The Catalog Problem: Clustering and Ordering Variable-Sized Sets

Mateusz Jurewicz [1 2 3] **Graham W. Taylor** [2 4] **Leon Derczynski** [3 5]

Abstract

Prediction of a **varying number** of **ordered clusters** from sets of **any cardinality** is a challenging task for neural networks, combining elements of set representation, clustering and learning to order. This task arises in many diverse areas, ranging from medical triage and early discharge, through machine part management and multi-channel signal analysis for petroleum exploration to product catalog structure prediction. This paper focuses on that last area, which exemplifies a number of challenges inherent to adaptive ordered clustering, referred to further as the eponymous *Catalog Problem*. These include learning variable cluster constraints, exhibiting relational reasoning and managing combinatorial complexity. Despite progress in both neural clustering and set-to-sequence methods, no joint, fully differentiable model exists to-date. We develop such a modular architecture, referred to further as Neural Ordered Clusters (NOC), enhance it with a specific mechanism for learning cluster-level cardinality constraints, and provide a robust comparison of its performance in relation to alternative models. We test our method on three datasets, including synthetic catalog structures and PROCAT, a dataset of real-world catalogs consisting of over $1.5\,\mathrm{M}$ products, achieving state-of-the-art results on a new, more challenging formulation of the underlying problem, which has not been addressed before. Additionally, we examine the network's ability to learn higher-order interactions.

[1]Tjek A/S, Copenhagen, Denmark [2]Vector Institute for Artificial Intelligence, Toronto, Canada [3]IT University of Copenhagen, Department of Computer Science, Copenhagen, Denmark [4]University of Guelph, Guelph, Canada [5]University of Washington, Seattle, USA. Correspondence to: Mateusz Jurewicz <mateusz.jurewicz@gmail.com>, Graham Taylor <gwtaylor@uoguelph.ca>, Leon Derczynski <ld@itu.dk>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

The ability to group members of a set and order these groups is key to many important real-world decision-making processes. It finds applications ranging from supply chain management (Wenzel et al., 2019) to prioritization in medical triage (Miles et al., 2020; Buchard et al., 2020). Other application domains include petroleum exploration (Rabiller et al., 2010), business process analytics (Le et al., 2014), parallelization and machine part management (Bakkelund, 2022) and product catalog structuring (Jurewicz & Derczynski, 2022), where the goal is to take a set of products, group them together and order these groups to form a coherent product catalog. We term this problem of simultaneously grouping and ordering a set of items the Catalog Problem.

This paper defines the Catalog Problem and presents an investigation into neural network approaches to it. To this end we introduce a fully-differentiable, deep learning (DL) model architecture that addresses the Catalog Problem. It clusters sets of items into groups, and yields an ordering of these groups. All of this is achieved in a *supervised* manner. While clustering methods are often unsupervised (Aljalbout et al., 2018; Ronen et al., 2022), the meaningful ordering of clusters often requires more knowledge than is available from the instance representation.

Similarly, learning to order is often framed as a supervised learning task (Vinyals et al., 2015; Yin et al., 2020; Shi, 2022). Referred to further as *set-to-sequence* (S2S), this area and its corresponding methods inspire the cluster-ordering aspect of our proposed Neural Ordered Clusters (NOC) model. Both neural clustering and set-to-sequence models have limitations. Element-wise neural clustering methods are $O(n)$, where n[1] is the cardinality of the input set. Cluster-wise and attention-based models are more computationally efficient, but exhibit a limited ability to learn cluster cardinality constraints (Pakman et al., 2020), integral to both the prototypical Catalog Problem and its practical instantiations. Set-to-sequence methods, on the other hand, are effective at learning constraints (Zhu et al., 2021) and generalizing to unseen distributions (Wen, 2022). However, they

[1]$O(n)$ can be prohibitive with large input sets ($n >= 1000$), which is often the case in many interesting set-input problems such as 3D point cloud tasks (Qi et al., 2017; Ge et al., 2018; Zhao et al., 2021).

are limited by their inability to predict an input-dependent number k of clusters without major adjustments (Fernández-González, 2022), two of which are proposed in Section 5.1. These S2S variants suffer from noisy in-cluster order and cascading errors (Gan et al., 2020; Vial et al., 2022).

To address these challenges, we implement a unified clustering and cluster ordering method. NOC is capable of predicting ordered, partitional cluster assignments for elements of sets of varying cardinality. It infers a flexible, input-dependent number of k diverse clusters, maintains $O(k)$ complexity and utilizes a jointly learned representation of set elements to find the target cluster order. Unlike existing neural clustering methods, it exhibits the ability to learn cluster cardinality constraints through supervision. To our knowledge, no other neural-based method exists to address such challenges in an end-to-end, jointly trainable way, instead performing clustering and ordering as two separate tasks, sometimes with the separate addition of a representation learning step (Aljalbout et al., 2018). All code, hyper-parameters and datasets required for reproducing our results are made available and detailed via the appendix. To summarize, our contributions are as follows:

- We introduce the Catalog Problem, a novel joint clustering and cluster ordering problem over sets of elements, which is a challenging variant of the set-to-sequence domain with multiple aspects that are not handled by existing neural methods. We exemplify and tackle this problem on three datasets, including a real-world dataset of over $1.5\,\mathrm{M}$ products grouped and ordered into product catalogs by human experts.
- We propose a novel, fully differentiable, joint neural clustering and cluster ordering model, Neural Ordered Clusters (NOC), capable of predicting an adaptive, input-dependent number of ordered, partitional clusters from sets of any cardinality.
- We provide a robust comparison of existing and proposed neural methods on the Catalog Problem using diverse datasets, providing insights into the models' capacity to learn higher-order relational rules of cluster composition and ordered structure.

2. The Catalog Problem

Many problems require predicting an adaptive, input-dependent number (k) of partitional clusters from sets of varying cardinality (n) and consequently ordering these clusters according to a target preference. These include but are not limited to vitally important challenges related to triage-like tasks in the medical, military and crisis relief contexts (Kennedy et al., 1996), which are only beginning to be tackled via machine learning methods in supervised or semi-supervised settings (Buchard et al., 2020). Such challenges fall under the umbrella term of the Catalog Problem.

Multiple datasets from various areas of application either lend themselves directly to this formulation or can be adjusted to it, such as the PROCAT dataset which contains a supervision target in the form of the human-designed structure of product catalogs (Jurewicz & Derczynski, 2021) or the MIMIC-III dataset of electronic health records (Johnson et al., 2016). Other curated datasets exist in the field of large-scale image preference grouping (Chang et al., 2016), mortality risk ranking (Kwon et al., 2018) and early warning systems for influenza (Espino et al., 2003), with further supervised datasets being potentially obtainable in such areas as grouping and ordering sets of tasks for parallel runs on a finite number of processors (Bakkelund, 2022), placement in physical design of integrated circuits (Lin et al., 2018) and graph clustering (Tian et al., 2014).

In the Catalog Problem the input is an unordered set of unique elements $\mathbb{X} = \{x_1, \ldots, x_n\}$, with a varying cardinality $n = |\mathbb{X}|$. The target output is a sequence of clusters $\mathbf{y} = (\mathbb{C}_1, \ldots, \mathbb{C}_k)$, where k is an input-dependent number of clusters and each cluster is defined by the elements assigned to it, whose number can differ per cluster. For example, given an input set $\mathbb{X} = \{x_1, x_2, x_3, x_4, x_5\}$ the target can take the form of the sequence $\mathbf{y} = (\{x_3, x_4\}, \{x_1\}, \{x_2, x_5\})$. This example thus requires the prediction of the following set of clusters: $\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3\}$, such that if $\mathbb{C}_1 = \{x_1\}$, $\mathbb{C}_2 = \{x_3, x_4\}$ and $\mathbb{C}_3 = \{x_2, x_5\}$ then $\mathbf{y} = (\mathbb{C}_2, \mathbb{C}_1, \mathbb{C}_3)$. All elements must be assigned to a cluster, an element can only belong to one cluster and empty clusters are not allowed. The number of clusters to be predicted (k) is not given to the model ahead of time and can vary between examples.

The problem requires learning a parameterized function ρ_θ, capable of taking the input set, finding the optimal number of clusters to partition it into (k from the supervision target), assigning each element to one of k clusters and ordering these clusters. Thus $\rho_\theta(\mathbb{X}) = \hat{\mathbf{y}}$, with k being predicted indirectly. The optimal assignments and order of clusters is provided by supervision. This is a general problem that, as is shown by experiments later in this paper, is non-trivial. For a visual explanation, see Figure 1. Although the Catalog Problem is so named because it models the task of creating a catalog of items, e.g. products, no specific application is prescribed; the problem only defines input and output types and a relation between these two. The difficulty lies in learning the relationships between both input elements and groups thereof. This difficulty can be compounded by the uniqueness of input elements, making learning representations difficult, due to the scarcity of distributional information.

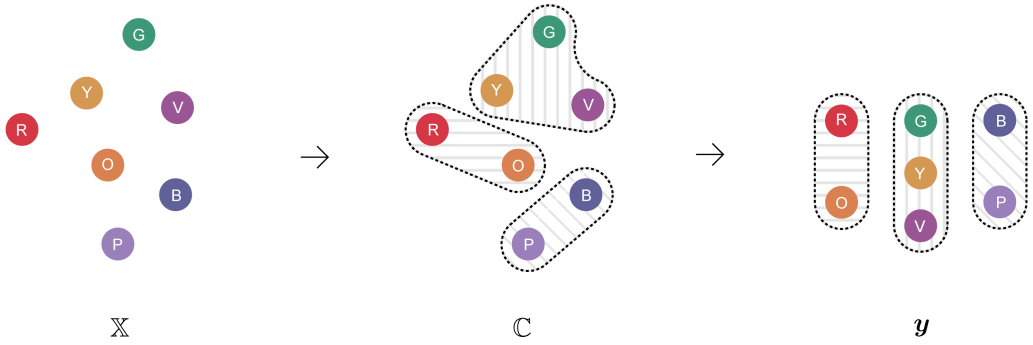

Figure 1. **The Catalog Problem**. From left to right: a set of input elements ($\mathbb{X}$); target partitional clustering of those elements ($\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3\}$); and a target ordering over those clusters ($\mathbf{y} = (\mathbb{C}_2, \mathbb{C}_1, \mathbb{C}_3)$). The targets presented here were chosen arbitrarily, roughly following the colour spectrum for the purposes of this demonstration. In used datasets they contain learnable patterns. A candidate model may perform all these tasks using information about inter-element relations and intra-cluster relations in order to characterise a cluster, and inter-cluster relations to generate the final, ordered clustering.

2.1. Related Work

There have been many machine learning (ML) approaches to clustering with some notion of order, albeit often aimed at preventing the impact of this order on the clusters (Fisher et al., 1992). In the more common, *unsupervised* setting these range from hierarchical clustering (Johnson, 1967; Chu, 1974), through ordinal clustering (Janowitz, 1978) and incremental conceptual clustering (Fisher, 1987) to Markov clustering (Van Dongen, 2000) and other, more recent methods (Ankerst et al., 1999; Turowski et al., 2020). Certain unsupervised clustering methods without the ordering element, like affinity propagation (Frey & Dueck, 2007; Vlasblom & Wodak, 2009), are also capable of outputting an adaptive, input-dependent number of partitional clusters.

Closer to the supervised setting of interest, there have been attempts to leverage instance labels to augment k-means (Ergun et al., 2022), improve the interpretability thereof (Peng et al., 2022) and to cluster labelled data to facilitate permutation learning (Lee & Kim, 2020). Similarly, contrastive clustering utilizes soft labels to maximize the similarities of positive pairs while minimizing those of negative ones (Li et al., 2021), in an approach reminiscent of the pairwise order prediction modules that resulted in increased performance on strictly set-to-sequence tasks (Yin et al., 2020). However, these supervised clustering methods do not yield an ordering of clusters.

3. Background

We identify three classes of neural approaches towards solving aspects of the Catalog Problem: set representation; neural clustering; and ordering through pointer attention. Firstly, learning permutation invariant **set representations** that can encode higher-order interactions is vital, due to the complex relational factors among set elements that determine the target output. Deep learning advances in set representation focus primarily on being able to effectively learn such relations, starting with *Deep Sets* (Zaheer et al., 2017), through the *Set Transformer* (Lee et al., 2019) to modifications thereof (Girgis et al., 2021; Jurewicz & Derczynski, 2022). These methods can be used for both encoding elements and representing clusters. In the Set Transformer, given an unordered set ($\mathbb{X}$), we obtain the representations of set elements ($\mathbf{E}_\pi$) and then of the entire input set ($\mathbf{s}$) via:

$$\mathbf{E}_\pi = \text{MAB}(\mathbb{X}, \mathbb{X}) = \text{LN}(\mathbf{H} + \phi(\mathbf{H})), \tag{1}$$

$$\text{where } \mathbf{H} = \text{LN}(\mathbf{X} + \text{MHA}(\mathbf{X}_q, \mathbf{X}_k, \mathbf{X}_v)), \tag{2}$$

$$\mathbf{s} = \text{PMA}(\mathbf{E}_\pi) = \text{MAB}(\mathbf{r}, \mathbf{E}_\pi). \tag{3}$$

Here, multihead, intra-set attention (denoted MHA) is performed by casting the input set $\mathbb{X}$ to query, key, and value matrices $\mathbf{X}_q, \mathbf{X}_k, \mathbf{X}_v$ according to an arbitrary permutation π, and adding a residual connection as defined by Vaswani et al. (2017), without positional encoding. This operation is incorporated into a multihead self-attention block (MAB) by the inclusion of a row-wise feed-forward neural network (NN) ϕ, with layer normalization (LN) after each block (Ba et al., 2016), resulting in a permutation equivariant[2] matrix of per-element representations ($\mathbf{E}_\pi$). These are then aggregated into a permutation invariant representation of the entire set ($\mathbf{s}$) by performing pooling by multihead attention (PMA) between the per-element representations and a learned seed vector $\mathbf{r}$. These operations are used extensively in our method for encoding both the initial input set and the predicted clusters.

Secondly, supervised **neural clustering** obtains per-element cluster assignments ($\hat{c}_i$) through a number of modular func-

[2]For a formal proof, see Section 3.1 and supplementary material of Lee et al. (2019).

tions parameterized by NNs. These networks leverage set representation methods to encode the set of currently available, unassigned elements ($\mathbf{U}_j$), each previously completed cluster ($\mathbf{g}_j$) and consequently all clustered elements jointly ($\mathbf{G}_j$), at each step j. This is paired with an algorithm for selecting the next j-th cluster (if clusterwise) or element (if pointwise) to consider until all elements are assigned.

In the $O(k)$ clusterwise formulation each cluster assignment is the output of another NN (ρ), in the form of a binary choice ($\hat{c}_i$) per encoded element ($\mathbf{x}_i$), conditioned on these representations and trained in a teacher-forced manner, with loss calculated only for the elements belonging to the current cluster. In the attention-based, clusterwise framework of the *Attentive Clustering Process* (Pakman et al., 2020) a random anchor element ($\mathbf{x}_a$) is selected at each step j, along with a latent variable ($\mathbf{z}_j$) sampled from a Normal distribution via learned mean and standard deviation, on which the final predictions are conditioned for the current j-th cluster:

$$p(\mathbf{z}_j \mid \mathbb{X}_j) = \mathcal{N}(\mathbf{z}_j \mid \mathbf{x}_a, \mathbf{U}_j, \mathbf{G}_j), \tag{4}$$

$$p_{\theta,i}(\hat{c}_i = 1 \mid \mathbb{X}_j) = \text{sigmoid}(\rho(\mathbf{x}_i, \mathbf{x}_a, \mathbf{z}_j, \mathbf{U}_j, \mathbf{G}_j)). \tag{5}$$

Thirdly, **pointer attention** can be used to select a single element from a set of any cardinality n, common in set-to-sequence NNs. At each step $m \in \{1, \dots, k\}$ it outputs an attention vector ($\mathbf{a}_m$) over all obtained clusters $\mathbb{C}$. As the clusters are selected sequentially, this represents their predicted order, with highest attention value pointing to the index of the m-th cluster in that order:

$$\mathbf{a}_m = \text{softmax}(\mathbf{v}^\top \tanh(\mathbf{W}_1\mathbb{C} + \mathbf{W}_2\mathbf{h}_m^d)), \tag{6}$$

where $\mathbf{v}$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are model parameters, $\tanh$ is the hyperbolic tangent nonlinearity, and $\mathbf{h}_m^d$ is customarily the hidden state of the pointer network at current selection step m. The first hidden state $\mathbf{h}_0^d$ can be initialized from the permutation invariant set representation $\mathbf{s}$. In our context, this in principle enables us to sequentially select predicted clusters according to their learned target order.

4. The Neural Ordered Clusters Model

This section introduces the proposed Neural Ordered Clusters (NOC) model (Figure 2).

The NOC model consists of three modular parts, each with a corresponding loss factor. These components take the form of partitional neural clustering, per-cluster cardinality prediction, and cluster ordering via pointer attention. The learned representations of elements and the set in its entirety are transformed by each of these modules and continuously adjusted during training in a fully differentiable way.

The **first step** is to obtain a partitional clustering (NOC_1). We propose to achieve this through an adjusted neural clustering module, building on the process described in equations 4 and 5. First, we utilize the *Set Interdependence Transformer*, or SIT (Jurewicz & Derczynski, 2022), to obtain both the representations of the individual elements ($\mathbf{E}_\pi$) and the permutation-invariant representation of the entire set ($\mathbf{s}$). SIT consists of a stack of MAB layers described in equations 1 and 3, except that the second layer's input takes the form of an augmented matrix, in which the vector representation of the set is concatenated to $\mathbf{E}_\pi$ as if it was an additional set element $\mathbf{e}_i$. This is intended to enable learning of higher-order interactions in fewer layers. At each cluster prediction step j the representations of unassigned elements ($\mathbf{U}_j$) and each previously completed cluster ($\mathbf{g}_{1:j}$) are adjusted through a stack of SIT transformations and used to make cluster assignments $\hat{c}_i$ per unassigned element i:

$$\text{NOC}_1(e_i, \mathbb{X}_j) = p_{\theta,i}(\hat{c}_i = 1 \mid \mathbb{X}_j), \tag{7}$$

$$\text{NOC}_1(e_i, \mathbb{X}_j) = \sigma(\phi_1(\mathbf{e}_i, \mathbf{e}_a^j, \mathbf{z}_j, \text{SIT}(\mathbf{U}_j), \text{SIT}(\mathbf{g}_{1:j}))). \tag{8}$$

The **second step** (NOC_2) is to adjust the cluster assignments via the predicted cardinality t_j of the j-th cluster. At each step a function, parameterized by a fully-connected neural network ϕ_2, is used to predict the cardinality of the current cluster as a regression task. The obtained cardinality, conditioned on the available elements and previously predicted clusters, is used as a threshold for the maximum number of elements to assign to current cluster. If the number of elements assigned by NOC_1 exceeds this threshold, the elements with lower values of $\hat{c}_i$ are excluded from $\mathbb{C}_j$.

$$t_j = \text{NOC}_2(\mathbb{C}_j, \mathbb{X}_j) = \phi_2(\text{PMA}(\mathbf{e}_a^j, \text{SIT}(\mathbb{U}_j), \text{SIT}(\mathbf{g}_{1:j}))), \tag{9}$$

$$\mathbb{C}_j = \begin{cases} \mathbf{C}^j, & \text{if } |\mathbb{C}_j| \leq t_j \\ \mathbf{C}^j_{1:t_j}, & \text{otherwise} \end{cases}. \tag{10}$$

Steps one and two are repeated until we have obtained k partitional clusters ($\mathbb{C}_1, \dots \mathbb{C}_k$) with individual cardinalities. Set-to-sequence methods expect fixed-length vector representations, therefore we use SIT and PMA to obtain them ($\mathbf{C}_\pi = [\mathbf{c}_1, \dots, \mathbf{c}_k]$ where $\mathbf{c}_j = \text{PMA}(\text{SIT}(\mathbb{C}_j))$). In the **third and final stage** of NOC_3 an Enhanced Pointer Network (Yin et al., 2020) is used to output an attention vector $\mathbf{a}_m$ at each step $m \in \{1, \dots, k\}$. The highest attention value points to the cluster to be placed at m-th position in the output sequence of ordered clusters:

$$\mathbf{a}_m = \text{softmax}(\mathbf{v}^\top \tanh(\mathbf{W}_1\mathbf{M}_m + \mathbf{W}_2\mathbf{h}_m^d)), \tag{11}$$

$$\mathbf{h}_m^d = \text{LSTM}(\mathbf{h}_{m-1}^d, \mathbf{c}_{m-1}). \tag{12}$$

This largely resembles the process outlined in Equation 6, with the exception of matrix $\mathbf{M}_m$, specific to the Enhanced Pointer Network, explained in more detail in appendix A.1.

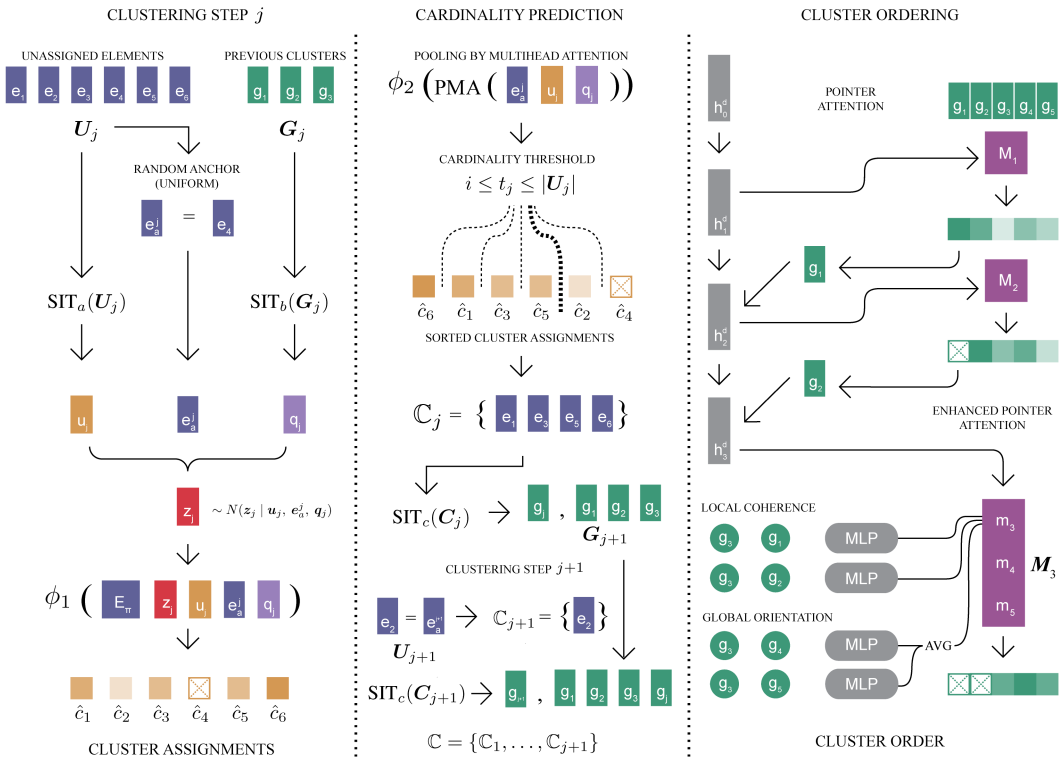

Figure 2. **NOC Architecture**. An overview of how NOC completes the clustering of the input set (in two steps, j and $j+1$), followed by ordering of the predicted clusters (rightmost panel). Starting at the top of the leftmost panel and moving to the bottom before switching to the next panel to the right, at clustering step j the representations of unassigned elements ($\mathbf{U}_j = \mathbf{e}_{1:6}$), previously created clusters ($\mathbf{G}_j = \mathbf{g}_{1:3}$) and a randomly selected anchor element ($\mathbf{e}_a^j$) are used to obtain initial cluster assignments' probabilities, which represent how likely each unassigned element is to become part of the current, j^{th} cluster ($\hat{c}_{1:6} \approx p_\theta(\hat{c}_{1:6} = j) = p_\theta(e_{1:6} \in \hat{\mathbb{C}}_j)$), with color opacity indicating higher predicted probability. In the middle panel the current cluster cardinality ($t_j = 4$) is predicted and used to adjust the j^{th} cluster ($\hat{\mathbb{C}}_j$), which is then transformed via $\text{PMA}_c(\text{SIT}_c(\hat{\mathbb{C}}_j))$ into its embedded representation $\mathbf{g}_j$, which becomes part of the $\mathbf{G}_{j+1}$ matrix and is used during the remaining clustering steps. In the rightmost panel, after k iterations of the NOC_1 and NOC_2 steps, the predicted clusters ($\hat{\mathbb{C}} \approx \mathbf{G}_{j+1} = \mathbf{G}_k$) are ordered via NOC_3's Enhanced Pointer attention (A.1).

Together, these three elements of NOC allow prediction of an adaptive number k of partitional clusters with varying, learned cardinalities. Learning is achieved through a weighted sum of the loss from each of the three stages of NOC, with teacher-forcing (Williams & Zipser, 1989).

For further clarity, we outline the progression over all three stages of the proposed Neural Ordered Clusters method in Algorithm 1 . Steps 1-23 jointly describe the processing within NOC_1 and NOC_2 (steps 12-16 specifically for the latter). The third module, NOC_3 is described by steps 24-30. We begin with an unordered set $\mathbb{X} \in \mathbb{R}^d$ (of any cardinality), and assume that it has at least two elements, which can then potentially belong to separate clusters. This set is represented as a matrix of d-dimensional elements, ordered according to some arbitrary permutation π into $\mathbf{X}_\pi$. The initial, intermediate output comes in the form of individual clusters of elements at each j-th iteration, which ultimately form the set of all predicted clusters ($\mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$). Each candidate cluster goes through a final cardinality prediction step, resulting in the threshold value of t_j, through which some elements may be excluded from their original cluster. Finally, an Enhanced Pointer Network (EPN) performs k iterations, selecting a single cluster to be placed next in the final output sequence $\hat{\mathbf{y}}$ by the index of the highest value in the predicted attention vector $\mathbf{a}_m$.

5. Experiments

In this section we examine approaches to the Catalog Problem, including baseline methods adapted to this output structure as well as the NOC model, evaluating over both synthetic and real-world datasets. **All datasets, hyperparameters and code are freely available** and described in detail in Appendix A.4. The provided code includes all data preprocessing and generation steps.

Algorithm 1 Neural Ordered Clusters

Require: $|\mathbb{X}| = n \geq 2$ // At least two elements, otherwise single cluster
Ensure: $x_i \neq x_j \; \forall \, i, j \neq i \in \{1, \ldots, n\}$ // No repeated elements
$\mathbf{E}_\pi \leftarrow \text{SIT}(\mathbf{X}_\pi \sim \mathbb{X}),\ j \leftarrow 1$
$r \leftarrow n - 1$ // Track number of unassigned elements
$\mathbf{e}_a^j \leftarrow \mathbf{E}_\pi$ // Randomly chosen anchor for initial cluster
$\mathbf{U}_j \leftarrow \text{SIT}(\mathbb{E} \setminus \{\mathbf{e}_a^j\})$ // Initialize unassigned representations
$\mathbf{q}_j \leftarrow \varnothing$ // No previous clusters
while $r > 1$ **do**
  $\mathbf{z}_j \sim \mathcal{N}(\text{SIT}(\mathbf{e}_a^j, \mathbf{U}_j, \mathbf{g}_{1:j}))$
  for $i \leftarrow 1 \ldots r$ **do**
    $\hat{c}_i \leftarrow \phi_1(\mathbf{e}_i, \mathbf{e}_a^j, \mathbf{z}_j, \text{SIT}(\mathbf{U}_j), \text{SIT}(\mathbf{g}_{1:j}))$ // j-th cluster assignments per element
  end for
  $\mathbf{C}_j \leftarrow \mathbf{E}^\pi_{i:\hat{c}_i=1}$ // Cluster j from assignments (sorted)
  $t_j \leftarrow \phi_2(\text{PMA}(\mathbf{e}_a^j, \text{SIT}(\mathbf{U}_j), \text{SIT}(\mathbf{g}_{1:j})))$ // Predict cluster cardinality
  if $|\mathbb{C}_j| \leq t_j$ **then**
    $\mathbb{C}_j \leftarrow \mathbf{C}_j$
  else
    $\mathbb{C}_j \leftarrow \mathbf{C}_{1:t_j}$ // Adjust j-th cluster's cardinality
  end if
  $j \leftarrow j + 1$
  $\mathbf{U}_j \leftarrow \text{SIT}(\mathbb{E} \setminus \mathbb{C}_j)$ // Update unassigned representations
  $\mathbf{e}_a^j \leftarrow \mathbf{U}_j$ // Randomly chosen anchor for next cluster (from unassigned)
  $\mathbf{q}_j \leftarrow \mathbb{C} = \{\mathbb{C}_1, \ldots, \mathbb{C}_j\}$ // Update preceding clusters' representations
  $r \leftarrow r - |\mathbb{C}_j| - 1$ // Adjust number of unassigned elements
end while
$\mathbf{h}_1^d \leftarrow \text{SIT}(\mathbf{C}_\pi \sim \mathbb{C})$ // First hidden state from all clusters
$\hat{\mathbf{y}} = (\varnothing_1, \ldots, \varnothing_j)$ // Final prediction placeholder
for $m \leftarrow 1 \ldots k$ **do**
  $\mathbf{a}_m, \mathbf{h}_m^d \leftarrow \text{EPN}(\mathbf{C}_\pi, \mathbf{h}_m^d)$ // Enhanced pointer attention over k predicted clusters
  $l \leftarrow \arg\max(\mathbf{a}_m)$
  $\hat{y}_m \leftarrow \mathbb{C}_l$ // Next cluster by highest attention index
end for

The models' exact layer dimensions are given in Appendix A.4, with the number of learnable parameters of each model varying by less than 5% per task. The AdamW (Loshchilov & Hutter, 2017) optimizer was used with weight decay coefficient 1e-3, learning rate (α) 1e-4, dropout rate of 0.05 and batch size 64, for 50–100 epochs. Experiments were performed on cloud-based GPU instances, with NVIDIA Quadro P6000 graphics cards (24 GB) and 8 CPU cores. To represent natural language entities in Section 5.4 we use the concatenated and averaged output of the last 4 layers of the cased, large version of BERT (Devlin et al., 2019), frozen at training to isolate the effect of compared clustering and permutation methods on performance.

The best performance is reported in **bold** and second best is underlined. Reported results are averaged over three full training runs, standard deviation is reported after the ± sign. We use V-Measure (Rosenberg & Hirschberg, 2007) and Kendall's Rank Correlation Coefficient (τ) as the primary clustering and permutation metrics respectively, scaled by a factor of a hundred for readability, following convention (Wang & Wan, 2019; Pandey & Chowdary, 2020).

5.1. Baselines

We present two groups of baselines for addressing the Catalog Problem. **i) Neural clustering methods with an added set-to-sequence module:** the module takes the predicted clusters and outputs their order via attention-based pointing. These methods include the pointwise Neural Clustering Process (NCP), the Clusterwise Clustering Process (CCP) and the Attentive Clustering Process (ACP) developed by (Pakman et al., 2020) and (Wang et al., 2021). **ii) Proposed variants of the set-to-sequence architecture:** these S2S variants enhance the pointer mechanism with the notion of predicting ordered *clusters*, as opposed to ordered *elements*. The first variant, called S2S-B (for "break"), adds a secondary decision of whether or not to start a new cluster in

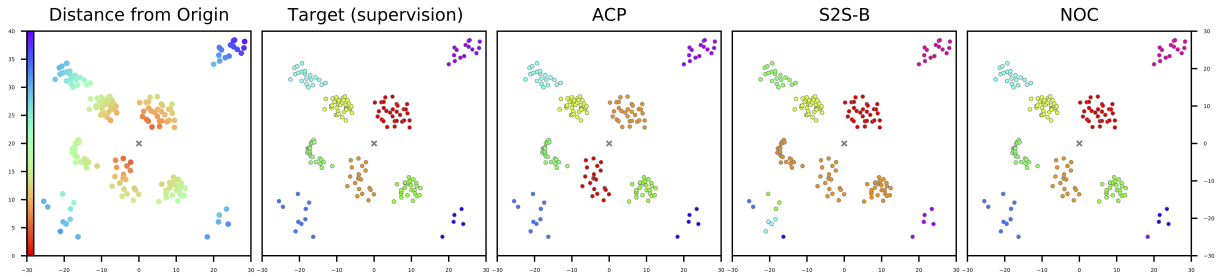

Figure 3. **Example of predicted ordered clusters**. Target (supervision) shows clusters and their order through colour, red being closest to the origin point (marked with a gray ×), dark blue and violet being furthest. Heat map (leftmost) indicates distance for individual points. The ACP prediction exhibits good clustering, but errs in the ordering (mistaken red and orange clusters). S2S-B exhibits good ordering, but incorrect clustering in the bottom-left quarter. NOC (ours) is closest to the target.

parallel to the selection of the set element to be placed next in the predicted sequence. The second variant, called S2S-C (for "clusterwise"), uses a threshold mechanism to select multiple elements forming a single cluster at each step. For details, see Appendix A.3.

5.2. Ordered Mixtures of Gaussians

This dataset consists of 2D coordinates for a number of points, generated from a mixture of a finite number of Gaussian distributions. The points should be clustered and the clusters ordered **by distance from the origin point (0,0)**, as in the supervision target. Following convention from probabilistic models for clustering (McLachlan & Basford, 1988), we introduce a random variable c_i signifying the cluster to which each data point x_i is assigned. The generation process creates a random number of clusters k, each with their own parameter vector μ_j controlling the distribution of the j-th cluster. For comparison with prior work (Pakman et al., 2020), we use a Chinese Restaurant Process with a single modification — the addition of a target order of the clusters, based on the Euclidean distance of their *centroids* from the origin point. An example of the joint prediction of per-element cluster assignments and predicted cluster order can be seen in Figure 3. The predicted order is denoted through colour gradient, with a bright red to deep blue and violet scale. In the figure, three separate predictions are displayed, one from the ACP model, one from a modification of set-to-sequence methods in the form of S2S-B and finally one from the proposed NOC model.

As shown in the rightmost column of Table 1, NOC outperforms other methods on both the clustering task, according to V-Measure, and the cluster ordering task, measured with Kendall's τ. Specifically it improves by +1.18 points over the second-best clustering method (ACP) and +2.51 over the second-best set-to-sequence method (S2S-B). Its performance appears relatively consistent, showing a smaller standard deviation over three full training runs.

5.3. Procedurally Generated Catalogs

The second experiment uses synthetic catalogs. These catalogs consist of varying-length sequences of clusters of elements, with repetition. Elements are colour-coded. These catalogs form the supervised training targets $\mathbf{y}_i$, with the unordered set of available atomic elements forming the inputs $\mathbb{X}_i$. The correct composition of individual sections and the structure of the overall catalog, in the form of the order of its sections, depends on n-th order interactions between the input elements.

For procedural generation, these interactions are formalized

	2D Gaussians		Synthetic Catalogs		PROCAT	
Method	V-Measure	Kendall's τ	V-Measure	Kendall's τ	V-Measure	Kendall's τ
NCP + S2S	91.52 ± 3.30	75.31 ± 4.5	63.12 ± 4.12	74.82 ± 5.1	25.42 ± 5.14	21.94 ± 4.3
CCP + S2S	93.94 ± 2.13	83.88 ± 4.2	79.41 ± 3.76	81.10 ± 3.9	37.41 ± 3.10	25.24 ± 4.0
ACP + S2S	<u>96.63 ± 1.82</u>	90.13 ± 3.7	<u>87.66 ± 3.91</u>	85.73 ± 3.2	<u>41.38 ± 3.88</u>	31.73 ± 3.1
S2S-B	89.37 ± 4.21	<u>95.89 ± 2.3</u>	78.39 ± 1.64	<u>92.13 ± 2.0</u>	39.01 ± 3.35	<u>44.39 ± 3.7</u>
S2S-C	92.45 ± 2.01	93.41 ± 2.1	75.83 ± 4.91	91.55 ± 3.3	36.71 ± 4.26	40.22 ± 4.2
NOC	**97.81 ± 0.92**	**98.40 ± 0.5**	**96.13 ± 1.28**	**95.84 ± 0.9**	**52.84 ± 3.15**	**56.67 ± 2.8**

Table 1. Clustering and permutation results of baselines and NOC over all three datasets. Best-performing results per column are in **bold**, second-best are <u>underlined</u>.

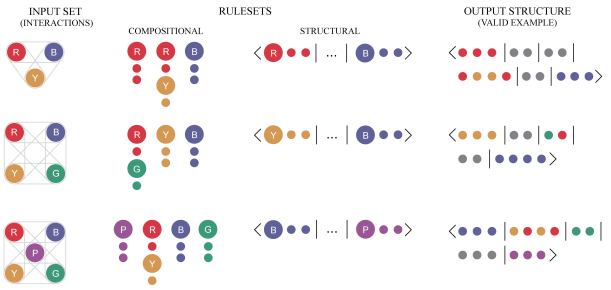

Figure 4. **Procedurally generated catalogs.** Relations between elements of the input set define the compositional and structural rules, which inform the generation of these synthetic datasets. A successful model should learn these rules from supervised exposure to the resulting synthetic datasets, and then be able to order new sets of elements according to the learned rules. One valid example is given for each input set (wrapped over 2 lines). See the second paragraph of Section 5.3 for a written description of the compositional and structural ruleset portrayed in the top row.

as compositional (intra-cluster) and structural (inter-cluster) rules. A simplified example of a compositional rule would be: *"if the input set contains only red, blue and yellow elements, a section containing red and yellow elements in 1:1 ratio is a valid section"*. An example of a structural rule could specify that (given the same input) the catalog has to begin with an all-red section or end on an all-blue one (top row of Figure 4). Compositional rules also include upper cluster cardinality constraints for valid sections. We use the tool made by Jurewicz & Derczynski (2022) to generate catalogs.

As shown in Table 2, neural clustering methods appear to be better at composing valid catalog sections but struggle with ordering sections into valid catalogs. This is indicated through two metrics – a *compositional score*, which is the percentage of predicted sections that were valid in accordance with the applicable n-th order ruleset, and a *structural score*, which is the percentage of valid predicted catalog structures (i.e. section ordering). By contrast, the adapted S2S models outperform neural clustering methods at correctly ordering sections, as measured via the structural score. NOC outperforms both methods on each of the two scores. This improvement is also reflected on the same test set in the more general but related V-Measure and Kendall's τ, shown in Table 1, where NOC surpasses the next-best models by +8.47 and +3.71 percentage points.

Among the sections predicted by neural clustering methods (NCP, CCP, ACP), the predominant error (present in 74% of invalid sections) stemmed from incorrect cluster cardinality, even though the models correctly predict the composition (15%) and ratio (11%) of elements to include. This error occurs despite the presence of mechanisms that could, in principle, allow for the learning of max-cardinality constraints: NCP constructs clusters element-by-element, further transforming the candidate cluster at each element's addition; CCP and ACP obtain a representation of the current candidate cluster before assigning candidate elements.

NOC overcomes this limitation through the addition of a cluster-level cardinality prediction mechanism and corresponding loss. It outperforms the second best method on the section composition task by +11.96, +13.01 and +15.65 percentage points with regards to the 3rd, 4th and 5th order relational ruleset respectively. It also performs better with regards to the structural score, offering a smaller but consistent improvement over the S2S-C and S2S-B methods by +4.54, +3.59 and +3.61 points, with respect to increasing n-th order rulesets.

5.4. PROCAT

The final experiment was performed on the PROCAT dataset (Jurewicz & Derczynski, 2021), using its provided training and testing split. All models were trained on approximately 9K product catalogs and tested on a separate set of 2K catalogs. Unlike the benchmarks provided with the PROCAT dataset, this formulation of the task mirrors the Catalog Problem exactly, with no information about the target number of sections (clusters) being available to the models. Individual elements were transformed into vector

	Compositional score			Structural score		
Method	$n=3$	$n=4$	$n=5$	$n=3$	$n=4$	$n=5$
NCP + S2S	64.13 ± 3.9	55.81 ± 4.6	51.82 ± 5.2	56.49 ± 4.0	51.87 ± 5.1	49.70 ± 6.8
CCP + S2S	75.40 ± 3.2	71.49 ± 4.3	65.11 ± 4.5	70.21 ± 3.5	68.39 ± 4.7	66.55 ± 5.4
ACP + S2S	<u>87.05 ± 1.7</u>	81.33 ± 1.9	<u>76.83 ± 2.2</u>	81.09 ± 2.2	76.34 ± 3.4	73.86 ± 3.8
S2S-B	84.99 ± 0.5	<u>82.90 ± 0.7</u>	74.82 ± 0.6	92.33 ± 1.5	<u>89.83 ± 2.1</u>	<u>87.31 ± 2.0</u>
S2S-C	82.03 ± 1.8	78.74 ± 2.1	72.13 ± 2.4	<u>92.49 ± 1.6</u>	87.41 ± 2.2	85.05 ± 2.3
NOC	**99.01 ± 0.3**	**95.91 ± 0.4**	**92.48 ± 0.4**	**97.03 ± 0.9**	**93.42 ± 1.0**	**90.92 ± 1.2**

Table 2. Results over procedurally generated catalogs, by n-th order relational ruleset. Best-performing results per column are in **bold**, second-best are <u>underlined</u>.

Figure 5. **PROCAT**. An example of three sequential sections predicted by the NOC as part of a larger catalog, from an input set of products from the PROCAT dataset. The prediction groups elements into complementary sections (the three pages shown above) and orders them into a rendered catalog. Here NOC correctly groups vegetables, meats and beverages together into consecutive sections.

representations via a pre-trained, frozen language model as described at the beginning of Section 5, removing its effect on the variation in performance on the downstream task. Figure 5 displays a sample catalog predicted by NOC from a PROCAT input set of product offers.

As the two rightmost columns of Table 1 show, the PROCAT structure prediction task is more difficult than the previous tasks. The best results in terms of both the clustering quality (via V-Measure) and section order (measured indirectly via Kendall's τ with regards to element order) are approximately 40% below the corresponding scores on the procedural task in its default configuration. One possible explanation stems from the existence of a higher number of reasonable substitutions for each element in any given section from the entire input set of initially available products. While also present in the procedural catalogs, this challenge becomes harder as the cardinality of the input increases from tens in the procedural case to hundreds in PROCAT.

NOC outperforms both neural clustering methods and the adjusted set-to-sequence models. While the overall pattern of neural clustering methods outperforming S2S-B and S2S-C in V-Measure is upheld, it is less pronounced (+2.37 points between ACP and S2S-B on PROCAT compared to +9.27 and +4.18 on the procedural and 2D Gaussian task respectively). The adjusted set-to-sequence models continue to outperform NCP, CCP and ACP on the ordering aspect of the task, with a margin of +12.66 points. NOC yields the best performance in terms of both partitional clustering and ordering, exceeding the relevant second-best methods by +11.46% and +12.28% respectively.

6. Conclusion

The posited Catalog Problem consists of learning to group elements and to order the groups. It poses a more difficult challenge than its individual components. Our work defined benchmark tasks representing this problem and presented approaches for them, including both adjusted baselines and a candidate approach, Neural Ordered Clusters (NOC). Existing neural clustering methods appear ineffective at learning cluster-level cardinality constraints. Our method offers an improvement in this area through its cluster cardinality prediction module. NOC outperforms adjusted S2S methods in terms of both clustering quality and accuracy of the predicted cluster order, indicating that structuring models to address adaptive ordered clustering leads to improved performance over standard S2S prediction.

The complexity and fluidity of intra- and inter-cluster relations result in the Catalog Problem remaining significantly more challenging than S2S processing. We proposed a predictive solution, where we trained the model to yield a single "ground truth" human-generated catalog given a set of products. Future work could consider a fully generative formulation of the problem that respects an unlimited number of valid solutions for both clustering and ordering.

References

Aljalbout, E., Golkov, V., Siddiqui, Y., Strobel, M., and Cremers, D. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bakkelund, D. Order preserving hierarchical agglomerative clustering. *Machine Learning*, 111(5):1851–1901, 2022.

Birhane, A. The impossibility of automating ambiguity. *Artificial Life*, 27(1):44–61, 2021.

Buchard, A., Bouvier, B., Prando, G., Beard, R., Livieratos, M., Busbridge, D., Thompson, D., Richens, J., Zhang, Y., Baker, A., et al. Learning medical triage from clinicians using deep q-learning. *arXiv e-prints*, pp. arXiv–2003, 2020.

Buolamwini, J. Limited vision: The undersampled majority. In *AI Now*. MIT Media Lab, 2017.

Chang, H., Yu, F., Wang, J., Ashley, D., and Finkelstein, A. Automatic triage for a photo series. *ACM Transactions on Graphics (TOG)*, 35(4):1–10, 2016.

Chu, K. C. Applications of artificial intelligence to chemistry. use of pattern recognition and cluster analysis to determine the pharmacological activity of some organic compounds. *Analytical chemistry*, 46(9):1181–1187, 1974.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, 2019.

Ergun, J., Feng, Z., Silwal, S., Woodruff, D. P., and Zhou, S. Learning-augmented k-means clustering. *arXiv preprint arXiv:2110.14094*, 2022.

Espino, J. U., Hogan, W. R., and Wagner, M. M. Telephone triage: a timely data source for surveillance of influenza-like diseases. In *AMIA Annual Symposium Proceedings*, volume 2003, pp. 215. American Medical Informatics Association, 2003.

Fernández-González, D. Multitask pointer network for multi-representational parsing. *Knowledge-Based Systems*, 236:107760, 2022.

Fisher, D., Xu, L., and Zard, N. Ordering effects in clustering. In *Machine Learning Proceedings 1992*, pp. 163–168. Elsevier, 1992.

Fisher, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172, 1987.

Frey, B. J. and Dueck, D. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

Gan, C., Zhou, R., Yang, J., and Shen, C. Cost-aware cascading bandits. *IEEE Transactions on Signal Processing*, 68:3692–3706, 2020.

Ge, L., Cai, Y., Weng, J., and Yuan, J. Hand pointnet: 3d hand pose estimation using point sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8417–8426, 2018.

Girgis, R., Golemo, F., Codevilla, F., Weiss, M., D'Souza, J. A., Kahou, S. E., Heide, F., and Pal, C. Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Janowitz, M. F. An order theoretic model for cluster analysis. *SIAM Journal on Applied Mathematics*, 34(1):55–72, 1978.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Johnson, S. C. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Jurewicz, M. and Derczynski, L. PROCAT: Product catalogue dataset for implicit clustering, permutation learning and structure prediction. In *Proceedings of the conference on Neural Information Processing Systems: Datasets and Benchmarks Track*, 2021.

Jurewicz, M. and Derczynski, L. Set interdependence transformer: Set-to-sequence neural networks for permutation learning and structure prediction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2022.

Kennedy, K., Aghababian, R. V., Gans, L., and Lewis, C. P. Triage: techniques and applications in decisionmaking. *Annals of emergency medicine*, 28(2):136–144, 1996.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Kwon, J.-m., Lee, Y., Lee, Y., Lee, S., Park, H., and Park, J. Validation of deep-learning-based triage and acuity score using a large national dataset. *PloS one*, 13(10): e0205836, 2018.

Le, M., Nauck, D., Gabrys, B., and Martin, T. Sequential clustering for event sequences and its impact on next process step prediction. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 168–178. Springer, 2014.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. Set transformer. In *International Conference on Machine Learning*, 2019.

Lee, S.-H. and Kim, C.-S. Deep repulsive clustering of ordered data based on order-identity decomposition. In *International Conference on Learning Representations*, 2020.

Li, Y., Hu, P., Liu, Z., Peng, D., Zhou, J. T., and Peng, X. Contrastive clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8547–8555, 2021.

Lin, Y., Li, M., Watanabe, Y., Kimura, T., Matsunawa, T., Nojima, S., and Pan, D. Z. Data efficient lithography modeling with transfer learning and active data selection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(10):1900–1913, 2018.

Loshchilov, I. and Hutter, F. Fixing weight decay regularization in adam. In *CoRR, abs/1711.05101, 2017.*, 2017.

Lupinacci Amaral, L. 'absentmindedly scrolling through nothing': liveness and compulsory continuous connectedness in social media. *Media, Culture & Society*, 43:016344372093945, 07 2020. doi: 10.1177/0163443720939454.

McLachlan, G. J. and Basford, K. E. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York, 1988.

Miles, J., Turner, J., Jacques, R., Williams, J., and Mason, S. Using machine-learning risk prediction models to triage the acuity of undifferentiated patients entering the emergency care system: a systematic review. *Diagnostic and prognostic research*, 4(1):1–12, 2020.

Noë, B., Turner, L. D., Linden, D. E., Allen, S. M., Winkens, B., and Whitaker, R. M. Identifying indicators of smartphone addiction through user-app interaction. *Computers in human behavior*, 99:56–65, 2019.

Pakman, A., Wang, Y., Mitelut, C., Lee, J., and Paninski, L. Neural clustering processes. In *International Conference on Machine Learning*, pp. 7455–7465. PMLR, 2020.

Pandey, D. and Chowdary, C. R. Modeling coherence by ordering paragraphs using pointer networks. *Neural Networks*, 126:36–41, 2020.

Peng, X., Li, Y., Tsang, I. W., Zhu, H., Lv, J., and Zhou, J. T. Xai beyond classification: Interpretable neural clustering. *Journal of Machine Learning Research*, 23(6): 1–28, 2022.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Rabiller, P., Boles, P., and Boashash, B. Ordered clustering: A way to simplify analysis of multichannel signals. In *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, pp. 237–242. IEEE, 2010.

Raji, D. How our data encodes systematic racism. *MIT Technology Review*, 10, dec 2020.

Ronen, M., Finder, S. E., and Freifeld, O. Deepdpm: Deep clustering with an unknown number of clusters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9861–9870, 2022.

Rosenberg, A. and Hirschberg, J. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pp. 410–420, 2007.

Shi, C. Pointer network solution pool: Combining pointer networks and heuristics to solve tsp problems. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pp. 1236–1242. IEEE, 2022.

Stevens, J. L. and Shanahan, K. J. Structured abstract: Anger, willingness, or clueless? understanding why women pay a pink tax on the products they consume. In *Creating Marketing Magic and Innovative Future Marketing Trends*, pp. 571–575. Springer, 2017.

Tian, F., Gao, B., Cui, Q., Chen, E., and Liu, T.-Y. Learning deep representations for graph clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

Turowski, K., Sreedharan, J. K., and Szpankowski, W. Temporal ordered clustering in dynamic networks. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 1349–1354. IEEE, 2020.

Van Dongen, S. M. *Graph clustering by flow simulation*. PhD thesis, Utrecht University, 2000.

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

Vial, D., Sanghavi, S., Shakkottai, S., and Srikant, R. Minimax regret for cascading bandits. *arXiv preprint arXiv:2203.12577*, 2022.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Vlasblom, J. and Wodak, S. J. Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC bioinformatics*, 10(1):1–14, 2009.

Wallach, D. and Goffinet, B. Mean squared error of prediction as a criterion for evaluating and comparing system models. *Ecological modelling*, 44(3-4):299–306, 1989.

Wang, T. and Wan, X. Hierarchical Attention Networks for Sentence Ordering. *AAAI*, 33:7184–7191, 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33017184.

Wang, Y., Lee, Y., Basu, P., Lee, J., Teh, Y. W., Paninski, L., and Pakman, A. Amortized probabilistic detection of communities in graphs. *arXiv preprint arXiv:2010.15727*, 2021.

Wen, J. Spectral-pointer network: Pre-sort leads the pointer network to elude the tsp vortex. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pp. 578–586. IEEE, 2022.

Wenzel, H., Smit, D., and Sardesai, S. A literature review on machine learning in supply chain management. In *Artificial Intelligence and Digital Transformation in Supply Chain Management: Innovative Approaches for Supply Chains. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 27*, pp. 413–441. Berlin: epubli GmbH, 2019.

Williams, R. J. and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Yin, Y., Meng, F., Su, J., Ge, Y., Song, L., Zhou, J., and Luo, J. Enhancing pointer network for sentence ordering with pairwise ordering predictions. In *AAAI*, volume 34, pp. 9482–9489, 2020.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

Zhu, Y., Zhou, K., Nie, J.-Y., Liu, S., and Dou, Z. Neural sentence ordering based on constraint graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 14656–14664, 2021.

Acknowledgements

This work was partly supported by an Innovation Fund Denmark research grant (number 9065-00017B) and by Tjek A/S.

Ethics Statement

Given the e-commerce context of the third presented dataset, we must highlight the wider problem of *endless scroll* user interfaces in product presentation apps and social media (Lupinacci Amaral, 2020). Although the PROCAT dataset is tailored to the prediction of cluster sequences of finite lengths, we cannot rule out the possibility of extending the proposed adaptive clustering and cluster ordering models to non-finite sets. It is also in principle possible to retrain the proposed models with additional inputs such as embedded personal preferences, making the predicted catalogs tailored to specific individuals, which has previously been linked to mental health issues in relation to smartphone addiction (Noë et al., 2019).

As with many machine learning systems, the results are not perfect, and sub-optimal predictions from NOC could silently disadvantage an end-user; for example a business may produce catalogs that don't make it easier for the reader to discover relevant, cost-saving offers, or an individual may receive an inaccurate medical analysis (in the case of the hypothesized medical triage use case).

Applying this tool may impact the employment of people performing creative catalog-related tasks, and further, might not even do the task as well as them. Product catalog design is considered something of an art among its practitioners, and there may be deep interactions not clearly evinced in training data that are lost by transiting the ownership of the catalog construction task from human subject matter experts to a machine learning model. Attempting to completely replace a human at this task may lead to both unsatisfactory and marginalizing results (Birhane, 2021).

We do not see any direct way for the presented methods to exacerbate bias against people of a certain gender, race, sexuality, or who have other protected characteristics. However, bias inherent to the marketing decisions made by people who have designed the catalogues contained in the PROCAT dataset, will be propagated by models trained on it. Negative biases in this particular scenario include as the *pink tax* (Stevens & Shanahan, 2017). In general, learning from socially-biased data and making predictions based on it will propagate those biases (Buolamwini, 2017; Raji, 2020).

Reproducibility Statement

In order to ensure reproducibility all code and datasets needed for repeated experiments have been made freely available, as described in detail in Appendix A.4 as part of the provided supplementary materials. The anonymized code repository includes a comprehensive readme.md file describing the necessary steps to set up the execution environment, download, generate and preprocess the datasets and run each of the experiments discussed in Section 5. The exact hyperparameters per experiment are stated both in the Appendix (A.4.1, A.4.2, A.4.3) and in the provided configuration files in the linked code repository. Additionally, a detailed description of the NOC algorithm is provided (1) to ensure that the method can be reimplemented, if necessary.

A. Appendix

A.1. Enhanced Pointer Network

In all reported experiments we use the same set-to-sequence module, the Enhanced Pointer Network (Yin et al., 2020), which is a pointer-attention based method inspired by the popular Pointer Network (Vinyals et al., 2015). It offers a performance improvement by leveraging two additional mechanisms for pairwise ordering predictions towards improved global and local coherence of the output sequence. Formally, the conditional probability of a predicted order $\hat{\mathbf{y}}$ is calculated as:

$$p_\theta(\hat{\mathbf{y}} \mid \mathbb{C}) = \prod_{j=1}^{K} p_\theta(\hat{y}_j \mid \hat{\mathbf{y}}_{<j}, \mathbf{C}_\pi, \mathbf{s}_c) \; ; \; \mathbf{s}_c = \text{PMA}(\text{SIT}(\mathbf{C}_\pi)) \tag{13}$$

$$p_\theta(\hat{y_j} \mid \hat{\mathbf{y}}_{<j}, \mathbb{C}) = \text{softmax}(\mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{h}_j^d + \mathbf{W}_2 \mathbf{M}_j)) \tag{14}$$

$$\mathbf{h}_j^d = \text{LSTM}(\mathbf{h}_{j-1}^d, \mathbf{c}_{j-1}) \; , \; \mathbf{h}_0^d = \mathbf{s}_c \tag{15}$$

where $\mathbf{v}$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are model parameters, k is the total number of clusters, tanh is the hyperbolic tangent nonlinearity, $\mathbf{g}_{j-1}$ is the fixed-length embedding of the cluster selected at the preceding step $j-1$ and $\mathbf{h}_j^d$ is the hidden state of the permutation module at current step i. The first hidden state $\mathbf{h}_0^d$ is initialized from the permutation invariant set representation of all previously predicted clusters $\mathbf{s}_c$, obtained via SIT and PMA. The $\mathbf{M}_j$ matrix provides additional context consisting of 2 kinds of information. The first is global orientation relating all remaining unordered clusters to one another. The second is local coherence between previously selected clusters and remaining candidates. This contextual information is obtained via HISTORY and FUTURE sub-modules from the original matrix of all cluster representations ($\mathbf{G}_k \approx \mathbb{C}$), which form the elements to be ordered. These two sub-modules output pairwise ordering predictions in relation to each candidate cluster, which are then combined to form $\mathbf{M}_j$. For exact implementation details, we refer the reader to Yin et al. (2020), but also provide more detail regarding these two sub-modules below.

The FUTURE sub-module calculates the probability of every currently (at j^{th} decoder step) unordered element x_u appearing before and after every other unordered element $x_{u\prime}$, denoted here as $p_\theta(\textit{before}, \textit{after} \mid x_u, x_{u\prime})$, which is represented by a softmaxed vector consisting of two elements ($\mathbf{p}_{u,u\prime} \in \mathbb{R}^2$) obtained in the following way:

$$\mathbf{z}_{u,u\prime} = \text{ReLU}(\mathbf{W}_c \times \text{ReLU}(\mathbf{W}_a \mathbf{e}_u + \mathbf{W}_b \mathbf{e}_{u\prime})), \tag{16}$$

$$\mathbf{p}_{u,u\prime} = \text{softmax}(\mathbf{W}_d \times \mathbf{z}_{u,u\prime}), \tag{17}$$

where $\times$ denotes matrix multiplication, ReLU() is the rectified linear unit nonlinear activation function, $\mathbf{e}_u$ and $\mathbf{e}_{u\prime}$ are the embedded representations of the unordered element x_u and $x_{u\prime}$ respectively (obtained by the encoder). Bias terms have been omitted for readability. All pairs of unordered elements are processed in this way and then transformed into a single vector $\mathbf{m}_{f,u}$, per unordered element:

$$\mathbf{m}_{f,u} = \frac{1}{|\mathbb{X}_u|} \left(\sum_{x_{u\prime} \in \mathbb{X}_u} [\mathbf{z}_{u,u\prime} \mid \mathbf{p}_{u,u\prime}] \right), \tag{18}$$

where $\mathbb{X}_u$ denotes the set of unordered elements except x_u and $[\mathbf{z}_{u,u\prime} \mid \mathbf{p}_{u,u\prime}]$ is a concatenation of the two listed vectors. This enables the model to exploit global relative orientation of other unordered elements when considering x_u.

The HISTORY sub-module performs analogous transformations for each unordered element x_u, but this time with regards to two previously ordered elements x_{j-1} and x_{j-2} (placed in the output sequence at the $j-1^{\text{th}}$ and $j-2^{\text{th}}$ index during preceding decoder steps). For x_{j-1} this takes the form of:

$$\mathbf{z}_{u,j-1} = \text{ReLU}(\mathbf{W}_c \times \text{ReLU}(\mathbf{W}_a \mathbf{e}_u + \mathbf{W}_b \mathbf{e}_{j-1})), \tag{19}$$

$$\mathbf{p}_{u,j-1} = \text{softmax}(\mathbf{W}_d \times \mathbf{z}_{u,j-1}). \tag{20}$$

The $\mathbf{z}_{u,j-1}$ and $\mathbf{p}_{u,j-2}$ vectors are obtained analogously, but through different learned $\mathbf{W}_*$ weight matrices (which the authors of the method justify by the fact that they refer to a relative distance of two places, instead of a single one). These are then used to obtain the $\mathbf{m}_{h1}$ and $\mathbf{m}_{h2}$ vectors by concatenating the corresponding $\mathbf{z}$ and $\mathbf{p}$ vectors, which are taken to encode left-side, local coherence (local referring to the fact that it's just two elements to the left of the current candidate that are being considered, as opposed to the entire left-side context encoded in the decoder's hidden state $\mathbf{h}_j^d$). Finally, the $\mathbf{m}_{f,u}$ vector (obtained by the FUTURE sub-module) and the $\mathbf{m}_{h1}$ and $\mathbf{m}_{h2}$ vectors (obtained by the HISTORY module) are concatenated into the $\mathbf{m}_u$ vector, which encodes both relative orientations of other unordered elements with respect to x_u and measures local coherence between two previously ordered elements and x_u. Such $\mathbf{m}_u$ vectors are obtained for all unordered elements at each j^{th} decoder step and packed into the $\mathbf{M}_j$ matrix.

A.2. NOC Training

NOC is trained in a teacher-forced way through a combined, weighted loss. The first loss factor comes from NOC_1 and is equivalent to the clustering loss of CCP (Pakman et al., 2020). At every j^{th} clustering step NOC_1 outputs a vector of cluster assignment probabilities for all unassigned set elements ($\mathbf{o}_j = (o_j^1, \ldots, o_j^m)$, where m is the number of remaining elements, which have not been assigned to any of the preceding clusters):

$$o_j^i = \text{NOC}_1(\mathbf{e}_i, \mathbb{X}_j) = p_\theta(x_i \in \hat{\mathbf{C}}_j) = p_\theta(\hat{c}_i = j), \tag{21}$$

where $\mathbb{X}_j$ represents the input set $\mathbb{X}$ at the j^{th} clustering step with regards to which elements have been assigned to which clusters, $\mathbf{e}_i$ is the embedded representation of the element x_i and θ represents all trainable model parameters. As per Equation 7, the $\mathbf{o}_j$ vector is obtained using the sigmoid activation function, making each o_j^i be a real number within the $[0, 1]$ range, which are treated as probabilities. During training the labelled examples are used to optimize an evidence lower bound (ELBO) thereof, by minimizing the expected KL divergence as described in the appendix of the paper introducing CCP (Pakman et al., 2020).

The second loss factor comes from NOC_2, which predicts a cardinality threshold for each j^{th} cluster as a positive integer ($t_j \in \mathbb{Z}^+$). This is compared to the target cardinality of the current cluster $|\mathbb{C}_j|$ (identified by the anchor element's cluster assignment) to obtain the mean squared error (Wallach & Goffinet, 1989). The third loss factor comes from NOC_3 and is equivalent to the Enhanced Pointer Network's loss. Given a batch $\mathbb{B}$ of m examples of the form $(\mathbb{X}, \mathbf{y})$:

$$\mathcal{L}(\mathbf{y}; \theta) = -\frac{1}{m} \sum_{(\mathbb{X},\, \mathbf{y}) \,\in\, \mathbb{B}} (\log\ p_\theta(\mathbf{y} \mid \mathbf{X}_\pi) + \lambda\, \mathcal{L}_{FH}), \tag{22}$$

where θ is the set of all model parameters and λ is a hyperparameter that balances the first term of the loss with $\mathcal{L}_{FH}$, a cross-entropy loss calculated from the pairwise predictions made by these FUTURE and HISTORY sub-modules (as outlined in Appendix A.1). For a visual explanation, the reader is referred to Figure 2 from the original paper by Yin et al. (2020). These three loss factors are weighted and the combined loss is minimized using the stochastic Adam optimizer (Kingma & Ba, 2015).

A.3. Set-to-Sequence Baselines

In this subsection, a more detailed description of the proposed S2S variants is given. The **S2S-B variant** utilizes pointer attention to select individual remaining set elements at each step, following the convention of Pointer Networks (Vinyals et al., 2015) and their enhancements (Yin et al., 2020). What distinguishes S2S-B from these models is an added prediction target which requires making $n - 1$ binary decisions, where n is the cardinality of the input set. At each step of the predicted permutation sequence, S2S-B indicates whether the currently selected element should be the last one of the current, open cluster. If so, this would indicate a *"break"* in the sequence, reminiscent of a page break in a product catalog. Once the last available element is reached, any remaining opened clusters are closed by default, hence $n - 1$. All previously pointed-to elements since the last break are considered members of the current open cluster.

The S2S-B model is thus capable of predicting a clustering where each element is assigned its own cluster and one where all elements belong to a single cluster. It is guaranteed to assign a cluster to every single element and can handle varying cardinality input sets, like all pointer networks. The first difficulty faced due to this particular modification stems from

highly skewed class distribution. Namely, we never complete (or break) a cluster after each element. This is mitigated via a class-weighted binary cross-entropy loss function:

$$\mathcal{L}_{\text{BCE-w}}(\theta) = -\frac{1}{m}\sum_{i=1}^{m}(w_b \times y_m \times \log(\hat{y}_m) + (1 - y_m) \times \log(1 - \hat{y}_m) \tag{23}$$

Where m is the number of training examples, w_b is the adjusted weight for the positive class, and y_i and $\hat{y}_i$ are the target and prediction respectably. This loss factor is then scaled and added to the negative loss likelihood loss used to train the pointer selection mechanism. The main disadvantage of this model is that it predicts meaningless in-cluster order, making the loss signal noisy. The order of elements within each cluster is meaningless within the confines of the presented Catalog Problem.

To mitigate this disadvantage, a second variant was developed. Referred to as **S2S-C** (for "clusterwise"), this model predicts the entire next cluster of elements at each step, instead of pointing to a single next element in the output sequence. Instead of performing n transformation steps in a loop over the entire input set, it outputs an attention vector over all available elements until there are none left. Thus it is also bound between assigning all elements to a single cluster or every element to its own cluster, much like S2S-B, guaranteeing cluster assignment for each element of the input set.

In order to predict clusters of adaptive, input-dependent cardinality, the formula for obtaining the pointer-attention vector over available elements had to be adjusted. The softmax operator was replaced with the sigmoid function (σ) and a threshold (t_a) of 0.5 was adopted. At each step $j \in \{1, 2 \ldots, n\}$ every element with a corresponding attention value (a_i^j) above the threshold is thus assigned to the next cluster:

$$\mathbf{a}_i = \sigma(\mathbf{v}^\top \tanh(\mathbf{W}_2\mathbf{E}_\pi + \mathbf{W}_1\mathbf{h}_i^d)) \tag{24}$$

$$\hat{y}_i^j = \begin{cases} 0, & \text{if } a_i^j < t_a \\ 1, & \text{otherwise} \end{cases} \tag{25}$$

During training, the S2S-C model was teacher-forced (Williams & Zipser, 1989) to prevent the cascading impact of incorrect initial cluster assignment on subsequent computation steps, which is a known challenge in certain areas of machine learning, such as the multi-armed bandit problem (Gan et al., 2020). This is not a departure from the other tested models (with the exception of S2S-B), as all neural clustering baselines are also teacher-forced during training, as per author implementations of the papers that originally introduced them.

A.4. Code, Datasets and Parameters

The code required for all three of the main experiments can be found in a fully anonymized repository under the following link:

`https://github.com/mateuszjurewicz/neural-ordered-clusters`

Follow the instructions provided in the readme.md document to set up the necessary environment locally, via the requirements.txt file listing all necessary packages and their versions.

In the following sections we describe each dataset in more detail, including how to download or generate it. All datasets are freely available under publicly accessible links. Additionally, each section contains the specific hyperparameters used for repeated experiments as well as the exact number of layers and parameters per tested NOC model.

A.4.1. Ordered Mixtures of 2D Gaussians

Data. The dataset for predicting ordered clusters of 2D Gaussians (based on their distance from the origin point) is synthetically generated when running the experiment via the linked run_gauss2D.py file. The full, default configuration is given in the parser arguments (nothing should require adjustment to run the equivalent experiment). This includes a default seed, which should help ensure repeatability. In the provided experiments we generate 30K batches of 64 examples each, for a total of just under 2 million individual training examples for a full run. Each example is a set of 5 to 100 individual points characterized by their coordinates, generated through the Chinese Restaurant Process with dispersion parameter α set to 0.7 for all experiments. Unlike the batch generation process used by Pakman et al. (2020), we generate batches with diverse number of clusters and cluster cardinalities in each example.

Hyperparameters. The training regimen includes a learning rate adjustment from 1e-4 to 5e-5 at the 15K-th batch and 1e-5 at the 20K-th batch. The AdamW (Loshchilov & Hutter, 2017) optimizer was used with a weight decay coefficient of 1e-3. Additionally, the default weights per loss factor are provided. The main clustering loss factor λ_c is equal to 1.0, the cluster ordering loss factor is set to λ_o = 4.0 and the cardinality prediction loss factor λ_k = 3e-3. A 100 inferences samples is generated by default during validation, final metrics being calculated for the clustering prediction with the highest probability.

Model parameters. The NOC model with reported performance had over 12mil trainable parameters. The element and cluster encoding functions, each consisting of three stacked ISAB layers had the input and hidden dimensions of 128. The set pooling functions consisted of two stacked ISAB layers followed by a PMA layer, also with 128 dimensions. The NOC_1 clustering module consistently uses a Parametric Rectified Linear Unit (PReLU) as the nonlinearity (He et al., 2015).

A.4.2. Procedurally Generated Catalogs

Data. The dataset for predicting the cluster composition (sections of offer tokens) and structure (order of these sections) of synthetic catalogs is automatically generated when running the linked run_synthetic.py experiment script with default parser arguments. This script loads the provided configuration file synthetic_rulesets.json which specifies all compositional and structural rulesets to which the generated synthetic catalogs will adhere. In all reported experiments we refer to this default set of rulesets, but encourage researchers to treat it as an easy-to-edit, flexible configuration that can be adjusted for other exploratory experiments.

For the experiments, we generate 300K synthetic catalogs for the training set and 75K for the validation and test sets (split into 15 data-loaders). Each example consists of 35-50 offer tokens, each batch consists of 64 examples with varied number of clusters and cluster cardinalities in each batch. The NOC model is trained over 250K batch iterations, the equivalent of 50 epochs.

Hyperparameters. The procedurally generated catalog training regimen includes a learning rate adjustment from 1e-4 to 5e-5 at the 100K-th batch iteration and 1e-5 at the 200K-th. The AdamW (Loshchilov & Hutter, 2017) optimizer was used with a weight decay coefficient of 1e-3. Additionally, the default weights per loss factor are provided. The main clustering loss factor λ_c is equal to 1.0, the cluster ordering loss factor is set to λ_o = 15.0 and the cardinality prediction loss factor λ_k = 0.1. A hundred inferences samples is generated by default during validation, final metrics being calculated for the clustering prediction with the highest probability.

Model parameters. The NOC model with reported performance had over 18mil trainable parameters. The element and cluster encoding functions, each consisting of four stacked ISAB layers had the input and hidden dimensions of 128. The set pooling functions consisted of three stacked ISAB layers followed by a PMA layer, also with 128 dimensions. The NOC_1 clustering module consistently uses a Parametric Rectified Linear Unit (PReLU) as the nonlinearity (He et al., 2015).

A.4.3. PROCAT

Data. The PROCAT dataset is freely available under the following link:

https://figshare.com/articles/dataset/PROCAT_Product_Catalogue_Dataset_for_Implicit_Clustering_Permutation_Learning_and_Structure_Prediction/14709507

We follow the provided train - test split of 8K - 2K catalogs and all pre-processing steps from the original paper (Jurewicz & Derczynski, 2021). The provided section break tokens are removed in the pre-processing to enable the prediction of input-dependent number of sections. Elements are by default truncated to 512 dictionary tokens for the language-specific BERT model, available in the linked hugging face repository and the suggested max-offer threshold of 200 per catalog is followed. Batches of 64 catalogs are used. The proposed NOC model is trained for 12.5K batch-iterations, the equivalent of 100 epochs.

Hyperparameters. The PROCAT training regimen includes a learning rate adjustment from 1e-4 to 5e-5 at the 5K-th batch iteration and 1e-5 at the 10K-th. The AdamW (Loshchilov & Hutter, 2017) optimizer was used with a weight decay coefficient of 1e-3. Additionally, the default weights per loss factor are provided. The main clustering loss factor λ_c is equal to 1.0, the cluster ordering loss factor is set to λ_o = 10.0 and the cardinality prediction loss factor λ_k = 0.5. A hundred inferences samples is generated by default during validation, final metrics being calculated for the clustering prediction with the highest probability.

Model parameters. The NOC model with reported performance had 23mil trainable parameters (not including the BERT model, which was frozen during training). The element and cluster encoding functions, each consisting of 5 stacked ISAB layers had the input and hidden dimensions of 128. The set pooling functions consisted of 4 stacked ISAB layers followed by a PMA layer, also with 128 dimensions. The NOC_1 clustering module consistently uses a Parametric Rectified Linear Unit (PReLU) as the nonlinearity (He et al., 2015).