Order-Based Pre-training Strategies for Procedural Text Understanding

Abhilash Nandy **Yash Kulkarni** **Pawan Goyal** **Niloy Ganguly**
nandyabhilash@kgpian.iitkgp.ac.in
Indian Institute of Technology Kharagpur
India

Abstract

In this paper, we propose sequence-based pre-training methods to enhance procedural understanding in natural language processing. Procedural text, containing sequential instructions to accomplish a task, is difficult to understand due to the changing attributes of entities in the context. We focus on recipes, which are commonly represented as ordered instructions, and use this order as a supervision signal. Our work is one of the first to compare several 'order-as-supervision' transformer pre-training methods, including Permutation Classification, Embedding Regression, and Skip-Clip, and shows that these methods give improved results compared to the baselines and SoTA LLMs on two downstream Entity-Tracking datasets: NPN-Cooking dataset in recipe domain and ProPara dataset in open domain. Our proposed methods address the non-trivial Entity Tracking Task that requires prediction of entity states across procedure steps, which requires understanding the order of steps. These methods show an improvement over the best baseline by 1.6% and 7-9% on NPN-Cooking and ProPara Datasets respectively across metrics.[1]

1 Introduction

Procedural text comprises a series of sequential instructions aimed at guiding individuals through a task by presenting information in a step-by-step manner. A procedure describes a step-wise interaction between multiple participating entities and their attribute changes. For instance, "Photosynthesis" as a procedure consists of interaction between entities such as water, light, CO2, sugar, etc. Recently, there has been an increase in the number of studies in NLP that use procedural texts. Procedural text is common in natural language in recipes (Marin et al., 2018a; Bień et al., 2020a; Chandu et al., 2019; Majumder et al., 2019; Bosselut et al., 2017), how-to guides (Nandy et al., 2021), and scientific processes (Mishra et al., 2018). In this study, we focus on recipes as they are commonly represented as ordered instructions. We utilize this order as a supervision signal to develop customized pre-training techniques to solve non-trivial tasks that require anticipating the implicit effects of actions on entities.

Understanding procedural text is difficult due to the changing attributes of entities in the context. Previous works such as Lee et al. (2020) used Sentence-level Language Modeling (SLM) to learn contextualized sentence-level representation by training a hierarchical transformer to reconstruct the original order of a shuffled sequence, Tang et al. (2020) proposed Interactive Entity Network (IEN) to model different types of entity interactions using a recurrent network with memory for state tracking, and Zhang et al. (2021) combined external knowledge with a BERT model to improve entity tracking. However, such works do not compare pre-training techniques which consider sequential order of the steps of the procedure.

In this paper, we try to solve the non-trivial Entity Tracking Task that requires prediction of entity states across procedure steps. Solving such a task requires understanding the *sequential nature/order of the steps*. Explicitly learning the order within data has been shown to enhance performance of tasks such as Video Representation Learning, solving Jigsaw Puzzles to learn image representations Noroozi and Favaro (2016), etc. Similarly, ALBERT (Lan et al., 2020) shows that sentence-order prediction between two sentences is a useful pre-training objective to improve performance on various downstream NLP tasks. Inspired by such works, our work is one of the first to introduce and compare several novel 'order-as-supervision' pre-training methods such as Permutation Classification, Skip-Clip, and Embedding Regression to enhance procedural understanding.

[1]Code is available at https://github.com/abhi1nandy2/Order_As_Supervision

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers), pages 821–828

These pre-training methods give a significant improvement of 1.6% and 7 − 9% compared to baselines across metrics on two downstream Entity-Tracking datasets, namely, NPN-Cooking dataset (Bosselut et al., 2017) in the recipe domain and ProPara dataset (Mishra et al., 2018) in the open domain. Our methods also outperform SoTA LLMs in terms of Average Accuracy on ProPara.

2 Pre-training Methods

We propose three new pre-training methods that help in learning sequential context for procedural texts: *Permutation Classification*, *Embedding Regression*, and *Skip-Clip*. For all the methods, a set of recipes with the same number of steps are sampled. Such a recipe of N steps can be represented by $x = (x_1, x_2, ..., x_N)$.

2.1 Permutation Classification

In this method, the original recipe is shuffled by permuting its steps by some index permutation $\psi_i = (\psi_{i1}, \psi_{i2}, ..., \psi_{iN})$. The set of all possible permutations ψ^* contains $N!$ elements. If, for example, N = 9 the total number of possible permutations equals 9! = 362,880. For practical reasons, as a pre-processing step, we reduce the set of all possible permutations by sampling a set ψ of maximally diverse permutations from ψ^*. Following Noroozi and Favaro (2016), we iteratively include the permutation with the maximum Hamming distance to the already chosen ones. For every recipe, we select a random permutation from this set and assign its index as a label. To solve the permutation classification task, we input the permuted sequence into a transformer and use the $< s >$ (classification token) embedding to perform sequence multi-class classification. The number of output classes is equal to the size of the permutation set. Figure 1 shows the Permutation Classification Architecture.

2.2 Embedding Regression

Following Korba et al. (2018), we modify the permutation classification method defined above. Thus, instead of predicting the index, we convert the permutation into an embedding vector and perform a regression task on this embedding. We experiment with 2 different embedding constructions, considering ψ as a permutation of length N - Hamming and Lehmer Embedding. Hamming Embedding (h) is a vector of size N^2 formed by concatenating one-hot vectors for each value of the permutation - $h_{N.i+j} = I\{\psi(i) = j\}$, where I is the Identity Function. Lehmer Embedding (l) is a vector of size N, where the value at i^{th} index is the number of indices less than i with a greater permutation value - $l_i = \#\{j : j < i, \psi(j) > \psi(i)\}_{1 \leq i \leq N}$. E.g. for the permutation (4,3,1,2) the Hamming Embedding is (0,0,0,1,0,0,1,0,1,0,0,0,0,1,0,0) and Lehmer Embedding is (0,1,2,2). We use Mean Squared Error (MSE) as the loss function. It can be theoretically shown that minimizing this loss on the selected embeddings is equivalent to optimizing a ranking metric like Kendall's Tau (Kendall, 1938) or Hamming (Hamming, 1950) distance. Figure 1 shows the Embedding Regression Architecture.

2.3 Skip-Clip

In this method inspired by El-Nouby et al. (2019), given the first few steps as the context, other steps closer to the context have more similar representations to that of the context than the ones that are farther. Here, we sample the first K steps of a recipe with N steps, as the context $c = (x_1, x_2, ..., x_K)$ and randomly sample M target steps $(x_{t_1}, x_{t_2}, ..., x_{t_M})$, where t_i is the index in the original recipe for the i^{th} target step, with $M, K < N$ and $t_1 > K$. Using transformer model f, we get latent representations of the context $h = f(c)$ and each step in the target, $z_i = f(x_{t_i})$. We also define a scoring function, $\Gamma(h, z_i)$, e.g. cosine similarity, representing the relationship between the context and the target steps. The objective is hinge rank loss formulated as: $L = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} max(0, -\Gamma(h, z_i) + \Gamma(h, z_j) + \delta)$, where the constant δ is the margin. Figure 2 shows the Skip-Clip architecture.

3 Downstream tasks

Entity Tracking consists of two sub-tasks - entity state and location tracking. Both tasks output the result for every entity at each step of the process. Entity state tracking task is a 4-way classification task that predicts if the entity is created, moved, unchanged, or destroyed at that step of the process. Entity Location tracking is formulated as a span-based question-answering problem that outputs the location of the entity at a particular step taking the entire text of the process as the input. We perform this task on NPN-Cooking and ProPara datasets[2]. NPN-Cooking dataset (Bosselut et al., 2017) consists of 65,816 training, 175 development, and 700 evaluation recipes. ProPara Dataset (Mishra et al.,

[2]Datasets are in the English Language

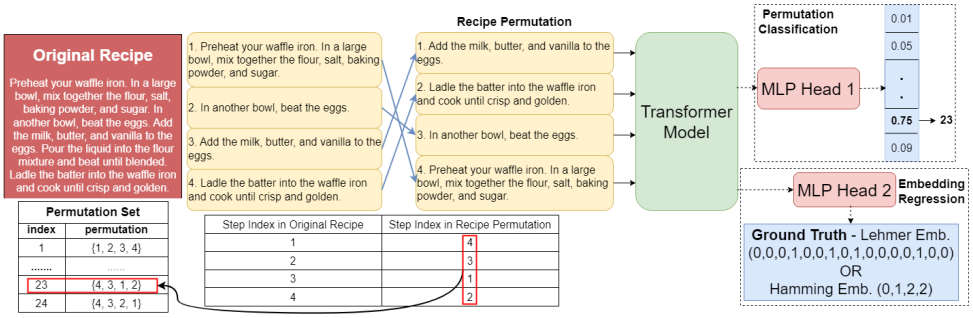

Figure 1: Permutation Classification and Embedding Regression for a 4-step recipe. Recipe steps are reordered via a randomly chosen permutation from a predefined permutation set and then fed to the transformer model. The Permutation Classification Task is to predict the index of the chosen permutation which in this case is 23, and Embedding Regression Task is to predict the corresponding Lehmer/Hamming Embedding.

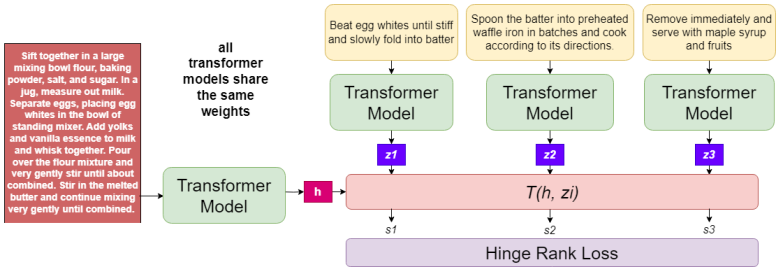

Figure 2: Skip-Clip model with a 6-step context and 3 target steps. The task is to rank the target steps based on scores obtained from a scoring function and their order in the recipe using hinge rank loss.

2018) consists of 488 human-authored procedures (split 80/10/10 into train/dev/test) with 81k annotations regarding changing states (existence and location) of entities in those paragraphs. The model is evaluated in 3 ways corresponding to a given entity e. **Category 1**: which of the three transitions - created, destroyed, or moved undergone by e over the lifetime of a procedure; **Category 2**: steps at which e is created, destroyed, and/or moved; and **Category 3**: the location (span of text) which indicates e's creation, destruction or movement. Following Faghihi and Kordjamshidi (2021), Entity Tracking is formulated as a question-answering problem. Fine-tuning Hyperparameters are the same as in the default open-source implementation[3].

4 Experiments and Results

4.1 Pre-training Setup

Parameter Initialization: For fast convergence, we initialize the transformer in each pre-training method with RoBERTa-BASE (Liu et al., 2019)[4]. **Dataset:** We use a dataset of 2.5 million+ recipes in total collected from various different sources on the internet such as Recipe1M+ dataset (Marin et al., 2018b), RecipeNLG dataset (Bień et al., 2020b), datasets collected by Majumder et al. (2019) and Chandu et al. (2019). For each recipe in the dataset, a sentence with the ingredients is also added as a step before the original recipe. The dataset is filtered to include recipes with more than 4 steps. The statistics of the dataset is shown in Table 5 ***in Section D.1 of Appendix***. Permutation Classification and Embedding Regression require all recipes to have the same number of steps per recipe. Hence, we use a subset of recipes that have a certain, fixed number of steps. **Hyperparameters:** We pre-trained for 1 epoch using AdamW optimizer with batch size of 32, learning rate of 5e-5, weight decay of 0.01, and 500 warmup steps.

Model	Dev Acc	Test Acc
NPN-Model	-	51.3
KG-MRC	-	51.6
DYNAPRO	-	62.9
RoBERTa-BASE	<u>65.07</u>	64.28
Permutation Classfn.	**65.48**	64.75
$Emb_{Hamming}$	65.03	<u>64.92</u>
Emb_{Lehmer}	63.96	64.29
Skip-Clip	63.87	**65.33**

Table 1: Results on NPN-Cooking Dataset. Numbers in bold and underlined are the highest and the second-highest scores, respectively.

[3] https://github.com/HLR/TSLM

[4] Compute details are ***in Section D.1 of Appendix***

Model	Location Acc.	Status Acc.	Cat1 Acc.	Cat2 Acc.	Cat3 Acc.	Avg. Cat. Acc.
Rule-based	-	-	57.14	20.33	2.4	26.62
Feature-based	-	-	58.64	20.82	9.66	29.71
ProLocal	-	-	62.7	30.5	10.4	34.53
ProGlobal	-	-	63	36.4	35.9	45.1
EntNet	-	-	51.6	18.8	7.8	26.07
QRN	-	-	52.4	15.5	10.9	26.27
RoBERTa-BASE	56.27	65.71	71.33	31.78	34.05	45.72
Permutation Classfn.	57.71	**70.57**	**73.72**	**43.16**	32.72	**49.87**
$Emb_{Hamming}$	53.05	61.36	68.5	30.43	**36.16**	45.03
Emb_{Lehmer}	**60.61**	68.11	73.3	38.82	35.05	49.06
Skip-Clip	58.19	63.4	66.94	34.46	32.73	44.71

Table 2: Results on ProPara Dataset. Numbers in bold and underlined are the highest and the second-highest scores respectively.

4.2 Fine-tuning

We fine-tune and evaluate models pre-trained using techniques mentioned in Section 2 on ProPara and NPN-Cooking Datasets. Note that Embedding Regression has two variants based on the type of embedding used - Hamming Embedding ($Emb_{Hamming}$) and Lehmer Embedding (Emb_{Lehmer}). We use hyperparameter grid search on development sets corresponding to each pre-training variant to get the best set of hyperparameters, as mentioned ***in Section D.2 of Appendix***.

4.3 Baselines

NPN-Cooking Dataset: We use Neural Process Network Model (**NPN-Model**) (Bosselut et al., 2017), **KG-MRC** (Das et al., 2018), **DYNAPRO** (Amini et al., 2020), and RoBERTa-BASE (Liu et al., 2019) as baselines.

ProPara Dataset: We use a **rule-based** method called **ProComp** (Clark et al., 2018), a **feature-based** method (Mishra et al., 2018) using Logistic Regression and CRF, **ProLocal** (Mishra et al., 2018), **ProGlobal** (Mishra et al., 2018), **EntNet** (Henaff et al., 2016), **QRN** (Seo et al., 2016), and RoBERTa-BASE (Liu et al., 2019) as baselines.

4.4 Analysis of Results

Table 1 shows results of proposed methods and baselines on NPN-Cooking Dataset. We see that all proposed methods perform better than baselines w.r.t Test Accuracy. Permutation Classification gives the best dev set result, but falls behind on the test set, as classification on 100 classes leads to overfitting. Skip-Clip gives best test accuracy, with an improvement of 1.6% compared to RoBERTa-BASE, suggesting that predicting next step from a given context helps in Entity Tracking in Recipe Domain. $Emb_{Hamming}$ gives the second-highest test accuracy, showing that predicting permutation as an embedding is useful for this task.

Table 2 shows results of proposed methods and baselines on ProPara. RoBERTa-BASE is the best baseline. Most proposed methods beat baselines. Skip-Clip does not perform as well, suggesting that this pre-training method of predicting a future step in recipes does not transfer to open domain. Permutation Classification and Embedding Regression perform much better. Emb_{Lehmer} performs better on 5 out of 6 metrics compared to $Emb_{Hamming}$. Permutation Classification has the best Status Accuracy and Average Category Score and gives an improvement of 7.4% and 9% respectively compared to RoBERTa-BASE, showing that predicting a permutation helps in a task in another domain.

Comparison with LLMs: We compare with the following LLMs ***in Table 8 in Section D.4 of Appendix*** - (1) Open-source LLMs such as Falcon-7B-Instruct (instruction-fine-tuned Falcon-7B) (Penedo et al., 2023), Llama 2-7B-Chat (instruction-fine-tuned Llama 2-7B) (Touvron et al., 2023) (2) OpenAI's GPT-3.5 (OpenAI, 2021). The LLMs are used in a 1-shot and 3-shot In-Context Learning Setting (Dong et al., 2022). Table 8 shows that - (1) even though Falcon-7B-Instruct and Llama 2-7B-Chat have almost 14x the number of parameters compared to the proposed permutation-based methods, they perform considerably worse in comparison (2) the proposed methods outperform GPT-3.5 in 1-shot setting across all metrics, and GPT-3.5 in 3-shot setting across 3 out of 4 metrics, even though the number of parameters and pre-training data is just a small fraction of that of GPT-3.5.

Additionally, we compare predictions of Permutation Classification and well-performing baseline RoBERTa-BASE on a procedure ***in Table 9 in Section D.4 of Appendix***. We infer that Permutation Classification is able to better predict the step when an entity ceases to exist, compared to the baseline.

5 Combination of different pre-training strategies

In this section, we explore sequential combinations of pre-training strategies. As Permutation Classification performs consistently well, we experiment with one of either Skip-Clip, $Emb_{Hamming}$, or Emb_{Lehmer}, followed by Permutation Classification.

Tables 3 and 4 reveal combinations of pre-

Model	Test Acc
$Permutation\ Classfn.$	64.75
$Emb_{Hamming}$	64.92
Emb_{Lehmer}	64.29
$Skip\text{-}Clip$	**65.33**
$Emb_{Hamming} + Permutation\ Classfn.$	61.88
$Emb_{Lehmer} + Permutation\ Classfn.$	62.66
$Skip\text{-}Clip + Permutation\ Classfn.$	0.01

Table 3: Results of sequential combination of different pre-training strategies on NPN-Cooking Dataset.

Model	Cat1 Acc.	Cat2 Acc.	Cat3 Acc.	Avg. Cat. Acc.
$Permutation\ Classfn.$	**73.72**	**43.16**	32.72	**49.87**
$Emb_{Hamming}$	68.5	30.43	**36.16**	45.03
Emb_{Lehmer}	73.3	38.82	35.05	49.06
$Skip\text{-}Clip$	66.94	34.46	32.73	44.71
$Emb_{Hamming} + Permutation\ Classfn.$	63.14	22.74	34.94	40.27
$Emb_{Lehmer} + Permutation\ Classfn.$	59.89	23.95	33.83	39.22
$Skip\text{-}Clip + Permutation\ Classfn.$	44.63	12.44	5.44	20.84

Table 4: Results of sequential combination of different pre-training strategies on ProPara Dataset.

training strategies being inferior to using individual strategies, possibly because these strategies use different supervision cues. For instance, while Permutation Classification treats each permutation as an independent target class, Skip-Clip pushes representations of nearby steps closer, and vice versa. Hence, Skip-Clip + Permutation Classification performs poorly. Emb_{Lehmer}, unlike Permutation Classification, uses distance between each step before and after permuting as target encoding, hence, different permutations are not independent, making methods slightly inconsistent. $Emb_{Hamming}$, like Permutation Classification, has different targets for each permutation, but has a larger target vector than PC. Hence, $Emb_{Hamming}$ + Permutation Classification is reasonably good, but is inferior to each.

6 Conclusion

Our work is one of the first to propose order-based in-domain pre-training for procedural data to enhance Entity Tracking performance. We introduce 3 pre-training tasks - Permutation Classification, Embedding Regression, Skip-Clip to learn sequential nature of procedures. Skip-Clip performs best on the in-domain NPN-Cooking Task, while Permutation Classification and Embedding Regression perform best on the open-domain ProPara Task. We believe such methods could be extended to procedures in E-Manuals, manufacturing guides, etc.

Limitations

1. Our work focuses on recipes as a type of procedural text. It would require further study to see if the results can be generalized to other types of procedural text such as science processes or how-to guides.

2. The Entity Tracking Task is only one aspect of understanding procedural text. Other aspects such as identifying entities and their attributes, and understanding causal relationships between entities may also be important for some applications.

3. Our work evaluates the proposed methods on two specific datasets, which may not be representative of all possible scenarios. The performance of the methods on other datasets or real-world applications may vary.

Ethics Statement

The proposed methodology can be used for any type of procedural text, including user-generated procedures. However, before applying the model to such procedures, it is important to consider exposure bias patterns. Additionally, the interpretability of the model's output is limited, so users should exercise caution when using it.

References

Aida Amini, Antoine Bosselut, Bhavana Dalvi Mishra, Yejin Choi, and Hannaneh Hajishirzi. 2020. Procedural reading comprehension with attribute-aware context flow.

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020a. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020b. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2017. Simulating action dynamics with neural process networks.

Khyathi Raghavi Chandu, Eric Nyberg, and Alan W. Black. 2019. Storyboarding of recipes: Grounded contextual generation. In *ACL*.

Peter Clark, Bhavana Dalvi, and Niket Tandon. 2018. What happened? leveraging verbnet to predict the effects of actions in procedural text.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018. Building dynamic knowledge graphs from text using machine reading comprehension.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Alaaeldin El-Nouby, Shuangfei Zhai, Graham W Taylor, and Joshua M Susskind. 2019. Skip-clip: Self-supervised spatiotemporal representation learning by future clip order ranking. *arXiv preprint arXiv:1910.12770*.

Hossein Rajaby Faghihi and Parisa Kordjamshidi. 2021. Time-stamped language model: Teaching language models to understand the flow of events.

R. W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks.

M. G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Anna Korba, Alexandre Garcia, and Florence d'Alché Buc. 2018. A structured prediction approach for label ranking. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9008–9018, Red Hook, NY, USA. Curran Associates Inc.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Haejun Lee, Drew A. Hudson, Kangwook Lee, and Christopher D. Manning. 2020. Slm: Learning a discourse language representation with sentence unshuffling.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2018a. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2018b. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: A challenge dataset and models for process paragraph comprehension.

Abhilash Nandy, Soumya Sharma, Shubham Maddhashiya, Kapil Sachdeva, Pawan Goyal, and NIloy Ganguly. 2021. Question answering over electronic devices: A new benchmark dataset and a multi-task learning based QA framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4600–4609, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mehdi Noroozi and Paolo Favaro. 2016. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI*, pages 69–84. Springer.

OpenAI. 2021. Gpt-3.5 turbo documentation.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Query-reduction networks for question answering.

Jizhi Tang, Yansong Feng, and Dongyan Zhao. 2020. Understanding procedural text using interactive entity networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7281–7290, Online. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Zhihan Zhang, Xiubo Geng, Tao Qin, Yunfang Wu, and Daxin Jiang. 2021. Knowledge-aware procedural text understanding with multi-stage training. In *Proceedings of the Web Conference 2021*. ACM.

Appendix

A Introduction

B Pre-training Methods

C Downstream Tasks

D Experiments and Results

D.1 Pre-training Setup

Compute Details: Number of trainable parameters for each pre-training method is 488,126,475. We use Tesla V100 GPUs for our experiments. Permutation Classification and Embedding Regression Methods take about 24 GPU-Hours, while Skip-Clip takes about 8 GPU-Hours (GPU-Hours is the number of GPUs used multiplied by the training time in hours).

Pre-training Data: The statistics of Pre-training Data is elaborated in Table 5.

Dataset	No. of Recipes	No. of words (only steps)	No. of words (only ingredients)
Recipe1M+	1,029,720	137,364,594	54,523,219
RecipeNLG	1,643,098	147,281,977	73,655,858
Majumder et al. (2019)	179,217	23,774,704	3,834,978
Chandu et al. (2019)	33,720	26,243,714	-
Total	2,885,755	334,664,989	132,014,055

Table 5: Statistics of Pre-training Data

D.2 Fine-tuning

The set of hyperparameters used for performing Grid Search are mentioned in Tables 6 and 7.

Hyperparameter	Set of values
Number of recipe steps	{4, 6, 9}
Size of the permutation set	{2, 10, 50, 100}

Table 6: Set of hyperparameters used for grid search for Permutation Classification and Embedding Regression

Hyperparameter	Set of values
Number of steps used as input context	{3, 4}
Number of target steps	{3, 4}

Table 7: Set of hyperparameters used for grid search for Skip-Clip

The best set of hyperparameters obtained are as follows - (1) **Permutation Classification:** No. of recipe steps = 6, Size of permutation set = 100 (2) **Embedding Regression:** No. of recipe steps = 6, Size of permutation set = 50 (3) **Skip-Clip:** No. of steps used as input context = 4, No. of target steps = 4.

D.3 Baselines

D.4 Analysis of Results

Comparison with LLMs: Table 8 compares performance of our proposed methods with that of LLMs in 1 and 3-shot In-Context Learning setting.

	Cat1 Acc.	Cat2 Acc.	Cat3 Acc.	Avg. Cat Acc.
Falcon-7B-Instruct (1-shot)	50.42	5.42	0.38	18.74
Falcon-7B-Instruct (3-shot)	48.44	3.15	1.94	17.84
Llama 2-7B-Chat (1-shot)	47.88	9.74	6.44	21.35
Llama 2-7B-Chat (3-shot)	51.27	13.98	11.97	25.74
GPT-3.5 (1-shot)	53.25	24.66	11.37	29.76
GPT-3.5 (3-shot)	62.43	34.66	15.81	37.63
Permutation Classfn.	**73.72**	**43.16**	32.72	**49.87**
$Emb_{Hamming}$	68.5	30.43	**36.16**	45.03
Emb_{Lehmer}	73.3	38.82	35.05	49.06
Skip-Clip	66.94	34.46	32.73	44.71

Table 8: Results on the ProPara Dataset - LLMs vs. proposed permutation-based methods

Table 9 shows annotated ground truth, predictions of Permutation Classification, and well-performing baseline RoBERTa-BASE for an entity in the procedure.

	Ground Truth	Permutation Classification	RoBERTa-BASE
Procedure	flower	flower	flower
(Before the process starts)	-	-	-
1. A seed is planted.	-	-	-
2. It becomes a seedling.	-	-	-
3. It grows into a tree.	-	-	-
4. The tree grows flowers.	tree	tree	tree
5. The flowers become fruit.	-	-	tree
6. The fruits contain seeds for new trees.	-	-	tree

Table 9: Analysis of the ground truth and the predictions of Permutation Classification vs. a well-performing baseline on a sample from the ProPara Dataset.