Revisiting Bundle Recommendation for Intent-aware Product Bundling

ZHU SUN, A*STAR Centre for Frontier AI Research, Nanyang Technological University, Singapore, Singapore
KAIDONG FENG, Yanshan University, Qinhuangdao, China
JIE YANG, Delft University of Technology, Delft, the Netherlands
HUI FANG, Shanghai University of Finance and Economics, Shanghai, China
XINGHUA QU, Shanda AI Lab, Singapore, Singapore
YEW-SOON ONG, A*STAR Centre for Frontier AI Research, Nanyang Technological University, Singapore, Singapore
WENYUAN LIU, Yanshan University, Qinhuangdao, China

Product bundling represents a prevalent marketing strategy in both offline stores and e-commerce systems. Despite its widespread use, previous studies on bundle recommendation face two significant limitations. Firstly, they rely on noisy datasets, where bundles are defined by heuristics, e.g., products co-purchased in the same session. Secondly, they target specific tasks by holding unrealistic assumptions, e.g., the availability of bundles for recommendation directly. This paper proposes to take a step back and considers the process of bundle recommendation from a holistic user experience perspective. We first construct high-quality bundle datasets with rich metadata, particularly bundle intents, through a carefully designed crowd-sourcing task. We then define a series of tasks that together, support all key steps in a typical bundle recommendation process, from bundle detection, completion and ranking, to explanation and auto-naming, whereby 19 research questions are raised correspondingly to guide the analysis. Finally, we conduct extensive experiments and analyses with representative recommendation models and large language models (LLMs), demonstrating the challenges and opportunities, especially with the emergence of LLMs. To summarize, our study contributes by introducing novel data sources, paving the way for new research avenues, and offering insights to guide product bundling in real e-commerce platforms.

We greatly acknowledge the support of National Natural Science Foundation of China (Grant No. 72371148 and 72192832), the Shanghai Rising-Star Program (Grant No. 23QA1403100), and the Natural Science Foundation of Shanghai (Grant No. 21ZR1421900). It was partly supported by the Delft Design@Scale AI Lab. It was also supported by A*Star Center for Frontier Artificial Intelligence Research and in part by the Data Science and Artificial Intelligence Research Centre, School of Computer Science and Engineering at the Nanyang Technological University (NTU), Singapore.
Authors' addresses: Z. Sun, Institute of High Performance Computing; Centre for Frontier AI Research, A*STAR, 1 Fusionopolis Way, Singapore, Singapore, 138632; e-mail: sunzhuntu@gmail.com; K. Feng (Corresponding author) and W. Liu, Yanshan University, 438 Hebei Ave, Qinhuangdao, China, 066000; e-mails: fengkaidong@stumail.ysu.edu.cn, wyliu@ysu.edu.cn; J. Yang, Delft University of Technology, Mekelweg 5, Delft, the Netherlands, 2628; e-mail: j.yang-3@tudelft.nl; H. Fang, Shanghai University of Finance and Economics, 100Wudong Road, Shanghai, China, 200433; e-mail: fang.hui@mail.shufe.edu.cn; X. Qu, Shanda AI Lab, 25 Ubi Road 4, Singapore, Singapore, 408621; e-mail: quxinghua17@gmail.com; Y.-S. Ong, A*STAR Centre for Frontier AI Research; Nanyang Technological University, 1 Fusionopolis Way and 50 Nanyang Ave, Singapore, Singapore, 138632 and 639798; e-mails: ong_yew_soon@hq.a-star.edu.sg, asysong@ntu.edu.sg.

ACM 2770-6699/2024/06-ART24
https://doi.org/10.1145/3652865

CCS Concepts: • **Information systems** → **Recommender systems**; • **Computing methodologies** → *Neural networks;*

Additional Key Words and Phrases: Product bundling, crowd-sourcing task, bundle datasets, bundle recommendation

ACM Reference Format:
Zhu Sun, Kaidong Feng, Jie Yang, Hui Fang, Xinghua Qu, Yew-Soon Ong, and Wenyuan Liu. 2024. Revisiting Bundle Recommendation for Intent-aware Product Bundling. *ACM Trans. Recomm. Syst.* 2, 3, Article 24 (June 2024), 34 pages. https://doi.org/10.1145/3652865

1 INTRODUCTION

Product bundling plays a vital role as a marketing strategy to support promotional campaigns, attract customers, and drive sales revenue, in the realm of e-commerce [11]. This strategy groups together a collection of related products that users can purchase as a cohesive unit, due to factors like a limited total price [25] or specific intentions [31]. To be specific, Figure 1 illustrates three example bundles in the domains of Clothing, Electronics, and Food. For instance, (a) showcases a party clothing bundle comprising items like handbags, dresses, heels, necklaces, and earrings. Suppose a customer is shopping online with the intent of purchasing fashion clothing for an upcoming party. In such a scenario, they would anticipate a bundle consisting of well-coordinated fashion products rather than individual items with no apparent connections.

The aforementioned demonstration highlights the role of bundles in organizing and presenting products that are highly correlated, alleviating customers from the tedium of decision-making. This practice benefits both customers and sellers for several reasons. Firstly, bundles contribute to enhancing the user experience in various ways. For instance, alternative items bring together related products, enabling customers to make better comparisons. Additionally, complementary items broaden the user's scope of options, offering an escape from monotonous decision-making processes. Secondly, bundles have the potential to increase sales revenue for sellers. By exposing users to new items that they may not have considered individually, bundles can drive additional purchases. Lastly, buying or selling bundles can be more cost-effective compared to purchasing or selling individual products separately. E-commerce systems often offer promotions and waive delivery fees if the products in a single order exceed a certain threshold [4]. In this way, product bundling serves as a win-win solution for both customers and sellers.

Although many approaches have been recently proposed for bundle recommendation, they suffer from several inherent limitations. **Firstly**, most approaches are built upon noisy datasets, where bundles are defined by unverified heuristics. For instance, most studies [22, 44, 98] simply treat co-purchased products as synthetic bundles, despite the fact that lots of such products are co-purchased with no common underlying intents. Others directly treat user-generated lists as bundles in specific domains such as music and books [9, 11, 13, 30, 31, 46]. Apart from being limited in specific domains, these studies have not made an attempt to gain an understanding of the rationale behind product bundling. Some other work [18, 25, 45, 77] leverages bundles defined by retailers. Obtaining such bundles is however a long, laborious and expensive process; and consequently, the sizes of such bundle datasets are limited. **Secondly**, existing studies are restricted to specific tasks, e.g., they often directly dive into the task of bundle recommendation, with the unrealistic assumption that the historical bundles of a user are observable to the system. We argue that to support bundle recommendation, intermediate steps need to be in place such as detecting bundles from user sessions [82]; furthermore, a set of auxiliary tasks, e.g., bundle completion (i.e., adding more relevant products with the same intent for bundle expansion), need to be considered for successful bundle recommendation in real applications.

Fig. 1. Three running examples for bundles in the domains of Clothing, Electronic, and Food.

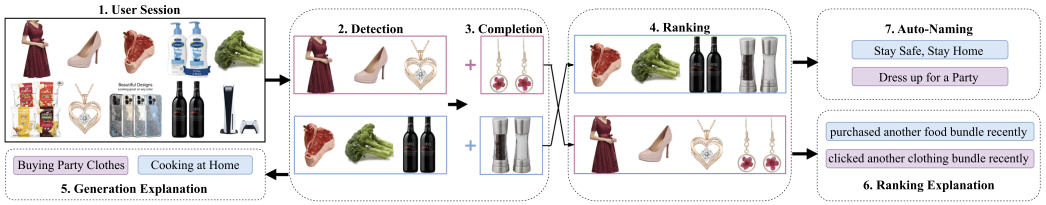


Fig. 2. The interrelated tasks for product bundling.

Datasets Construction. In this study, we take a step back and consider the process of bundle recommendation from a holistic user experience perspective. First, to better understand the rationale behind product bundling, we delicately design a crowd-sourcing task with the goal of leveraging crowd intelligence to help label potential bundles and corresponding intents hidden in the user session in three domains (Electronic, Clothing, and Food). Our design draws inspiration from recent studies [82] showing that users tend to explore highly correlated, e.g., alternative or complementary [67], products with a common intent in a session. As a result, we obtain three high-quality bundle datasets with rich meta information, particularly the bundle intents.

Task Formulation. Using such data, we then propose a series of important interrelated tasks to support bundle recommendation, as illustrated in Figure 2. In reality, a user may interact (e.g., click) with multiple items in one session [82], with or without common intents. An essential task is therefore *bundle detection*, aiming to efficiently detect potential bundles hidden in the session. Accordingly, *bundle completion* is a subsequent task, which seeks to expand existing bundles by adding more relevant products for broader choices. Afterwards, *bundle ranking* then ranks these enriched bundles based on user preference for more accurate bundle recommendation. Meanwhile, *bundle explanation* could help interpret the results of both bundle generation (i.e., detection and completion) and bundle ranking. As such, it consists of *bundle generation explanation* and *bundle ranking explanation*, where the former task aims to infer the intents (topics) reflected by the bundle, thus gaining a better understanding of the rationale behind product bundling; and the latter task seeks to figure out the reason for recommending certain bundles, thus it can better understand the motivations behind users' consumption behavior. Both tasks could enable enhanced system transparency and increased user trust. With the inferred bundle intents, *bundle auto-naming* could help further generate attractive bundle names, e.g., 'Stay Safe, Stay Home' for the food bundle in Figure 1(c), thus providing a better user experience.

Task Exploration. Lastly, to understand the research needed for better product bundling, we critically examine and analyze a set of state-of-the-art methods for our defined tasks through extensive experiments, especially the efficiency of prevalent **large language models (LLMs)** [35].

Specifically, we tailor the *a priori* algorithm [1] for bundle detection at the item category level and devise novel metrics for evaluation. We then adopt and adapt a number of representative recommendation methods ranging from simple conventional ones to recent deep learning advancements for bundle completion, ranking, and ranking explanation tasks. Furthermore, we take advantage of prevalent LLMs (e.g., T5 [57] and ChatGPT [49]) to accommodate the bundle generation explanation and auto-naming tasks. To better guide our analysis, we raise 19 research questions ranging from the effectiveness of specific method design (e.g., graph-based neural methods) to general training strategies (e.g., pre-training). By answering these questions, we aim to point out potential research challenges and opportunities, especially with the emergence of LLMs, as well as to provide insights and recommendations for future studies.

Important Findings. Several important conclusions are drawn accordingly: (a) pattern mining allows to generate high-quality, new bundles; (b) the generative model VAE [41] performs well in bundle completion; (c) graph-based neural methods [11] show the efficacy and necessity of capturing complex relations among users, products and bundles for bundle ranking; (d) the **sequence-to-sequence (seq2seq)** language model BART [38] is most effective in explaining bundle generation; (e) the graph-based neural method KGAT [83] shows superiority on bundle ranking explanation.; and (f) the LLMs (e.g., ChatGPT) gain satisfying performance on bundle auto-naming. Yet, (g) data sparsity remains a major challenge confronted by all tasks, which calls for more research, such as fusing side information (e.g., knowledge graphs [68, 83]) or exploiting LLMs (with rich knowledge and remarkable capability on few-shot recommendation scenarios [6]), for further enhancement.

Our Contributions. In summary, our main contributions lie in three folds. (1) We introduce three new bundle datasets with enriched meta data (e.g., bundle intents) contributed by crowds, thereby facilitating future research on intent-aware product bundling. (2) We define a series of tasks for product bundling, from bundle detection, completion, and ranking, to explanation and auto-naming, pointing to new research directions to bring forward product bundling. (3) We perform extensive experiments and in-depth analysis on the defined tasks via state-of-the-art methods (prevalent LLMs included), highlighting the arising research challenges and opportunities, particularly in light of the emergence of LLMs. The resulting insights can help provide guidance, and ultimately promote product bundling in real e-commerce platforms.[1]

[1]Our preliminary study was published at SIGIR'22 [72]. In this paper, we extend it by investigating the unexplored bundle explanation and auto-naming tasks in four aspects. First, we perform a comprehensive literature review on both explainable recommendation and bundle auto-naming tasks; meanwhile, we also complement the latest related works regarding bundle recommendation. Second, we refine the proposed interrelated tasks by dividing the bundle explanation task into bundle generation explanation and bundle ranking explanation tasks in Figure 2. The former task aims to interpret the results of bundle generation (i.e., bundle detection and bundle completion) by inferring the intent (topic) reflected by the bundle, thus it can better understand the rationale behind product bundling; whereas the goal of the latter task is to uncover the reason for recommending certain bundles, thus it can better understand the motivations behind users' consumption behavior. Both tasks could enable enhanced system transparency and increased user trust. Third, we formulate bundle generation explanation, bundle ranking explanation, and bundle auto-naming tasks, whereby new research questions are raised accordingly to guide the analysis. To answer these questions, extensive experiments are conducted to examine the performance of explanation and auto-naming approaches, including but not limited to prevalent LLMs, through both automatic and human evaluation. Especially for the bundle auto-naming task, which is quite new in the context of recommendation, little academic research has been done so far. We thus seek inspirations from other related domains, e.g., computer vision and natural language processing, and resort to image captioning models and LLMs (e.g., ChatGPT) for potential solutions. Lastly, we perform in-depth analysis based on the experimental results, whereby new insightful conclusions, research challenges, and opportunities are provided, particularly in light of the emergence of LLMs.

2 RELATED WORK

In this section, we first conduct a comprehensive literature review regarding the bundle recommendation. Then, we provide an overview of the explainable recommendation, which could be adopted and adapted for the bundle explanation task. Finally, we present relevant studies offering potential solutions for the bundle auto-naming task.

2.1 Bundle Recommendation

Bundle recommendation methods are typically classified into nine categories. (1) *Constraint based methods* consider various constraints, e.g., cost and revenue in e-commerce [98]; season, price and ratings for travel package recommendation [45, 88]; and degree requirements for course recommendation [51]. (2) *Data mining based methods* leverage, e.g., association rule mining [22] and probabilistic models [44] for bundle recommendation. (3) *Preference elicitation based methods* are proposed to learn utility functions [21, 89] that capture user preference among various features (e.g., cost and quality) over bundles via user feedback. (4) *Factorization based methods* decompose, e.g., user-item and/or user-bundle matrices, to learn user preference over items and bundles, such as LIRE [46], BBPR [53] and EFM-Joint [9].

Stemming from the success in the related domains (e.g., computer vision), neural network based approaches have been widely applied in bundle recommendation. In particular, (5) *sequence based neural methods* exploit, e.g., the sequence generation model in BGN [4] and LSTM [33] in ComEmb [37], for bundle recommendation. Later, (6) *attention based neural methods* integrate the attention mechanism to attentively learn item affinity or user preference towards bundles, e.g., DAM [13], AttList [30], CAR [31] and BRUCE [3], to name a few. After that, (7) *graph based neural methods*, such as BundleNet [18] and BGCN [11] apply **graph convolutional network (GCN)** [83] on the user-item-bundle tripartite graph to facilitate bundle recommendation. Recently, (8) *constrastive learning based neural methods* have been proposed for more effective bundle recommendation, including MIDGN [97], CrossCBR [48] and Conna [85]. Moreover, (9) conversational based methods have emerged for dynamic bundle recommendation, for instance, BUNT [32] adapts multi-round conversational recommendation settings to bundle recommendation scenarios by designing the Markov decision processes with multiple agents.

Despite the prosperity of current research on bundle recommendation, most studies are based on unverified datasets. They directly treat either co-purchased products [44, 98] or user-generated lists [11, 13, 21, 30, 31, 46] as synthetic bundles, ignoring the fact that (1) co-purchased products may not always reflect common intents; and (2) user-generated lists are generally limited to specific domains (e.g., music and books); moreover, they also fail to unveil the rationale behind product bundling. Studies relying on bundles pre-defined by retailers [4, 18, 22, 25, 45, 77] are restricted by limited size of datasets due to the high cost on producing such bundles. Furthermore, they merely focus on specific tasks, e.g., directly diving into bundle recommendation assuming the availability of bundles to the system. Prior to that, a series of tasks need to be in place to support bundle recommendation, e.g., detecting well-defined bundles from user sessions, and completing them for more choices. Other auxiliary tasks, e.g., bundle explanation and auto-naming are also essential to promote bundle recommendation in real e-commerce platforms.

2.2 Explainable Recommendation

The goal of explainable recommendation is to provide interpretation for the recommended products, thereby enhancing system transparency and increasing user trust [96]. In this subsection, we provide an overview of representative explainable recommendation studies from three perspectives, namely *when*, *how*, and *what* explanations are generated. Although explainable recommendation has been widely studied in recent years, it has been rarely touched in the complex scenario

of bundle recommendation. Therefore, in this study, we seek to explore how to explain bundle recommendation on the basis of existing investigation.

With regard to 'when explanations are generated', two types of approaches are identified, including (1) *inherently explainable models*, which adopt the same model to simultaneously generate recommendation and explanations [83, 95]; and (2) *post-hoc explainability methods*, which employ an independent model to provide explanations for the recommendation model after the recommendations have been provided [54, 74].

Regarding 'how explanations are generated', there are typically three types of methods. (1) *Rule mining based methods* [42] adopt association rules to mine frequent patterns to generate and interpret the recommendations. (2) *Factorization based methods* extract explicit item features and decompose user- and item-feature matrices to infer user interest and generate feature-level explanation [2, 16, 95]. (3) *Deep learning based methods* have attracted much attention recently, where seq2seq models and **knowledge graph (KG)** based models are the representatives. In particular, the former generally adopts seq2seq approaches to generate natural language explanation based on the review information along with the rating prediction task, e.g., [39, 47, 65]; whereas the latter mainly exploits KGs to mine connected paths between users and items as the recommendation explanation, e.g., [26, 52, 83, 87].

In terms of 'what explanations are generated', five types of methods have been taken into account. (1) *Pattern explanation* presents the mined frequent patterns by the rule mining based methods as the explanation [42]. (2) *Feature explanation* mainly adopts users' interested product features as explanations, and are generally adopted by the factorization based methods [2, 95]. (3) *Natural language explanation* usually employs the natural sentence generated by the seq2seq models as the interpretation [47, 52]. (4) *Path explanation* makes use of the extracted paths from the KGs to represent explanations [26, 83]. (5) *Visual explanation* extracts visual features from the product images to explain the recommendations [43].

2.3 Bundle Auto-Naming

Bundle auto-naming aims to automatically generate attractive names for given bundles based on their content (e.g., items contained and intents). Since this is a new task in the context of recommendation, little academic research has been done so far. We, therefore, seek inspiration from other related domains, e.g., **computer vision (CV)** and **natural language processing (NLP)**.

One promising solution is to adopt and adapt the approaches for image captioning, i.e., describing images with syntactically and semantically meaningful sentences [64]. Thanks to the availability of the images of items inside bundles, we can formulate the auto-naming task into the image captioning task, that is, given images of items within a specific bundle, the goal is to generate semantically meaningful sentences to describe the bundle, i.e., naming the bundle. Specifically, existing studies (e.g., [12, 28, 92]) usually adopt CNN for image understanding and RNN for text generation. Due to the significant improvements of the Transformer [78] architecture and pre-training strategies in CV and NLP, the Transformer has been recently adopted in the image captioning, which utilizes a pre-trained transformer-based vision model as the encoder to extract image features and a pre-trained language model as the decoder to generate textual caption, such as TrOCR [40]. Inspired by this, we utilize the VisionEncoderDecoderModel[2] built by Hugging Face[3] to generate names by providing the images of bundles. *Another potential direction* is to take advantage of existing large language models (LLMs), e.g., BERT [19], GPT3 [8] and LLaMa [75], which have achieved remarkable success in natural language understanding and generation. The availability of bundle intents

[2]https://huggingface.co/docs/transformers/model_doc/vision-encoder-decoder
[3]https://huggingface.co

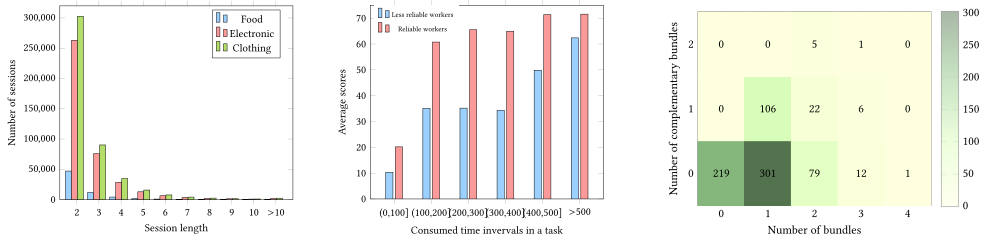


Fig. 3. (a) session length distribution for the three domains; (b) score distribution on various time intervals; and (c) session distribution regarding the number of detected bundles and complementary bundles.

in our study allows us to leverage existing dialogue systems, such as ChatGPT [49] and Bard [27] to generate attractive bundle names, that is, we first design a general instruction format for describing, e.g., the items inside a bundle, the bundle intent, and the task form in natural language, so that LLMs can understand and further execute the instruction for fulfilling the auto-naming task [93]. It is noteworthy that current dialogue systems, e.g., ChatGPT, cannot accommodate image input, so we do not include item images when designing instructions for the auto-naming task. Furthermore, the multi-product summarization task proposed in [76] is quite relevant to our bundle auto-naming task. Specifically, it aims to automatically generate a summary (i.e., widget caption) for a list of products based on textual information (e.g., product title and category) with a weakly supervised learning approach. Unfortunately, the unavailability of the source code prevents us from re-implementing this method for our bundle auto-naming task.

3 CONSTRUCTING BUNDLE DATASETS

This section describes the construction process for our new bundle datasets, considering specifically the necessity of having a consistent intent in the bundle. To obtain such intent information, we design and deploy a crowd-sourcing task to identify potential bundles in a user session and label them with user intents.

3.1 Crowdsourced Annotation Task

3.1.1 Data Preparation. Starting off with the Amazon datasets [29] widely used in recommendation research, we select three domains (Electronic, Clothing, and Food) and chronologically order the records of each user based on the timestamp information, and then divide them into sessions by days. Due to the long history of the original data (from May 1996 to July 2014), some products may be outdated. As such, we only consider records that occurred in the last year of the data, i.e., from July 2013 to July 2014. We sample 2,000 sessions for each domain with lengths from 2 to 10 considering that (1) most session lengths are in the range of $[2, 10]$ (see Figure 3(a)); and that (2) longer sessions may contain noisy items and increase the difficulty of annotation. To account for the imbalanced session length, we adopt the stratified sampling strategy. In particular, for each domain, we divide the sessions into four groups based on their lengths – $[2, 3]$, $[4, 5]$, $[6, 7]$, $[8, 9, 10]$, and then sample 500 sessions from each group. As a result, we obtain 2,000 sessions with balanced length distribution from each domain.

3.1.2 Task Design and Execution. Our task asks workers to identify potential bundles from user sessions and label them with corresponding user intents. It mainly consists of three parts, shown in Figure 4. In the beginning, each worker is asked to provide their basic information at the first page, including age, gender, country, occupation, education and online shopping frequency (see Figure 4(a)). Next, the worker will then be navigated to the second page with the task overview,

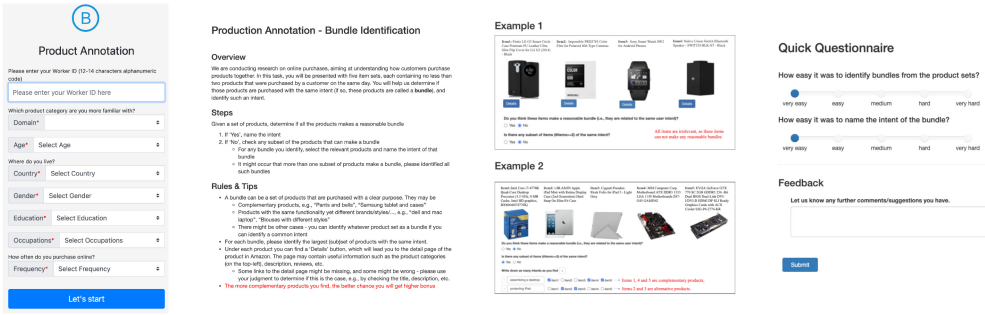


Fig. 4. The three web-pages for our designed task: (a) first page; (b,c) second page; and (d) third page.

instructions and examples (see Figures 4(b)–(c)), followed by five user sessions that need to be annotated. For each user session, the worker first answers the question 'Do you think these items make a reasonable bundle (i.e., they are related to the same user intent)?'. If yes, the worker is asked to name the bundle intent; otherwise, a following question shows up 'Is there any *subset* of items of the same intent?'. If yes, the worker is asked to select items that form the subsets (i.e., bundles) and name the corresponding intents. After finishing the annotation, the worker is asked to provide their feedback on our task (see Figure 4(d)).

We launch our task on Amazon Mechanical Turk (www.mturk.com/). With a 10$ per-hour standard (higher than the minimum wage 7.5$ per hour in the US), we pay each annotation task 1$, where 0.3$ is the base reward, and the rest 0.7$ is the bonus for high-quality contributions.

3.1.3 Quality Control Mechanisms. A key concern of crowdsourced data annotation is the annotation quality. That is, workers usually seek to maximize their monetary rewards by doing tasks as fast as possible, potentially de-prioritizing the quality of their work [23]. To deal with this issue, we adopt some widely-used quality control mechanisms, and beyond that, develop our own customized mechanisms in this study.

First, each sampled session is annotated by three workers, so that we can obtain higher quality annotations via results aggregation. Second, we use honeypot sessions to detect low-quality annotations. Within the five sessions prepared in each annotation task, we include one session with ground-truth bundles and intents available—workers who fail to provide correct annotation for this session are deemed to be less reliable. To create such ground-truth sessions, we randomly sample 90 extra sessions (10 for each bundle size) from each domain and have three authors manually annotate them. By carefully cross-checking with each other, a consensus is reached for all the annotations w.r.t. both the bundles and intents. Third, we record the time that each worker spends in completing the task: those who spend too short or too long time may indicate low-quality work [91].

While helping to build the first line of defense, those quality control mechanisms might be too general. For example, it remains unclear how to best set the threshold of time elapsed for filtering annotations, and use it together with the honeypot mechanism. Besides, given a limited budget, we seek to maximize the diversity of the bundles, in terms of properties that go beyond bundle sizes. With these in mind, we decide to serially launch our task in two batches, using the first batch to get preliminary feedback and do necessary adjustments in the second batch.

3.2 Bundle Collection Procedure

In the first batch, we randomly select 300 sessions for each domain and obtain the results contributed by 577 workers in total.

3.2.1 Quality Assessment. To filter out low-quality annotations, we assess the annotation quality from two aspects, namely *intent coverage* and *item coverage*, denoting to what extent a worker can find out 'all intents' from a session and 'all items' for an individual intent, respectively. Formally,

$$intent\_score = \#correct\_intents/\#all\_intents * 100,$$
$$item\_score = (\#correct\_items - \#wrong\_items)/\#all\_items * 100,$$

where $\#correct\_intents$ is the number of correct intents labelled by the worker; $\#correct\_items$ and $\#wrong\_items$ are the number of correct and wrong items selected by the worker w.r.t. a specific intent, respectively. We average the intent and item score to measure the reliability of each worker.

3.2.2 Filtering Annotations. We divide all workers from the first batch into two groups, viz., workers who provide correct answers for the honeypot sessions are considered reliable, and otherwise less reliable. Based on the above metrics, our goal is to find a sweet spot for the threshold of the time used by workers, such that we can keep as many high-quality annotations as possible – even from less reliable workers – while excluding low-quality ones.

To do so, we first manually filter out workers who provide obviously wrong answers with the extremely short time consumed (e.g., $< 50s$). With our categorization, we are left with 451 workers with 277 being reliable and 174 being less reliable. Figure 3(b) depicts the average score for the two groups of workers in different time intervals (i.e., time spent in completing the task). As expected, reliable workers perform better (i.e., gain higher scores) than less reliable workers across all intervals. The scores climb up as the time spent increases for all workers. Accordingly, we devise a rule to help filter high-quality annotations, viz., those from reliable workers with more than $100s$ spent and from less reliable workers with more than $500s$ spent. After the filtering, we are left with 752 sessions having valid annotations for the three domains, where 533 sessions contain bundles and the corresponding intents, and the remaining 219 sessions do not contain items that can form bundles.

3.2.3 Item Relation Analysis. To further examine the quality of the collected data, we analyze the number of bundles detected by the workers and the relations (alternative or complementary [67]) among items inside bundles for each session. Arguably, an ideal dataset should contain more complementary items to avoid monotonous choices. As shown in Figure 3(c), we find most sessions do not contain any complementary bundles: there are only 140 (out of 752) sessions containing complementary bundles, indicating the scarcity of such bundles. To increase the number of complementary bundles under a limited budget, we investigate how to sample sessions from the remaining 1,700 sessions (each domain) for our second batch.

3.2.4 Filtering Complementary Sessions. We use a data-driven approach to find characteristics of item pairs that can help us find more sessions with complementary items. To achieve this goal, we first manually label item relations—complementary and non-complementary—in the 533 sessions from our first batch, and then use these data to train a **decision tree (DT)** model to help automatically mine item relations in a session. Overall, we identify 976 pairs of complementary items and 11,811 pairs of non-complementary items. Considering the unbalanced data distribution, we randomly sample an equal number of item pairs for each relation type, and use 1,500 pairs to train and test the DT model. We construct five features using the meta data (i.e., category hierarchies and item images), including image distance calculated via ImageHash,[4] the number of common categories, the ratio of common categories, the depth between common category and leaf node, and the depth between root and common category. With careful parameter tuning, the DT

[4] pypi.org/project/ImageHash/

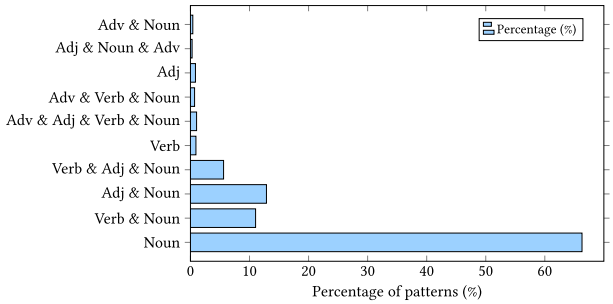


Fig. 5. The distribution of intent wording patterns.

Table 1. Examples for Intent Wording Patterns

Patterns	Examples
Noun	Earrings and pendants set; Accessories for Apple products;
Verb	Cooking; Baking;
Verb & Noun	Traveling and taking photos; Making a meal;
Adj & Noun	Men's jackets with different styles; High scale photography equipment;
Verb & Adj & Noun	Make hot drinks; Make digital photos;
Adv & Adj & Verb & Noun	The intent would be to upgrade Personal PC or utilizing PC more effectively.

reaches Precision, Recall, F1 and AUC on the test data (30%, cross-validation) with 52.6%, 4.3%, 8%, and 69%, respectively.

Afterwards, we use the well-trained DT to classify pair-wise item relations in the 1,700 sessions for each domain: a session is classified as a complementary session as long as it contains one complementary item pair. As a result, we obtain 506, 338, and 463 complementary sessions in the domain of Electronic, Clothing, and Food, respectively. We further sample non-complementary sessions to ensure a total of 1,000 sessions in each domain for the second batch while maintaining a higher ratio of complementary sessions.

3.2.5 Annotation Aggregation. In the above process, semantically similar intents are detected using hierarchical clustering. We use our manually-checked data to find the optimal thresholds for grouping intents, which results in an accuracy of 90%, 92%, and 96% for the three domains.

3.3 Resulting Datasets

After aggregation, we analyze the wording patterns of intents labelled by the workers according to POS tagging.[5] Distribution of the wording patterns is shown in Figure 5, with examples of the patterns shown in Table 1. Note that we pre-process all intents by removing stopwords and conducting lemmatization. The results show that our task allows us to collect a diverse set of intent patterns; in comparison, related studies such as [7] only consider two intent patterns, activity (verb) and audience (noun). Table 2 presents the statistics of our collected datasets.[6] We also analyze

[5] www.nltk.org/api/nltk.tag.html
[6] Our datasets are available at https://github.com/BundleRec/bundle_recommendation

Table 2. Statistics of Bundle Datasets

	Electronic	Clothing	Food
#Users	888	965	879
#Items	3,499	4,487	3,767
#Sessions	1,145	1,181	1,161
#Bundles	1,750	1,910	1,784
#Intents	1,537	1,590	1,323
Average Bundle Size	3.52	3.31	3.58
#User-Item Interactions	6,165	6,326	6,395
#User-Bundle Interactions	1,753	1,912	1,785
Density of User-Item Interactions	0.20%	0.15%	0.19%
Density of User-Bundle Interactions	0.11%	0.10%	0.11%

Fig. 6. (a) Difficulty feedback of two batches; (b) General feedback for the first batch; and (c) General feedback for the second batch.

worker feedback to our annotation task and present the results in Figure 6. Results show that more than 80% of the workers believe the task has an easy or medium difficulty; only around 10% of them feel it is hard to complete the task. Besides, we also obtain numerous positive feedback comments, e.g., good/interesting survey, and enjoyable/fun/easy task, shown in Figures 6(b)–(c). As a final remark, our data is contributed by workers with diverse backgrounds, where Figure 7 shows the distribution of their basic information.[7]

3.4 Related Datasets and Discussion

Comparison with Related Datasets. The dataset most akin to ours is iFashion [14], featuring outfits created by Taobao's fashion experts. A detailed comparison is presented in Table 3. Compared to iFashion, our datasets hold three advantages. (1) Ours contain richer side information, e.g., bundle intents and session records, fostering exploration into diverse tasks. For instance, the bundle intents and semantic category information enable research into explainable bundle recommendation, facilitating advancements in explainable AI. Additionally, session data supports novel tasks like bundle detection. (2) Ours span more diverse domains, revealing comprehensive product bundling patterns across various sectors and enabling studies with enhanced generalizability. (3) Ours involve more direct preference signals for accurate user modelling, as the interaction type between users and bundles is *purchase*; whereas iFashion considers the *click* behaviors. Studies have shown that *purchase* serves as stronger indicators of user preferences compared to *click* [55]. However, our datasets also exhibit limitations compared to iFashion. (1) iFashion surpasses our dataset in scale, recording 127,169 interactions among 3,569,112 users and 1,013,136 outfits, with 4,463,302

[7]In order to maintain the conciseness of the figures, we have abbreviated the occupation. The specific names are provided as follows: 'O & AS' means 'Office and administrative support', 'B & F' means 'Business and financial', 'T & MM' means 'Transportation and material moving', 'IM & R' means 'Installation, maintenance and repair', 'C & E' means 'Construction and extraction' and 'F & F' means 'Farming, fishing and forestry'.

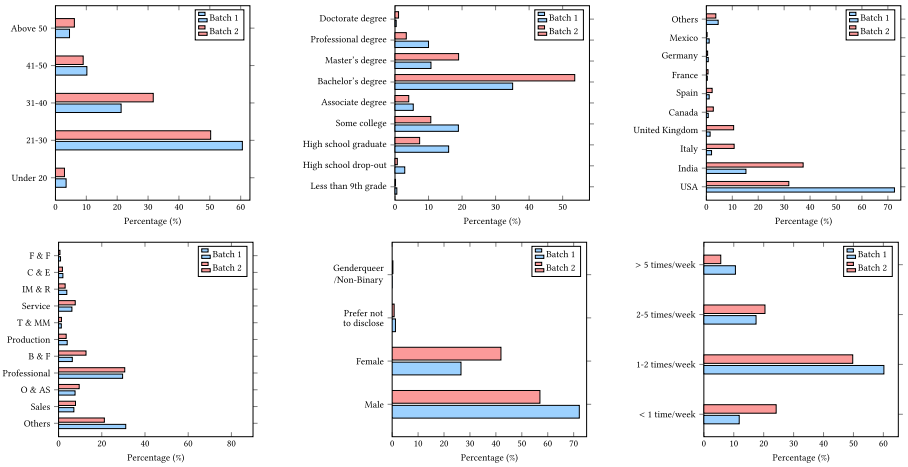


Fig. 7. Worker basic information in the first and second batch. (a) Distribution of age; (b) Distribution of education; (c) Distribution of country; (d) Distribution of occupation; (e) Distribution of gender; and (f) Distribution of shopping frequency.

Table 3. The Comparison between iFashion and Our Constructed Bundle Datasets

	Title	Image	Category	Intent	Session	Avg. Size	Behavior	Domain	Annotator
iFashion	✔	✔	✔ (ID)	✘	✘	3.86	Click	Clothing	Expert
Bundle	✔	✔	✔ (Text)	✔	✔	3.31∼3.58	Purchase	Electronic, Clothing, Food	Worker

items involved. (2) iFashion may possess higher-quality bundles due to expert annotations, while ours rely on crowdsourcing for labeling [50].

Further Discussion. While our current constructed datasets are relatively small in scale, they represent an invaluable *starting point* for revisiting bundle recommendation. Notably, we identify limitations in existing bundle datasets, often defined by unverified heuristics such as treating co-purchased products or user-generated lists as bundles. To overcome these limitations, we meticulously design crowdsourcing tasks aimed at creating high-quality bundle datasets with enriched metadata as summarized in Table 3. These datasets aim to catalyze investigations into numerous new tasks beyond bundle recommendation, such as bundle detection and explanation, benefiting both academia and industry within this domain. To augment their sizes, various potential strategies exist, for instance: (1) manually labeling bundles through crowdsourcing tasks, (2) automatic annotation using LLMs, or (3) a combined approach leveraging both strategies to complement each other. Exploring these possibilities further remains a focus for our future investigations.

4 THE PROPOSED INTERRELATED TASKS

In this section, we first formulate interrelated tasks in the context of product bundling and then explore the corresponding solutions with the goal of answering 19 raised research questions, summarized in Table 4.

4.1 Task Formulation

As analyzed in Section 2, existing studies mainly focus on bundle recommendation. In fact, a number of essential tasks need to be in place, e.g., detecting bundles from user sessions, to support bundle recommendation. Based on our collected datasets, we propose a series of interrelated tasks (Figure 2), described below in detail.

Table 4. The Interrelated Tasks and Corresponding 19 Research Questions Raised

Tasks	Research Questions
Detection	1. Does category-level pattern mining help detect high-quality bundles from user sessions?
	2. Does category-level pattern mining help generate new bundles?
Completion	3. Does neural-based bundle completion methods defeat traditional ones?
	4. Which operation (mean-pooling or concatenation) performs better regarding VAE-based methods?
	5. Does pre-training help achieve better bundle completion?
Ranking	6. Do neural-based bundle ranking methods outperform traditional ones?
	7. Do self-attention mechanisms facilitate better bundle ranking?
	8. Is it sufficient to model user-bundle interactions only for accurate bundle ranking?
	9. Does bundle ranking benefit from the pre-trained item representations?
Generation Explanation	10. Does bi-directional architecture outperform a single-directional one?
	11. Does Transformer perform better than LSTM?
	12. Do pre-trained language models boost the performance?
	13. Do all methods perform consistently in automatic and human evaluation?
Ranking Explanation	14. Do KG based methods outperform non-KG based ones?
	15. Do all methods achieve consistent performance regarding automatic and human evaluation?
	16. What do the recommendation and explanations generated by each method look like (qualitatively)?
Auto-Naming	17. What is the overall quality of bundle names generated by different methods, including ImageCap and LLMs (quantitatively)?
	18. Is there any difference in bundle names generated by various LLMs (i.e., Bard and ChatGPT)?
	19. What do the bundle names generated by different methods look like (qualitatively)?

Definition 4.1 (Bundle Detection). With the boom of e-commerce, a tremendous number of users shop online with e-commerce platforms (e.g., Amazon) and leave their footprints (e.g., view, click, and purchase products) in their daily lives. During a short period (i.e., a session), users are more likely to explore highly correlated (either alternative or complementary) products with certain intents [82]. That is to say, these user-generated session data could provide rich data sources for generating high-quality and more diverse product bundles. Hence, one essential task is *bundle detection*, viz., given user session data, it aims to efficiently detect the potential bundles.

Definition 4.2 (Bundle Completion). Given the detected bundles, *bundle completion* aims to add more relevant products with the same intent for bundle expansion. It can serve a wide range of applications via expanding bundles in an automatic fashion. For instance, if the system detects a user browsing dress and heels in a session, this could form a bundle. With bundle completion, it can quickly further enrich it with, e.g., jewelry, to broaden user interest.

Definition 4.3 (Bundle Ranking). Given the completed bundles, *bundle ranking* aims to rank these bundles according to user preference and demand for final recommendation. For instance, given a user session, several potential bundles (e.g., bundles for party clothing and cooking) may be produced via bundle detection and completion shown in Figure 2. In this case, bundle ranking will assist in sorting bundles on the basis of user interest, thus best satisfy user needs.

Definition 4.4 (Bundle Generation Explanation). Given the completed bundles, *bundle generation explanation* is to infer the corresponding intents (topics). For example, given a bundle of products in Figure 1(a), it can interpret as 'buying party clothing'. In doing so, it helps gain a better understanding of the rationale behind product bundling.

Definition 4.5 (Bundle Ranking Explanation). Given the ranked (recommended) bundles, *bundle ranking explanation* aims to provide explanations for the recommendation, e.g., the reason for

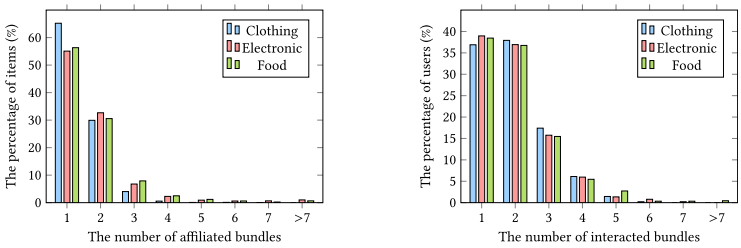


Fig. 8. (a) item-bundle distribution; (b) user-bundle distribution.

recommending the food bundle in Figure 2 is that 'the user purchased another food bundle recently'. This would help enhance system transparency and increase user trust.

Definition 4.6 (Bundle Auto-Naming). Given the completed bundles and inferred intents, *bundle auto-naming* seeks to automatically generate attractive bundle names. For example, regarding the food bundle in Figure 1(c), it could generate fancy names, e.g., 'Stay Safe, Stay Home'. In this sense, bundle auto-naming would extensively save human labour on manual naming and promote product bundling in real e-commerce platforms.

4.2 Task Exploration

We now explore solutions for each defined task. Instead of proposing new methods, our goal is to gain a better understanding of these tasks by leveraging and adapting existing techniques (including prevalent LLMs, e.g., ChatGPT [49]), and critically examining and analyzing their strengths and weaknesses. To guide our analysis, we raise a number of research questions; by answering these questions, we aim to point out potential research challenges and opportunities, as well as to provide insights and recommendations for future studies.

4.2.1 Bundle Detection. Given a user session, bundle detection aims to discover potential bundles within the session. This inspires us to mine frequent patterns on our collected bundles, and then perform pattern matching on user sessions to identify potential bundles. However, the data sparsity issue makes it impractical to directly mine patterns at the item level, viz., more than 50% of items belong to one bundle in the three domains as shown in Figure 8(a). As such, we consider pattern mining at the item category level. Specifically, we first map items in each bundle (e.g., $b_1 = \{i_1, i_2, i_3\}$) into the corresponding categories (e.g., $b_1 = \{c_1, c_2, c_3\}$), and then employ the *a priori* algorithm [1] to discover frequent patterns (e.g., $p_1 = \{c_1, c_2\}$ or $p_2 = \{c_1, c_2, c_3\}$) for bundling products with the two core parameters, namely *support* and *confidence*.

As patterns are mined at the category level, some patterns may only involve one category (e.g., $p_3 = \{c_1, c_1\}$), indicating alternative products inside a bundle; meanwhile, the order of categories within each pattern does not matter, viz., we do not consider antecedent and consequent. Consequently, we adapt the original support and confidence in the *a priori* algorithm to seamlessly accommodate the unique properties in our study, given by,

$$Sup(p = \{c_1, c_2, \cdots, c_n\}) = \frac{count(p = \{c_1, c_2, \cdots, c_n\})}{|\mathcal{B}|},$$

$$Con(p = \{c_1, c_2, \cdots, c_n\}) = \frac{count(p = \{c_1, c_2, \cdots, c_n\})}{\sum_{k=1}^{n} count(c_k)},$$

where $|\mathcal{B}|$ is the total number of bundles; $count(p)$ is the frequency of all categories within pattern p co-occurring in $\mathcal{B}$; n is the number of categories in p; $count(c_k)$ is the frequency of category c_k

appearing in $\mathcal{B}$, and is only counted once within a bundle even if it appears multiple times within that bundle. As such, a pattern is valid only if its *support* and *confidence* are no less than the predefined thresholds. With such valid patterns, we can perform pattern matching on any given user sessions for bundle detection.

In this task, we aim to answer two research questions: (**RQ1**) does category-level pattern mining help detect high-quality bundles from user sessions? and (**RQ2**) does category-level pattern mining help generate new bundles?

4.2.2 Bundle Completion. Given partial products in a bundle as context, bundle completion seeks for more relevant products to further expand the bundle. However, one major challenge confronted is: a considerable number of items only appear in one bundle as shown in Figure 8(a), which aggravates the data sparsity issue and increases the difficulty of this task. To combat this challenge, we design a strategy to obtain pre-trained item representations via BPRMF [59] for model initialization, the details of which are deferred to 'Our Strategy for Pre-training' at the end of this section. With such pre-trained item representations, we ultimately adapt seven methods for this task.

(1) *ItemKNN* [61] calculates item similarity via pre-trained item representations, where the most similar items with the ones in the bundle are selected for completion. Note that we do not calculate item similarity via the bundle-item affinity matrix due to its high sparsity. (2) *BPRMF* [59] factorizes the item-bundle affinity matrix to learn item and bundle representations and then selects the most similar items with the target bundle for completion. (3) *mean-VAE* [41] is inspired by VAE-CF [41], which takes the averaged representation of existing items in a bundle as the input and then selects the items with the highest probability for completion. (4) *concat-VAE* [41] is similar to mean-VAE where the only difference is the input, viz., concatenating representations of existing items in a bundle as input. Due to varied bundle sizes, we align the input dimension by the padding operation. (5) *mean-CVAE* [63] upgrades mean-VAE by averaging the embedding of bundle intent and the learned latent variable $\mathbf{z}$ in VAE. It aims to leverage the bundle intent as a condition to guide the bundle completion task. Note that the embedding of bundle intent is generated through SentenceTransformers,[8] which is a Python framework for state-of-the-art sentence, text and image embeddings [58]. (6) *concat-CVAE* [63] upgrades concat-VAE by concatenating the embedding of bundle intent with the learned latent variable $\mathbf{z}$ in VAE. (7) ***TSF (Transformer)*** [19] is a BERT-like model inspired by [32], which adopts the Transformer encoder to represent existing items within a bundle and predict the most suitable item for bundle completion.

Accordingly, in this task, we answer three research questions: (**RQ3**) does neural-based bundle completion methods defeat traditional ones? (**RQ4**) which operation (mean-pooling or concatenation) performs better regarding VAE-based methods? and (**RQ5**) does pre-training help achieve better bundle completion?

4.2.3 Bundle Ranking. Given a number of completed bundles, bundle ranking aims to sort them based on user preference for recommendation. It is still facing the data sparsity issue, as most users only interact with a small number of bundles as shown in Figure 8(b). To ease this issue, the pre-trained item representations are also unitized for model initialization. As such, we examine the performance of six state-of-the-art methods on bundle ranking.

(1) *ItemKNN* [61] calculates bundle similarity with bundle representations, viz., the averaged representation of items in the bundle. Note that we do not calculate bundle similarity via the user-bundle interaction matrix due to its high sparsity. (2) *BPRMF* [59] factorizes the user-bundle interaction matrix to learn user and bundle representations. (3) *DAM* [13] designs a factorized attention

[8] https://www.sbert.net/

network to aggregate item representations in a bundle for bundle representation and jointly models user-bundle and user-item interactions in a multi-task manner. (4) *AttList* [30] learns user and bundle representations via the self-attention mechanism, whereby it aggregates items to characterize the bundle that they belong to, and then aggregates bundles to estimate user preference. (5) *GCN* applies the graph convolutional network [84] on the user-bundle-item tripartite graph to learn user and bundle representations. (6) *BGCN* [11] constructs a heterogeneous graph by unifying user-item, user-bundle interactions, and item-bundle affinity, and then applies propagation at both item and bundle levels to learn user and bundle representations. It is noteworthy that we do not consider EFM-Joint [9], as the source code for constructing its PMI matrix is not available and our re-implementation does not achieve promising results.

Given these methods, we answer four research questions: (**RQ6**) do neural-based bundle ranking methods outperform traditional ones? (**RQ7**) do self-attention mechanisms facilitate better bundle ranking? (**RQ8**) is it sufficient to model user-bundle interactions only for accurate bundle ranking? and (**RQ9**) does bundle ranking benefit from the pre-trained item representations?

4.2.4 Bundle Generation Explanation. Given the completed bundles, the goal of bundle generation explanation is to describe the corresponding intents, thus uncovering the rationale behind product bundling. As each bundle has its well-labeled intents by the workers, and the titles of all items inside each bundle are available, we adapt the seq2seq models to generate intents for bundles with item titles as input. We consider the following six state-of-the-art approaches.

(1) *LSTM* [33] uses LSTM as encoder and decoder; (2) ***BiLSTM*** [62] adopts **bidirectional LSTM** as encoder and LSTM as decoder; (3) *Transformer* (TSF) [78] is a seq2seq model considering specifically the relationship between a given word and the context via self-attention; (4) ***BertGeneration*** **(BertG)** [60] is a Transformer-based model where its parameters (221M) of encoder and decoder are initialized with BERT, while the self-attention mechanism in the decoder is masked to look only at the left context; (5) *BART* [38] is a Transformer-based architecture with a bidirectional encoder trained over corrupted text and a left-to-right autoregressive decoder (we use the version with 120M parameters); and (6) *T5* [57] is also a Transformer-based seq2seq model, pre-trained on a multi-task mixture of unsupervised and supervised tasks all converted to the text-to-text format (we use the version with 220M parameters). We adopt the tokenizer provided in torchtext[9] to process item titles for the first three methods; for the last three pre-trained models with their own vocabularies, we use the built-in tokenizers in the Hugging Face implementations.

In this task, we seek to answer four research questions: (**RQ10**) does bi-directional architecture outperform a single-directional one? (**RQ11**) does Transformer perform better than LSTM? (**RQ12**) do pre-trained language models boost the performance? and (**RQ13**) do all methods perform consistently in automatic and human evaluation?

4.2.5 Bundle Ranking Explanation. The goal of bundle ranking explanation is to explain the ranking of the recommended bundles. As introduced in Section 2.2, explicit features (e.g., product attributes or reviews) are generally required for this task. Therefore, in our study, we extract features for each bundle from its item titles and intent via the natural language processing toolkit - NLTK.[10] Accordingly, we adopt and adapt four classic explainable recommenders for this task.

(1) *RM* is the rule mining based method, which uses association rule learning to mine frequently co-occurred bundle features to rank bundles and provide explanations accordingly; (2) *EFM* [95] is the factorization based method, which decomposes user-feature and bundle-feature matrices to infer user preference and generate feature-level explanations; (3) *PGPR* [87] and (4) *KGAT* [83] are

[9] pytorch.org/text/stable/index.html
[10] www.nltk.org/

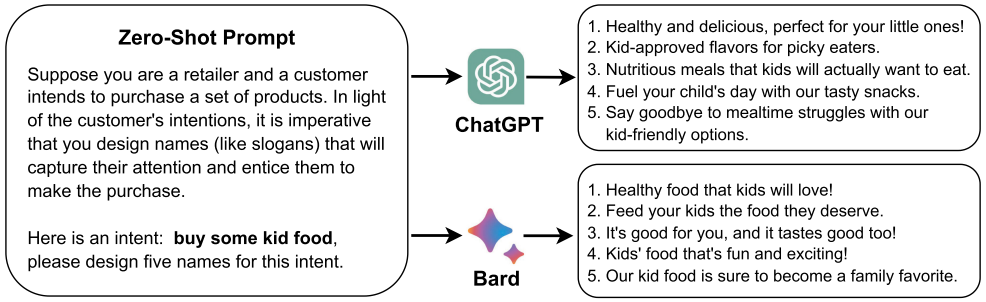


Fig. 9. The example of zero-shot prompt and corresponding responses.

KG based methods, which mine connected paths between users and bundles from KGs as explanations. Particularly, there are respectively three types of entities (user, bundle, and feature) and relations (purchase, tagged_by and described_by) in the KG of PGPR, where *purchase* means the user purchases the bundle; *tagged_by* means the user is tagged by the feature of interacted bundles; and *described_by* means the bundle is described by the feature. For KGAT, three types of entities (user, bundle, and feature) and two types of relations (purchase and described_by) are involved in the KG. Note that the bundle review information is usually not available as in our study, thus we do not consider seq2seq models in this task.

In this task, we seek answers to the following three research questions: (**RQ14**) do KG based methods outperform non-KG based ones? (**RQ15**) do all methods achieve consistent performance regarding automatic and human evaluation? and (**RQ16**) what do the recommendation and explanations generated by each method look like (qualitatively)?

4.2.6 Bundle Auto-Naming. Given the completed bundles and inferred intents, bundle auto-naming is to generate attractive bundle names. Given the limited research that has been done in the context of recommendation, we adapt state-of-the-art approaches in CV and NLP to fulfill this task.

In particular, we explore and exploit two types of methods, including **image captioning (ImageCap)** models and large language models (LLMs). To be specific, *ImageCap* initializes an image-to-text model by utilizing the pre-trained transformer-based vision model as the encoder and the pre-trained transformer-based language model as the decoder, which can generate names using item images in a bundle as input. To implement ImageCap, we utilize the ViT [20] as the encoder and the GPT-2 [56] as the decoder with the VisionEncoderDecoderModel of the Hugging Face. The second type mainly leverages abundant knowledge and superior inference capability of LLMs, making LLMs understand the description of the task and generate names for the given bundles by designing suitable prompts (instructions). Two LLMs respectively developed by OpenAI and Google (i.e., *ChatGPT* [49] and *Bard* [27]) are considered in our study. In order to make these LLMs understand the bundle auto-naming task, we design the zero-shot prompt for generating names for given bundles: *Suppose you are a retailer and a customer intends to purchase a set of products. In light of the customer's intentions, it is imperative that you design names (like slogans) that will capture their attention and entice them to make the purchase. Here are {{intentions}}, please design five names for each intention.* The {*intentions*} is the variable representing the bundle intents, which should be injected when generating bundle names. In addition, we randomly sample one from the five generated names for each bundle to ensure the diversity of generated names. Figure 9 demonstrates an example of the zero-shot prompt and the corresponding responses from ChatGPT and Bard.

Table 5. Statistics of Datasets for Pre-training

	#Users	#Items	#Interactions
Electronic	745,911	178,443	1,951,397
Clothing	223,571	236,387	749,423
Food	143,390	54,323	325,169

In this task, our goal is to answer three research questions: (**RQ17**) what is the overall quality of bundle names generated by different methods, including ImageCap and LLMs (quantitatively)? (**RQ18**) is there any difference in bundle names generated by various LLMs (i.e., Bard and ChatGPT)? (**RQ19**) what do the bundle names generated by different methods look like (qualitatively)?

4.2.7 Our Strategy for Pre-training. As observed in Table 2, the user-bundle interaction matrix is quite sparse. To ensure the quality of learned representations, we adopt BPRMF to pre-train item representations to initialize all methods; meanwhile the bundle representation is initialized with the averaged representation of items inside this bundle. To avoid data leakage, we sample extra user-item interactions from each domain with the same time span as the bundle collection (i.e., July 2013 - July 2014) for pre-training. In particular, for each item in Table 2, we first find out all users who have interacted with it excluding those in Table 2, and then leverage these users' records to construct the user-item interaction matrix for pre-training. By doing so, the constructed interaction data covers all items but excludes all users in Table 2, therefore, they do not leak any interactions in our collected data in Table 2. The statistics of datasets for pre-training across the three domains are shown in Table 5, where 80% of the interactions in each domain are treated as training data, and the rest as test data. The model is trained until optimal performance is observed on the test data regarding NDCG@10 to obtain better item representations.

5 INSIGHTS, CHALLENGES AND OPPORTUNITIES

In this section, we aim to answer the 19 research questions raised in Section 4.2, thereby providing insights, and showing both research challenges and opportunities w.r.t. the defined tasks.

5.1 Bundle Detection

5.1.1 Experimental Setup. We use the three collected datasets in Table 2. For each dataset, every user session corresponds to its ground-truth bundles labelled by the workers. We divide the sessions into training, validation and test sets with the ratio of 8:1:1, where the training set is used to mine frequent patterns with size ranging from 2 to 5; the validation set is used to tune the hyper-parameters, namely *support* and *confidence* for best performance. Specifically, we apply a grid search in the range of $\{0.0001, 0.001, 0.01\}$ to find out the optimal parameter settings, and both are set as 0.001 across the three domains. Finally, we apply the valid patterns obtained from the training set to sessions in the test set for bundle detection.

We evaluate our method proposed in Section 4.2 at both session and bundle levels. In particular, at the session level, we adopt precision and recall to measure how many patterns have been correctly predicted for each session, given by,

$$Precision = \frac{1}{|\mathcal{S}_t|} \sum_{s \in \mathcal{S}_t} \frac{\#hit\_patterns}{\#predicted\_patterns},$$

$$Recall = \frac{1}{|\mathcal{S}_t|} \sum_{s \in \mathcal{S}_t} \frac{\#hit\_patterns}{\#ground\_truth\_patterns},$$

where $\#ground\_truth\_patterns$ is the number of ground-truth patterns for each session s; $\mathcal{S}_t$ is all sessions in the test set; and $\#hit\_patterns$ is the number of correctly predicted patterns that match

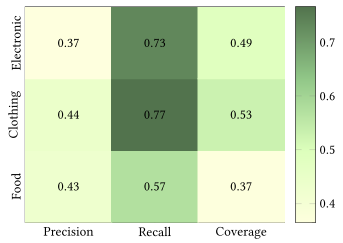


Fig. 10. Performance on bundle detection.

Fig. 11. The detected bundles from sessions.

ground-truth patterns (subsets included) for each session. For each session, if multiple predicted patterns are all subsets of a same ground-truth pattern, we only retain the one with the maximum size. At bundle level, we utilize coverage to measure how many categories are correctly covered by each predicted bundle pattern compared with the ground-truth pattern, given by,

$$Coverage = \frac{1}{\sum_{s \in \mathcal{S}_t} |\mathcal{B}_s|} \sum_{s \in \mathcal{S}_t} \sum_{b \in \mathcal{B}_s} \frac{\#hit\_categories}{\#ground\_truth\_categories},$$

where $\mathcal{B}_s$ is the set of predicted bundle patterns for session s; $\#ground\_truth\_categories$ is the number of categories in the ground-truth pattern for bundle b; and $\#hit\_categories$ is the number of correctly predicted categories, i.e., categories included in the ground truth pattern.

5.1.2 Analysis and Insights. (**RQ1**) As reported in Figure 10, the higher recall indicates that our method is able to identify majority ground-truth bundles hidden in the sessions, whereas the relatively low coverage suggests that not all categories within ground-truth bundles are discovered, showing the necessity of bundle completion. Figure 11 visualizes six sessions and the corresponding detected bundles across the three domains, where we can find that our method can help detect interesting patterns with complementary products, e.g., [‘Tablets’, ‘Cases’], [‘Jeans’, ‘Shoes’, ‘T-shirts’] and [‘Cleaning & Repair’, ‘Watches’]. (**RQ2**) It is easy to infer only partial predicted patterns match the ground truth according to the results on Precision. This naturally leads us to ask a question: *are the predicted but non-hit bundle patterns reasonable*? To this end, we randomly sample 30 such patterns from each domain where three examples per domain are visualized in Figure 12. To quantitatively examine the quality of the non-hit patterns, we invite 15 annotators in total and ensure each pattern with 5 annotators assigned to check whether these non-hit patterns are reasonable. After that, the voting method is adopted to aggregate the 5 annotations for

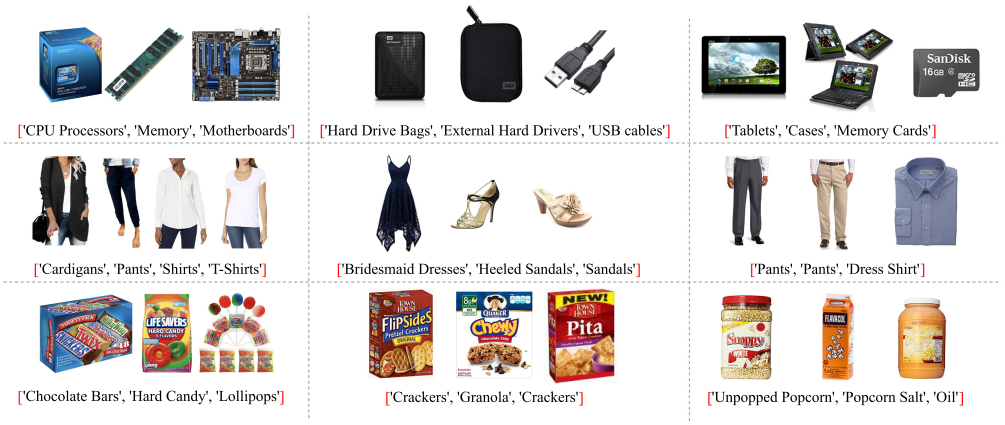


Fig. 12. Case study of non-hit bundle patterns.

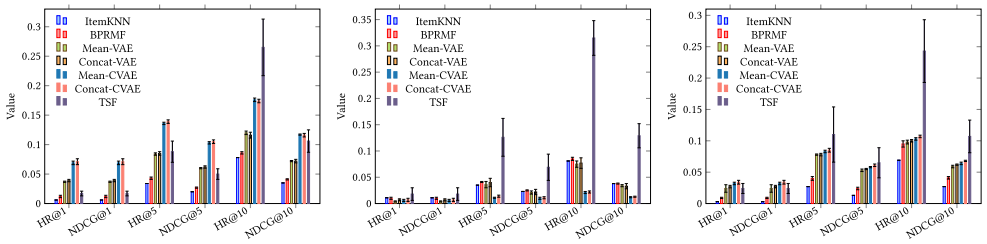
each pattern, that is, if more than 3 annotators provide positive scores, then the pattern is considered as reasonable one. Overall, we find that 73.3%, 93.3%, and 90.0% of them are reasonable bundles in Electronic, Clothing and Food, respectively. This demonstrates such pattern mining indeed facilitates the generation of high-quality new bundles.

5.2 Bundle Completion

5.2.1 Experimental Setup. For each domain, we split the bundles in Table 2 into training, validation, and test sets with the ratio of 8:1:1. In our current study, we consider the scenario, viz., removing one item from each bundle and deem the rest as context. Consequently, the goal is to complete the missing item based on the context for each bundle as accurately as possible. For a more solid evaluation, we expand our test set by taking one different item from a bundle as the missing one, and the rest as context each time. As such, one test bundle b will be expanded to $|b|$ test samples. To improve test efficiency, for each test bundle, we pair 99 randomly-sampled items with its missing (ground-truth) item, and then rank the 100 items for completion. We adopt HR@N and NDCG@N as the evaluate metrics. Generally, higher metric values indicate better ranking accuracy. We test each method ten times and report the results in the format of $(mean \pm std)$.[11]

5.2.2 Analysis and Insights. Figure 13 presents the results of bundle completion. Overall, as the size of the recommendation list (N) increases, the values of HR and NDCG consistently go up for all methods across the three domains. This is natural as a larger N enhances the probability of correct items being selected for completion. (**RQ3**) First, comparing different methods, the factorization-based BPRMF outperforms the memory-based ItemKNN, whilst being defeated by the neural-based VAE and TSF in most cases, suggesting the superiority of deep learning based methods on capturing bundle context for more accurate completion. Second, the performance of concat-CVAE exceeds that of concat-VAE on Electronic and Food, indicating the usefulness of bundle intents in guiding better bundle completion. However, an opposite scenario is held on Clothing, where concat-VAE significantly defeats concat-CVAE. The possible reason is that the dimension of the bundle intent embeddings generated by SentenceTransform is 384, whereas the dimension of the latent variable $\mathbf{z}$ is only 50 (note that 50 is the optimal value using grid search). Therefore, directly concatenating the embedding of bundle intent and $\mathbf{z}$ could potentially diminish the impact of $\mathbf{z}$, rendering it

[11]For the sake of reproducibility [70, 71], the optimal parameter settings for different methods regarding all tasks are reported in our GitHub Repo.



(a) Electronic (b) Clothing (c) Food

Fig. 13. Performance on bundle completion, where the short line on each bar denotes the standard deviation.

Table 6. Performance on Bundle Ranking Task in the Domain of Electronic, where the Best Results are Highlighted in Bold; and '-Pre' Means without Pre-training

	HR@1	HR@5	HR@10	NDCG@1	NDCG@5	NDCG@10
ItemKNN	0.032±0.000	0.084±0.000	0.154±0.000	0.032±0.000	0.058±0.000	0.081±0.000
BPRMF	0.026±0.005	0.088±0.015	0.141±0.013	0.026±0.005	0.057±0.009	0.074±0.007
BPRMF-pre	0.001±0.002	0.014±0.005	0.043±0.013	0.001±0.002	0.008±0.003	0.017±0.005
DAM	0.013±0.009	0.076±0.027	0.152±0.035	0.013±0.009	0.044±0.018	0.068±0.018
DAM-pre	0.001±0.001	0.005±0.005	0.015±0.008	0.001±0.001	0.003±0.002	0.006±0.003
AttList	0.051±0.045	0.134±0.092	0.171±0.107	0.051±0.045	0.094±0.068	0.106±0.073
AttList-pre	0.022±0.012	0.069±0.032	0.130±0.038	0.022±0.012	0.046±0.021	0.066±0.024
GCN	0.133±0.029	0.300±0.038	0.382±0.038	0.133±0.029	0.219±0.032	0.245±0.031
GCN-pre	0.067±0.021	0.211±0.043	0.316±0.050	0.067±0.021	0.141±0.031	0.174±0.032
BGCN	**0.179±0.042**	**0.388±0.052**	**0.522±0.044**	**0.179±0.042**	**0.287±0.045**	**0.331±0.042**
BGCN-pre	0.065±0.016	0.176±0.025	0.282±0.035	0.065±0.016	0.122±0.017	0.156±0.018

overshadowed by the dominant bundle intent embedding. To address this issue, one potential solution involves introducing additional neural layers aimed at reducing the dimensionality of the bundle intent embedding. We leave this aspect as an area in our future exploration. Third, TSF generally outperforms VAE across all datasets, revealing the superior capability of Transformer in the bundle completion task. (**RQ4**) Regarding the VAE-based methods, both concat-VAE and concat-CVAE respectively perform better than their counterparts, mean-VAE and mean-CVAE, which indicates concatenation can preserve more useful information for bundle completion. (**RQ5**) Figure 13 only reports the results of all compared methods with pre-trained representations, which far exceed those without pre-trained representations. This helps confirm the usefulness of pre-training for the bundle completion task.

5.3 Bundle Ranking

5.3.1 Experimental Setup. We adopt the user-bundle interaction data in Table 2. Leave-one-out is used for evaluation, viz., for each user, we randomly hold out one bundle for validation, one bundle for testing, and the rest for training. To boost test efficiency, we pair 99 randomly-sampled bundles that are not purchased by the target user with the ground-truth bundle, and rank the 100 bundles for recommendation. HR@N and NDCG@N are adopted as the evaluation metrics. We test each method ten times and report the results in the format of ($mean \pm std$).

5.3.2 Analysis and Insights. Tables (6–8) present the results of bundle ranking, where several interesting findings can be noted. (**RQ6**) ItemKNN, though simple, outperforms BPRMF and even the attention-based DAM. This is in line with the conclusion drawn in [17], where traditional

Table 7. Performance on Bundle Ranking Task in the Domain of Clothing

	HR@1	HR@5	HR@10	NDCG@1	NDCG@5	NDCG@10
ItemKNN	0.021±0.000	0.074±0.000	0.139±0.000	0.021±0.000	0.049±0.000	0.070±0.000
BPRMF	0.011±0.004	0.055±0.009	0.107±0.013	0.011±0.004	0.033±0.006	0.049±0.006
BPRMF-pre	0.002±0.002	0.015±0.006	0.048±0.013	0.002±0.002	0.008±0.004	0.018±0.006
DAM	0.017±0.021	0.075±0.054	0.167±0.066	0.017±0.021	0.046±0.037	0.075±0.041
DAM-pre	0.000±0.000	0.001±0.002	0.002±0.002	0.000±0.000	0.000±0.001	0.001±0.001
AttList	0.030±0.025	0.085±0.049	0.138±0.066	0.030±0.025	0.057±0.037	0.074±0.042
AttList-pre	0.027±0.009	0.073±0.014	0.135±0.024	0.027±0.009	0.050±0.009	0.070±0.012
GCN	0.182±0.015	0.336±0.020	0.431±0.024	0.182±0.015	0.262±0.017	0.293±0.018
GCN-pre	0.081±0.022	0.233±0.052	0.350±0.037	0.081±0.053	0.158±0.037	0.196±0.044
BGCN	**0.255±0.033**	**0.447±0.047**	**0.568±0.059**	**0.255±0.033**	**0.354±0.038**	**0.394±0.043**
BGCN-pre	0.058±0.011	0.151±0.016	0.242±0.021	0.058±0.011	0.104±0.013	0.134±0.014

Table 8. Performance on Bundle Ranking Task in the Domain of Food

	HR@1	HR@5	HR@10	NDCG@1	NDCG@5	NDCG@10
ItemKNN	0.055±0.000	0.133±0.000	0.188±0.000	0.055±0.000	0.095±0.000	0.113±0.000
BPRMF	0.016±0.004	0.081±0.012	0.141±0.019	0.016±0.004	0.049±0.007	0.068±0.008
BPRMF-pre	0.001±0.001	0.019±0.007	0.059±0.014	0.001±0.001	0.010±0.003	0.022±0.005
DAM	0.012±0.011	0.077±0.039	0.163±0.046	0.012±0.011	0.043±0.025	0.071±0.026
DAM-pre	0.001±0.002	0.003±0.004	0.004±0.004	0.001±0.002	0.002±0.002	0.002±0.002
AttList	0.058±0.045	0.122±0.079	0.169±0.094	0.058±0.045	0.091±0.064	0.107±0.068
AttList-pre	0.025±0.011	0.089±0.029	0.149±0.040	0.025±0.011	0.057±0.020	0.076±0.023
GCN	0.170±0.022	0.336±0.026	0.439±0.026	0.170±0.022	0.257±0.020	0.291±0.020
GCN-pre	0.066±0.023	0.205±0.067	0.314±0.085	0.066±0.023	0.136±0.044	0.171±0.049
BGCN	**0.201±0.064**	**0.405±0.066**	**0.526±0.053**	**0.201±0.064**	**0.306±0.064**	**0.345±0.060**
BGCN-pre	0.053±0.025	0.161±0.024	0.251±0.028	0.053±0.025	0.106±0.023	0.134±0.022

recommenders may defeat neural models with well-tuned parameters. (**RQ7**) Regarding attention-based methods, AttList achieves better performance than DAM due to (a) it learns hierarchical user preference via both item- and bundle-level aggregation; and (b) it adopts the self-attention mechanism to refine the item/bundle representation by fusing the consistency of neighboring items/bundles before aggregation. (**RQ8**) Graph-based neural methods (GCN and BGCN) perform better than others, and BGCN consistently achieves the best performance. It helps confirm (a) the necessity of capturing complex relations among users, items, and bundles with the heterogeneous graph; and (b) the efficacy of convolutional propagation on better representation learning. Such a compatible combination (i.e., graph as data structure and GCN as technical support) greatly assists in mitigating the scarcity of user-bundle interactions. (**RQ9**) Generally, pre-training helps ensure better performance, for example, the performance of BPRMF far exceeds that of BPRMF-pre.

5.4 Bundle Generation Explanation

5.4.1 Experimental Setup. For better model training, we merge bundles in the three domains in Table 2 and split them into training, validation, and test sets with the ratio of 8:1:1. The goal is to generate an intent for each bundle given its item titles. To comprehensively examine all methods, we perform both automatic and human evaluations. For automatic evaluation, we adopt the widely-used ROUGE [60] to evaluate n-grams of the generated intent with ground truth and report precision, recall, and F1 scores on Rouge-1, -2 and $-L$. For human evaluation, three metrics are used. Specifically, *Naturalness* indicates whether the intent is easy to read and understand with a 1-3 scale (1 - it is difficult to read and understand; 2 - it is fair to read and understand; and

Table 9. Automatic Evaluation on Generation Explanation

	ROUGE-1			ROUGE-2			ROUGE-L		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
LSTM	4.40	1.92	2.42	0.53	0.23	0.27	4.28	1.75	2.28
BiLSTM	7.99	4.40	5.21	0.83	0.50	0.59	7.93	4.33	5.14
TSF	17.59	16.92	15.78	4.24	3.83	3.83	17.41	16.81	15.64
BertG	26.36	22.36	22.32	6.62	5.98	6.03	25.99	22.06	22.03
BART	28.54	23.17	23.72	7.86	6.99	7.07	28.14	22.88	23.42
T5	**29.57****	**24.11*****	**24.39*****	**8.05****	**7.39****	**7.78****	**29.31*****	**23.87*****	**24.16****

The statistical significance of pairwise differences of the winner T5 vs. the runner-up BART is determined by a paired t-test (** for $p < 0.05$ and *** for $p < 0.001$).

3 - it is easy to read and understand). *Coverage* implies to what extent the items in the bundle can be covered by the intent with a 1-3 scale (1 - only a few items in the bundle are covered by the intent; 2 - around half items in the bundle are covered by the intent; 3 - most items in the bundle are covered by the intent). *Motivation* suggests whether the intent contains motivational description, i.e., describing the purpose of the bundle by activities, events, or actions. For example, the intent 'assembling computer' is motivational, whilst 'different computer accessories' is not. Its scale ranges from 1 to 2, where 1 refers to the intent containing no motivational description; whilst 2 means the intent contains a motivational description. In our study, we asked 15 participants to rate the intents generated by the workers and three representative methods (i.e., TSF, BART, and T5) for 300 bundles via the three metrics. To enhance the reproducibility of human evaluation and reduce labor, we investigate the effectiveness of LLMs in this task. Specifically, we initiate three ChatGPT agents, referred to as GPT-Agent, to evaluate the generated intents using the three metrics.

5.4.2 Analysis and Insights. Table 9 shows the bundle generation explanation performance regarding the automatic evaluation, where the best results are highlighted in bold.[12] **(RQ10)** BiLSTM outperforms LSTM, which confirms the advantages of bi-directional architecture over single-directional one. **(RQ11)** The results are largely improved via TSF compared with BiLSTM, verifying that the self-attention mechanism can better capture the relationship of words in the sequence. **(RQ12)** The three pre-trained language models (i.e., BertG, BART and T5) generally defeat TSF. This showcases the superiority of pre-trained language models on bundle generation explanation task. Besides, BART achieves higher accuracy than BertG, but falls behind T5, which is mainly attributed to two aspects: (1) BART and T5 are end-to-end and unified pre-trained seq2seq model, whereas BertG is simply initialized with BERT; and (2) T5 is a pre-trained model with more parameters (220M) compared with BART (around 120M), thus can better capture semantic features encoded in the sequence. Table 10 displays the corresponding results w.r.t. human and GPT-Agent evaluation. **(RQ13)** Overall, T5 outperforms TSF, BART, and even Workers on Naturalness and Coverage, indicating the consistent performance of all methods w.r.t. both automatic and human evaluation. However, the worse performance on Motivation achieved by T5 relative to Worker exhibits the superior ability of humans to provide motivational descriptions for bundle intent generation. Additionally, the evaluations offered by both humans and GPT-Agents exhibit similar trends across various metrics. For instance, BART outperforms TSF but lags behind T5, and Workers demonstrate superiority in Motivation. The sole discrepancy appears in Naturalness: humans

[12]The $F1$ score is in the range of $[\min(Pre, Rec)], \max(Pre, Rec)]$ for each user. However, the reported scores are the mean values across all users, thus the $F1$ scores may be out of that range in some cases.

Table 10. Human and GPT-Agent Evaluation on Generation Explanation, where the Statistical Significance of Pairwise Differences of T5 vs. Workers is Determined by a Paired t-test (** for $p < 0.05$ and *** for $p < 0.001$)

	Naturalness		Coverage		Motivation	
	Human	GPT-Agent	Human	GPT-Agent	Human	GPT-Agent
Intents annotated by Workers	2.52	**2.66****	2.39	2.38	**1.26*****	**1.77*****
Intents generated by TSF	2.46	2.02	2.11	1.82	1.13	1.44
Intents generated by BART	2.67	2.23	2.47	2.33	1.18	1.69
Intents generated by T5	**2.68*****	2.44	**2.48****	**2.39****	1.20	1.73

Table 11. Automatic Evaluation on Ranking Explanation

	Metrics	RM	EFM	PGPR	KGAT
Electronic	HR@5	0.017±0.000	0.011±0.008	0.309±0.013	**0.377±0.026**
	HR@10	0.027±0.000	0.020±0.010	0.430±0.013	**0.454±0.021**
	NDCG@5	0.010±0.000	0.006±0.004	0.221±0.008	**0.301±0.018**
	NDCG@10	0.014±0.000	0.009±0.004	0.260±0.007	**0.326±0.016**
Clothing	HR@5	0.025±0.000	0.006±0.003	0.321±0.010	**0.324±0.023**
	HR@10	0.034±0.000	0.015±0.005	**0.429±0.007**	0.389±0.024
	NDCG@5	0.016±0.000	0.003±0.002	0.221±0.009	**0.255±0.020**
	NDCG@10	0.019±0.000	0.007±0.003	0.256±0.008	**0.276±0.019**
Food	HR@5	0.019±0.000	0.010±0.006	0.296±0.014	**0.314±0.024**
	HR@10	0.029±0.000	0.021±0.005	**0.401±0.016**	0.384±0.019
	NDCG@5	0.017±0.000	0.005±0.003	0.186±0.009	**0.249±0.021**
	NDCG@10	0.020±0.000	0.009±0.002	0.220±0.011	**0.271±0.019**

believe T5 delivers the best performance, while GPT-Agents perceive Workers as the superior choice. This demonstrates the potential of LLMs in crowdsourcing tasks [50].

5.5 Bundle Ranking Explanation

5.5.1 Experimental Setup. We use the collected data in Table 2, and adopt leave-one-out as the evaluation protocol. Following [52, 80], we perform both automatic and human evaluation to examine the quality of ranking accuracy and generated explanation. In particular, automatic evaluation is used to measure the ranking accuracy, where HR@N and NDCG@N are applied as evaluation metrics. We test each method ten times and report the results in the format of ($mean \pm std$). Meanwhile, the human evaluation is to examine the quality of both ranking accuracy and generated explanation. Specifically, we randomly sample 10 cases from each domain (30 cases in total), where each case contains three parts: a user's historically purchased bundles, the recommended bundle, and the generated explanation of each method. Inspired by [80], we design a questionnaire that covers five questions for each method: (1) *satisfaction* - "are you satisfied with this recommendation?" (2) *easy to understand* - "is the explanation easy to understand?" (3) *transparency* - "does the explanation help you understand why the system recommends this bundle?" (4) *persuasiveness* - "does the explanation make the recommendation more convincing?" (5) *effectiveness* - "does the explanation help you make the decision (i.e., purchase the recommended bundle or not)?" Based on this, we then recruit 10 participants to take the survey, where each participant is asked to rate the 30 cases by answering every question on a 1-5 scale (i.e., strongly negative, negative, neutral, positive, and strongly positive).

5.5.2 Analysis and Insights. Table 11 reports the results of bundle ranking explanation w.r.t. automatic evaluation. (**RQ14**) Overall, KG based methods (i.e., PGPR and KGAT) perform better than non-KG based methods (i.e., RM and EFM), indicating that KGs allow to learn better

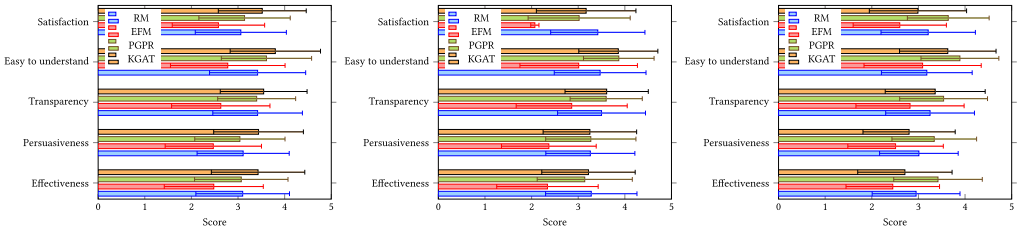


Fig. 14. Human evaluation on ranking explanation for the three domains, i.e., Electronic, Clothing and Food.

representations for users, bundles and features. For the non-KG based methods, the rule mining based RM surpasses the factorization based EFM. This may be attributed to the data sparsity issue, which leads to insufficient learning of user, bundle and feature representations by EFM. Regarding the KG based methods, the performance of KGAT surpasses that of PGPR. Possible reasons are: (1) PGPR leverages reinforcement learning to mine the connected path between user and target bundles, the performance of which is largely affected by the size of action space. Pruning the action space may filter out target bundles, whilst expanding the action space may increase the difficulty of reaching the target bundles; and (2) the extra user-feature connection in the KG of PGPR may also set another barrier to reach the target bundles. Furthermore, we also compare the ranking accuracy of bundle ranking methods (see Tables 6, 7 and 8) and that of ranking explanation methods (see Table 11). Note that the former methods are specially designed for bundle ranking; whereas the latter methods are originally proposed for individual item ranking explanation, and then adapted for the bundle ranking explanation task. As such, we observe that the winner for the bundle ranking task (i.e., BGCN) generally outperforms the one for the ranking explanation task (i.e., KGAT), which suggests room for further improving the ranking performance of bundle explanation approaches specifically designed for ranking explanation.

The performance w.r.t. human evaluation is presented in Figure 14, where we can get the following observations. (**RQ15**) As a whole, the KG-based methods (at least one method, either PGPR or KGAT) outperform the non-KG based methods on the five metrics across the three domains; among the non-KG based methods, RM outperforms EFM on all domains; and for the KG based methods, KGAT surpasses PGPR on Electronic and Clothing. These findings align with the ones obtained via automatic evaluation. However, an opposite observation is gained on Food domain, where PGPR performs better than KGAT. To this end, we review the 10 cases and find that the extracted features connecting the purchased and recommended bundles for KGAT are mainly related to product size (e.g., 6-pack, 16-ounce), which severely hurts the quality of the generated explanation paths – a recommendation for 6-pack K-cups coffee is meaningless if the only reason is because the user purchased 6-pack yogurt in history. This implies the necessity of using more advanced feature extraction tools and fusing more meaningful side information besides item titles and bundle intents. (**RQ16**) We visualize three examples to showcase the recommendation and explanation generated by each method across the three domains, as shown in Figure 15, where each example involves four parts, namely, the user's purchased bundles, the recommended bundle and generated explanation by each method (i.e., RM, EFM, PGPR and KGAT), as well as the ground truth bundle (i.e., label). We note that in most cases, KGAT can provide accurate bundle recommendation and generate reasonable explanation accordingly.

5.6 Bundle Auto-Naming

5.6.1 Experimental Setup. We use the collected bundles in Table 2. To fine-tune ImageCap with sufficient data, we merge bundles in the three domains and split them into training, validation,

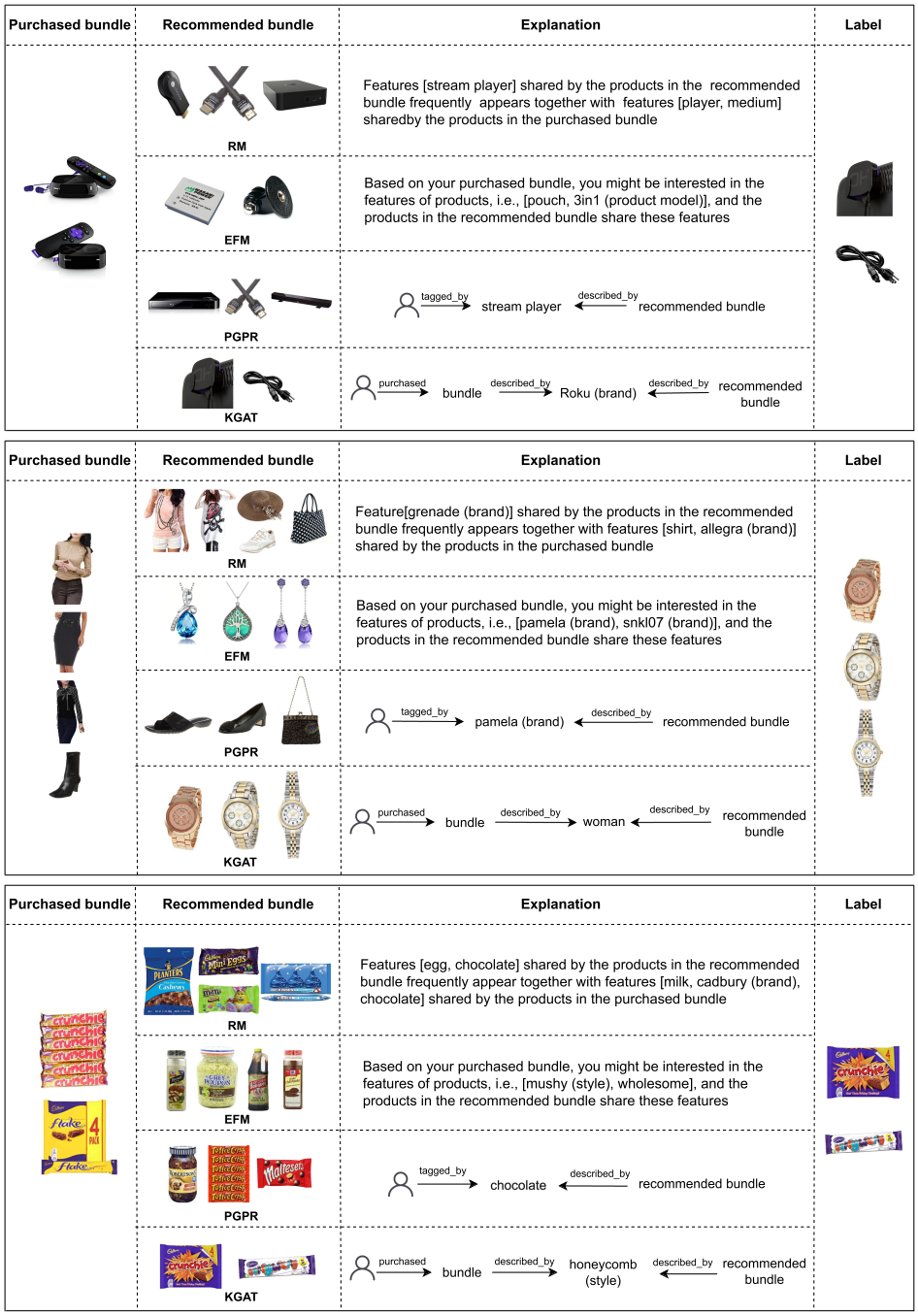


Fig. 15. Examples of the generated explanation for the three domains, i.e., Electronic, Clothing and Food.

and test sets with the ratio of 8:1:1. Due to lacking bundle names, we adopt the annotated bundle intents as labels for ImageCap. For LLMs models, we design the zero-shot prompts as introduced in Section 4.2.6 to ensure the outputs of ChatGPT and Bard meet our expectation. We then carefully design human evaluation to examine the quality of the generated names with three metrics. To

Table 12. Human and LLM-Agent Evaluation on Auto-naming, where the Statistical Significance of Pairwise Differences of ChatGPT vs. Bard is Determined by a Paired t-test (** for $p < 0.05$ and *** for $p < 0.001$)

	Metrics	Human			GPT-Agent			Bard-Agent		
		ImageCap	Bard	ChatGPT	ImageCap	Bard	ChatGPT	ImageCap	Bard	ChatGPT
Electronic	Naturalness	2.56	2.56	**2.68****	1.44	2.31	**2.62****	2.51	2.71	**2.91****
	Relatedness	1.78	2.57	**2.77****	1.51	2.13	**2.83*****	1.59	2.78	**2.89****
	Attractiveness	1.37	**2.57****	2.55	1.36	2.37	**2.85*****	1.23	2.68	**2.87****
Clothing	Naturalness	2.19	2.69	**2.83****	1.32	2.66	**2.92****	2.13	2.53	**2.63****
	Relatedness	1.79	2.62	**2.85****	1.13	2.51	**2.93*****	1.25	2.44	**2.71****
	Attractiveness	1.27	2.36	**2.81*****	1.19	2.54	**2.89*****	1.17	2.27	**2.59****
Food	Naturalness	2.72	**2.92****	2.82	1.42	2.52	**2.79****	2.23	2.81	**2.93****
	Relatedness	1.98	2.73	**2.89****	1.24	2.53	**2.93*****	1.63	2.83	**2.89*****
	Attractiveness	1.08	2.52	**2.76*****	1.21	2.25	**2.91*****	1.02	2.11	**2.45****

be specific, *Naturalness* refers to whether the name is easy to read and understand and its scale ranges from 1 to 3 (1 - it is difficult to read and understand; 2 - it is fair to read and understand; and 3 - it is easy to read and understand). *Relatedness* indicates the extent to which the name can accurately describe the level of relevance of bundles, with scale ranging from 1 to 3 (1 - the name has no relatedness to the bundle; 2 - the name has some relation to the bundle; and 3 - the name accurately describes the bundle). *Attractiveness* suggests whether the name is appealing enough to entice consumers to purchase the bundle, with a scale in the range of $[1, 3]$ (1 - the name is not appealing enough to encourage the purchase of the bundle; 2 - the name has a certain appeal that encourages consumers to purchase it; and 3 - the name is highly appealing to consumers to purchase the bundle). Finally, we employ 15 annotators (8 males and 7 females) to rate the generated names for 20 randomly-sampled bundles per domain by different methods. Similarly, to enhance the reproducibility of human evaluation and reduce labor, we investigate the effectiveness of LLMs in this task. Specifically, we initiate three ChatGPT agents (GPT-Agent) and three Bard agents (Bard-Agent), to evaluate the generated bundle names using the three defined metrics.

5.6.2 Analysis and Insights. Table 12 demonstrates the results of human and LLM-Agent evaluation on the auto-naming task. (**RQ17**) The scores from both human and LLM-Agent exhibit overall consistency. Across the three domains, ImageCap consistently lags behind LLMs (Bard and ChatGPT) on all metrics. This is credited to the ample training corpus data that enables LLMs to precisely comprehend user intentions and swiftly retrieve relevant responses from their knowledge base. Furthermore, the alignment between human and LLM-Agent evaluations further confirms the substantial potential of LLMs in crowdsourcing tasks [50]. (**RQ18**) With regard to the LLMs, ChatGPT generally outperforms Bard regarding both human and LLM-Agent evaluation. A possible explanation is that the **reinforcement learning from human feedback (RLHF)** enables ChatGPT to generate content that is more aligned with human expressions. However, there are exceptions where Bard performs better than ChatGPT with human evaluation on certain metrics, for example, on 'Naturalness' in the Food domain. The reason is that Bard can generate more concise names for food bundles (e.g., Snacks that are delicious and healthy) when compared with the ones generated by ChatGPT (e.g., From sweet to savory, we have a snack for every mood), which may make it easier for humans to read and understand though being less attractive, as demonstrated in Figure 16. The difference between human and LLM evaluations suggests that there's still room for further enhancement of LLMs to better align with human workers in crowdsourcing tasks [10]. (**RQ19**) In order to explore and examine the appearance of bundle names produced by various

Bundle		
Intent	Canon Camera accessories	Assembling a desktop
ImageCap	Camera and accessories	Different style of tablets
Bard	The perfect accessories for your Canon camera	Build your dream PC
ChatGPT	Discover our comprehensive collection of Canon camera accessories	Build Your Dream Desktop with Our Comprehensive Range of Components
Bundle		
Intent	Girls frog dresses	Baby clothing
ImageCap	Women dress	T-shirt
Bard	Dress your little one in style with our frog dresses	Cuteness overload
ChatGPT	Our frog dresses will bring a smile to your little girl's face	Make memories with our collection of adorable baby clothing
Bundle		
Intent	Snacks	K-cups, coffee/tea
ImageCap	Different food of snack	Different brands of coffee
Bard	Snacks that are delicious and healthy	The perfect way to start your day
ChatGPT	From sweet to savory, we have a snack for every mood	Start your day off right with our premium coffee and tea options

Fig. 16. Examples of the generated bundle names for the three domains.

methods, we randomly sample two bundles per domain with their generated names and annotated intents as shown in Figure 16. From the results, we can note that (1) the names generated by ImageCap are quite simple, short, and plain similar to the annotated intents, but sometimes have less relatedness to the bundle, e.g., 'T-shirt' for the second clothing bundles; (2) the quality of names generated by Bard is much improved on the three metrics in comparison with ImageCap, e.g., 'Women dress' vs. 'Dress your little one in style in our frog dresses'. (3) Overall ChatGPT produces names that are of superior quality compared to the other methods. However, there are instances, such as in the Food domain, where Bard outperforms ChatGPT specifically in terms of naturalness.

5.7 Discussion on Challenges and Opportunities

According to our experimental analysis in Sections 5.1–5.6, we summarize the challenges and opportunities for the defined tasks, especially with the emergence of large language models (LLMs).

In general, data sparsity is a major challenge confronted by all tasks. To this end, we perform pattern mining at the category level rather than the item level for bundle detection; recurrently sample items to expand the test set for bundle completion; design a pre-training strategy for model initialization; and extract features from available metadata (i.e., item titles, images, and bundle intents) for bundle explanation and auto-naming. While the presence of such a challenge also brings forth new opportunities for other potential solutions as elaborated in what follows.

5.7.1 Bundle Detection. The major challenges confronted include: (1) how to detect bundles with extremely sparse item-bundle affinity records; (2) how to detect bundles with consistent yet diverse items; and (3) how to detect and evaluate new bundles. Accordingly, to address these challenges, we may (1) leverage side information, e.g., category hierarchy [67], to detect bundles with different granularities; (2) exploit techniques, e.g., self-attention and DPP [4] to respectively maintain item consistent and diversity; (3) adopt semi-supervised learning [77] for bundle detection; and (4) design crowdsourcing tasks for new bundle evaluation.

5.7.2 Bundle Completion. The main challenges lie in (1) how to design high-quality completion models with sparse item-bundle affinity records; (2) how to complete bundles with new items; and (3) how to learn accurate context representations for bundles. Correspondingly, we could (1) adopt data augmentation techniques [81] to address the sparsity issue; (2) leverage side information, e.g., knowledge graph [69] for completion with (new) items; (3) design, e.g., multi-modal algorithms [66], to consider item category, title, and image, and so forth; and (4) employ techniques, such as CVAE [36], to consider bundle intents as conditions (context).

5.7.3 Bundle Ranking. There are four main obstacles for this task: (1) how to build accurate ranking models with sparse user-bundle interaction records; (2) how to rank new bundles for users; (3) how to mine and leverage the complex relations among users, items, and bundles; and (4) how to learn accurate user, item, and bundle representations. Accordingly, we could (1) leverage user-item interaction records as complements; and graph-structured data (e.g., bipartite/knowledge/hyper graphs [83, 84, 90]) to represent the relations among users, items, and bundles; (2) use meta-learning [15] to transfer knowledge from multiple source domains to the target domain; (3) adopt techniques, e.g., GCN [84], hypergraph convolution [5] to model high-order relations; and (4) employ contrastive learning on graphs [48] to reinforce representation learning via self-discrimination.

5.7.4 Bundle Explanation. The major issues that need to be eased include: (1) how to devise better explanation models with sparse user-bundle interaction records; (2) how to mine proper and high-quality features from the metadata for explanation; (3) how to organize features of different types for explanation; and (4) how to automatically evaluate the quality of explanation results. Particularly for the evaluation, a notable limitation of our study lies in the reliance on human assessment for evaluating the generated explanations. This approach is costly, time-intensive, and prone to subjectivity and biases, thereby impeding large-scale evaluation. Consequently, the possible solutions may include: (1) leveraging multi-modal knowledge, e.g., review, category, image, catalog, customer Q&A, and so on; (2) adopting more advanced tools to extract features, e.g., Textrank (github.com/chartbeat-labs/textacy); (3) employing multi-modal graph attention technique [66] to conduct information propagation; (4) using conversational recommenders [94] to understand user immediate interest for a better explanation; and (5) designing automatic evaluation metrics for explanation results assessment instead of user study only. It is noteworthy that in our study, we explore the possibility of using LLMs to evaluate the generated explanations with the defined metrics, thus reducing the reliance on human evaluation. The overall alignment observed between

human and LLM-Agent evaluations underscores the potential of LLM-Agent in enabling automatic and large-scale evaluation of the generated explanations.

5.7.5 Bundle Auto-Naming. The dilemma for bundle auto-naming lies in two-fold. On the one hand, the lack of well-labeled attractive bundle names increases the difficulty in developing well-performed supervised models. On the other hand, it is also significant to determine how to systematically evaluate the quality of the generated bundle names on a large scale. To ease these issues, we could design crowdsourcing tasks to respectively generate bundle names as labels and evaluate the quality of the bundle names via crowd intelligence. With a small amount of well-labeled bundle names, we may leverage the weakly supervised learning approach for auto-naming [76]. Meanwhile, we could employ the capability of LLMs in crowdsourcing tasks for automatic and large-scale evaluation on the generated bundle names [50].

5.7.6 Exploration of LLMs on Product Bundling. The emergence of LLMs also offers potential solutions for addressing the issue of product bundling. With extensive knowledge and superior capabilities in natural language understanding, generation, and logical reasoning, LLMs have demonstrated remarkable performance in the few-shot recommendation scenarios [6, 24, 34, 79]. Leveraging LLMs to mitigate the data sparsity problem in product bundling shows promise [73], which can be achieved through two approaches: in-context learning and instruction tuning [86]. In-context learning involves providing demonstrations within the prompt to guide LLMs in generating the desired outputs effectively. However, several challenges need to be addressed, such as how to appropriately represent users, items, and bundles within the prompt, as well as how to select representative and impactful demonstration examples. Overcoming these challenges is crucial for optimizing the performance of in-context learning and maximizing the potential benefits of utilizing LLMs in generating desired outputs. On the other hand, instruction tuning focuses on fine-tuning LLMs using well-constructed tuning data specifically designed for product bundling tasks. This process equips LLMs with the relevant abilities required for effective product bundling, that is, aligning LLMs with product bundling tasks. However, challenges remain, such as how to construct tuning samples and design parameter-efficient fine-tuning strategies. Furthermore, LLMs can also offer assistance in conducting large-scale evaluations of various product bundling tasks, e.g., bundle explanation and auto-naming [50]. This eliminates the reliance on hiring workers from crowdsourcing platforms, which can be costly and labor-intensive. Nevertheless, there is still room for further enhancement of LLM-Agent to better align with human workers [10]. For instance, an important concern arises regarding the fairness of the evaluation by LLMs. It is crucial to ensure that LLMs do not exhibit a bias towards assigning higher scores to their generated outputs, thus guaranteeing impartial evaluation across all methods. Addressing these concerns will enhance the reliability and objectivity of the evaluation when employing LLMs for product bundling tasks.

6 CONCLUSION

This paper seeks more fundamental contributions in the area of bundle recommendation. To achieve this, we first delicately design a crowdsourcing task to help construct high-quality bundle datasets with rich meta data, particularly the bundle intents, thus facilitating a better understanding of the rationale behind product bundling. With such data, a series of interrelated tasks have been defined and their potential solutions have been explored. Lastly, we perform extensive experiments and in-depth analysis to draw insightful conclusions, and point out challenges and opportunities. Our study delivers new data sources, opens up new research directions and provides useful guidance to promote product bundling in real-world platforms.

REFERENCES

[1] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *VLDB*, Vol. 1215. 487–499.

[2] Faisal M. Almutairi, Nicholas D. Sidiropoulos, and Bo Yang. 2021. XPL-CF: Explainable embeddings for feature-based collaborative filtering. In *CIKM*. 2847–2851.

[3] Tzoof Avny Brosh, Amit Livne, Oren Sar Shalom, Bracha Shapira, and Mark Last. 2022. BRUCE: Bundle recommendation using contextualized item embeddings. In *RecSys*. 237–245.

[4] Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. 2019. Personalized bundle list recommendation. In *WWW*. 60–71.

[5] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637.

[6] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. TALLRec: An effective and efficient tuning framework to align large language model with recommendation. *arXiv preprint arXiv:2305.00447* (2023).

[7] Adrian Boteanu, Emily Dutile, Adam Kiezun, and Shay Artzi. 2020. Subjective search intent predictions using customer reviews. In *CHIIR*. 303–307.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. 2020. Language models are few-shot learners. *NeurIPS* 33 (2020), 1877–1901.

[9] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*. 585–594.

[10] Jan Cegin, Jakub Simko, and Peter Brusilovsky. 2023. ChatGPT to replace crowdsourcing of paraphrases for intent classification: Higher diversity and comparable model robustness. *arXiv preprint arXiv:2305.12947* (2023).

[11] Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Bundle recommendation with graph convolutional networks. In *SIGIR*. 1673–1676.

[12] Fuhai Chen, Rongrong Ji, Xiaoshuai Sun, Yongjian Wu, and Jinsong Su. 2018. GroupCap: Group-based image captioning with structured relevance and diversity constraints. In *CVPR*. 1345–1353.

[13] Liang Chen, Yang Liu, Xiangnan He, Lianli Gao, and Zibin Zheng. 2019. Matching user with item set: Collaborative bundle recommendation with deep attention network. In *IJCAI*. 2095–2101.

[14] Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized outfit generation for fashion recommendation at Alibaba iFashion. In *KDD*. 2662–2670.

[15] Yudong Chen, Xin Wang, Miao Fan, Jizhou Huang, Shengwen Yang, and Wenwu Zhu. 2021. Curriculum meta-learning for next POI recommendation. In *KDD*. 2692–2702.

[16] Weiyu Cheng, Yanyan Shen, Linpeng Huang, and Yanmin Zhu. 2019. Incorporating interpretability into latent factor models via fast influence analysis. In *KDD*. 885–893.

[17] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *RecSys*. 101–109.

[18] Qilin Deng, Kai Wang, Minghao Zhao, Zhene Zou, Runze Wu, Jianrong Tao, Changjie Fan, and Liang Chen. 2020. Personalized bundle recommendation in online games. In *CIKM*. 2381–2388.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, and Sylvain Gelly. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[21] Paolo Dragone, Giovanni Pellegrini, Michele Vescovi, Katya Tentori, and Andrea Passerini. 2018. No more ready-made deals: Constructive recommendation for telco service bundling. In *RecSys*. 163–171.

[22] Yan Fang, Xinyue Xiao, Xiaoyu Wang, and Huiqing Lan. 2018. Customized bundle recommendation by association rules of product categories for online supermarkets. In *DSC*. 472–475.

[23] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. 2015. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *CHI*. 1631–1640.

[24] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, HaofenWang, and Jiawei Zhang. 2023. Chat-REC: Towards interactive and explainable LLMs-augmented recommender system. *arXiv preprint arXiv:2303.14524* (2023).

[25] Yong Ge, Hui Xiong, Alexander Tuzhilin, and Qi Liu. 2014. Cost-aware collaborative filtering for travel tour recommendations. *TOIS* 32, 1 (2014), 1–31.

[26] Shijie Geng, Zuohui Fu, Juntao Tan, Yingqiang Ge, Gerard De Melo, and Yongfeng Zhang. 2022. Path language modeling over knowledge graphs for explainable recommendation. In *WWW*. 946–955.

[27] Google. 2023. Bard: A Large Language Model from Google AI. https://bard.google.com

[28] Jiuxiang Gu, Gang Wang, Jianfei Cai, and Tsuhan Chen. 2017. An empirical study of language CNN for image captioning. In *ICCV*. 1222–1231.
[29] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.
[30] Yun He, Jianling Wang, Wei Niu, and James Caverlee. 2019. A hierarchical self-attentive model for recommending user-generated item lists. In *CIKM*. 1481–1490.
[31] Yun He, Yin Zhang, Weiwen Liu, and James Caverlee. 2020. Consistency-aware recommendation for user-generated item list continuation. In *WSDM*. 250–258.
[32] Zhankui He, Handong Zhao, Tong Yu, Sungchul Kim, Fan Du, and Julian McAuley. 2022. Bundle MCR: Towards conversational bundle recommendation. In *RecSys*. 288–298.
[33] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[34] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023. Large language models are zero-shot rankers for recommender systems. *arXiv preprint arXiv:2305.08845* (2023).
[35] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. 2023. Do LLMs understand user preferences? Evaluating LLMs on user rating prediction. *arXiv preprint arXiv:2305.06474* (2023).
[36] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *ICLR*.
[37] Pigi Kouki, Ilias Fountalis, Nikolaos Vasiloglou, Nian Yan, Unaiza Ahsan, Khalifeh Al Jadda, and Huiming Qu. 2019. Product collection recommendation in online retail. In *RecSys*. 486–490.
[38] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
[39] Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate neural template explanations for recommendation. In *CIKM*. 755–764.
[40] Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and FuruWei. 2021. TrOCR: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282* (2021).
[41] Dawen Liang, Rahul G. Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *WWW*. 689–698.
[42] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. 2002. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery* 6, 1 (2002), 83–105.
[43] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten De Rijke. 2019. Explainable outfit recommendation with joint outfit matching and comment generation. *TKDE* 32, 8 (2019), 1502–1516.
[44] Guannan Liu, Yanjie Fu, Guoqing Chen, Hui Xiong, and Can Chen. 2017. Modeling buying motives for personalized product bundle recommendation. *TKDD* 11, 3 (2017), 1–26.
[45] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *ICDM*. 407–416.
[46] Yidan Liu, Min Xie, and Laks V. S. Lakshmanan. 2014. Recommending user generated item lists. In *RecSys*. 185–192.
[47] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: Multi-task learning for recommendation and explanation. In *RecSys*. 4–12.
[48] Yunshan Ma, Yingzhi He, An Zhang, Xiang Wang, and Tat-Seng Chua. 2022. CrossCBR: Cross-view contrastive learning for bundle recommendation. In *KDD*. 1–9.
[49] OpenAI. 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt
[50] Lidiia Ostyakova, Veronika Smilga, Kseniia Petukhova, Maria Molchanova, and Daniel Kornev. 2023. ChatGPT vs. crowdsourcing vs. experts: Annotating open-domain conversations with speech functions. In *SIGDIAL*. 242–254.
[51] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. 2011. Recommendation systems with complex constraints: A course recommendation perspective. *TOIS* 29, 4 (2011), 1–33.
[52] Sung-Jun Park, Dong-Kyu Chae, Hong-Kyun Bae, Sumin Park, and Sang-Wook Kim. 2022. Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation. In *WSDM*. 784–793.
[53] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and personalizing bundle recommendations on steam. In *SIGIR*. 1073–1076.
[54] Georgina Peake and Jun Wang. 2018. Explanation mining: Post Hoc interpretability of latent factor models for recommendation systems. In *KDD*. 2060–2069.
[55] Huihuai Qiu, Guibing Guo, Jie Zhang, Zhu Sun, Hai Thanh Nguyen, and Yun Liu. 2016. TBPR: Trinity preference based Bayesian personalized ranking for multivariate implicit feedback. In *UMAP*. 305–306.
[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[57] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR* 21, 140 (2020), 1–67.
[58] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese BERT-networks. *arXiv preprint arXiv:1908.10084* (2019).
[59] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[60] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *TACL* 8 (2020), 264–280.
[61] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.
[62] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.
[63] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NIPS*, Vol. 28.
[64] Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Giuseppe Fiameni, and Rita Cucchiara. 2022. From show to tell: A survey on deep learning-based image captioning. *TPAMI* 45, 1 (2022), 539–559.
[65] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. 2020. Dual learning for explainable recommendation: Towards unifying user preference prediction and review generation. In *WWW*. 837–847.
[66] Rui Sun, Xuezhi Cao, Yan Zhao, Junchen Wan, Kun Zhou, Fuzheng Zhang, Zhongyuan Wang, and Kai Zheng. 2020. Multi-modal knowledge graphs for recommender systems. In *SIGIR*. 1405–1414.
[67] Zhu Sun, Jie Yang, Jie Zhang, and Alessandro Bozzon. 2017. Exploiting both vertical and horizontal dimensions of feature hierarchy for effective recommendation. In *AAAI*, Vol. 31. 189–195.
[68] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*. 297–305.
[69] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *ECRA* 37 (2019), 100879.
[70] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are we evaluating rigorously? Benchmarking recommendation for reproducible evaluation and fair comparison. In *RecSys*. 23–32.
[71] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. 2022. DaisyRec 2.0: Benchmarking recommendation for rigorous evaluation. *TPAMI* (2022).
[72] Zhu Sun, Jie Yang, Kaidong Feng, Hui Fang, Xinghua Qu, and Yew Soon Ong. 2022. Revisiting bundle recommendation: Datasets, tasks, challenges and opportunities for intent-aware product bundling. In *SIGIR*. 2900–2911.
[73] Zhu Sun, Kaidong Feng, Jie Yang, Xinghua Qu, Hui Fang, Yew-Soon Ong, andWenyuan Liu. 2023. Dynamic in-context learning from nearest neighbors for bundle generation. *arXiv preprint arXiv:2312.16262* (2023).
[74] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual explainable recommendation. In *CIKM*. 1784–1793.
[75] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, and Faisal Azhar. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[76] Quoc-Tuan Truong, Tong Zhao, Changhe Yuan, Jin Li, Jim Chan, Soo-Min Pantel, and Hady W Lauw. 2022. AmpSum: Adaptive multiple-product summarization towards improving recommendation captions. In *WWW*. 2978–2988.
[77] Hen Tzaban, Ido Guy, Asnat Greenstein-Messica, Arnon Dagan, Lior Rokach, and Bracha Shapira. 2020. Product bundle identification using semi-supervised learning. In *SIGIR*. 791–800.
[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[79] Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153* (2023).
[80] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *SIGIR*. 165–174.
[81] Qinyong Wang, Hongzhi Yin, Hao Wang, Quoc Viet Hung Nguyen, Zi Huang, and Lizhen Cui. 2019. Enhancing collaborative filtering with generative augmentation. In *KDD*. 548–556.
[82] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet Orgun, and Longbing Cao. 2019. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *IJCAI*. 3771–3777.
[83] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *KDD*. 950–958.

[84] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.
[85] Penghui Wei, Shaoguo Liu, Xuanhua Yang, Liang Wang, and Bo Zheng. 2022. Towards personalized bundle creative generation with contrastive non-autoregressive decoding. In *SIGIR*. 12–16.
[86] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, and Qi Liu. 2023. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860* (2023).
[87] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *SIGIR*. 285–294.
[88] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. 2010. Breaking out of the box of recommendations: From items to packages. In *RecSys*. 151–158.
[89] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. 2014. Generating top-k packages via preference elicitation. *VLDB Endowment* 7, 14 (2014), 1941–1952.
[90] Dingqi Yang, Bingqing Qu, Jie Yang, and Philippe Cudre-Mauroux. 2019. Revisiting user mobility and social relationships in LBSNs: A hypergraph embedding approach. In *WWW*. 2147–2157.
[91] Jie Yang, Judith Redi, Gianluca Demartini, and Alessandro Bozzon. 2016. Modeling task complexity in crowdsourcing. In *HCOMP*. 249–258.
[92] Quanzeng You, Hailin Jin, ZhaowenWang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *CVPR*. 4651–4659.
[93] Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023. Recommendation as instruction following: A large language model empowered recommendation approach. *arXiv preprint arXiv:2305.07001* (2023).
[94] Lu Zhang, Zhu Sun, Jie Zhang, Yiwen Wu, and Yunwen Xia. 2022. Conversation-based adaptive relational translation method for next POI recommendation with uncertain check-Ins. *TNNLS* (2022).
[95] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. 83–92.
[96] Yongfeng Zhang and Xu Chen. 2020. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval* 14, 1 (2020), 1–101.
[97] Sen Zhao, Wei Wei, Ding Zou, and Xianling Mao. 2022. Multi-view intent disentangle graph networks for bundle recommendation. In *AAAI*. 1–9.
[98] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. 2014. Bundle recommendation in ecommerce. In *SIGIR*. 657–666.

Received 11 July 2023; revised 6 March 2024; accepted 11 March 2024