for Feed Forward Network, and pe_i for learnable positional encoding.

this case, the problem can be rewritten as

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \sum_{q \in \mathcal{Q}} \mathcal{L}(\mathbf{y}, \phi(\mathbf{X}, \mathbf{PM}, \mathcal{S}_q))$$

where $\mathcal{L}$ is the loss function, $\mathbf{X}$ is the feature matrix of all items in the list, $\mathbf{PM}$ is the personalized matrix, and $\mathcal{S}_q$ is the initially ranked list.

3.2 Model Architecture

In this section, we describe the architecture of the adopted self-attention based re-ranking model, which is a slight adaptation of the PRM model introduced by Pei et al. [2019]. The model can be split into three components: the *input layer*, the *encoding layer* and the *output layer*. Figure 1 shows the detailed model structure. Each component is described separately in the following subsections.

3.2.1 Input layer

The input layer is supposed to produce embeddings for each item in the initial list $S_q = [i_1, \ldots, i_n]$ before passing them to the encoding layer, starting from the raw item feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{item}}}$ and the user-specific personalized matrix $\mathbf{PM} \in \mathbb{R}^{n \times d_{\text{pv}}}$. The first step consists in concatenating $\mathbf{X}$ and $\mathbf{PM}$ to produce an intermediate embedding matrix $\mathbf{U} \in \mathbb{R}^{n \times (d_{\text{item}} + d_{\text{pv}})}$

$$\mathbf{U} = [\mathbf{X}, \mathbf{PM}] = \begin{bmatrix} \mathbf{x}_{i_1} & \mathbf{pv}_{i_1} \\ & \vdots \\ \mathbf{x}_{i_n} & \mathbf{pv}_{i_n} \end{bmatrix}$$

Next, the row vectors of matrix $\mathbf{U}$ are normalized using layer normalization [Ba et al., 2016], obtaining another intermediate embedding matrix $\mathbf{U}' \in \mathbb{R}^{n \times (d_{\text{item}} + d_{\text{pv}})}$. The hidden representations grouped in matrix $\mathbf{U}'$ are then passed through a shared multi-layer feed-forward network of input size $(d_{\text{item}} + d_{\text{pv}})$ and output size d_{enc}, where d_{enc} represents the input dimension of the encoding layer, producing a new matrix $\mathbf{U}'' \in \mathbb{R}^{n \times d_{\text{enc}}}$. As previously mentioned, in the re-ranking problem we assume that the input list of items is the result of a previous ranking layer. By itself, self-attention based encoder is permutation-equivariant, and therefore does not discern the order of input items. In order to use such sequential information, we add learnable positional encodings $\mathbf{PE} \in \mathbb{R}^{n \times d_{\text{enc}}}$ to matrix $\mathbf{U}''$

$$\mathbf{E} = \mathbf{U}'' + \mathbf{PE}$$

The matrix $\mathbf{E} \in \mathbb{R}^{n \times d_{\text{enc}}}$ is the final embedding matrix which is passed to the encoding layer.

3.2.2 Encoding layer

The encoding layer is the main component of the model, aimed at modeling the mutual influences between items in the input list. It is based on the Transformer self-attention mechanism, which fits well with the re-ranking problem since it is able model the interaction between each pair of items regardless of the distance between them.

Self-attention mechanism The goal of the attention mechanism is to compare each item in a list (*queries*) to a collection of other items (*keys*) and capture their relevance in the current context (*values*). The output of the attention mechanism for a given query is a weighted sum of the value vectors, where the weights represent how relevant to the query is the key of the corresponding value vector. We use the so-called *scaled dot-product attention*, defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$$

where matrices $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ represent queries, keys and values, respectively, and d is the query vector dimension. Self-attention consists in the attention mechanism where queries, keys and values are all projected from the same item embedding matrix. In order to model more complex mutual influences, it is useful to perform the self-attention operation multiple times and combine the outputs. The outputs are first concatenated and then linearly projected to a lower-dimensional space with respect to the dimension of the concatenated vector. The entire procedure is called *multi-head self-attention* and is described by the following equations

$$\begin{aligned}\text{MultiHead}(\mathbf{M}) &= \text{Concat}(\text{head}_1, \dots, \text{head}_h)\mathbf{W}^O \\ \text{head}_i &= \text{Attention}(\mathbf{M}\mathbf{W}^Q, \mathbf{M}\mathbf{W}^K, \mathbf{M}\mathbf{W}^V)\end{aligned}$$

where $\mathbf{M} \in \mathbb{R}^{n \times d_{\text{enc}}}$ is the item embedding matrix passed to the encoder block, $\mathbf{W}^Q \in \mathbb{R}^{d_{\text{enc}} \times d_q}$, $\mathbf{W}^K \in \mathbb{R}^{d_{\text{enc}} \times d_k}$, $\mathbf{W}^V \in \mathbb{R}^{d_{\text{enc}} \times d_v}$ are the query, key and value projection matrices, respectively, and $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{\text{enc}}}$ is the output projection matrix.

Layer structure The embedding matrix $\mathbf{E} \in \mathbb{R}^{n \times d_{\text{enc}}}$ produced by the input layer is passed through N stacked encoder blocks, where the input and output dimensions of each block are matched. The first block gets $\mathbf{E}$ as input and each subsequent block gets as input the output of the previous one. Each encoder block consists in a multi-head self-attention layer with h attention heads and a position-wise feed-forward network, each followed by residual connections [He et al., 2016] and layer normalization, applying dropout before performing summation in residual blocks. We let $\mathbf{Z}^{(N)} \in \mathbb{R}^{n \times d_{\text{enc}}}$ be the output of the last encoder block which is passed to the output layer.

3.2.3 Output Layer

The goal of the output layer is to compute a score vector $\mathbf{s} \in \mathbb{R}^n$, where s_j represents the re-ranking score of item i_j, starting from the encoding layer output $\mathbf{Z}^{(N)}$. To do so, $\mathbf{Z}^{(N)}$ is passed through a fully-connected layer shared across all items, i.e.

$$\mathbf{s} = \mathbf{Z}^{(N)}\mathbf{w} + \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix}$$

where $\mathbf{w} \in \mathbb{R}^{d_{\text{enc}} \times 1}$ and $b \in \mathbb{R}$ are the (shared) projection matrix and bias, respectively. The model is trained in an end-to-end manner by optimizing the ListNet loss [Cao et al., 2007] defined as

$$\mathcal{L}(\mathbf{s}, \mathbf{y}) = -\sum_j \text{softmax}(\mathbf{y})_j \cdot \log(\text{softmax}(\mathbf{s})_j)$$

where $\mathbf{s}$ are the computed scores and $\mathbf{y}$ are the ground-truth relevance labels.

4 Experimental Evaluation

In this section we first introduce the dataset used and the baselines considered for the evaluation. We then compare the performances of the adopted PRM model with respect to the baselines and the initial ranking generated by Zillow. Finally, to quantify the importance of user personalization, we conduct a feature ablation study.

4.1 Data

The experiments have been conducted on the datasets provided by Zillow Group. The datasets are divided into: *users search sessions* dataset, *user item features* dataset, and *item features* dataset. The *users search sessions* dataset contains user search sessions over a day's span. For each session, we are provided with a list of items in the order they have been shown to the user as well as the binary user's interactions with such items: $y_i^{\text{click}}, y_i^{\text{favorite}}, y_i^{\text{submit}}$. The *user item features* dataset contains the user-item interaction features ($\mathbf{PM}$), i.e. learned representation of user historic behavior for each (user, item) pair, while the *item features* dataset contains the item-specific features ($\mathbf{X}$) for each item.

First, we discard the features in *item features* dataset for which we observe the proportion of null values to exceed 20%. For the remaining nulls, we impute them by considering the mode when the feature class is categorical or the median when the feature class is numerical. Once we made sure that all datasets didn't present any null values or further irregularities, joined the *users search sessions* dataset with the other two datasets based on unique identifiers, obtaining a single dataset. In the resulting dataset, each row represents a search query for which we have the list of items in the order that they have been shown to the user, and for each item we have the item-specific features, the user-item interaction features, and the corresponding interaction labels. As for dataset statistics, the obtained dataset contains 1,265,042 unique search sessions with 40 as the maximum length of the item list (i.e., a maximum of 40 items have been shown to the user following a search query). For each item, we have 8 user-item interaction features and 59 item-specific features; as these were anonymized, we were not informed of their actual meaning. Among all sessions, we observe that: 1,121,762 have at least one click interaction, 25,704 have at least one favorite interaction, and 2,472 have at least one submit interaction.

It is important to highlight that we kept the provided timestamp for each search session, as we used this for the dataset when splitting into train, validation, and test portions. We thus get 60% of the least recent search queries in the train set, the following 30% in the validation set, and the most recent 10% in the test set.

4.2 Methodology

4.2.1 Baselines

To compare the performances of PRM, three baselines have been selected:

- **No Re-rank** Refers to the initially ranked list obtained by Zillow's preexisting ranking model. We use this to quantify the impact of the additional L_3 layer.
- **Popularity** Refers to the re-ranking obtained by sorting the items based on the click-through rate, where the click-through rate for each item is defined as the number of times such item has been clicked over the total number of times it has been shown across sessions. This allows to have an easy rule-based method to compare to more complex ones.
- **LambdaMART** Traditional LTR method proposed by Burges [2010], an ensemble model that is built on the MART (Multiple Additive Regression Trees) structure coupled with the concept of swap values called lambdas. LambdaMART is a widely used method in commercial search engines.

4.2.2 Evaluation Metrics

To evaluate the performances of the models produced during our work, we used the following metrics:

Normalized Discounted Cumulative Gain (NDCG@k) The Discounted Cumulative Gain (DCG) is a measure of the usefulness, or gain, of a document based on its position in the ranked list. The gain is then discounted based on the rank of the item in the list. The DCG accumulated at a rank position k is then defined as

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}$$

where rel_i is the relevance of item in position i of the list. In order to make this measure invariant to the length of the ranked list, the DCG is *normalized* by the Ideal Discounted Cumulative Gain

Model	lr	n_estimators	max_depth	N	batch_size
LambdaMART	1e-4, 1e-2, <u>0.1</u>	<u>200</u>, 300	<u>8</u>, 10	-	-
PRM (Binary)	-	-	-	4, <u>6</u>, 8	<u>128</u>, 256

Table 1: Search range per hyper-parameter. For LambdaMART, we consider the learning rate (lr), the number of boosted trees (n_estimators), and the maximum depth of each tree (max_depth). For PRM, we consider the number of encoder blocks (N) and the size of the training batch (batch_size). The best value is underlined.

(IDCG), which is the maximum possible DCG obtained by sorting the list of documents by their relevance and computing the DCG. The NDCG formula at a given position k is then given by

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}$$

Mean Average Precision (MAP@k) The Precision@k is defined as the fraction of relevant documents in the first k positions of the ranked list. Since Precision is invariant to the order of the items in the list, we define Average Precision (AP) to weight the items based on their rank. The AP at a rank position k is then given by

$$\text{AP@k} = \frac{\sum_{i=1}^{k} \text{Precision@}i \cdot \text{rel}_i}{\sum_{i=1}^{k} \text{rel}_i}$$

where rel_i is a binary relevance. Then the MAP@k for a set of queries $\mathcal{Q}$ is the average of the AP@k over all queries

$$\text{MAP@k} = \frac{\sum_{q=1}^{|\mathcal{Q}|} \text{AP}_q\text{@k}}{|\mathcal{Q}|}$$

4.2.3 Experimental Setup

Regarding the target labels, we considered two different training settings for PRM. With PRM (Binary) we refer to the model trained on a binary ground-truth relevance based on the click interactions: $\text{rel}_i = y_i^{\text{click}}$. Conversely, with PRM (Multi) we refer to the model trained on a multi-level ground-truth relevance constructed as follows: $\text{rel}_i = y_i^{\text{click}} + 3y_i^{\text{favorite}} + 5y_i^{\text{submit}}$.

For LambdaMART and PRM (Binary), we performed cross-validation to determine the best parameter configurations. For PRM (Multi), we used the parameters obtained by tuning PRM (Binary). See Table 1 for the considered grid. For LambdaMART, the parameters not considered in the grid have been left to the default values set by the XGBoost implementation of SparkXGBRanker (inheriting them from the XGBRanker)[1]. In all experiments, each PRM encoder block has been configured to have 8 attention heads (h), 1024 hidden units, and 0.1 as p_{dropout}. In the input layer, the hidden sizes of the Feed-Forward-Network have been set to [512, 1024, 512]. The starting learning rate of the Adam optimizer is set to 3e-5, with a scheduler halving this every 25 epochs. The PRM max number of epochs is 250, with an early-stopping patience of 15 epochs.

Data preprocessing and all baselines have been implemented using Spark with a custom singularity instance on the NYU Greene cluster. Concerning PRM, we followed the PyTorch code-base provided by Pobrotyn et al. [2020] for multi-GPU training.

4.3 Results

The evaluation results are shown in Table 2, Table 3, and Table 4. We compare the performances of PRM (Binary) and PRM (Multi) against the baselines, on the three different binary interaction labels (clicks, favorites, submits). We can see how, for all target labels, PRM shows improvements with respect to the selected baselines in both binary and multi-level relevance training settings. In particular, from Table 2 it is possible to notice how we have been able to obtain a lift of 2% on the

[1] https://xgboost.readthedocs.io/en/latest/python/python_api.html#xgboost.XGBRanker

Model	NDCG@				MAP@			
	5	10	15	20	5	10	15	20
No Re-rank	34.18	39.82	43.24	45.35	27.79	30.57	32.08	32.92
Popularity	23.73	29.95	33.77	36.44	17.91	20.75	22.32	23.31
LambdaMART	28.61	34.96	38.69	41.11	22.54	25.62	27.25	28.20
PRM (Binary)	**35.78**	**41.79**	**45.13**	**47.21**	**38.31**	**38.22**	**37.47**	**36.89**
PRM (Multi)	35.61	41.62	44.96	47.02	38.11	38.03	37.30	36.71

Table 2: Ranking metrics computed on the test set for the click labels. All values have been multiplied by 100.

Model	NDCG@				MAP@			
	5	10	15	20	5	10	15	20
No Re-rank	27.35	33.71	37.52	39.93	25.80	27.75	28.29	28.50
Popularity	18.23	21.44	26.71	30.19	15.09	17.77	19.02	19.89
LambdaMART	21.56	24.56	29.12	31.49	19.66	21.80	22.37	23.04
PRM (Binary)	29.02	36.25	39.89	42.38	27.74	29.67	30.18	30.12
PRM (Multi)	**29.37**	**36.57**	**40.26**	**42.73**	**27.96**	**29.90**	**30.21**	**30.34**

Table 3: Ranking metrics computed on the test set for the favorite labels. Only the search sessions with at least one favorite interaction have been considered. All values have been multiplied by 100.

NDCG@10 (our target metric) with respect to Zillow initial ranking model (No Re-rank case). The significance of such improvement is supported by the fact that both Popularity and LambdaMART obtain substantially lower performances with respect to the No Re-rank case, meaning that Zillow ranking model is already very strong. Tables 3 and 4 demonstrate how PRM is able to obtain better performances also for favorite and submit interactions. See Figure 2 and 3 for a visual representation of the effect of PRM on the initially ranked list. In both cases, it is clear how after re-ranking the items in the list based on the score produced by PRM, the most relevant items get positioned at the top of the list.

4.4 Discussion

Comparing the performances of PRM against the selected baselines, the effectiveness of re-ranking is unequivocal. Both PRM (Binary) and PRM (Multi) outperform the initial ranker by a considerable margin for all types of user interaction labels. This shows how the list-wise context and self-attention mechanism actively encode the user-specific mutual influences between items, generating more relevant search suggestions. Analyzing the results of PRM on both binary and multi-level training settings, from Tables 2 and 4 we notice that PRM (Binary) is the best-performing model when we evaluate on click and submit interaction labels, while from Table 3 we notice how PRM (Multi) is the best-performing model when we conduct the evaluation on favorite ones. The reason for this might be the increased sparsity of the labels or the fact that we used the best parameter configuration found in the PRM (Binary) cross-validation for PRM (Multi). Some possible remedies could then be varying the weights for each type of interaction defined in the multi-level relevance, assigning more importance to "stronger" signals, and conducting a thorough hyperparameter exploration for PRM (Multi). Additionally, increasing the amount of training data should help. Due to time constraints, we leave these further experiments as a possible future direction. Finally, it is worth noticing how the self-attention method takes into consideration the fact that the input list should already be reasonably ranked. We can see this in Figure 4, where the highest weights get assigned to the items placed on the top of the input list, and they get smaller as we descend the list. This indicates that the information passed from the L_2 layer both gets considered by the model in computing the re-ranking scores and doesn't get lost.

Model	NDCG@				MAP@			
	5	10	15	20	5	10	15	20
No Re-rank	31.36	36.75	40.50	41.73	28.26	29.88	31.00	31.29
Popularity	22.29	27.64	30.06	32.31	17.24	21.19	22.68	23.44
LambdaMART	25.38	29.91	33.07	34.93	21.89	24.11	25.75	26.97
PRM (Binary)	**39.89**	**45.90**	**48.51**	**50.38**	**36.72**	**38.75**	**39.26**	**39.39**
PRM (Multi)	39.00	44.89	47.42	49.86	35.47	37.35	37.96	38.38

Table 4: Ranking metrics computed on the test set for the submit labels. Only the search sessions with at least one submit interaction have been considered. All values have been multiplied by 100.

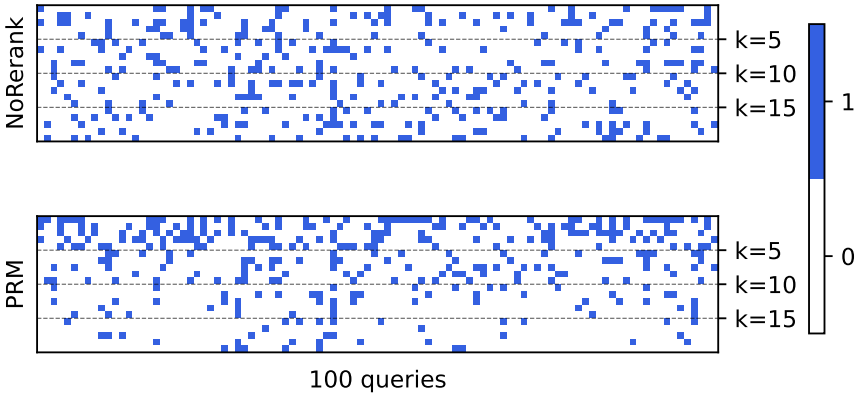

Figure 2: Visualization of PRM (Binary) re-ranking results against the No Re-rank baseline for the top-20 items on 100 randomly sampled test queries. The color encodes the item's ground-truth binary relevance based on clicks.

4.5 Feature Ablation

As we believe in the importance of personalization, we conducted a feature ablation study to quantify the impact of modeling users' preference behaviour. We investigated whether, and to what degree, it is beneficial to include both the user-specific personalization matrix $\mathbf{PM}$ and raw item feature matrix $\mathbf{X}$ in the input layer, compared to only considering the raw feature matrix $\mathbf{X}$ as input. With PRM, we refer to the model in which both $\mathbf{PM}$ and $\mathbf{X}$ are used, while with PRM-ITEM to the model using only $\mathbf{X}$. For both models we use the best parameter configuration obtained in the cross-validation highlighted in Table 1 and described in Section 4.2.3. Table 5 displays the results of this experiment. It is possible to see how, by including the personalized component, we obtain an improvement in both NDCG and MAP at all levels of k. In particular, with respect to our target metric NDCG@10, we outperform both No Re-rank and PRM-ITEM by 2 percentage points. We can thus state that personalization has a positive contribution in the re-ranking problem.

5 Conclusions and Future Directions

Utilizing the current state-of-the-art for context-aware re-ranking, we have been able to obtain improvements in both NDCG and MAP metrics with respect to Zillow's generated ranking. We have seen that by including additional information in the target used for training (multi-level relevance) we obtain slightly lower performances in both NDCG and MAP. This possibly stems from the fact that the relative weight of each interaction type hasn't been investigated or that no hyperparameter fine-tuning has been conducted on PRM (Multi). Finally, the experimental results obtained in the feature ablation study prove the effectiveness of personalization for re-ranking.

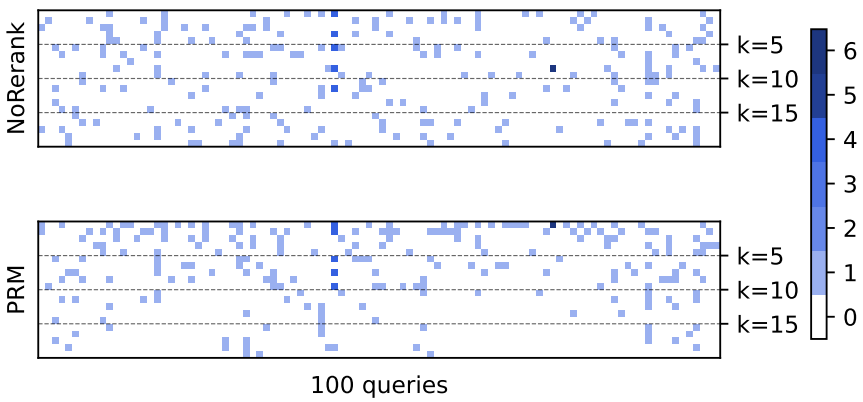

Figure 3: Visualization of PRM (Multi) re-ranking results against the No Re-rank baseline for the top-20 items on 100 randomly sampled test queries. The color encodes the item's ground-truth multi-level relevance.

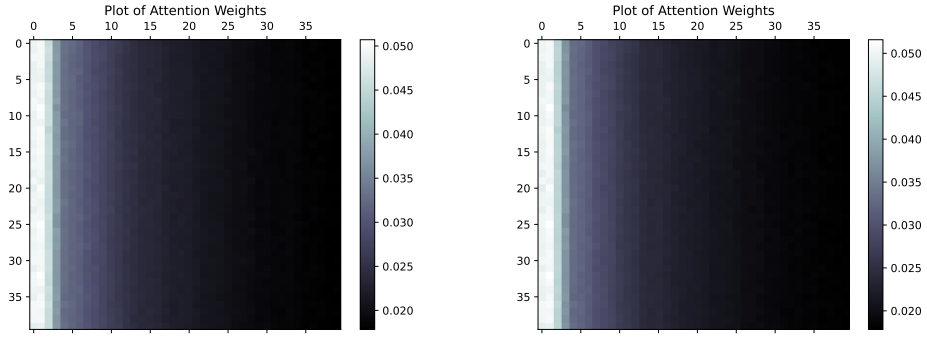

Figure 4: Average attention weights of 8 attention heads of the last transformer block for PRM (Binary) on the left, and PRM (Multi) on the right.

In regards to future directions, we note that we did not experiment with qualitative metrics such as novelty, diversity and fairness, which may yield other interesting avenues of investigation for multi-objective optimization. Moreover, this would pave the way for further personalization, as different users may have different preferences regarding these more qualitative aspects. In this case, the need for learning a framework to model the relationship between the multiple objectives would arise. A solution to it could then be learning a trade-off parameter to weight the different metrics, or directly use metrics combining multiple objectives like $\alpha-$NDCG [Clarke et al., 2008] or subtopic-recall (S-recall) [Zhai et al., 2003].

Another aspect that might be worth looking into is the supervision relevance signal used to train a re-ranking model. Feedback on item relevance is only available for the displayed initial ranking lists, not for the remaining $n! - 1$ configurations (*counterfactual lists*). Due to the mutual influence between items in the displayed list, different permutations of the input items may generate different relevance labels. A popular method used to learn by counterfactual signals is the *evaluator-generator paradigm*, where the generator produces feasible permutations and the evaluator gives feedback for each permutation Xi et al. [2021].

Finally, we have not explored the "echo chamber" effect studied by Ge et al. [2020] when considering how recommendations influence user preferences and behaviors. It is possible that users intents were influenced by repeated exposure to similar suggestions. Different metrics would need to be

Model	NDCG@				MAP@			
	5	10	15	20	5	10	15	20
No Re-rank	34.18	39.82	43.24	45.35	27.79	30.57	32.08	32.92
PRM	**35.78**	**41.79**	**45.13**	**47.21**	**38.31**	**38.22**	**37.47**	**36.89**
PRM-ITEM	34.32	39.94	43.24	45.36	27.92	31.42	32.17	33.08

Table 5: Ranking metrics computed on the test set for the click labels. All values have been multiplied by 100. Both PRM and PRM-ITEM have been trained on the binary relevance based on clicks.

considered to capture this phenomenon when investigating click, favorite, and submit tendencies over time.

6 Lessons Learned

Solely relying on native Pandas dataframes when handling large datasets on single machines presented us with low performance and insufficient memory use. To better extract and manipulate the dataframe's underlying vectors in a big data ecosystem, code must be written in a manner more directly interfacing the cluster. In our case this was done via PySpark, which served as the interface to the Greene cluster's Apache Spark. Scalable computing became enough a concern to warrant the use of distributed computation altogether: we leverage XGBoost PySpark implementation for the training of LambdaMART, while the other baselines inherently leveraged parallelization in the Spark ecosystem, all facilitated by the Slurm resource manager. Had we more time (and resources), implementing distributed training of PRM would've been considered in a multi-node, multi-GPU environment using PyTorch's Distributed Data parallel library.

We believe this project has been an incredible opportunity to measure our ability of adapting and learning about a new domain in a short period of time, as well as a great proving ground for the data science skills we developed throughout our course of study.

7 Student Contributions

Giacomo Bugli Literature review, environment setup for distributed Spark processing and PyTorch multi-GPU training on GCP, data preprocessing, Popularity and No Re-rank baselines implementation, PRM implementation, ran model experiments, implemented evaluation pipeline for model testing and visualization of results, poster creation, report writing, produced presentations to Zillow Applied Science Team

Luigi Noto Literature review, environment setup for distributed Spark processing and PyTorch multi-GPU training on GCP, data preprocessing, LambdaMART baseline implementation, PRM implementation, ran model experiments, implemented evaluation pipeline for model testing and visualization of results, poster creation, report writing, produced presentations to Zillow Applied Science Team

Guilherme Albertini Literature review, data exploration, investigated trade-offs of different code bases, PRM implementation, ran model configurations as needed by team, poster creation, report writing, produced presentations to Zillow Applied Science Team

References

Qingyao Ai, Keping Bi, Jiafeng Guo, and W. Bruce Croft. Learning a deep listwise context model for ranking refinement. volume abs/1804.05936, 2018. URL `http://arxiv.org/abs/1804.05936`.

Qingyao Ai, Xuanhui Wang, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning groupwise scoring functions using deep neural networks. *CoRR*, abs/1811.04415, 2019. URL `http://arxiv.org/abs/1811.04415`.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Huai-hsin Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. Seq2slate: Re-ranking and slate optimization with rnns. volume abs/1810.02019, 2018. URL http://arxiv.org/abs/1810.02019.

Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. 2010. URL http://research.microsoft.com/en-us/um/people/cburges/tech_reports/MSR-TR-2010-82.pdf.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 129–136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, page 659–666, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581644. doi: 10.1145/1390334.1390446. URL https://doi.org/10.1145/1390334.1390446.

Yingqiang Ge, Shuya Zhao, Honglu Zhou, Changhua Pei, Fei Sun, Wenwu Ou, and Yongfeng Zhang. Understanding echo chambers in e-commerce recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, jul 2020. doi: 10.1145/3397271.3401431. URL https://doi.org/10.1145%2F3397271.3401431.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. Setrank: Learning a permutation-invariant ranking model for information retrieval. *CoRR*, abs/1912.05891, 2019. URL http://arxiv.org/abs/1912.05891.

Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. Personalized re-ranking for recommendation. RecSys '19, page 3–11, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362436. doi: 10.1145/3298689.3347000. URL https://doi.org/10.1145/3298689.3347000.

Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. Context-aware learning to rank with self-attention. *CoRR*, abs/2005.10084, 2020. URL https://arxiv.org/abs/2005.10084.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/29921001f2f04bd3baee84a12e98098f-Paper.pdf.

Yunjia Xi, Weiwen Liu, Xinyi Dai, Ruiming Tang, Weinan Zhang, Qing Liu, Xiuqiang He, and Yong Yu. Context-aware reranking with utility maximization for recommendation, 2021. URL https://arxiv.org/abs/2110.09059.

Cheng Xiang Zhai, William W. Cohen, and John Lafferty. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. SIGIR '03, page 10–17, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136463. doi: 10.1145/860435.860440. URL https://doi.org/10.1145/860435.860440.

Tao Zhuang, Wenwu Ou, and Zhirong Wang. Globally optimized mutual influence aware ranking in e-commerce search. volume abs/1805.08524, 2018. URL http://arxiv.org/abs/1805.08524.