Combining Bundle Economy and Relevancy through Content Recommendations in Candy Crush Saga

Styliani Katsarou[1,†], Francesca Carminati[1,†], Martin Dlask[1,*], Marta Braojos[1], Lavena Patra[1], Richard Perkins[1], Carlos Garcia Ling[1] and Maria Paskevich[1]

[1]*King, Stockholm, Sweden*

Abstract

Recommender systems traditionally focus on maximizing user satisfaction by suggesting preferred products to the users. This approach aims to increase the likelihood of immediate sales and relevancy for users. However, this focus can often neglect broader business strategic goals crucial for stakeholders. This paper presents a novel dual-objective Bundle Recommendation system tailored for King's "Candy Crush Saga" designed to balance the relevancy of recommendations for players with game economy returns. We propose a two-step methodology combining an attention-based model to accurately capture diverse player behaviors and preferences, followed by a clustering algorithm that defines a representative pool of in-game bundles, tailored to the game platform's constraints. The efficacy of our approach is assessed through a controlled A/B testing framework, measuring the take rate (TR) for economic impact and the click rate (CR) for user engagement. We report significant performance gains, with a 41% increase in TR and a 33% increase in CR, effectively balancing user satisfaction with economic returns.

Keywords

Personalization, Recommender Systems, Bundle Recommendation, Attention models, Productionization, Click Rate, Engagement, TabNet

1. Introduction

Recommender systems have become ubiquitous across various sectors, enhancing decision-making processes by facilitating efficient discovery and access to new products, services, and content. Traditional systems predominantly adopt a receiver-centric (user-centric) approach, focusing on maximizing user satisfaction without considering the broader strategic or business objectives that are crucial for system stakeholders.

However, the landscape of recommender systems is evolving. The distinction between "organic" recommender systems, which prioritize personalized user experiences, and "strategic" or "utility-aware" recommender systems is becoming more pronounced. The latter aims to balance user relevance with additional utilities, be they economic, social, or ethical, to optimize overall system value [1]. This approach is particularly relevant in multi-sided platforms where the interaction between user and utility adds layers of complexity to the recommendation process [2, 3].

In the gaming industry, especially in systems featuring in-game economies and virtual marketplaces, utility-aware recommender systems aim to consider both user engagement and economic returns. This paper explores a method employed by King to achieve this alignment in Candy Crush Saga: a strategic *Bundle Recommendation* of in-game items, designed to appeal to players' desires and keep the bundle economy balanced, while increasing transactions from the recommended items.

Bundle Recommendation (BR) [4, 5] is a specific type of recommendation systems where the goal is to suggest combinations or sets of items (bundles) that are likely to be of interest to the user. BR is a complex problem due to the following reasons: In contrast to the conventional recommendation

Workshop on Strategic and Utility-aware REcommendations (SURE) @RecSys2024 October 14-18, 2024, Bari, Italy

[*]Corresponding author.

[†]These authors contributed equally.

stella.katsarou@king.com (S. Katsarou); francesca.carminati@king.com (F. Carminati); martin.dlask@king.com (M. Dlask); marta.braojos@king.com (M. Braojos); lavena.patra@king.com (L. Patra); richard.perkins@king.com (R. Perkins); carlos.garcia2@king.com (C. G. Ling); maria.paskevich@king.com (M. Paskevich)

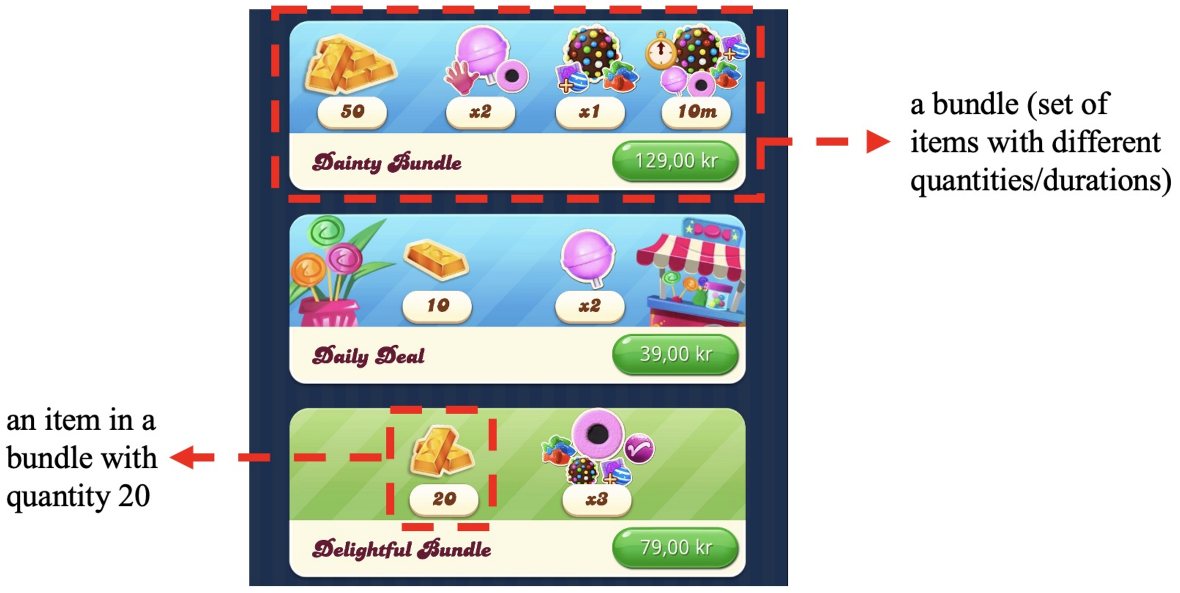

a bundle (set of items with different quantities/durations)
an item in a bundle with quantity 20
50
x2
x1
10m
Dainty Bundle
129,00 kr
10
x2
Daily Deal
39,00 kr
20
x3
Delightful Bundle
79,00 kr

Figure 1: In-game bundle recommendation.

(CR) [6, 7], where the task involves selecting one or multiple items from a fixed, but large list, BR includes various combinations of items with arbitrary quantities. While both CR and BR lead to data sparsity problems, the number of combinations in BR is several orders of magnitude higher, deepening the problem with sparsity further on. To further add to the complexity, diversity should be ensured both across, as well as within the bundles, so that the recommended results positively contribute to the game economy. In this work, we design our BR system with the dual objective of optimizing for user relevancy and the game economy, whilst considering constraints of the game system.

Our approach employs a two-step, game economy-aware methodology for in-game bundle recommendation: We first employ an attention-based recommendation model to accurately capture individual user preferences. This step focuses on diverse player behaviors and preferences at a granular level. Given the constraints of the gaming platform, where only a limited number of bundles can be available in the game, in the second step we employ a clustering algorithm to define a pool of the most statistically representative in-game bundles. To evaluate the effectiveness of these bundles with respect to the game economy, we monitor the take rate (TR). User relevance is assessed in an offline setting by measuring the model's performance using cosine distance, and in an online setting, monitored through click rate (CR).

The contributions of our work are outlined below:

- We introduce a novel two-step approach to Bundle Recommendation systems, that employs a combination of a supervised and an unsupervised approach. This utility-aware BR system is designed to consider both user preferences and the game economy.
- We validate the online performance of our method using a controlled A/B testing framework, demonstrating the correlation between user relevancy and game economy. Click rate is employed to measure user relevancy, while take rate serves as a measure of economic impact.
- We present how this solution is deployed in the real-world in order to serve millions of users on a daily basis, detailing the design of data collection, inference, training, and monitoring pipelines, ensuring maintainability.

2. Related Work

Current recommender algorithms mainly suggest individual items by analyzing user-item interactions. However, recommending item sets, or bundles, particularly within online gaming environments, has received less focus. The integration of Bundle Recommendation (BR) in online games, which aims to simultaneously cater to user preferences and business goals, is an emerging research area. This field is still in its early stages of development, especially in the industrial applications of online games.

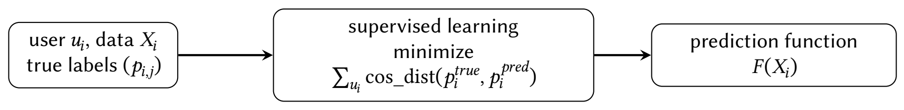
Within the context of bundle recommendations, several methodologies have been explored. [8] solve the problem of bundle recommendations for suggesting booklists, by using a latent factor-based Bayesian Personalized Ranking (BPR) model which would consider users' interactions with both item lists and individual items. Later, this approach was extended by [9] who introduced the Embedding Factorization Model (EFM), an approach that jointly models user-item and user-list interactions, incorporating Bayesian Personalized Ranking[10] and word2vec models [11]. In [12], existing bundles were suggested to users based on constituent items, and personalized new bundles were generated using a bundle-level BPR model. A graph-based approach introduced by [13], unified user-item interaction, user-bundle interaction, and bundle-item affiliation into a heterogeneous graph. In [14], a factorized attention network was employed to aggregate item embeddings within a bundle, addressing user-bundle and user-item interactions in a multi-task manner. More recently, in [15], the Bundle Graph Transformer model (BundleGT) was introduced, that utilized a token embedding layer and a hierarchical graph transformer layer to simultaneously capture strategy-aware user and bundle representations. BRUCE [16] is another method that adapted Transformers to the bundle recommendation problem, by leveraging the self-attention mechanism to capture latent relations between items within a bundle and users' preferences toward individual items and the entire bundle. [4] used a feature-aware softmax in an encoder-decoder framework and integrated masked beam search to generate high-quality and diverse bundle lists with appropriate sizes for E-commerce. [17] introduced Bundle Multi-Round Conversational Recommendation (Bundle MCR) that extended multi-round conversational recommendation (MCR) [18] to a bundle setting, by formulating Bundle MCR as Markov Decision Processes (MDPs) with multiple agents. Additional related work on bundle recommendation include [19, 20, 21, 22].

Despite these advancements, considering both broader user engagement and economic returns mentioned in the Introduction in bundle recommendations remains underexplored. In the gaming sector, previous research has primarily focused on recommending game titles based on historical user data [23, 24] or single in-game item recommendations [25, 26]. The exploration of bundle recommendations within online gaming is notably sparse. Deng et al. [27] is one of the few works that tackled BR in gaming by framing it as a link prediction problem within a tripartite graph and employing a neural network model for direct learning. However, their approach focused solely on user relevancy.

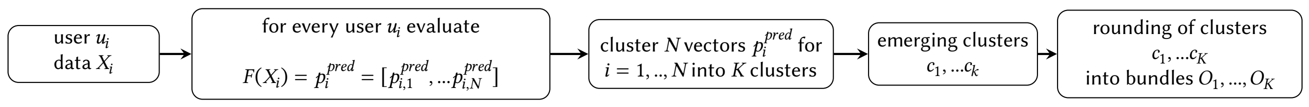
Our approach seeks to fill this gap by designing a Bundle Recommendation system with the dual objective of optimizing for user relevancy and considering the game economy and the constraints of the game system, while ensuring through careful monitoring that our recommendations not only achieve user satisfaction but also drive substantial business value.

3. Methodology

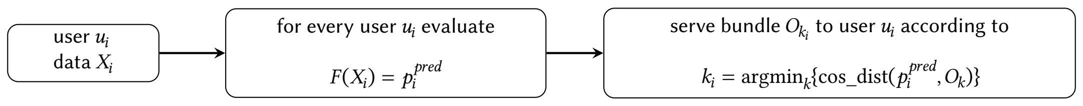
Candy Crush Saga developed by King— is an online mobile game with millions of users, where players advance through a sequential map of progressively challenging levels by solving match-3 puzzles. The pace of advancement is contingent upon the player's skill level, determined by their ability to strategically choose optimal moves, with the option of utilizing appropriate boosters and timing their usage effectively. As in other free-to-play games, players have the option to buy virtual items with real money through in-app purchases (IAPs). Users are presented with a range of bundles that consist of in-game currency and other in-game power-ups like boosters, time-limited boosters, and unlimited lives. The quantity of in-game currency and other in-bundle items can vary. An example of how bundle recommendations are presented to the users is depicted in Fig. 1.



(a) Training flow (step 1)


user u_i
data X_i
for every user u_i evaluate
$F(X_i) = p_i^{pred} = [p_{i,1}^{pred}, ...p_{i,N}^{pred}]$
cluster N vectors p_i^{pred} for
$i = 1, .., N$ into K clusters
emerging clusters
$c_1, ...c_k$
rounding of clusters
$c_1, ...c_K$
into bundles $O_1, ..., O_K$

(b) Clustering flow (step 2)


user u_i
data X_i
for every user u_i evaluate
$F(X_i) = p_i^{pred}$
serve bundle O_{k_i} to user u_i according to
$k_i = \text{argmin}_k\{\text{cos\_dist}(p_i^{pred}, O_k)\}$

(c) Inference flow (executed daily)

Figure 2: Training, Inference and Clustering flows

In this section we explain in two-steps how we optimize for user relevance in an offline setting, to ensure that the recommended bundles reflect the user preference on a global scale, using historical data. We also describe how we streamline our experimentation and deployment processes with a platform that supports the complete ML workflow.

3.1. Our Solution

Suppose we have users $U = \{u_i \mid i = 1, 2, \ldots N\}$ and items $I = \{i_j \mid j = 1, 2, \ldots D\}$. Our solution comprises two sequential steps.

Step 1. In the first step, we predict one D-dimensional vector per user u_i, denoted as $P_i = [p_{i,1}, p_{i,2}, \ldots, p_{i,D}]$, where each value $p_{i,j}$ represents the quantity of a bundle item j purchased by the user u_i. To predict this vector, we adopt a supervised learning approach. We formulate the task as a multi-output regression problem, where the target consists of D numerical values representing the quantities of each respective item purchased by the user. During training, we aim to minimize the cosine distance between true preference P_{true} and prediction P_{pred} as follows

$$\sum_{u_i} \text{cos\_dist}(P_i^{true}, P_i^{pred}) = d_p.$$

Given that the targets are normalised, we use cosine similarity as our evaluation metrics as it ignores the overall scale of the predicted vectors, which is beneficial if the magnitude of the model predictions is not directly comparable to the targets. Moreover, the proportionality of the items' values in the vectors matters to us in this use-case. The cosine distance metric enables a scale-invariant comparison of the proportions of the different items present in the predicted vector and the actual label vector. The training flow is depicted in Fig. 2.

Step 2. We expect to find many similar combinations of in-game item proportions in the predictions yielded from the model in Step 1, so in this step we employ an unsupervised clustering approach to define a discrete preference space. Since the quantities of the items in a bundle are discrete, the clustering approach serves the double purpose of discretizing the problem and resolving the data sparsity. The goal here is to define a set of preference clusters,

$$C = \{c_k \mid k = 1, 2, \ldots K\}.$$

Parameter K is usually chosen between 5 and 30 so we preserve meaningful differences between the bundles. These differences depend on user perception and can be only studied qualitatively. In the experiments section $K = 20$ is used. The distance from the raw prediction to the closest cluster centroid is:

$$\text{cos\_dist}(pred, clust) = d_c.$$

At this point we have K real-valued vectors, but given that the elements of the vectors represent actual in-game products, we need to round the values so that they describe the actual quantities of the various in-game items to be shown in the bundles. In this step, we convert the K clusters to bundles that will be recommended to our users. By the end of this step, we will have defined a set of bundles $\mathcal{O} = \{O_1, O_2, ...O_K\}$, $O_k = (v_1, ..., v_D) \in \mathbb{N}^D$ with $v_j \in \mathbb{N}$ is the volumes of each item i_j for every $j = 1, .., D$. The distance between the cluster centroid and the final product with rounded values is:

$$\text{cos\_dist}(clust, product) = d_o.$$

The error to be minimized from this whole procedure is:

$$\text{cos\_dist}(true, product) <= d_p + d_c + d_o.$$

The process of creating clusters and bundles is visualized in Fig. 2. This 2-step process enables the segregation of the model predictions and the delivery of personalized results through bundles, providing flexibility to easily modify and market different offers.

3.1.1. Model selection

Our ML model of choice for Step 1 is TabNet [28]. TabNet uses a structured attention mechanism to highlight important features during each decision step, which enables transparency and interpretability of the model's predictions, as well as efficient handling of sparse features. In our dataset, each row represents a distinct user during a specific time period. Multiple rows can belong to the same user if their activity spans over larger periods, reflecting their interactions at different points of time. To prevent data leakage, the same user does not appear in both the training and evaluation sets. Given the diverse user-base in terms of skill and playing style, and the dynamic nature of user playing behavior, with rapid progress and style changes over short periods, each row is unique across users and even for the same user from day to day, so not all features are expected to be relevant for every example. TabNet's capability to handle sparsity and operate on an instance-wise basis is advantageous for our use case, as it allows the model to independently determine the features to pay attention to for each example. Regarding TabNet hyperparameters, we primarily adhere to the default settings as provided in the PyTorch implementation[29]. We use a progressively decreasing learning rate schedule to enhance the stability of the model's performance. In Step 2, we chose to employ an unsupervised k-means clustering algorithm. This decision was based on its simplicity and efficiency, making it an ideal choice for scalability and speed—crucial factors when deploying for millions of predictions.

3.2. System Overview

To enhance the ease of experimenting and deploying machine learning models, King has developed a platform designed to support and automate various aspects of the ML workflow. Employing a self-service approach, the platform provides machine learning practitioners with a range of modular components and tools that streamline the modeling workflow. These resources are integrated under a unified system, akin to previously described ML systems [30] [31]. Figure 3 details the structure of our system.

The system in Fig.3 is composed of four distinct pipelines: a data pipeline for daily data extraction, a training pipeline, an inference pipeline, and a monitoring pipeline. Our internal data notification service notifies our in-house pipelines orchestrator of new data availability and in turn, the orchestrator triggers all pipelines.

We use infrastructure-as-code tools, leveraging standardized modules that we apply to all environments to ensure consistency of the environments in development and production, allowing for more rigorous testing and minimizing one of the most common pain points of ML practitioners [32]. As we only need to make daily predictions, we opt for a batch prediction system that executes all of the pipelines daily.

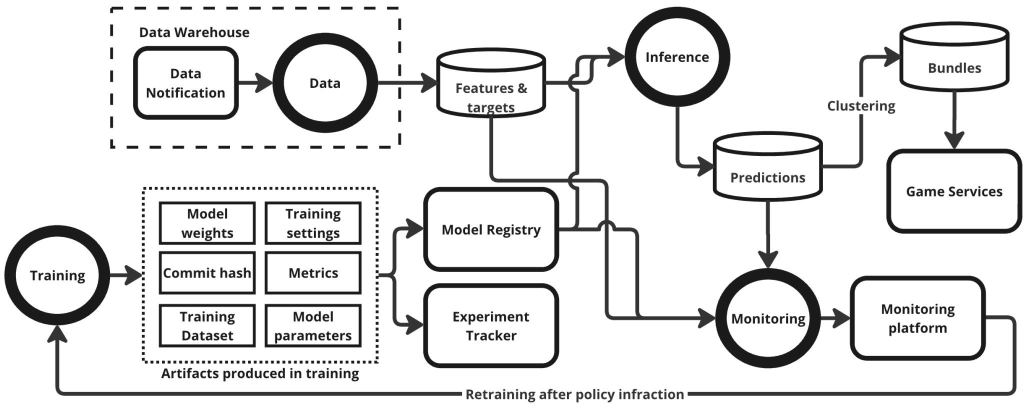


Figure 3: ML System Overview

3.2.1. Data Extraction

The initial data extraction, including the retrieval of raw model features and the computation of labels, is handled by the data pipeline which draws from the data warehouse. The feature transformation pipeline is configured to ensure that the feature generation process is idempotent, even in the case of data backfilling.

3.2.2. Training & Inference

The training pipeline fetches data prepared by the data pipelines to train models and generates artifacts which are consumed by an experiment tracker and the monitoring and inference pipelines. The type of data we hold within the artifacts are model weights, parameters and metadata, the git hash of the training code, evaluation metrics, training datasets and settings (e.g. learning rate, seed, optimizer parameters, etc.). This artifact choice enables us to tackle the reproducibility challenge inherent in operationalizing ML projects [33] by providing all the necessary elements to recreate the model.

The inference pipeline produces predictions for each feature for the next day and stores them in a database. After we have generated our predictions, they are converted to the bundles using the unsupervised clustering approach described in Step 2 of 3.1.

To prevent issues related to training/inference skew, we rely on parameterized queries as the input to the model for both training and inference pipeline. Together with a common data pipeline, this allows for a common source of truth, removing discrepancies in the data preparation.

3.2.3. Serving

Once the personalized bundles have been generated, they are uploaded to the game services to be delivered to our players. The recommendation is cached in the game for the whole user session and the user is displayed the recommended bundle consistently through the assigned placements. To ensure reliability, the game system incorporates two fail-safe mechanisms: first, if there is no prediction available for a user in the most recent batch of bundles, the latest available bundle for that user is displayed. Second, if no bundle is available at all, we employ a fallback, non-personalized bundle. Once the recommendations are live, we validate their performance following relevant metrics with an A/B test setup. To avoid issues stemming from inconsistent definitions [32], we have a standard validation process for all models at various stages of the ML process (which is also version-controlled). Models undergo the same A/B test setup, and once in production, we monitor the same business metrics for all of them, this ensures reliable comparisons between models and an increase in iteration speed.

3.2.4. Monitoring

Model monitoring is essential for reliability and production-level machine learning systems [34]. Our system monitoring relies on a third-party platform. The monitoring pipeline is responsible for uploading daily predictions, features, and labels to this external platform. In addition to the standard monitoring policies to address training/serving skew, changes in feature distributions or relationship between features and labels [35], we track key business metrics to ensure the model's relevance to the business [36]. In particular we monitor the bundles' take rate, click rate, and recommendation diversity associated with the model's usage. Furthermore, we implement feature importance monitoring to ensure that the contributions of features remain consistent during serving, fostering transparency in understanding the correlation between input data and model outcomes.

Upon any monitoring policy violation, an alert prompts an investigation, followed by the model retraining; post-retraining, a detailed manual investigation informs the decision to promote the updated model to production, utilizing CI/CD pipelines for automatic deployment.

4. Experiments

4.1. Existing baselines

In this section, we describe the experimental setup and outcomes of both the offline training of TabNet and the online A/B testing of bundles suggested by our two-step approach introduced in Section 3.1. For the offline analysis, we evaluate TabNet against XGBoost to determine effectiveness w.r.t. cosine distance. In the online scenario, we extend the comparison to include both TabNet and XGBoost, alongside a heuristic approach. The heuristic approach, known as *heuristics*, is crafted manually by subject matter experts and is based on game domain knowledge rather than personalized user data. Additionally, in the second online experiment, we investigate the impact of injecting varying levels of randomness—referred to as contamination—into our model recommendations.

4.2. Offline Experiments

The training process is carried out in batches, with the input being a matrix $X = \left\{x^{(m)}\right\}^{M}{}_{m=1} \in R^{M \times F}$, where M represents the number of samples in each batch, and F denotes the number of input features in each sample. The input features include data on player behavior. We do not use or compare with public datasets as they do not have relevant properties and user actions that are required for our solution architecture, moreover, they cannot be used for online experiments. In our dataset, we only keep users who have been active for more than or equal to 30 days and aggregate all features by averaging them over an N-day period. The target consists of D numerical values $[p_1, p_2, \ldots, p_D]$ representing the quantities of each respective item purchased by the user on their next active day after the N-day period. We train our models over hundreds of thousands of users, for D=13. To simulate the production setting where users exhibit diverse activity levels, we do not aggregate the test set over a N-day period. Instead, we include all users regardless of the amount of active days they have had. If a user has been active for less than N days, we aggregate their corresponding input features over as many days as they have been active for. We train two distinct models:

- TabNet with $N = 15$ days
- TabNet with $N = 30$ days

These two models are differentiated by their respective number of aggregation days. We evaluate the models' performance using the mean cosine distance as our evaluation metric, as outlined in Section 3.1. A lower value of this metric indicates better model performance. Using cosine distance for both optimization and evaluation ensures consistency since there is a clear alignment between what the model is learning during training and how it's judged in evaluation. It also simplifies performance interpretation, as the model is directly optimized to minimize this metric. However, it may overly specialize the model, potentially missing broader patterns that enhance user satisfaction or real-world performance, since cosine distance focuses on direction rather than magnitude or other factors that could affect user preferences.

Based on the results shown in Table 1, we proceed with the TabNet model that uses a 30-day aggregation period. It is worth mentioning that the mean cosine similarity is not normally distributed on [0,1]. Statistical tests suggest it follows a Beta distribution across all offline experiments. Additionally, the mean of the distribution significantly decreases with the use of the TabNet architecture, improving similarity across a broader range of users, including those who initially had higher cosine distance.

Model	Mean Cosine Distance
XGBoost baseline	0.234
TabNet 15day	0.124
TabNet 30day	**0.103**

Table 1
Offline experimentation results.

4.3. Online Experiments

4.3.1. Metrics

In our setting we want to study especially the relationship between the relevancy and game economy. Therefore, we define a set of online metrics that can quantify the relevancy of the recommendation.

Click volume: Click volume $CV(P,d)$ is the total number of clicks on product P and day d.

Acceptance volume: Acceptance volume $AV(P,d)$ denotes the number of takes of product P and day d.

Click rate: Click rate $CR(d)$ is the proportion between $CV(d)$ and number of impressions on day d.

Take rate: Take rate $TR(d)$ is the proportion between $AV(d)$ and number of impressions on day d.

We are using the click rate CR to measure the relevancy of our recommendation, while the take rate TR measures the performance of the game economy. Both of these metrics are scaled according to the number of impressions per day, since those can vary from user to user. We want to validate that increased relevancy for the user translates into increase in CR, while the overall effect on the game economy increases TR.

4.3.2. A/B experimentation

To be able to understand the online performance of our approaches to recommendation, we have tested the predictive models using A/B experiments. The A/B testing methodology allows us to compare the performance of key metrics between the treatment group and the control group in a single experiment. We denote the uplift of metric M as a percentage difference of the absolute values of M in the treatment group and control, respectively, scaled to the size of these groups. We denote the aggregate uplift in metric M as ΔM.

We define a pool of bundles as a set $\mathcal{O} = \{O_1, O_2, ...O_k\}$, where O_i is a bundle. We define a random bundle B_R as a discrete random variable uniformly distributed on $\mathcal{O}$, i.e. $B_R \sim \mathrm{U}(\mathcal{O})$. Recommended bundle R is defined as $\mathrm{argmin}\{\mathrm{cos\_dist}(O_i, x) \, : \, O_i \in \mathcal{O}\}$ for model prediction x and pool of bundles $\mathcal{O}$. Let X be a uniformly distributed continuous random variable on $[0,100]$ and let B_R be a random recommendation. The recommendation with contamination $p\%$ is defined as a random variable $N_p = \mathrm{I}(p < X) \cdot B + I(p \geq X) \cdot P$ where I is the indicator function, i.e. $\mathrm{I}(\mathrm{true}) = 1$ and $\mathrm{I}(\mathrm{false}) = 0$ and R is recommended bundle. Throughout the experiments section, we'll be using four kinds of treatment groups that we denote T:

- TabNet model recommendation (T_0)
- Recommendation with 10% contamination (T_{10})
- Recommendation with 30% contamination (T_{30})

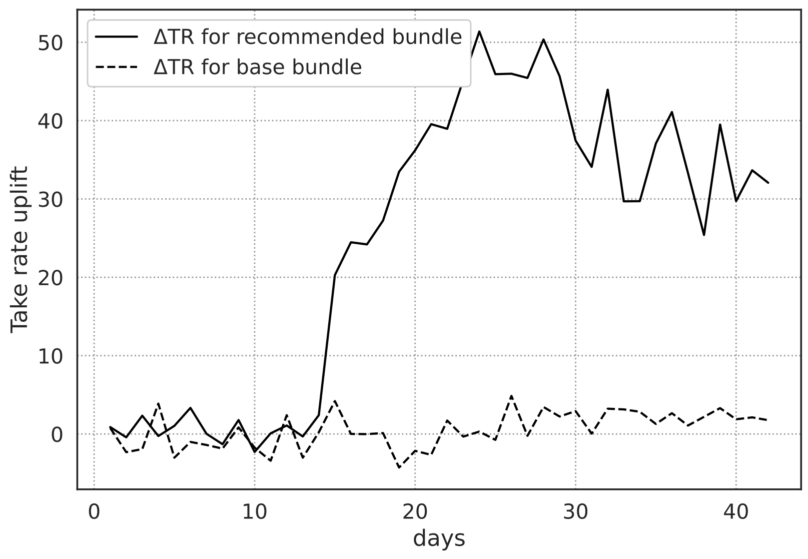


Figure 4: Take rates uplifts in experiment 1.

4.3.3. Experiments

Prior to conducting experiments, Candy Crush Saga had a heuristic (rule-based) approach to serve bundle content to players. Here we conducted two experiments, where we tested the performance of the recommendation against the heuristic approach, which is our primary control group. Alongside this, we want to understand the relationship between the model accuracy and relevancy. We conducted experiment with random group as a source treatment, while serving contaminated recommendation in the target treatment. We use K=20 as the number of total bundles. The setting, including the source treatment and target treatment, is shown in Tab. 2.

Experiment	Source treatment	Target treatment
1	heuristic	T_0
2	random	T_0, T_{10}, T_{30}

Table 2
Experimental setting overview.

Experiment 1 This experiment tests the model recommendation against a heuristic recommendation. Our source treatment serves heuristic bundle recommendation, while the target treatment is decided using the TabNet recommendation model. While the take rate uplift has increased significantly by 41% and the corresponding click rate has increased by 33%, as shown in Tab. 3

We visualize the trend in the increase of take rate of the recommended product in Fig. 4. The experiment started on day 11, and while the recommended product gradually gained popularity, the novelty effect stabilized roughly 20 days after the experiment started. This experiment has demonstrated the potential of varying bundle content with respect to the user engagement metrics. While the user engagement increased, the game economy metric TR increased even more, resulting in more successful transactions. However, it is necessary to understand further the relationship of how quickly TR decreases if the relevance deteriorates. This is the objective of the next experiment.

Experiment 2 In experiment 2 we study the relation between a random recommendation and model recommendations with different levels of contamination. This enables us to understand how the key

Experiment	Treatment group	ΔTR	ΔCR	$\Delta TR/\Delta CR$
1	T_0	41.32%	32.59%	1.27
2	T_0	131.41%	39.18%	3.35
2	T_{10}	117.74%	36.64%	3.21
2	T_{30}	91.17%	28.81%	3.16

Table 3
Changes in engagement metrics from experiments.

engagement metrics uplifts deteriorate when the model is contaminated with a random recommendation in some cases, being the source treatment a random recommendation in all cases. We take advantage of the artificially created treatment group with a random recommendation as a method to compare to other treatment groups T_0, T_{10}, T_{30}. The changes in key metrics are presented in Tab. 3. We can see that with increasing levels of contamination, the TR and CR uplifts decrease, however, at a non-linear pace. The level of contamination $p \in [0, 10, 30]$ does not guarantee a proportional decrease in the engagement metrics.

Results interpretation The experiments aimed to study the relationship between the click rate as a measure of relevancy together with the take rate as a measure of game economy. In the first experiment, we've observed significant increase in both metrics, while the take rate uplift was higher than the click rate uplift. We've observed similar trend also in experiment 2, while we've achieved a much higher uplift in take rates when compared to the random group.

The question one could be asking is why we didn't obtain a higher click-rate uplift when compared to the random group in experiment 2. The proportion of $\Delta TR/\Delta CR$ is about 1.27 in experiment 1, while for T_0 in experiment 2 it is around 3.35. We get almost three times higher TR in comparison to CR in experiment 2 is because the random control group created different recommendations for a user every day, which has increased clicks on that particular bundle, without increasing its relevancy. Additionally, these numbers are cleaned up from the novelty effect, reported after the initial 20-day period. We proved experimentally when serving different content every day, regardless if relevant or not, the resulting metrics achieve higher click rates than static or heuristic recommendations, which are the same every day. Therefore, to preserve the importance of click rate as a relevancy metric, experimentation with contamination is necessary, to prove a diminishing relationship between the recommendation performance and corresponding online metrics in experiment 2.

5. Discussion

5.1. TabNet

TabNet [28] has been our initial take on a Tabular Neural network for this approach due to its flexibility and interpretability.

5.1.1. Interpretability

The model's architecture uses a sequential attention mechanism that dynamically identifies and prioritizes important features for each sample. Specifically, by examining attention weights, we can ensure the model prioritizes relevant features. This helps identify and correct biases where the model might overemphasize features linked to unrelated targets, leading to more accurate, target-focused predictions.

5.1.2. Self-supervised pre-training

TabNet's self-supervised pre-training involves training on an unsupervised task without labels [28], allowing the model to discover data patterns and focus on important features before supervised training, thus improving its ability to identify underlying structures.

5.1.3. Other modeling approaches

In the current 3-step approach described in 3.1, we use TabNet as the model of choice. The downside of the solution is its heavier computational load, which slows down the training pipeline. While real-time retraining is not a constraint in the current formulation, we are interested in exploring lighter models and the balance between offline performance evaluation and computation costs for the training across different architectures. XGBoost or other vanilla Neural Networks like a simple feed-forward neural network, have been considered, and initial experiments reveal comparable results in our key metric, cosine distance, during offline evaluation.

5.2. Data and Loss Function enhancements

Adding smart processing to the same data can improve how our models identify player preferences. This section explores a few approaches for achieving this.

5.2.1. Feature Aggregation

When exploring player's preferences in historical data, it is important to capture both short-term patterns as well as more long-term in-game habits and preferences of a user. Currently, our model's 30-day input feature aggregation captures long-term player behavior but doesn't give higher weight to more recent events. Incorporating shorter aggregation periods may improve performance. Alternatively, a hierarchical model could automatically capture short-term patterns in lower levels and aggregate long-term behavior in higher levels.

5.2.2. The Cold Start Problem

The Cold Start Problem: Our training data focuses on in-game purchases, targeting users who made a purchase on a given day. We aim to determine the preference vector for items in the purchased bundle, so only paying and new users are included, while non-paying players are excluded. This creates a cold start problem, as recommendations require at least one purchase. Addressing this by developing models for underrepresented players could be included in the scope of a future iteration.

5.2.3. Loss function: imbalance between targets

The loss function uses cosine distance between output and true label vectors but doesn't account for weight differences between items, which can cause overestimation when some items are over-represented in training. Our analysis shows a 60% drop in performance when excluding one target, with in-game currency included in training but not in evaluation. This suggests in-game currency contributes 60% of the cosine distance. To better understand the players' preferences and cater to individual play styles, we could explore target-specific weighting, separate models for distinct targets, or fixed allocation for specific targets, and focusing the model on the preferences for the remaining.

6. Conclusions

In this paper, we present a novel two-step approach to item recommendations in mobile games, which was applied and tested on a bundle recommendation problem in Candy Crush Saga. First, we've defined the general methodology and architecture of the solution, which was specially designed for the mobile game environment. Apart from offline validation, the architecture was also tested in several online experiments, empirically modeling the relationship between the click- and take rates and model accuracy. The robust architecture and technical debt prevention strategies allowed the system to be deployed in two in-game placements, one of them being illustrated in Figure 1.

The novelty of this approach lies not only in the item recommendation methodology, which is subsequently applied to bundle recommendation but also in the implementation. The robust and fail-safe pipeline is designed to scale for millions of players and implements many policies that can prevent the delivery of inaccurate recommendations. We continuously monitor the system performance, both in the offline and online environment, where we focus on understanding the click change- and take rates, but also other underdeveloped metrics, such as the impact of degenerate feedback loops and a corresponding deterioration in recommendation diversity.

The scale-invariant system defined in the methodology is an efficient tool both for the generalization of the system for other tasks and presents a responsible AI solution that makes sure fairness resulting from the generation of the recommendation is in place regardless of user's level of activity or spending.

7. Acknowledgements

We're grateful for the support from King we've been given while preparing the manuscript and the openness we work in that contributes to a good level of insight sharing. In addition to the authors who have contributed hands-on to the success of this solution, we also thank other teams in King for their help and support, specifically ML Special Projects, MLOps Accelerator, Core Data, AI Labs, CCS IAP&E and CCS Operators teams. We also thank Pradyumna Prasad for his significant support and contributions.

References

[1] A. De Biasio, A. Montagna, F. Aiolli, N. Navarin, A systematic review of value-aware recommender systems, Expert Systems with Applications (2023) 120131.

[2] C. Pei, X. Yang, Q. Cui, X. Lin, F. Sun, P. Jiang, W. Ou, Y. Zhang, Value-aware recommendation based on reinforcement profit maximization, in: The World Wide Web Conference, 2019, pp. 3123–3129.

[3] Q. Wu, H. Wang, L. Hong, Y. Shi, Returning is believing: Optimizing long-term user engagement in recommender systems, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1927–1936.

[4] J. Bai, C. Zhou, J. Song, X. Qu, W. An, Z. Li, J. Gao, Personalized bundle list recommendation, in: The World Wide Web Conference, 2019, pp. 60–71.

[5] J. Chang, C. Gao, X. He, D. Jin, Y. Li, Bundle recommendation and generation with graph neural networks, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 2326–2340. doi:10.1109/TKDE.2021.3114586.

[6] Xiao, Benbasat, E-commerce product recommendation agents: Use, characteristics, and impact, MIS Quarterly 31 (2007) 137. URL: http://dx.doi.org/10.2307/25148784. doi:10.2307/25148784.

[7] Y. Zhang, J. R. Jiao, An associative classification-based recommendation system for personalization in b2c e-commerce applications, Expert Systems with Applications 33 (2007) 357–367. URL: http://dx.doi.org/10.1016/j.eswa.2006.05.005. doi:10.1016/j.eswa.2006.05.005.

[8] Y. Liu, M. Xie, L. V. Lakshmanan, Recommending user generated item lists, in: Proceedings of the 8th ACM Conference on Recommender systems, 2014, pp. 185–192.

[9] D. Cao, L. Nie, X. He, X. Wei, S. Zhu, T.-S. Chua, Embedding factorization models for jointly recommending items and user generated lists, in: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, 2017, pp. 585–594.

[10] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint arXiv:1205.2618 (2012).

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Advances in neural information processing systems 26 (2013).

[12] A. Pathak, K. Gupta, J. McAuley, Generating and personalizing bundle recommendations on steam, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 1073–1076.

[13] J. Chang, C. Gao, X. He, D. Jin, Y. Li, Bundle recommendation with graph convolutional networks, in: Proceedings of the 43rd international ACM SIGIR conference on Research and development in Information Retrieval, 2020, pp. 1673–1676.

[14] L. Chen, Y. Liu, X. He, L. Gao, Z. Zheng, Matching user with item set: Collaborative bundle recommendation with deep attention network., in: IJCAI, 2019, pp. 2095–2101.

[15] Y. Wei, X. Liu, Y. Ma, X. Wang, L. Nie, T.-S. Chua, Strategy-aware bundle recommender system, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 1198–1207.

[16] T. Avny Brosh, A. Livne, O. Sar Shalom, B. Shapira, M. Last, Bruce: Bundle recommendation using contextualized item embeddings, in: Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 237–245.

[17] Z. He, H. Zhao, T. Yu, S. Kim, F. Du, J. McAuley, Bundle mcr: Towards conversational bundle recommendation, in: Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 288–298.

[18] Y. Deng, Y. Li, F. Sun, B. Ding, W. Lam, Unified conversational recommendation policy learning via graph-based reinforcement learning, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1431–1441.

[19] S. Qi, N. Mamoulis, E. Pitoura, P. Tsaparas, Recommending packages to groups, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 449–458.

[20] Y. He, J. Wang, W. Niu, J. Caverlee, A hierarchical self-attentive model for recommending user-generated item lists, in: Proceedings of the 28th ACM international conference on information and knowledge management, 2019, pp. 1481–1490.

[21] R. Garfinkel, R. Gopal, A. Tripathi, F. Yin, Design of a shopbot and recommender system for bundle purchases, Decision Support Systems 42 (2006) 1974–1986.

[22] M. Beladev, L. Rokach, B. Shapira, Recommender systems for product bundling, Knowledge-Based Systems 111 (2016) 193–206.

[23] S. M. Anwar, T. Shahzad, Z. Sattar, R. Khan, M. Majid, A game recommender system using collaborative filtering (gambit), in: 2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST), IEEE, 2017, pp. 328–332.

[24] R. Sifa, A. Drachen, C. Bauckhage, Large-scale cross-game player behavior analysis on steam, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 11, 2015, pp. 198–204.

[25] V. Araujo, F. Rios, D. Parra, Data mining for item recommendation in moba games, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 393–397.

[26] P. Bertens, A. Guitart, P. P. Chen, A. Perianez, A machine-learning item recommendation system for video games, in: 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, 2018, pp. 1–4.

[27] Q. Deng, K. Wang, M. Zhao, Z. Zou, R. Wu, J. Tao, C. Fan, L. Chen, Personalized bundle recommendation in online games, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 2381–2388.

[28] S. Ö. Arik, T. Pfister, Tabnet: Attentive interpretable tabular learning, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 6679–6687.

[29] Tabnet : Attentive interpretable tabular learning, 2023. URL: https://pypi.org/project/pytorch-tabnet/, accessed: 2023-12-01.

[30] I. L. Markov, H. Wang, Looper: An end-to-end ml platform for product decisions, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 3513–3523. URL: https://doi.org/10.1145/3534678.3539059. doi:10.1145/3534678.3539059.

[31] M. Haldar, M. Abdool, P. Ramanathan, T. Xu, S. Yang, H. Duan, Q. Zhang, N. Barrow-Williams, B. C. Turnbull, B. M. Collins, T. Legrand, Applying deep learning to airbnb search, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1927–1935. URL:

https://doi.org/10.1145/3292500.3330658. doi:10.1145/3292500.3330658.

[32] S. Shankar, R. Garcia, J. M. Hellerstein, A. G. Parameswaran, Operationalizing machine learning: An interview study, arXiv preprint arXiv:2209.09125 (2022).

[33] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, D. Dennison, Hidden technical debt in machine learning systems, Advances in neural information processing systems 28 (2015).

[34] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley, The ml test score: A rubric for ml production readiness and technical debt reduction, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 1123–1132.

[35] C. Huyen, Designing Machine Learning Systems: An Iterative Process for Production-ready Applications, O'Reilly Media, Incorporated, 2022. URL: https://books.google.se/books?id=YISIzwEACAAJ.

[36] T. Schröder, M. Schulz, Monitoring machine learning models: a categorization of challenges and methods, Data Science and Management 5 (2022) 105–116.