Conditional Set Generation Using SEQ2SEQ Models

Aman Madaan, Dheeraj Rajagopal, Niket Tandon[†]**,**
Yiming Yang, Antoine Bosselut[‡]
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[†] Allen Institute for Artificial Intelligence, Seattle, WA, USA
[‡] EPFL, Switzerland
amadaan@cs.cmu.edu

Abstract

Conditional set generation learns a mapping from an input sequence of tokens to a set. Several NLP tasks, such as entity typing and dialogue emotion tagging, are instances of set generation. SEQ2SEQ models, a popular choice for set generation, treat a set as a sequence and do not fully leverage its key properties, namely order-invariance and cardinality. We propose a novel algorithm for effectively sampling informative orders over the combinatorial space of label orders. We jointly model the set cardinality and output by prepending the set size and taking advantage of the autoregressive factorization used by SEQ2SEQ models. Our method is a model-independent data augmentation approach that endows any SEQ2SEQ model with the signals of order-invariance and cardinality. Training a SEQ2SEQ model on this augmented data (without any additional annotations) gets an average relative improvement of 20% on four benchmark datasets across various models: BART-base, T5-11B, and GPT3-175B.[1]

1 Introduction

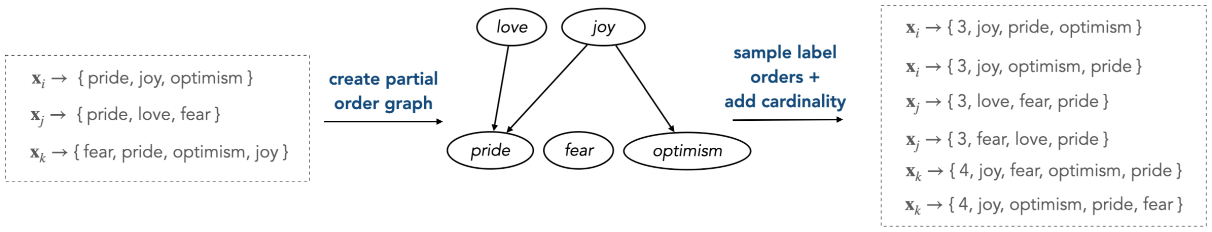
Conditional set generation is the task of modeling the distribution of an output set given an input sequence of tokens (Kosiorek et al., 2020). Several NLP tasks are instances of set generation, including open-entity typing (Choi et al., 2018; Dai et al., 2021), fine-grained emotion classification (Demszky et al., 2020), and keyphrase generation (Meng et al., 2017; Yuan et al., 2020; Ye et al., 2021). The recent successes of the pretraining-finetuning paradigm have encouraged a formulation of set generation as a SEQ2SEQ generation task (Vinyals et al., 2016; Yang et al., 2018; Meng et al., 2019; Ju et al., 2020).

In this paper, we posit that modeling set generation as a vanilla SEQ2SEQ generation task is suboptimal, because the SEQ2SEQ formulations do not explicitly account for two key properties of a set output: *order-invariance* and *cardinality*. Forgoing order-invariance, vanilla SEQ2SEQ generation treats a set as a sequence, assuming an arbitrary order between the elements it outputs. Similarly, the cardinality of sets is ignored, as the number of elements to be generated is typically not modeled.

Prior work has highlighted the importance of these two properties for set output through loss functions that encourage order invariance (Ye et al., 2021), exhaustive search over the label space for finding an optimal order (Qin et al., 2019; Rezatofighi et al., 2018; Vinyals et al., 2016), and post-processing the output (Nag Chowdhury et al., 2016). Despite the progress, several important gaps remain. First, exhaustive search does not scale with large output spaces typically found in NLP problems, thus stressing the need for an optimal sampling strategy for the labels. Second, cardinality is still not explicitly modeled in the SEQ2SEQ setting despite being an essential aspect for a set. Finally, architectural modifications required for specialized set-generation techniques might not be viable for modern large-language models.

We address these challenges with a novel data augmentation strategy. Specifically, we take advantage of the auto-regressive factorization used by SEQ2SEQ models and (i) impose an *informative* order over the label space, and (ii) explicitly model *cardinality*. First, the label sets are converted to sequences using informative orders by grouping labels and leveraging their dependency structure. Our method induces a partial order graph over label space where the nodes are the labels, and the edges denote the conditional dependence relations. This graph provides a natural way to obtain informative orders while reinforcing order-invariance. Specifically, sequences obtained via topological traversals of this graph allow independent labels to appear at different locations in the sequence, while restricting order for dependent labels. Next, we

[1]Code to use SETAUG available at: https://setgen.structgen.com



$x_i \rightarrow$ { pride, joy, optimism }
$x_j \rightarrow$ { pride, love, fear }
$x_k \rightarrow$ { fear, pride, optimism, joy }
create partial order graph
love
joy
pride
fear
optimism
sample label orders + add cardinality
$x_i \rightarrow$ { 3, joy, pride, optimism }
$x_i \rightarrow$ { 3, joy, optimism, pride }
$x_j \rightarrow$ { 3, love, fear, pride }
$x_j \rightarrow$ { 3, fear, love, pride }
$x_k \rightarrow$ { 4, joy, fear, optimism, pride }
$x_k \rightarrow$ { 4, joy, optimism, pride, fear }

Figure 1: An illustrative task where given an input $\boldsymbol{x}$, the output is a set of emotions. Our method first discovers a partial order graph (middle) in which specific labels (joy) come before more general labels (pride). Listing the specific labels first gives the model more clues about the rest of the set. Topological samples from this partial order graph are label sequences that can be efficiently generated using SEQ2SEQ models. The size of each set is also added as the first element for joint modeling of output with size.

jointly model a set with its cardinality by simply prepending the set size to the output sequence. This strategy aligns with the current trend of very large language models which do not lend themselves to architectural modifications but increasingly rely on the informativeness of the inputs (Yang et al., 2020; Liu et al., 2021).

Figure 1 illustrates the key intuitions behind our method using sample task where given an input $\boldsymbol{x}$ (say a conversation), the output is a set of emotions ($\mathbb{Y}$). To see why certain orders might be more meaningful, consider a case where one of the emotions is *joy*, which leads to a more general emotion of *pride*. After first generating *joy*, the model can generate *pride* with certainty (*joy* leads to *pride* in all samples). In contrast, the reverse order (generating *pride* first) still leaves room for multiple possible emotions (joy and love). The order [joy, pride] is thus more informative than [pride, joy]. The cardinality of a set can also be helpful. In our example, joy contains two sub-emotions, and love contains one. A model that first predicts the number of sub-emotions can be more precise and avoid over-generation, a significant challenge with language generation models (Welleck et al., 2020; Fu et al., 2021). We efficiently sample such informative orders from the combinatorial space of all possible orders and jointly model cardinality by leveraging the auto-regressive nature of SEQ2SEQ models.

Our contributions

(i) We show an efficient way to model sequence-to-set prediction as a SEQ2SEQ task by jointly modeling the cardinality and augmenting the training data with informative sequences using our novel SETAUG data augmentation approach. (§3.1, 3.2).

(ii) We theoretically ground our approach: treating the order as a latent variable, we show that our method serves as a better proposal distribution in a variational inference framework. (§3.1)

(iii) With our approach, SEQ2SEQ models of different sizes achieve a ∼20% relative improvement on four real-world tasks, with no additional annotations or architecture changes. (§4).

2 Task

We are given a corpus $\mathcal{D} = \{(\boldsymbol{x}_t, \mathbb{Y}_t)\}_{t=1}^m$ where $\boldsymbol{x}_t$ is a sequence of tokens and $\mathbb{Y}_t = \{y_1, y_2, \ldots, y_k\}$ is a set. For example, in multi-label fine-grained sentiment classification, $\boldsymbol{x}_t$ is a paragraph, and $\mathbb{Y}_t$ is a set of sentiments expressed by the paragraph. We use y_i to denote an output symbol, $[y_i, y_j, y_k]$ to denote an ordered sequence of symbols and $\{y_i, y_j, y_k\}$ to denote a set.

2.1 Set generation using SEQ2SEQ model

Task Given a corpus $\{(\boldsymbol{x}_t, \mathbb{Y}_t)\}_{t=1}^m$, the task of conditional set generation is to efficiently estimate $p(\mathbb{Y}_t \mid \boldsymbol{x}_t)$. SEQ2SEQ models factorize $p(\mathbb{Y}_t \mid \boldsymbol{x}_t)$ autoregressively (AR) using the chain rule:

$$\begin{aligned} p(\mathbb{Y}_t \mid \boldsymbol{x}_t) &= p(y_1, y_2, \ldots, y_k \mid \boldsymbol{x}_t) \\ &= p(y_1 \mid \boldsymbol{x}_t) \prod_{j=2}^{k} p(y_j \mid \boldsymbol{x}_i, y_1 \ldots y_{j-1}) \end{aligned}$$

where the order $\mathbb{Y}_t = [y_1, y_2, \ldots, y_k]$ factorizes the joint distribution using chain rule. In theory, any of the $k!$ orders can be used to factorize the same joint distribution. In practice, the choice of order is important. For instance, Vinyals et al. (2016) show that output order affects language modeling performance when using LSTM based SEQ2SEQ models for set generation.

Consider an example input-output pair $(\boldsymbol{x}_t, \mathbb{Y}_t = \{\mathrm{y}_1, \mathrm{y}_2\})$. By chain rule, we have the following equivalent factorizations of this sequence: $p(\mathbb{Y}_t \mid \boldsymbol{x}_t) = p(\mathrm{y}_1 \mid \boldsymbol{x})p(\mathrm{y}_2 \mid \boldsymbol{x}, \mathrm{y}_1) = p(\mathrm{y}_2 \mid \boldsymbol{x})p(\mathrm{y}_1 \mid \boldsymbol{x}, \mathrm{y}_2)$. However, order-invariance is only guaranteed with *true* conditional probabilities, whereas the conditional probabilities used to factorize a sequence are *estimated* by a model from a corpus. Thus, dependening on the order, the sequence factorizes as either $\hat{p}(\mathrm{y}_1 \mid \boldsymbol{x})\hat{p}(\mathrm{y}_2 \mid \boldsymbol{x}, \mathrm{y}_1)$ or $\hat{p}(\mathrm{y}_2 \mid \boldsymbol{x})\hat{p}(\mathrm{y}_1 \mid \boldsymbol{x}, \mathrm{y}_2)$, which are not necessarily equivalent. Further, one of these two factorizations may be better represented in the training data, and thus lead to better samples. For instance, if the training data always contains $\mathrm{y}_1, \mathrm{y}_2$ in the order $[\mathrm{y}_1, \mathrm{y}_2]$, $\hat{p}(\mathrm{y}_2 \mid \boldsymbol{x})\hat{p}(\mathrm{y}_1 \mid \boldsymbol{x}, \mathrm{y}_2)$ will be ~ 0.

Order will also be immaterial if the labels are conditionally independent given the input (Section B.3). However, this assumption is often not valid in practice, especially for NLP, where labels typically share a semantic relationship.

3 Method

This section expands on two critical components of our system, SETAUG. Section 3.1 presents TSAMPLE, a method to create informative orders over sets tractably. Section 3.2 presents our method for jointly modeling cardinality and set output.

3.1 TSAMPLE: Adding informative orders for set output

As discussed in Section 2, SEQ2SEQ formulation requires the output to be in a sequence. Prior work (Vinyals et al., 2016; Rezatofighi et al., 2018; Chen et al., 2021) has noted that listing the output in orders that have the highest conditional likelihood given the input is an optimal choice. Unlike these methods, we sidestep exhaustive searching during training using our proposed approach TSAMPLE.

Our core insight is that knowing the optimal order between pairs of symbols in the output drastically reduces the possible number of permutations. We thus impose pairwise order constraints for labels. Specifically, given an output set $\mathbb{Y}_t = \mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_k$, if $\mathrm{y}_i, \mathrm{y}_j$ are independent, they can be added in an arbitrary order. Otherwise, an order constraint is added to the order between $\mathrm{y}_i, \mathrm{y}_j$.

Learning pairwise constraints We estimate the dependence between elements $\mathrm{y}_i, \mathrm{y}_j$ using pointwise mutual information: $\texttt{pmi}(\mathrm{y}_i, \mathrm{y}_j) = \log p(\mathrm{y}_i, \mathrm{y}_j)/p(\mathrm{y}_i)p(\mathrm{y}_j)$. Here, $\texttt{pmi}(\mathrm{y}_i, \mathrm{y}_j) > 0$ indicates that the labels $\mathrm{y}_i, \mathrm{y}_j$ co-occur more than would be expected under the conditions of independence (Wettler and Rapp, 1993). We use $\texttt{pmi}(\mathrm{y}_i, \mathrm{y}_j) > \alpha$ to filter our such pairs of dependent pairs, and perform another check to determine if the order between them should be fixed. For each dependent pair $\mathrm{y}_i, \mathrm{y}_j$, the order is constrained to be $[\mathrm{y}_i, \mathrm{y}_j]$ (y_j should come after y_i) if $\log p(\mathrm{y}_j \mid \mathrm{y}_i) - \log p(\mathrm{y}_i \mid \mathrm{y}_j) > \beta$, and $[\mathrm{y}_j, \mathrm{y}_i]$ otherwise. Intuitively, $\log p(\mathrm{y}_j \mid \mathrm{y}_i) - \log p(\mathrm{y}_i \mid \mathrm{y}_j) > \beta$ implies that knowledge that a set contains y_i, increases the probability of y_j being present. Thus, in an autoregressive setting, fixing the order to $[\mathrm{y}_i, \mathrm{y}_j]$ will be more efficient for generating a set with $\{\mathrm{y}_i, \mathrm{y}_j\}$.

Generating samples To systematically create sequences that satisfy these constraints, we construct a topological graph G_t where each node is a label $\boldsymbol{y}_i \in \mathbb{Y}_t$, and the edges are determined using the $\texttt{pmi}$ and the conditional probabilities as outlined above (Algorithm 1). The required permutations can then be generated as topological traversals G_t (Figure 2). We begin the traversal from a different starting node to generate diverse samples. We call this method TSAMPLE. Our method of generating graphs avoids cycles by design (proof in B.4), and thus topological sort remains well-defined. Later, we show that TSAMPLE can be interpreted as a proposal distribution in variational inference framework, which distributes the mass uniformly over informative orders constrained by the graph.

Do pairwise constraints hold for longer sequences? While TSAMPLE uses pairwise (and not higher-order) constraints for ordering variables, we note that the pairwise checks remain relevant with extra variables. First, dependence between pair of variables is retained in joint distributions involving more variables ($\mathrm{y}_i \not\perp\!\!\!\perp \mathrm{y}_j \implies \mathrm{y}_i \not\perp\!\!\!\perp \mathrm{y}_j, \boldsymbol{y}_k$) for some $\boldsymbol{y}_k \in \mathbb{Y}$ (Appendix B.1). Further, if $\mathrm{y}_i, \mathrm{y}_j \perp\!\!\!\perp \boldsymbol{y}_k$, then it can be shown that $p(\mathrm{y}_i \mid \mathrm{y}_j) > p(\mathrm{y}_j \mid \mathrm{y}_i) \implies p(\mathrm{y}_i \mid \mathrm{y}_j, \boldsymbol{y}_k) > p(\mathrm{y}_j \mid \mathrm{y}_i, \boldsymbol{y}_k)$ (Appendix B.2). The first property shows that the pairwise dependencies hold in the presence of other set elements. The second property shows that an informative order continues to be informative when additional independent symbols are added. Thus, using pairwise dependencies between the set elements is still effective. Using higher-order dependencies might be suboptimal for

Algorithm 1 Generating permutations for $\mathbb{Y}_t$

Input: Set $\mathbb{Y}_t$, number of permutations n
Parameter: α, β
Output: n topological sorts over $G_t(V, E)$

```
1: Let $V = \mathbb{Y}_t, E = \emptyset$.
2: for $y_i, y_j \in \mathbb{Y}_t$ do
3:     if $pmi(y_i, y_j) > \alpha; \lg p(y_i \mid y_j) - \lg p(y_j \mid y_i) > \beta$ then
4:         $E = E \cup y_j \rightarrow y_i$
5:     end if
6: end for
7: return topo_sort$(G_t(V, E), n)$
```

practical reasons: higher-order dependencies (or including $\boldsymbol{x}_t$) might not be accurately discovered due to sparsity, and thus cause spurious orders.

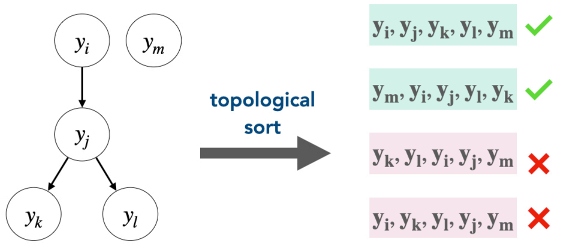
Finally, we note that if all the labels are independent, then the order is guaranteed not to matter (Lemma B.3, also shown empirically in Appendix G). Thus, our method will only be useful when labels have some degree of dependence.

Complexity analysis Let $\mathbb{Y}$ be the label space, $(\boldsymbol{x}_t, \mathbb{Y}_t)$ be a particular training example, N be the size of the training set, and c be the maximum number of elements for any set $\mathbb{Y}_t$ in the input. Our method requires three steps: i) iterating over training data to learn conditional probabilities and pmi, and ii) given a $\mathbb{Y}_t$, building the graph G_t (Algorithm 1), and iii) doing topological traversals over G_t to create samples for $(\boldsymbol{x}_t, \mathbb{Y}_t)$.

The time complexity of the first operation is $\mathcal{O}(Nc^2)$: for each element of the training set, the pairwise count for each pair y_i, y_j and unigram count for each y_i is calculated. The pairwise counts can be used for calculating joint probabilities. In principle, we need $\mathcal{O}(|\mathbb{Y}|^2)$ space for storing the joint probabilities. In practice, only a small fraction of the combinations will appear $|\mathbb{Y}|^2$ in the corpus.

Given a set $\mathbb{Y}_t$ and the conditional probabilities, the graph G_t is created in $\mathcal{O}(c^2)$ time. Then, generating k samples from G_t requires a topological sort, for $\mathcal{O}(kc)$ (or $\mathcal{O}(c)$ per traversal). For training data of size N, the total time complexity is $\mathcal{O}(Nck)$. The entire process of building the joint counts and creating graphs and samples takes less than five minutes for all the datasets on an 80-core Intel Xeon Gold 6230 CPU.

Interpreting TSAMPLE as a proposal distribution over orders We show that our method of augmenting permutations to the training data can be interpreted as an instance of variational inference with the order as a latent variable, and TSAMPLE as an instance of a richer proposal distribution.



y_i
y_m
y_j
y_k
y_l
topological sort
y_i, y_j, y_k, y_l, y_m ✓
y_m, y_i, y_j, y_l, y_k ✓
y_k, y_l, y_i, y_j, y_m ✗
y_i, y_k, y_l, y_j, y_m ✗

Figure 2: Our sampling method TSAMPLE first builds a graph G_t over the set $\mathbb{Y}_t$, and then samples orders from G_t using topological sort (topo_sort). The topological sorting rejects samples that do not follow the conditional probability constraints.

Let π_j be the j^{th} order over $\mathbb{Y}_t$ (out of $|\mathbb{Y}_t|!$ possible orders Π), and $\pi_j(\mathbb{Y}_t)$ be the sequence of elements in $\mathbb{Y}_t$ arranged with order π_j. Treating π as a latent random variable, the output distribution can then be recovered by marginalizing over all possible orders Π:

$$\log p_\theta(\mathbb{Y}_t \mid \boldsymbol{x}_t) = \log \sum_{\pi_z \in \Pi} p_\theta(\pi_z(\mathbb{Y}_t) \mid \boldsymbol{x}_t)$$

where p_θ is the SEQ2SEQ conditional generation model. While summing over Π is intractable, standard techniques from the variational inference framework allow us to write a lower bound (ELBO) on the actual likelihood:

$$\begin{aligned}\log p_\theta(\mathbb{Y}_t \mid \boldsymbol{x}_t) &= \log \sum_{\pi_{\boldsymbol{z}} \in \Pi} p_\theta(\pi_z(\mathbb{Y}_t) \mid \boldsymbol{x}_t) \\ &\geq \underbrace{\mathbb{E}_{q_\phi(\pi_{\boldsymbol{z}})}\left[\log \frac{p_\theta(\pi_{\boldsymbol{z}}(\mathbb{Y}_t) \mid \boldsymbol{x}_t)}{q_\phi(\pi_{\boldsymbol{z}})}\right]}_{\text{ELBO}}\end{aligned}$$

In practice, the optimization procedure draws k samples from the proposal distribution q to optimize a weighted ELBO (Burda et al., 2016; Domke and Sheldon, 2018). Crucially, q can be fixed (e.g., to uniform distribution over the orders), and in such cases only θ are learned (Appendix H). The data augmentation approach used for XL-NET (Yang et al., 2019b) can be interpreted with this framework. In their case, the proposal distribution q is fixed to a uniform distribution for generating orders over tokens. The variational interpretation also indicates that it might be possible to improve language modeling by using a different, more informative

q. Investigating such proposal distribution for language modeling is an interesting future work.

TSAMPLE can thus be seen as a particular proposal distribution that assigns all the support to the topological ordering over the label dependence graphs. We experiment with sampling from a uniform distribution over the samples (referred to as RANDOM experiments in our baseline setup). The idea of using an informative proposal distribution over space of structures to do variational inference has also been used in the context of grammar induction (Dyer et al., 2016) and graph generation (Jin et al., 2018; Chen et al., 2021). Our formulation is closest in spirit to Chen et al. (2021). However, the set of nodes to be ordered is already given in their graph generation setting. In contrast, we infer the order and the set elements jointly from the input.

3.2 Modeling cardinality

Let $m = |\mathbb{Y}_t|$ be the cardinality (number of elements) in $\mathbb{Y}_t$. Our goal is to jointly estimate m and $\mathbb{Y}_t$ (i.e., $p(m, \mathbb{Y}_t \mid \boldsymbol{x}_t)$). Additionally, we want the model to use the cardinality information for generating $\mathbb{Y}_t$. We add the cardinality information at the beginning of the sequence, thus converting a sample $(\boldsymbol{x}_t, \mathbb{Y}_t)$ to $(\boldsymbol{x}_t, [|\mathbb{Y}_t|, \pi(\mathbb{Y}_t)])$, and then train our SEQ2SEQ model as usual from $\boldsymbol{x} \rightarrow [|\mathbb{Y}_t|, \pi(\mathbb{Y}_t)]$. Here π is some ordering that converts the set $\mathbb{Y}_t$ to a sequence. As SEQ2SEQ models use autoregressive factorization, listing the order information first ensures that the distribution $p([|\mathbb{Y}_t|, \pi(\mathbb{Y}_t)] \mid \boldsymbol{x}_t)$ factorizes as $p([|\mathbb{Y}_t|, \pi(\mathbb{Y}_t)] \mid \boldsymbol{x}_t) = p(|\mathbb{Y}_t| \mid \boldsymbol{x}_t) p(\pi(\mathbb{Y}_t) \mid |\mathbb{Y}_t|, \boldsymbol{x}_t)$. Thus, the generation of $\mathbb{Y}_t$ is conditioned on the input and the cardinality $|\mathbb{Y}_t|$ (due to $p(\pi(\mathbb{Y}_t) \mid |\mathbb{Y}_t|, \boldsymbol{x}_t)$ term).

Why should cardinality help? Unlike models like deep sets (Zhang et al., 2019a), SEQ2SEQ models are not restricted by the number of elements generated. However, adding cardinality information has two potential benefits: i) it can help avoid over-generation (Welleck et al., 2020; Fu et al., 2021), and ii) unlike free-form text output, the distribution of the set output size ($p(|\mathbb{Y}_t| \mid \boldsymbol{x}_t)$) might benefit the model to adhere to the set size constraint. Thus, information on the predicted size can be beneficial for the model to predict the elements to be generated.

4 Experiments

SETAUG comprises: i) TSAMPLE, a way to generate informative orders to convert sets to sequences, and ii) CARD: jointly modeling cardinality and the set output. This section answers two questions:

RQ1: **How well does SETAUG improve existing models?** Specifically, how well SETAUG can take an existing SEQ2SEQ model and improve it just using our data augmentation and joint cardinality prediction, without making any changes to the model architecture. We also measure if these performance improvements carry across diverse datasets, model classes, and inference settings.

RQ2: **Why does our approach improve performance?** We study the contributions of TSAMPLE and joint cardinality prediction (CARD), and analyze where SETAUG works or fails.

4.1 Setup

Tasks We consider multi-label classification and keyphrase generation. These tasks represent set generation problems where the label space spans a set of fixed categories (multi-label classification) or free-form phrases (keyphrase generation).

1. **Multi-label classification task**: We have three datasets of varying sizes and label space:

- Go-Emotions classification (GO-EMO, Demszky et al. (2020)): generate a set of emotions for a paragraph.
- Open Entity Typing (OPENENT, Choi et al. (2018)): assigning open types (free-form phrases) to the tagged entities in the input text.
- Reuters-21578 (REUTERS, Lewis (1997)): labeling news article with the set of mentioned economic subjects.

2. **Keyphrase generation (KEYGEN)**: We experiment with a popular keyphrase generation dataset, KP20K (Meng et al., 2017) which involves generating keyphrases for a scientific paper abstract.

Table 1 lists the dataset statistics and examples from each dataset are shown in Appendix E. We treat all the problems as open-ended generation, and do not use any specialized pre-processing. For all the datasets, we filter out samples with a single label. For each training sample, we create n permutations using SETAUG.

Baselines We compare with two baselines:
i) **MULTI-LABEL**: As a non-SEQ2SEQ baseline, we train a multi-label classifier that makes independent predictions of the output labels. Encoder-only and encoder-decoder approaches can be adapted for MULTI-LABEL, and we experiment with BART (encoder-decoder) and BERT (encoder-

Task	Avg/min/max labels per sample	Unique labels	Train/test/dev samples per split
GO-EMO	3.03/3/5	28	0.6k/0.1k/0.1k
OPENENT	5.4/2/18	2519	2k/2k/2k
REUTERS	2.52/2/11	90	0.9k/0.4k/0.3k
KEYGEN	3.87/3/79	274k	156k/2k/2k

Table 1: Datasets used in our experiments.

only). This baseline represents a standard method for doing multi-label classification (e.g., Demszky et al. (2020)). During inference, top-k logits are returned as the predicted set. We search over $k = [1, 3, 5, 10, 50]$ and use k that performs the best on the dev set. Table 13 in Appendix F shows precision, recall, and F scores at each-k.

ii) **SET SEARCH**: each training sample $(\boldsymbol{x}, \{y_1, y_2, \ldots, y_k\})$ is converted into k training examples $\{(\boldsymbol{x}, y_i)\}_{i=1}^{k}$. We fine-tune BART-base to generate one training sample for input $\boldsymbol{x}$. During inference, we run beam-search with the maximum set size in the training data (Table 1). The unique elements generated by beam search are returned as the set output, a popular approach for one-to-many generation tasks (Hwang et al., 2021).

SETAUG can apply to *any* SEQ2SEQ model. We show results with models of various capacity:
iii) BART-base (Lewis et al., 2020) (110M),
iv) T5-11B (Raffel et al., 2020) (11B), and
v) GPT3-175B (Brown et al., 2020) (175B).

Training We augment $n = 2$ permutations to the original data using TSAMPLE. For all the results, we use three epochs and the same number of training samples (i.e., input data for the baselines is oversampled). This controls for models trained with augmented data improving only because of factors such as longer training time. All the experiments were repeated for three different random seeds, and we report the averages. We found from our experiments[2] that hyperparameter tuning over α, β did not affect the results in any significant way. For all the experiments reported, we use $\alpha = 1$ and $\beta = \log_2(3)$. We use a single GeForce RTX 2080 Ti for all our experiments on bart, and a single TPU for all experiments done with T5-11B. For GPT3-175B, we use the OpenAI completion engine (davinci) API (OpenAI, 2021).

[2]We conduct a one-tailed proportion of samples test (Johnson et al., 2000) to compare with the strongest baseline, and underscore all results that are significant with $p < 0.0005$. For Algorithm 1, we try $\alpha = \{0.5, 1, 1.5\}$ and $\beta = \{\log_2(2), \log_2(3), \log_2(4)\}$, and use networkx implementation of topological sort (Hagberg et al., 2008).

	GO-EMO	OPENENT	REUTERS
SET SEARCH (BART)	7.4	26.3	7.5
MULTI-LABEL (BART)	25.6	16.4	25.2
MULTI-LABEL (BERT)	25.7	16.2	25.5
BART	23.4	44.6	15.6
BART + SETAUG	**30.0**	<u>**53.5**</u>	<u>**26.7**</u>
T5-11B	47.8	53.6	45.3
T5-11B + SETAUG	**50.9**	<u>**57.0**</u>	<u>**48.5**</u>

Table 2: SETAUG improves SEQ2SEQ models by ~20% relative $F1$- points, on three multilabel classification datasets. BART and T5-11B are trained on the original datasets with a random order and no cardinality. "+ SETAUG" indicates augmented train data using TSAMPLE and cardinality is prepended to the output sequence.

Additional hyperparameter details in Appendix D. We use greedy sampling for all experiments. Our method remains effective across five different sampling techniques, incl. beam search, nucleus, top-k, and random sampling (Table 14, Appendix G).

4.2 SETAUG improves existing models

Our method helps across a wide range of models (BART, T5-11B, and GPT3-175B) and tasks.

4.2.1 Multi-label classification

Table 2 shows improvements across all datasets and models for the multi-label classification task (~20% relative gains). For brevity, we list macro F score, and include detailed results including macro/micro precision, recall, F scores in Table 9 (Appendix F). We attribute the comparatively lower performance of SET SEARCH baseline to two specific reasons - repeated generation of the same set of terms (e.g., *person, business* for OPENENT) and generating elements not present in the test set (see Section 4.3.4 for a detailed error analysis). We see similar trends with GPT3-175B (Section 4.2.4).

4.2.2 Keyphrase generation

To further motivate the utility of SEQ2SEQ models for set generation tasks, we experiment on KP-20k, which is an extreme multi-label classification dataset (Meng et al., 2017) with label space span-

Ye et al. (2021)	BART	BART + SETAUG
5.8	5.3	6.5
39.2	36.3	39.1

Table 3: SETAUG improves off-the-shelf BART-base for keyphrase generation task

ning over 257k unique keyphrases. Due to the large label space, training multi-class classification baselines is not computationally viable. In this dataset, the input text is an abstract from a scientific paper. We use the splits used by Ye et al. (2021). For a fair comparison with Ye et al. (2021), we use BART-base for this experiment. Table 3 shows the results. Similar to datasets with smaller label space, our method improves on vanilla SEQ2SEQ.

We want to emphasize that while specialized models for individual tasks might be possible, we aim to propose a general approach that shows that sampling informative orders can help efficient and general set-generation modeling.

4.2.3 Simulations

Following prior work on studying deep network properties effectively via simulation (Vinyals et al., 2016; Khandelwal et al., 2018), we design a simulation to study the effects of output order and cardinality on conditional set generation. The simulation reveals several key properties of our methods. We defer the details to Appendix G, and mention some key findings here. We find similar trends in simulated settings. Specifically, our method is (i) ineffective when labels are independent, (ii) helpful even when position embeddings are disabled, and (iii) helps across a wide range of sampling types.

4.2.4 Few-shot prompting with GPT3-175B

We fine-tune the generation models using augmented data for both BART and T5-11B. However, fine-tuning models at the scale of GPT3-175B is prohibitively expensive. Thus such models are typically used in *a few-shot prompting setup*. In a few-shot prompting setup, M ($\sim$10-100) input-output examples are selected as a prompt p. A new input x is appended to the prompt p, and $p||x$ is the input to GPT3-175B. Improving these prompts, sometimes referred to with an umbrella term prompt tuning (Liu et al., 2021), is a popular and emergent area of NLP. Our approach is the only feasible candidate for such settings, as it does not involve changing the model or additional post-processing. We apply our approach for tuning prompts for generating sets in few-shot settings.[3] We focus on GO-EMO and OPENENT tasks, as the relatively short input examples allow cost-effective experiments. We randomly create a prompt with $M = 24$ examples from the training set and run inference over the test set for each. For each example in the prompt, we order the set of emotions using our ordering approach TSAMPLE and compare the results with random orderings. Using TSAMPLE to arrange the labels outperforms random ordering for both OPENENT (macro F 34 vs. 39.5 with ours, 15% statistically significant relative improvement), and GO-EMO (macro F 16.5 vs. 14.5, 14% relative improvement). This suggests that ordering helps performance in resource-constrained settings e.g., few-shot prompting.

[3]We use the text-davinci-001 version of GPT3-175B available via the OpenAI API: https://beta.openai.com/

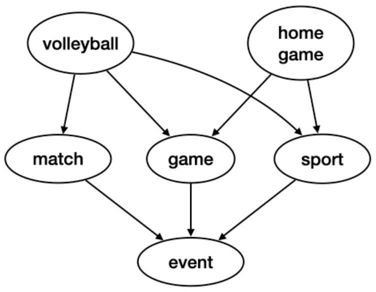


Figure 3: Label dependency discovered by TSAMPLE for OPENENT: specific entities (e.g., volleyball) precede generic ones (event). Appendix C has more examples

4.3 Why does SETAUG improve performance?

As mentioned in Section 3, our method of generating sets with SEQ2SEQ models consists of two components: i) a strategy for sampling informative orders over label space (TSAMPLE), and ii) jointly generating cardinality of the output (CARD). This section studies the individual contributions of these components in order to answer RQ2.

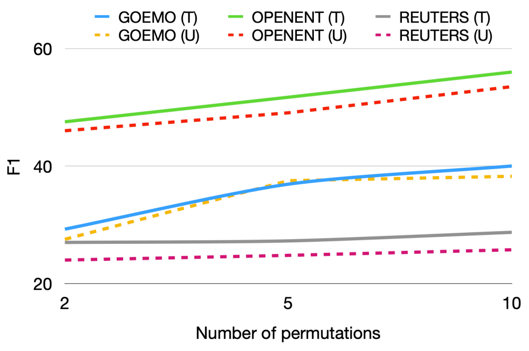


Figure 4: SETAUG (T) consistently outperforms RANDOM (U) as the number of permutations (n) is increased.

4.3.1 Ablation study

We ablate the two critical components of our system: cardinality (SETAUG - CARD) and order (SETAUG - TSAMPLE) and investigate the performance for each of these settings using BART for multi-label classification. Table 4 presents the results. Both the components individually help, but a larger drop is seen by removing cardinality. We also train using RANDOM orders, instead of TSAMPLE. RANDOM does not improve over SEQ2SEQ consistently (both with and without CARD), showing that merely augmenting with random permutations does not help. Further, Appendix F shows that cardinality is useful even with RANDOM order.

	GO-EMO	OPENENT	REUTERS
SETAUG	**30.0**	**53.5**	**26.7**
- CARD	23.3 (-22%)	48.0 (-10%)	15.8 (-40%)
- TSAMPLE	26.8 (-11%)	50.5 (-6%)	24.3 (-9%)
RANDOM	27.5 (-8%)	50.4 (-6%)	24.7 (-7%)
FREQ	19.03 (-36%)	49.9 (-7%)	23.4 (-12%)

Table 4: Ablations: modeling cardinality (CARD) and sampling informative orders (TSAMPLE) both help, with larger gains from CARD. RANDOM ordering hurts.

4.3.2 Role of order

Nature of permutations created by SETAUG SETAUG encourages highly co-occurring pairs (y_i, y_j) to be in the order y_i, y_j if $p(y_j \mid y_i) > p(y_i \mid y_j)$. In our analysis, this dependency in the datasets shows that the orders exhibit a pattern where *specific* labels appear before the *generic* ones. E.g., in entity typing, the more generic entity *event* is generated after the more specific entities *home game* and *match* (see Figure 3).

Increasing # permutations (n) helps: Fig. 4 shows that SETAUG and RANDOM improve as n is increased from $n = 2$ to 10; SETAUG outperforms RANDOM across n.

Reversing the order hurts performance In order to check our hypothesis of whether only informative orders helping with set generation, we invert the label dependencies returned by SETAUG for all the datasets and train with the same model settings. Across all datasets, we observe that reversing the order leads to an average of 12% drop in $F1$- score. The reversed order not only closes the gap between SETAUG and RANDOM, but in many instances, the performance is slightly worse than RANDOM.

Ordering by frequency Yang et al. (2018) use frequency ordering, where the most frequent label is placed first in the sequence. We compare with this baseline in Table 4 (FREQ). The results indicate that the performance of frequency-based ordering is dataset dependent. Relying on a fixed criteria like frequency might lead to skewed outputs, especially for datasets with a long-tail of labels. For instance, for OPENENT, one of the most significant failure modes of the freq method was generating the most common label in the corpus (person) for every input. TSAMPLE can be seen as a way to balance the most frequent and least frequent labels in the corpus using PMI and conditional likelihood (Algorithm 1, L3).

4.3.3 Role of cardinality

Cardinality is successfully predicted and used Table 4 shows that cardinality is crucial to modeling set output. To study whether the models learn to condition on predicted cardinality, we compute an *agreement* score - defined as the % of times the predicted cardinality matches the number of elements generated by the model. The model effectively predicts the cardinality almost exactly in GO-EMO and REUTERS datasets (avg. 95%). While the exact match agreement is low in OPENENT (35%), the model is within an error of ± 1 in 93% of the cases. These results show that cardinality predicts the end of sequence (EOS) token. The accuracy for predicting the exact cardinality is 61% across datasets, and it increases to 76% within an error of 1 (i.e., when the predicted cardinality is off by 1).

Information about cardinality improves multi-label classification MULTI-LABEL baseline uses different values of k for predicting labels. To test if knowledge of cardinality improves multi-class classification, we experiment with a setting where the true cardinality is available at inference (i.e., k is set to the true value of cardinality). Table 5 shows that cardinality improves performance.

	GO-EMO	OPENENT	REUTERS
MULTI-LABEL	**22.4**	14.3	21.7
MULTI-LABEL-K*	21.3(-4.9%)	**17.8**(+24.5%)	**25.6**(+18%)

Table 5: Cardinality improves multi-label classification.

4.3.4 Error analysis

We manually compare the outputs generated by the vanilla BART model with BART + SETAUG. For the open-entity typing dataset, we randomly sample 100 examples and find that vanilla SEQ2SEQ approach generates sets with an ill-formed element 22% of the time, whereas SETAUG completely avoids this. Examples of such ill-formed elements include *personformer*, *businessirm*, *polit*, *foundationirm*, *politplomat*, *eventlete*. This analysis indicates that training the model with an informative order infuses more information about the underlying type-hierarchy, avoiding the ill-formed elements.

5 Related work

Set generation in NLP Prior work has noted the impact of the order on the performance of text generation models (Vinyals et al., 2016), especially in the context of keyphrase generation (Meng et al., 2019). Approaches to explicitly model set properties for NLP tasks include either performing an exhaustive search to find the best order (Vinyals et al., 2016), changing the model training to modify the loss function (Qin et al., 2019), or applying post-processing (Nag Chowdhury et al., 2016). Notably, Ye et al. (2021) introduced One2Set, a method for training order-invariant models for generating set of keyphrases. Our main goal in this work is to provide a framework to identify useful orders for set generation, and show that such orders can help vanilla SEQ2SEQ models. SETAUG can work with any SEQ2SEQ model, and is complementary to these specialized methods.

Non-SEQ2SEQ set generation These include reinforcement learning for multi-label classification (Yang et al., 2019a) and combinatorial problems (Nandwani et al., 2020), and using pointer networks for keyphrase extraction (Ye et al., 2021). We focus on optimally adapting existing SEQ2SEQ models for set generation, without external knowledge (Wang et al., 2021; Zhang et al., 2019b).

Chen et al. (2021) explored the generation of an optimal order for graph generation *given* the nodes. They observed that ordering nodes before inducing edges improves graph generation. However, in our case, since the labels are unknown, conditioning on the labels to create the optimal order is not possible.

Connection with Janossy pooling Murphy et al. (2019) generalize deep sets by encoding a set of N elements by pooling permutations of $P(N, k)$ tuples. With $k = N$, their method is the same as pooling all $N!$ sequences, and reduces to deep sets with $k = 1$. We share similarity with tractable searching over $N!$ with Janossy pooling, but instead of iterating over all possible 2-tuples, we impose pairwise constraints on the element order.

Modeling set input A number of techniques have been proposed for encoding set-shaped inputs (Santoro et al., 2017; Zaheer et al., 2017; Lee et al., 2019; Murphy et al., 2019; Huang et al., 2020; Kim et al., 2021). Specifically, Zaheer et al. (2017) propose deep sets, wherein they show that pooling the representations of individual set elements and feeding the resulting features to a non-linear network is a principled way of representing sets. Lee et al. (2019) present permutation-invariant attention to encode shapes and images using a modified version of attention (Vaswani et al., 2017). Unlike these works, we focus on settings where the input is a sequence, and the output is a set.

6 Conclusion and Discussion

We present SETAUG, a novel data augmentation method for conditional set generation that incorporates informative orders and adds cardinality information. Our key idea is using the most likely order (vs. a randomly selected order) to represent a set as a sequence and conditioning the generation of a set on predicted cardinality. As a computationally efficient and general-purpose plug-in data augmentation algorithm, SETAUG improves SEQ2SEQ models for set generation across a wide spectrum of tasks. For future work, it would be interesting to investigate if the general ideas in this work have applications in settings beyond set generation. Examples include generating additional data to improve language modeling in low-resource scenarios and determining the ideal order of in-context examples in a prompt.

Acknowledgments

We thank Amrith Setlur for thoughtful discussion and the anonymous reviewers for valuable feedback. This material is partly based on research sponsored in part by the Air Force Research Laboratory under agreement number FA8750-19-2-0200. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. We also thank Google for providing the TPUs for conducting experiments.

Limitations

Ineffectiveness on independent sets SETAUG is *only* useful when the labels share some degree of dependence. For tasks where the labels are completely independent, SETAUG will not be effective. It can be shown that order will not affect learning joint distribution over labels if the labels are independent (Lemma B.3). Thus, in such settings, *any* method (including SETAUG) that seeks to leverage the relationship between labels will not be helpful. In addition to Lemma B.3, we conduct thorough simulation studies to verify this limitation (Figure 8).

Use of large language models We perform experiments with extremely large models, including T5-XXL and GPT-3 models. Particularly, GPT-3 is only available through OpenAI API; thus, all the details about its working are not publicly available. However, our experiments also show results using BART models that run on a single RTX 2080 GPU (please also see details on reproducibility in Appendix A). Further, such language models are typically trained on a large English corpora, which is also the focus of our work.

Focus on SEQ2SEQ A key limitation of our work is that it focuses on set-generation using SEQ2SEQ models. Thus the proposed insights may not apply to other settings (e.g., computer vision) where using language models is not directly feasible. Nevertheless, with the growing popularity of libraries like Huggingface (Wolf et al., 2019), we anticipate that SEQ2SEQ models will be applied to a growing number of use cases, even those that would traditionally be tackled using a non-SEQ2SEQ method. Further, we compare our method with representative non-SEQ2SEQ baselines (like multi-label classifier).

To our knowledge, our work does not directly use any datasets that contain explicit societal biases. Therefore, we anticipate that our work will not lead to any significant negative implications concerning real-world applications.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. 2016. Importance weighted autoencoders. In 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.

Xiaohui Chen, Xu Han, Jiajing Hu, Francisco JR Ruiz, and Liping Liu. 2021. Order matters: Probabilistic modeling of node sequence for graph generation. arXiv preprint arXiv:2106.06189.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

Hongliang Dai, Yangqiu Song, and Haixun Wang. 2021. Ultra-fine entity typing with weak supervision from a masked language model. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1790–1799, Online. Association for Computational Linguistics.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4040–4054, Online. Association for Computational Linguistics.

Justin Domke and Daniel R. Sheldon. 2018. Importance weighting and variational inference. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pages 4475–4484.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 199–209, San Diego, California. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long

Papers), pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 12848–12856.

Aric Hagberg, Pieter Swart, and Daniel S Chult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Qian Huang, Horace He, Abhay Singh, Yan Zhang, Ser-Nam Lim, and Austin R. Benson. 2020. Better set representations for relational reasoning. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 6384–6392.

Wengong Jin, Regina Barzilay, and Tommi S. Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, volume 80 of Proceedings of Machine Learning Research, pages 2328–2337. PMLR.

Richard A Johnson, Irwin Miller, and John E Freund. 2000. Probability and statistics for engineers, volume 2000. Pearson Education London.

Xincheng Ju, Dong Zhang, Junhui Li, and Guodong Zhou. 2020. Transformer-based label set generation for multi-modal multi-label emotion detection. In MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020, pages 512–520.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. 2021. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 15059–15068.

Adam R Kosiorek, Hyunjik Kim, and Danilo J Rezende. 2020. Conditional set generation with transformers. arXiv preprint arXiv:2006.16841.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, volume 97 of Proceedings of Machine Learning Research, pages 3744–3753. PMLR.

David Lewis. 1997. Reuters-21578 text categorization test collection, distribution 1.0. http://www. research/. att. com.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ArXiv.

Rui Meng, Xingdi Yuan, Tong Wang, Peter Brusilovsky, Adam Trischler, and Daqing He. 2019. Does order matter? an empirical study on generating multiple keyphrases as a sequence. arXiv preprint arXiv:1909.03590.

Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 582–592, Vancouver, Canada. Association for Computational Linguistics.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. 2019. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

Sreyasi Nag Chowdhury, Niket Tandon, and Gerhard Weikum. 2016. Know2Look: Commonsense knowledge for visual search. In Proceedings of the 5th Workshop on Automated Knowledge Base Construction, pages 57–62, San Diego, CA. Association for Computational Linguistics.

Yatin Nandwani, Deepanshu Jindal, Parag Singla, et al. 2020. Neural learning of one-of-many solutions for combinatorial problems in structured output spaces. arXiv preprint arXiv:2008.11990.

OpenAI. 2021. Openai completion engine (davinci) api.

Kechen Qin, Cheng Li, Virgil Pavlu, and Javed Aslam. 2019. Adapting RNN sequence prediction model to multi-label set prediction. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 3181–3190, Minneapolis, Minnesota. Association for Computational Linguistics.

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. Journal of Machine Learning Research, 21:1–67.

S Hamid Rezatofighi, Roman Kaskman, Farbod T Motlagh, Qinfeng Shi, Daniel Cremers, Laura Leal-Taixé, and Ian Reid. 2018. Deep perm-set net: Learn to predict sets with unknown permutation and cardinality using deep neural networks. arXiv preprint arXiv:1805.00613.

Adam Santoro, David Raposo, David G. T. Barrett, Mateusz Malinowski, Razvan Pascanu, Peter W. Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 4967–4976.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1405–1418, Online. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Manfred Wettler and Reinhard Rapp. 1993. Computation of word associations based on co-occurrences of words in large corpora. In Very Large Corpora: Academic and Industrial Perspectives.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771.

Pengcheng Yang, Fuli Luo, Shuming Ma, Junyang Lin, and Xu Sun. 2019a. A deep reinforced sequence-to-set model for multi-label classification. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5252–5258, Florence, Italy. Association for Computational Linguistics.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: Sequence generation model for multi-label classification. In Proceedings of the 27th International Conference on Computational Linguistics, pages 3915–3926, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yiben Yang, Chaitanya Malaviya, Jared Fernandez, Swabha Swayamdipta, Ronan Le Bras, Ji-Ping Wang, Chandra Bhagavatula, Yejin Choi, and Doug Downey. 2020. G-daug: Generative data augmentation for commonsense reasoning. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, pages 1008–1025.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019b. Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 5754–5764.

Jiacheng Ye, Tao Gui, Yichao Luo, Yige Xu, and Qi Zhang. 2021. One2Set: Generating diverse keyphrases as a set. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4598–4608, Online. Association for Computational Linguistics.

Xingdi Yuan, Tong Wang, Rui Meng, Khushboo Thaker, Peter Brusilovsky, Daqing He, and Adam Trischler. 2020. One size does not fit all: Generating and evaluating variable number of keyphrases. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7961–7975, Online. Association for Computational Linguistics.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep sets. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 3391–3401.

Yan Zhang, Jonathon S. Hare, and Adam Prügel-Bennett. 2019a. Deep set prediction networks. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pages 3207–3217.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019b. ERNIE: Enhanced language representation with informative entities. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

A Reproducibility

We take the following steps for reproducibility of our results:

1. All the experiments are performed for three different random seeds. In addition, we conduct a proportion of samples hypothesis test to establish the statistical significance of our results. We did not perform extensive hyperparameter tuning and used the same set of defaults for baselines and our proposed method.
2. For all data augmentation experiments, we match the number of training samples and epochs; all the models are trained for the same duration. This alleviates the concern that the models perform well with augmented data merely because of the longer training time.
3. We conduct a proportion of samples test for all the experiments conducted on real-world datasets and use a small $p = 0.0005$ to measure highly significant results, which are indicated with an underscore.

Our work aims to promote the usage of existing resources for as many use cases as possible. In particular, all our experiments are performed on the BASE-version of the model (BART) that can relatively lower parameter count to conserve resources and help lower our impact on climate change.

B Proofs

Let $\mathbb{Y}$ be the output space, $\mathrm{y}_i, \mathrm{y}_j, \mathrm{y}_k \in \mathbb{Y}$, and $\boldsymbol{y}_k \in \mathbb{Y} - \mathrm{y}_i - \mathrm{y}_j$ be a subset of the symbols excluding $\mathrm{y}_i, \mathrm{y}_j$. We assume that all the distributions are non-negative (i.e., $p(\boldsymbol{y}) > 0, \forall \boldsymbol{y} \in \mathbb{Y}$)

Lemma B.1. $\mathrm{y}_i \not\perp\!\!\!\perp \mathrm{y}_j \implies \mathrm{y}_i \not\perp\!\!\!\perp (\mathrm{y}_j\mathrm{y}_k)$

Proof Let $\mathrm{y}_i \perp\!\!\!\perp (\mathrm{y}_j\mathrm{y}_k)$ by contradiction. Then:

$$p(\mathrm{y}_i, \mathrm{y}_j\mathrm{y}_k) = p(\mathrm{y}_i)p(\mathrm{y}_j\mathrm{y}_k) \tag{1}$$

Also,

$$\begin{aligned} p(\mathrm{y}_i, \mathrm{y}_j) &= \sum_{\mathrm{y}_k \in \boldsymbol{Z}} p(\mathrm{y}_i, \mathrm{y}_j\mathrm{y}_k) \\ &= \sum_{\mathrm{y}_k \in \boldsymbol{Z}} p(\mathrm{y}_i)p(\mathrm{y}_j\mathrm{y}_k) \qquad \text{(equation 1)} \\ &= p(\mathrm{y}_i) \sum_{\mathrm{y}_k \in \boldsymbol{Z}} p(\mathrm{y}_j\mathrm{y}_k) \\ &= p(\mathrm{y}_i)p(\mathrm{y}_j) \qquad (2) \end{aligned}$$

However, $\mathrm{y}_i \not\perp\!\!\!\perp \mathrm{y}$ thus $\mathrm{y}_i \not\perp\!\!\!\perp \mathrm{y} \implies \mathrm{y}_i \not\perp\!\!\!\perp (\mathrm{y}_j\mathrm{y}_k)$.

Lemma B.2.

$$\begin{aligned} & p(\mathrm{y}_i \mid \mathrm{y}_j) > p(\mathrm{y}_j \mid \mathrm{y}_i) \\ \implies & p(\mathrm{y}_i \mid \mathrm{y}_j, \boldsymbol{y}_k) > p(\mathrm{y}_j \mid \mathrm{y}_i, \boldsymbol{y}_k) \end{aligned}$$

if $\mathrm{y}_i, \mathrm{y}_j \perp\!\!\!\perp \boldsymbol{y}_k$

Proof We have:

$$\begin{aligned} & p(\mathrm{y}_i \mid \mathrm{y}_j) > p(\mathrm{y}_j \mid \mathrm{y}_i) \\ \implies & p(\mathrm{y}_j) < p(\mathrm{y}_i) \end{aligned} \tag{3}$$

$$\begin{aligned} p(\mathrm{y}_j, \boldsymbol{y}_k) &= p(\boldsymbol{y}_k \mid \mathrm{y}_j)p(\mathrm{y}_j) \\ &< p(\boldsymbol{y}_k \mid \mathrm{y}_j)p(\mathrm{y}_i) \qquad \text{(Equation 3)} \\ &= p(\boldsymbol{y}_k \mid \mathrm{y}_i)p(\mathrm{y}_i) \\ & (\mathrm{y}_i, \mathrm{y}_j \perp\!\!\!\perp \boldsymbol{y}_k \implies p(\boldsymbol{y}_k \mid \mathrm{y}_j) = p(\boldsymbol{y}_k \mid \mathrm{y}_i) = p(\boldsymbol{y}_k)) \\ &= p(\mathrm{y}_i, \boldsymbol{y}_k) \qquad (4) \end{aligned}$$

Thus,

$$\begin{aligned} p(\mathrm{y}_i \mid \mathrm{y}_j, \boldsymbol{y}_k) &= \frac{p(\mathrm{y}_i, \mathrm{y}_j, \boldsymbol{y}_k)}{p(\mathrm{y}_j, \boldsymbol{y}_k)} \\ &> \frac{p(\mathrm{y}_i, \mathrm{y}_j, \boldsymbol{y}_k)}{p(\mathrm{y}_i, \boldsymbol{y}_k)} \\ &= p(\mathrm{y}_j \mid \mathrm{y}_i, \boldsymbol{y}_k) \qquad (5) \end{aligned}$$

Lemma B.3. *If* $\mathrm{y}_i \perp\!\!\!\perp \mathrm{y}_j \; \forall \mathrm{y}_i, \mathrm{y}_j \in \mathbb{Y}$*, the order is guaranteed to not affect learning.*

Proof Let π_j be the j^{th} order over $\mathbb{Y}$ (out of $|\mathbb{Y}|!$ possible orders Π), and $\pi_j(\mathbb{Y})$ be the sequence of elements in $\mathbb{Y}$ arranged with π_j. As $\mathrm{y}_i \perp\!\!\!\perp \mathrm{y}_j \; \forall \mathrm{y}_i, \mathrm{y}_j$, we have $p(\mathrm{y}_i \mid \mathrm{y}_j) = p(\mathrm{y}_i)$. This gives:

$$\begin{aligned} p(\mathrm{y}_i, \mathrm{y}_j, \mathrm{y}_k) &= p(\mathrm{y}_i)p(\mathrm{y}_j \mid \mathrm{y}_i)p(\mathrm{y}_k \mid \mathrm{y}_i, \mathrm{y}_j) \\ &= p(\mathrm{y}_i)p(\mathrm{y}_j)p(\mathrm{y}_k) \end{aligned}$$

Thus $\forall \pi_m, \pi_m \in \Pi$:

$$\begin{aligned} & p(\pi_m(\mathrm{y}_i, \mathrm{y}_j, \mathrm{y}_k)) \\ & \qquad = p(\pi_n(\mathrm{y}_i, \mathrm{y}_j, \mathrm{y}_k)) \end{aligned}$$

In other words, when all elements are mutually independent, all possible joint factorizations will simply be a product of the marginals, and thus identical.

Lemma B.4. *The graphs constructed to sample orders for* SETAUG *cannot have cycles.*

Proof Let y_i, y_j, y_k form a cycle: $y_i \rightarrow y_j \rightarrow y_k \rightarrow y_i$. By construction, the following conditions must hold for such a cycle to be present:

$$\log p(y_j \mid y_i) - \log p(y_i \mid y_j) > \beta \implies \log p(y_i) < \mathrm{lo}$$
$$\log p(y_k \mid y_j) - \log p(y_j \mid y_k) > \beta \implies \log p(y_j) < \mathrm{lo}$$
$$\log p(y_i \mid y_k) - \log p(y_k \mid y_i) > \beta \implies \log p(y_k) < \mathrm{lo}$$

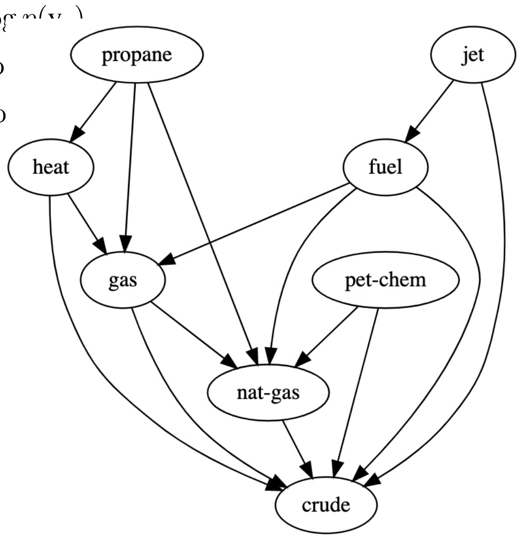


Figure 6: Label dependencies discovered by TSAMPLE for REUTERS

Putting the three implications together, we get $\log p(y_i) < \log p(y_j) < \log p(y_k) < \log p(y_i)$, which is a contradiction. Hence, the graphs constructed for SETAUG cannot have a cycle.

C Sample graphs

In this section, we present additional examples from REUTERS and GO-EMO datasets to illustrate the permutations generated by our method. As discussed in Section 3.1, SETAUG encourages highly co-occuring pairs (y_i, y_j) to be in the order y_i, y_j if $p(y_j \mid y_i) > p(y_i \mid y_j)$. In our analysis, this dependency in the datasets shows that the orders exhibit a pattern where *specific* labels appear before the *generic* ones. For example, in case of entity typing, the more GO-EMO, *sadness* is generated after the more specific emotion *remorse* and *fear* (Figure 5). Similarly, the entity *crude* is generated after the entities *gas* and *nat-gas*. (Figure 6 (right)).

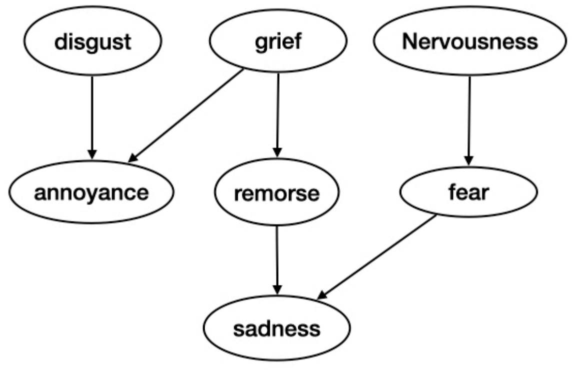


Figure 5: Label dependencies discovered by TSAMPLE for GO-EMO

Hyperparameter	Value
GPU	GeForce RTX 2080 Ti
gpus	1
auto_select_gpus	false
accumulate_grad_batches	1
max_epochs	3
precision	32
learning_rate	1e-05
adam_epsilon	1e-08
num_workers	16
warmup_prop	0.1
seeds	[15143, 27122, 999888]
add_lr_scheduler	true
lr_scheduler	linear
max_source_length	120
max_target_length	120
val_max_target_length	120
test_max_target_length	120

Table 6: List of hyperparameters used for all the experiments.

D Hyperparameters

We list all the hyperparameters in Table 6.

E Dataset

Table 7 shows examples for each of the datasets.

	Input	Output
Fine-grained emotion classification, [28] (Demszky et al., 2020)	*So there's hope for the rest of us! Thanks for sharing. What helped you get to where you are?*	{curiosity, gratitude, optimism}
Open-entity typing [2519] (Choi et al., 2018)	*Some 700,000 cubic meters of caustic sludge and water burst inundating* [*SPAN*] *three west Hungarian villages* [*SPAN*] *and spilling.*	{colony, region, location, hamlet, area, village, settlement, community}
Reuters [90] (Lewis, 1997)	*India is reported to have bought two white sugar cargoes for...* *...cargo sale, they said.*	{ship, sugar}
Keyphrase generation [270k] (Ye et al., 2021)	We analyze the impact of core affinity on both network and disk i/o performance...our dynamic core affinity improves the file upload throughput more than digit%	{big data, multi-core, process-scheduling}

Table 7: Real world tasks used for experiments

	GO-EMO	OPENENT	REUTERS
MULTI-LABEL	22.4	14.3	21.7
MULTI-LABEL @oracle-k	21.3	17.8	25.6
SETAUG + card	**30.0**	**53.5**	**26.7**

Table 8: Multi-label classification when the true cardinality is provided to the classifier. While providing the true cardinality helps the performance of multi-label classifiers, it still lags SETAUG.

F Additional results

This section presents detailed results that were omitted from the main paper for brevity. This includes macro and micro precision, recall, and F scores on all datasets, and additional ablation experiments.

1. Table 9 shows the detailed results from the four tasks.
2. Detailed results on GO-EMO, REUTERS, and OPENENT are present in Tables 10, 11, and 12, respectively.
3. Table 13 includes results from a multi-label classification baseline where bert-base-uncased is used as the encoder.

	GO-EMO			OPENENT			REUTERS			KEYGEN		
	p	r	F	p	r	F	p	r	F	p	r	F
MULTI-LABEL	20.8	42.4	22.4	16.4	25.1	14.3	19.7	43.4	21.7	-	-	-
MULTI-LABEL-K*	21.3	21.3	21.3	17.8	17.8	17.8	25.6	25.6	25.6	-	-	-
SET SEARCH	10.7	7.0	7.4	26.5	31.4	26.3	10.9	7.1	7.5	5.8	**7.4**	6.4
SEQ2SEQ	27.4	26.2	23.4	55.4	42.4	44.6	24.8	13.8	15.6	6.7	5.5	5.9
RANDOM	32.5	19.9	22.7	62.6	41.7	46.9	26.7	12.7	15.2	6.6	4.5	5.2
SETAUG	**36.7**	19.8	23.3	60.0	44.5	48.0	26.5	12.8	15.8	7.0	5.0	5.6
SEQ2SEQ + CARD	33.0	28.3	26.8	62.5	44.7	50.5	34.1	21.8	24.3	7.1	5.6	6.1
RANDOM + CARD	35.6	26.5	27.5	**68.6**	42.3	50.4	35.3	22.1	24.7	7.3	5.7	6.3
SETAUG + CARD	36.1	**30.5**	**30.0**	65.5	**47.5**	**53.5**	**36.7**	**24.1**	**26.7**	**7.7**	6.1	**6.6**

Table 9: Our main results in detail: using permutations generated by SETAUG and adding cardinality gives the best overall performance in terms of macro precision, recall, and $F1$- score. MULTI-LABEL is the standard multi-label classification approach. Statistically significant results (**p** = $5e^{-4}$) are underscored. CARD stands for cardinality.

	p_{micro}	p_{macro}	r_{micro}	r_{macro}	F_{micro}	F_{macro}	$jaccard$
SET SEARCH	47.17	10.68	13.09	7.02	10.7	7.36	7.4
SEQ2SEQ	41.65	27.39	35.19	26.21	27.4	23.41	23.4
SEQ2SEQ + CARD	39.77	33	38.02	28.31	33	26.79	26.8
RANDOM + CARD	44.77	35.6	32.96	26.54	35.6	27.53	27.5
SETAUG + CARD	43.37	36.08	34.51	30.54	36.1	30.01	30
RANDOM- CARD	48.85	32.45	27.75	19.86	32.5	22.67	22.7
SETAUG- CARD	50	36.68	29.84	19.84	36.7	23.31	23.3

Table 10: Results for GO-EMO.

	p_{micro}	p_{macro}	r_{micro}	r_{macro}	F_{micro}	F_{macro}	$jaccard$
SET SEARCH	70.04	10.92	34.9	7.1	46.56	7.54	37.49
SEQ2SEQ	66.36	24.74	42.28	13.78	51.64	15.58	44.3
SEQ2SEQ + CARD	73.02	34.17	53.8	21.85	61.95	24.28	59.08
RANDOM + CARD	74.26	35.31	54.33	22.13	62.75	24.74	58.95
SETAUG + CARD	75.65	36.67	55.54	24.13	64.05	26.66	61.14
RANDOM- CARD	69.56	26.68	38.15	12.71	49.27	15.2	42.24
SETAUG- CARD	76.55	26.49	41.78	12.77	54.06	15.78	47.34

Table 11: Results for REUTERS.

	p_{micro}	p_{macro}	r_{micro}	r_{macro}	F_{micro}	F_{macro}	$jaccard$
SET SEARCH	24.65	26.5	29.98	31.44	23.92	26.25	13.39
SEQ2SEQ	52.78	55.4	39.84	42.42	41.45	44.63	24.6
SEQ2SEQ + CARD	61.26	62.48	41.87	44.68	48.07	50.48	27.84
RANDOM + CARD	67.56	68.59	39.61	42.25	47.98	50.4	26.89
SETAUG + CARD	64.58	65.53	44.6	47.46	51.2	53.48	29.39
RANDOM- CARD	60.93	62.57	39.09	41.69	44.2	46.85	25.26
SETAUG- CARD	58.02	59.88	42.63	44.95	46.54	48.86	26.82

Table 12: Results for OPENENT.

	GO-EMO			OPENENT			REUTERS			
	p	r	F	p	r	F	p	r	F	
BERT @1	31.8	10.3	15.6	38.0	10.3	15.9	31.7	12.3	17.6	
BERT @3	23.8	23.4	23.6	19.7	14.0	16.1	23.4	28.3	25.5	
BERT @5	20.6	34.0	25.7	15.5	18.0	16.4	18.8	37.6	24.9	
BERT @10	16.5	54.3	25.3	11.8	26.0	16.0	15.1	61.8	24.2	
BERT @20	14.1	93.2	24.5	8.4	34.3	13.5	9.5	75.9	16.8	
BERT @50	-	-	-	2.6	**50.2**	4.9	8.9	-	-	-
BERT	21.4	43.0	22.9	16.0	25.5	13.8	19.7	43.2	21.8	
BART @1	31.7	10.3	15.5	38.0	10.3	15.6	31.8	12.3	17.6	
BART @3	21.2	21.0	21.0	19.7	14.0	15.8	23.1	28.1	25.2	
BART @5	14.1	33.4	25.6	15.5	18.0	16.2	18.7	37.6	24.8	
BART @10	16.3	53.4	25.0	11.7	26.0	15.9	15.1	62.0	24.1	
BART @20	14.1	**93.3**	24.5	8.4	34.3	13.4	9.6	**77.1**	17.1	
BART @50	-	-	-	4.9	48.0	8.9	-	-	-	
BART	20.8	42.4	22.4	16.4	25.1	14.3	19.7	43.4	21.7	
SET SEARCH	10.7	7.0	7.4	26.5	31.4	26.3	10.9	7.1	7.5	
SEQ2SEQ	27.4	26.2	23.4	55.4	42.4	44.6	24.8	13.8	15.6	
RANDOM	32.5	19.9	22.7	62.6	41.7	46.9	26.7	12.7	15.2	
SETAUG	**36.7**	19.8	23.3	60.0	44.5	48.0	26.5	12.8	15.8	
SEQ2SEQ +CARD	33.0	28.3	26.8	62.5	44.7	50.5	34.1	21.8	24.3	
RANDOM + CARD	35.6	26.5	27.5	**68.6**	42.3	50.4	35.3	22.1	24.7	
SETAUG + CARD	36.1	**30.5**	**30.0**	65.5	**47.5**	**53.5**	**36.7**	**24.1**	**26.7**	

Table 13: Our main results: using permutations generated by SETAUG and adding cardinality gives the best overall performance in terms of macro precision, recall, and $F1$--score score. Statistically significant results are underscored. CARD stands for cardinality. BERT @k / BART @k denotes the pointwise classification baseline using BERT/ BART where the top k labels are used as the model output. The average is denoted by BERT/ BART.

G Exploring the influence of order on SEQ2SEQ models with a simulation

We design a simulation to investigate the effects of output order and cardinality on conditional set generation, following prior work that has found simulation to be an effective for studying properties of deep neural networks (Vinyals et al., 2016; Khandelwal et al., 2018).

We note that a number of techniques have been proposed for encoding set-shaped inputs (Santoro et al., 2017; Zaheer et al., 2017; Lee et al., 2019; Murphy et al., 2019; Huang et al., 2020; Kim et al., 2021). Unlike these works, we focus on settings where the input is a sequence, and the output is a set, and design the data generation process accordingly.

Data generation We use a graphical model (Figure 7) to generate conditionally dependent pairs $(\boldsymbol{x}, \mathbb{Y})$, with different levels of interdependencies among the labels in $\mathbb{Y}$. Let $\mathbb{Y} = \{\mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_N\}$ be the set of output labels. We sample a dataset of the form $\{(\boldsymbol{x}, \boldsymbol{y})\}_{i=1}^{m}$. $\boldsymbol{x}$ is an N dimensional multinomial sampled from a dirichlet parameterized by α, and $\boldsymbol{y}$ is a sequence of symbols with each $y_i \in \mathbb{Y}$. The output sequence $\boldsymbol{y}$ is created in B *blocks*, each block of size k. A block is created by first sampling $k-1$ prefix symbols independently from Multinomial$(\boldsymbol{x})$, denoted by $\boldsymbol{y}_p$ The k^{th} suffix symbol (y_s) is sampled from either a uniform distribution with a probability = ϵ or is deterministically determined from the preceding $k-1$ prefix terms. For block size of 1 ($k = 1$), the output is simply a set of size B sampled from $\boldsymbol{x}$ (i.e., all the elements are independent). Similarly, $k = 2$ simulates a situation with a high degree of dependence: each block is of size 2, with the prefix sampled independently from the input, and the suffix determined deterministically from the prefix. Gradually increasing the block size increases the number of independent elements.

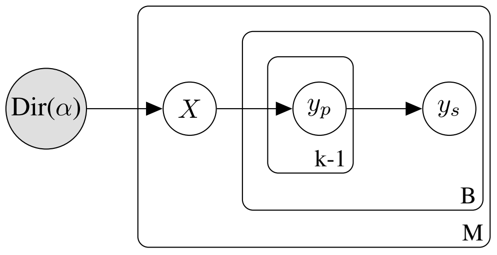


Figure 7: The generative process for simulation

G.1 Major Findings

We now outline our findings from the simulation. We use the architecture of BART-base (Lewis et al., 2020) (six-layers of encoder and decoder) without pre-training for all simulations. All the simulations were repeated using three different random seeds, and we report the averages.

Finding 1: SEQ2SEQ models are sensitive to order, but only if the labels are conditionally dependent on each other. We train with the prefix $\boldsymbol{y}_p$ listed in the lexicographic order. At test time, the order of is randomized from 0% (same order as training) to 100 (appendixly shuffled). As can be seen from Figure 8 the perplexity gradually increases with the degree of randomness. Further, note that perplexity is an artifact of the model and is independent of the sampling strategy used, showing that order affects learning.

Finding 2: Training with random orders makes the model less sensitive to order As Figure 9 shows, augmenting with random order makes the model less sensitive to order. Further, augmenting with random order keeps helping as the perplexity gradually falls, and the drop shows no signs of flattening.

Finding 3: Effects of position embeddings can be overcome by augmenting with a sufficient number of random samples Figure 9 shows that while disabling position embedding helps the baseline, similar effects are soon achieved by increasing the random order. This shows that disabling position embeddings can indeed alleviate some concerns about the order. This is crucial for pre-trained models, for which position embeddings cannot be ignored.

Finding 4: SETAUG leads to higher set overlap We next consider blocks of order 2 where the prefix symbol y_p is selected randomly as before, but the suffix is set to a special character y'_p with 50% probability. As the special symbol y'_p only occurs with y_p, there is a high pmi between each $(\mathrm{y}_p, \mathrm{y}'_p)$ pair as $p(\mathrm{y}_p \mid \mathrm{y}'_p) = 1$. Different from finding 1, the output symbols are now shuffled to mimic a realistic setup. We gradually augment the training data with random and topological orders and evaluate the learning and the final set overlap using training perplexity and Jaccard score, respectively. The results are shown in Figure 10. Similar trends hold for larger block sizes, and the results are included

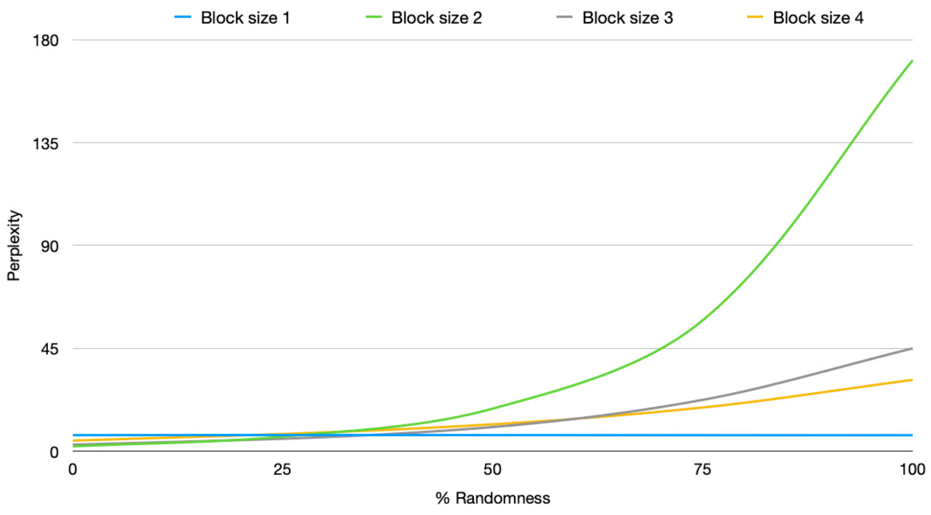


Figure 8: Perplexity vs. Randomness for varying block sizes. The degree of dependence between the labels is highest for block size = 2, where each label depends the preceding label. In such cases, the model is most affected by shuffling the order at test time. In contrast, with block size of 1, the perplexity is nearly unaffected by the order. This result complements Lemma B.3 in showing that order will not affect SEQ2SEQ models if all the labels are independent of each other.

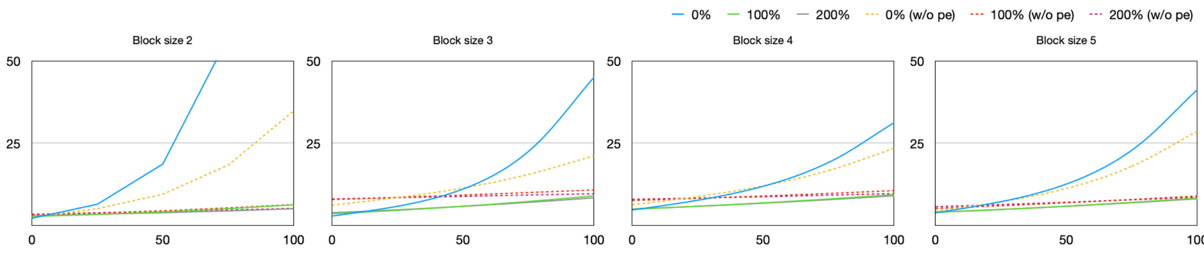


Figure 9: Augmenting dataset with multiple orders help across block sizes. Augmentations also overcome any benefit that is obtained by using position embeddings.

in the Appendix in the interest of space.

Finding 5: SETAUG helps across all sampling types We see from Table 14 that our approach is not sensitive to the sampling type used. Across five different sampling types, augmenting with topological orders yields significant gains.

Finding 6: SEQ2SEQ models can learn cardinality and use it for better decoding We created sample data from Figure 7 where the length of the output is determined by sum of the inputs X. We experimented with and without including cardinality as the first element. We found that training with cardinality increases step overlap by over 13%, from 40.54 to 46.13. Further, the version with cardinality accurately generated sets which had the same length as the target 70.64% of the times, as opposed to 27.45% for the version without cardinality.

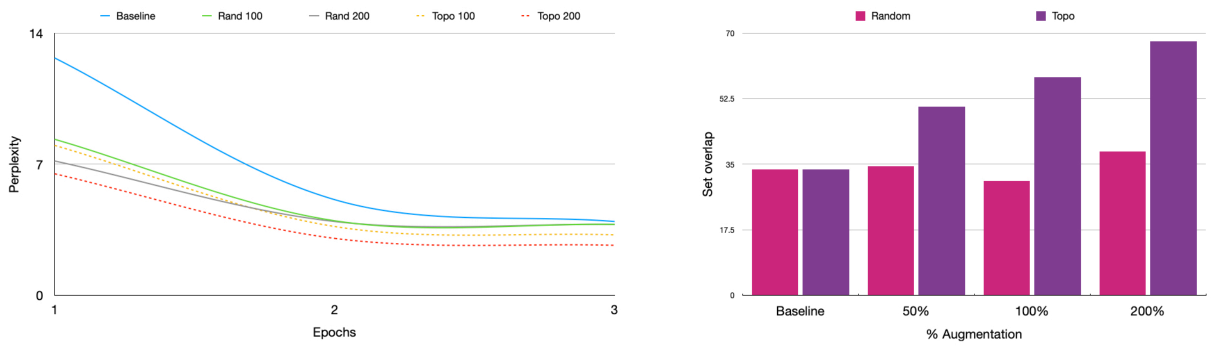


Figure 10: Effect of SETAUG on perplexity and set overlap. **Left:** Augmentations done SETAUG helps the model converge faster and to a lower perplexity. **Right:** Using SETAUG, the overlap between training and test set increases consistently, while consistently outperforming RANDOM.

	Beam	Random	Greedy	Top-k	Nucleus
RANDOM	0.39 ± 0.05	0.39 ± 0.02	0.35 ± 0.05	0.39 ± 0.02	0.39 ± 0.02
SETAUG	0.67 ± 0.05	0.67 ± 0.05	0.71 ± 0.04	0.67 ± 0.05	0.68 ± 0.05

Table 14: Set overlap for different sampling types with 200% augmentations. The gains are consistent across sampling types. Similar trends were observed for 100% augmentation and without positional embeddings. Top-k sampling was introduced by (Fan et al., 2018), and Nucleus sampling by (Holtzman et al., 2020).

H Fixing the proposal distribution in the VAE formulation

$$
\begin{aligned}
\log p_\theta(\mathbb{Y} \mid \boldsymbol{x}) &= \log \sum_{\pi_{\boldsymbol{z}} \in \Pi} p_\theta(\pi_z(\mathbb{Y}) \mid \boldsymbol{x}) \\
&= \log \sum_{\pi_{\boldsymbol{z}} \in \Pi} \frac{q_\phi(\pi_{\boldsymbol{z}})}{q_\phi(\pi_{\boldsymbol{z}})} p_\theta(\pi_z(\mathbb{Y}) \mid \boldsymbol{x}) \\
&= \log \mathbb{E}_{q_\phi(\pi_{\boldsymbol{z}})} \left[\frac{p_\theta(\pi_z(\mathbb{Y}) \mid \boldsymbol{x})}{q_\phi(\pi_{\boldsymbol{z}})} \right] \\
&\geq \mathbb{E}_{q_\phi(\pi_{\boldsymbol{z}})} \left[\log p_\theta(\mathbb{Y}, \pi_{\boldsymbol{z}} \mid \boldsymbol{x}) \right] - \mathbb{E}_{q_\phi(\pi_{\boldsymbol{z}})} \left[\log q_\phi(\pi_{\boldsymbol{z}}) \right] \\
\log p_\theta(\mathbb{Y} \mid \boldsymbol{x}) &= \log \sum_{\pi_{\boldsymbol{z}} \in \Pi} p_\theta(\pi_z(\mathbb{Y}) \mid \boldsymbol{x}) \\
&\geq \underbrace{\mathbb{E}_{q_\phi(\pi_{\boldsymbol{z}})} \left[\log \frac{p_\theta(\pi_{\boldsymbol{z}}(\mathbb{Y}) \mid \boldsymbol{x})}{q_\phi(\pi_{\boldsymbol{z}})} \right]}_{\text{ELBO}} = \mathcal{L}(\theta, \phi)
\end{aligned}
\tag{6}
$$

Where equation 6 is the evidence lower bound (ELBO). The success of this formulation depends on the quality of the proposal distribution q from which the orders are drawn. When q is fixed (e.g., to uniform distribution over the orders), learning only happens for θ. This can be clearly seen from splitting Equation 6 into terms that involve just θ and ϕ:

$$
\begin{aligned}
&\nabla_\phi \mathcal{L}(\theta, \phi) = 0 \\
&\nabla_\theta \mathcal{L}(\theta, \phi) = \nabla_\theta \mathbb{E}_{q_\phi(\pi_{\boldsymbol{z}})} \left[\log p_\theta(\mathbb{Y}, \pi_{\boldsymbol{z}} \mid \boldsymbol{x}) \right]
\end{aligned}
$$

A Example Appendix

This is an appendix.