Enhancing Neural Network Performance through Model-Generated Training Signals

Ruohong Zhang

CMU-LTI-25-002

February, 2025

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15123

Thesis Committee:
Yiming Yang, Chair
Chenyan Xiong
Daniel Fried
Guokun Lai (Moonshot AI)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Language and Information Technology.

Keywords: large language model, time series analysis, text classification, vision language model

Abstract

Neural networks have been widely applied to tasks such as natural language processing, time series analysis, and multimodal understanding. Their optimization typically requires large volumes of high-quality data, but human annotation is expensive and does not scale effectively. As an alternative, this thesis explores methods that *leverage model-generated training signals* to either replace or supplement manual labels, enabling robust neural network optimization while reducing reliance on human annotation. This contributes to the development of more automated and efficient artificial general intelligence.

This thesis demonstrates that leveraging model-generated signals can lead to superior performance across various domains. We validate this methodology through successful applications in three major areas, which form the core components of this thesis: time series forecasting and change-point detection, text classification with limited or no labeled data, and large language model alignment.

In Part Part I: We improve text classification by incorporating model-augmented document content and label descriptions in few-shot and zero-shot learning settings. In Part Part II: We utilize generated correlation graphs as augmented signals to enhance change-point detection in time series analysis. In Part Part III: We leverage self-enhancement techniques, such as reinforcement learning, to align large language models for developing more robust chatbots, improving instruction-following in video-language models, and enhancing reasoning in vision-language models.

Collectively, this work advances neural network optimization through model-generated signals across multiple domains, contributing to the development of intelligent AI systems with minimal human supervision.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Yiming Yang, for her invaluable guidance, encouragement, and support throughout my Ph.D. journey. Her insights and expertise have profoundly shaped my research and growth as a scholar. This thesis would not be possible without her guidance and support.

I am also grateful to my committee members Chenyan Xiong, Daniel Fried and Guokun Lai – for their constructive feedback and for challenging me to refine my work. Special thanks to my colleagues and friends at LTI, CMU, for their collaboration, stimulating discussions, and unwavering support.

I would also like to extend my gratitude to my collaborators in industry, who provided valuable computational resources that enabled meaningful experiments and shared business insights that contributed to building better LLM products. These experiences have reshaped my research direction and helped me discover my passion for product development, which I see as a vital part of my future career.

Last but not the least, I would like to extend my appreciation to my family members. To my spouse, Chen Chen, thank you for your patience, understanding, and encouragement. This journey would not have been possible without your support.

Contents

Chapter 1

Introduction

1.1 Motivation of Research

Neural networks have diverse applications, including natural language processing, time series analysis, and multimodal understanding. Optimizing these networks for specialized tasks, such as text classification or change-point detection in time series, often requires extensive domain-specific annotations. While large corpora like Wikipedia provide abundant human-annotated categories, their scale does not guarantee sufficient training data for all sub-tasks. For instance, Wiki10, a large-scale classification dataset with 31K labels, is highly imbalanced: 88.7% of labels appear in only 1–9 examples, 10.3% in 10–100 examples, and just 1% in 100+ examples. This means that if a network requires at least 10 labeled examples per class for effective training, only about 10% of labels are adequately represented. For the remaining 88%, the model must rely on fewer than 10 examples per label, which is often insufficient for robust learning. This illustrates how low-resource challenges naturally arise, motivating the need for advanced methods to improve training efficiency. One of the most effective approaches is leveraging model-generated signals to enhance network optimization.

While we propose leveraging model-generated signals as a general methodology for optimizing neural networks, different scenarios require distinct insights to address specific challenges. This thesis investigates the effective utilization of model-generated signals across various settings, particularly in tasks with limited task-relevant labeled data. It develops a series of methods spanning multiple domains, including text classification (Chapter 2, Chapter 3), time series analysis (Chapter 4), domain-specific chatbot training (Chapter 5), and image and video instruction-following (Chapter 6, Chapter 7).

By exploring different types of generated signals—such as graph-based generation, label descriptions, document content generation, and conversation data creation, reasoning chain generation—this work aims to reduce reliance on human annotations, enhancing AI system capabilities in data-scarce environments. Ultimately, this thesis contributes to the development of more automated and efficient artificial intelligence, fostering intelligent systems that require minimal human supervision.

1.2 Thesis Organization

This thesis explores the optimization of neural models using model-generated signals, aiming to enhance performance and reduce dependence on annotated data. It is structured into three main parts, each addressing a distinct aspect of this research area.

Part I: Text Classification with Limited Labeled Data As discussed in the introduction, label scarcity naturally arises, with low-frequency labels (fewer than 10 occurrences) dominating the label space. Traditional methods that rely on document-label co-occurrence for training often fail to provide sufficient supervision. This part investigates effective strategies for augmenting label or document information to enable robust few-shot and even zero-shot text classification.

Specifically in Chapter 2, we propose leveraging retrieval models for few-shot label augmentation in extreme classification and further enhancing label representations using statistical support vector machines (SVMs). We demonstrate that SVMs, as large-margin classifiers, improve robustness in tail-label prediction, while retrieval models can utilize semantic information from SVM label profiles for enhanced performance.

Beyond label-side augmentation, we explore document-side augmentation in Chapter 3 for zero-shot text classification. We show how instruction-aligned large language models (LLMs) enhance the semantic richness of short input documents, leading to more robust performance. Additionally, we employ LLMs to generate reliable training instances during self-training, improving the performance of a smaller topic classification model (e.g., SimCSE).

The success of few-shot extreme classification and zero-shot topic text classification demonstrates the effectiveness of leveraging model-generated signals for label and document augmentation. These findings highlight the potential of model-generated signals in mitigating label scarcity and improving text classification performance in data-constrained settings.

Part II: Auxiliary Graph Generation for Time Series Analysis The change-point detection task aims to identify significant and abrupt distributional shifts within a time series. Although time series data can span thousands of time steps, change-points—such as machine failures in sensor data—typically occur infrequently, if at all, resulting in naturally low-occurrence training data. Moreover, labeling change-points requires domain expertise, making it challenging to scale up training datasets.

To address this problem, in Chapter 4, we investigate unsupervised methods for change-point detection that do not rely on large labeled datasets. We are the first to study correlation changes in multivariate time series and propose leveraging generated correlation graphs as additional training signals for time series prediction. In our approach, predicted time series values are compared with observed values to detect distributional shifts, forming an unsupervised method for change-point detection. We demonstrate that augmenting graph signal to time series models enhances both forecasting and change-point detection, particularly in scenarios involving complex multivariate interactions. By integrating correlation graphs, we significantly improve the model's ability to detect change-points arising from shifts in correlation patterns. This advancement represents a step forward in applying neural models to time series analysis, highlighting the potential of auxiliary data sources in enhancing model performance.

Part III: Self-enhancement Methods for Large Language Models After the advancement of ChatGPT, chatbots have become valuable assistants for daily tasks. However, as highlighted in InstructGPT [111], conversation data is obtained through human annotation, which is both costly and challenging to control in terms of quality, especially when human annotators introduce factual errors. To address these issues, we propose generating large-scale conversation data automatically to train domain-specific textual chatbots (Chapter 5) and general video-language model chatbots (Chapter 6).

Specifically, in Chapter 5, we explore aligning LLMs to autonomously mine and learn from domain specific text corpus, enhancing their ability to acquire and utilize knowledge from documents. In Chapter 6, we leverage our textual conversation miners as an auxiliary model to generate large-scale video instruction-following data from long-form, detailed video captions. Additionally, we incorporate self-generated traces from a fine-tuned video model for reinforcement learning, significantly improving model performance.

More recently, to enhance the explainability and accuracy of chatbot outputs, chain-of-thought (CoT) prompting [139] has been employed to generate reasoning traces, improving both clarity and correctness. In Chapter 7, we investigate effective strategies for leveraging

self-generated reasoning traces in vision-language models and apply reinforcement learning on these traces to enhance the reasoning capabilities of vision-language models.

Summary of Contribution Each part of the thesis provides an in-depth examination of its respective topic, collectively contributing to the broader goal of advancing neural network optimization through model-generated signals. The goal of these studies presents a comprehensive understanding of how to design signals to effectively improve the model performance across various applications, potentially revolutionizing the field of artificial intelligence.

1.3 Related Publications

For the topic of extreme text classification discussed in Chapter 2, we have published DEPL in EACL 2023 [168] and an ArXiv paper on GloCalXML in 2023 [165]. For zero-shot text classification discussed in Chapter 3, we have published PESCO in ACL 2023 [138] and GenCO in EACL 2024 [167]. For time series change-point detection in Chapter 4, we have published CoRD-CPD in ICONIP 2020 [164]. For large language model alignment in Chapters 5 to 7, we have published LLMiner as an ArXiv preprint in 2023 [166], LLaVA-Hound in NAACL 2025 [160], and LLaVA-Reason, which is currently under submission for ACL 2025 [170].

Part I

Text Classification with Limited Labeled Data

Chapter 2

Pseudo Label Description Generation for XMTC

In this chapter, we explore methods for extreme Multi-label Text Classification (XMTC) that mitigate reliance on manually annotated text-label pairs by leveraging model-generated training signals. XMTC poses significant challenges due to its long-tailed label distribution and large label space, which exacerbate data scarcity for low-frequency labels. Traditional classification approaches struggle in this setting, as they depend on direct supervision from labeled examples. To address this limitation, we propose a neural retrieval-based framework that replaces explicit text-label supervision with semantic matching, mapping input documents to relevant label descriptions. Furthermore, we enhance label descriptions by generating pseudo label representations using a bag-of-words (BoW) classifier, which serves as an auxiliary training signal in data-scarce conditions. Our approach highlights the use of statistical model to generate useful signals to guide neural retrievers, significantly improving the prediction of tail labels and surpassing state-of-the-art models. Additionally, we provide a theoretical analysis that establishes a lower bound on model performance, offering insights into the interplay between statistical and neural approaches in label-scarce regimes.

Highlights We are the first to train a neural retriever specifically for the XMTC task, showcasing its efficacy in leveraging training signals generated from statistical model to enhance neural retriever, particularly for rare labels.

Model-generated Signals We leverage the pseudo label descriptions generated from a trained statistical (SVM) model. By integrating insights from statistical classifiers, which have tradi-

tionally leveraged manually defined features like bag-of-words to encapsulate statistical data, with neural models, our research uncovers a synergistic potential. This combination leverages the strengths of both model types, offering a composite view that enhances overall performance in XMTC tasks.

2.1 Introduction

Extreme multi-label text classification (XMTC) is the task of tagging documents with relevant labels in a very large and often skewed candidate space. It has a wide range of applications, such as assigning subject topics to news or Wikipedia articles, tagging keywords for online shopping items, classifying industrial products for tax purposes, etc.

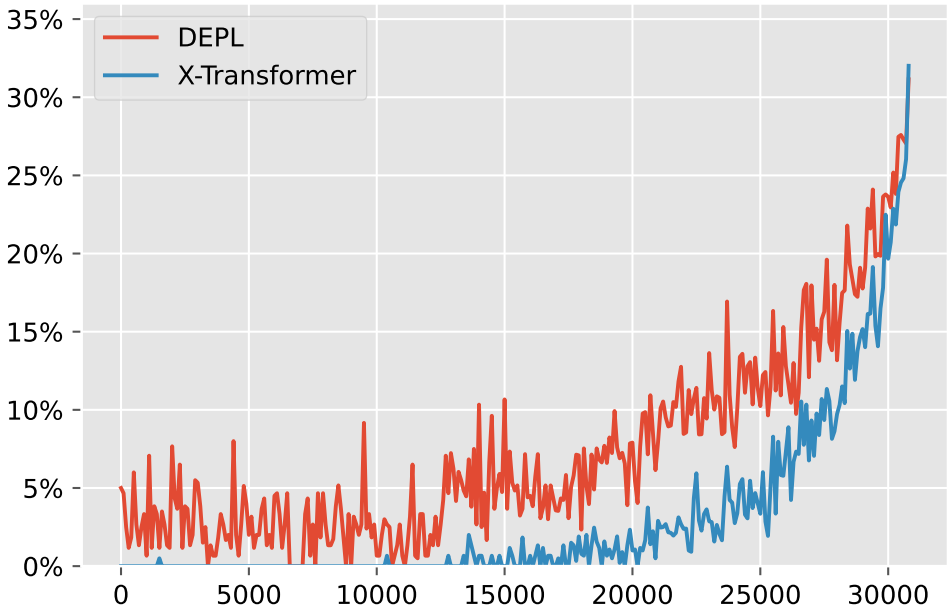


Figure 2.1: The classification performance of X-Transformer and DEPL (ours) measured in macro-averaged $F1@19$ on the Wiki10-31K dataset.

The most difficult part in solving the XMTC problem is to train classification models effectively for the rare labels in the long tail of highly skewed distributions, which suffers severely from the lack of sufficient training instances. Efforts addressing this challenge by the text classification community include Bayesian modeling of graphical dependencies among labels [47], novel loss or regularization of label embeddings [6, 140], clustering-based algorithms [16, 71, 112], and so on. Despite the remarkable progresses made so far, the problem is still

very far from being well solved. Figure 2.1 shows the performance of X-Transformer [16], one of the state-of-the-art (SOTA) XMTC models, on the Wiki10-31K benchmark dataset (with over 31k labels). The horizontal axis is the ranks of the labels sorted from rare to common and the vertical axis is the text classification performance measured in macro-averaged $F1@19$ (higher the better) for binned labels (100 labels per bin). The blue curve is the result of X-Transformer, which has the scores close to 0 (worst possible score) for nearly half of the total labels. In other words, SOTA methods in XMTC still perform poorly in tail label prediction.

In this chapter, we seek solutions for tail label prediction from a new angle: we introduce a novel framework, namely the Dual Encoder with Pseudo Label (DEPL). It treats each input document as a query and uses a neural network model to retrieve relevant labels from the candidate space based on the textual descriptions of the labels. The underlying assumption is, if the label descriptions are highly informative for text-based matching, then the retrieval system should be able to find relevant labels. The system would be particularly helpful for tail label prediction as the retrieval effectiveness does not necessarily rely on the availability of a large number of training instances, which is what the tail labels are lacking.

The next research question that we tackle is how to obtain highly informative descriptions for each label without human annotation. In reality, class names are often available but they are typically one or two words, which cannot be sufficient for retrieval-based label prediction. Therefore, we propose to augment the label description with statistical learning algorithms. Specifically, we train linear support vector machine (SVM) model with the bag-of-words (BoW) features, such as tf-idf, to automatically generate informative keywords for each label, which we call the *pseudo description* of the label. Since the learned label embeddings of the BoW classifier encode token importance information, it is natural and efficient to leverage them for keywords extraction. In sections 2.4 and 2.5.4, we further provide theoretical motivations and empirical evidence to show the advantage of unsupervised statistical features for classification under extreme scarce data conditions.

The result of our approach (DEPL) is shown as the red curve in Figure 1, which significantly outperforms the blue curve of X-Transformer not only in the tail-label region but also in all other regions. We also observed similar improvements by DEPL over strong baselines on other benchmark datasets (see § 2.5.4). Our main contributions are summarized as the following:

1. We propose DEPL, a retrieval-based model to alleviate the difficulty in tail label prediction by matching the semantics between documents and augmented label descriptions which are generated automatically by a statistical model with BoW features.

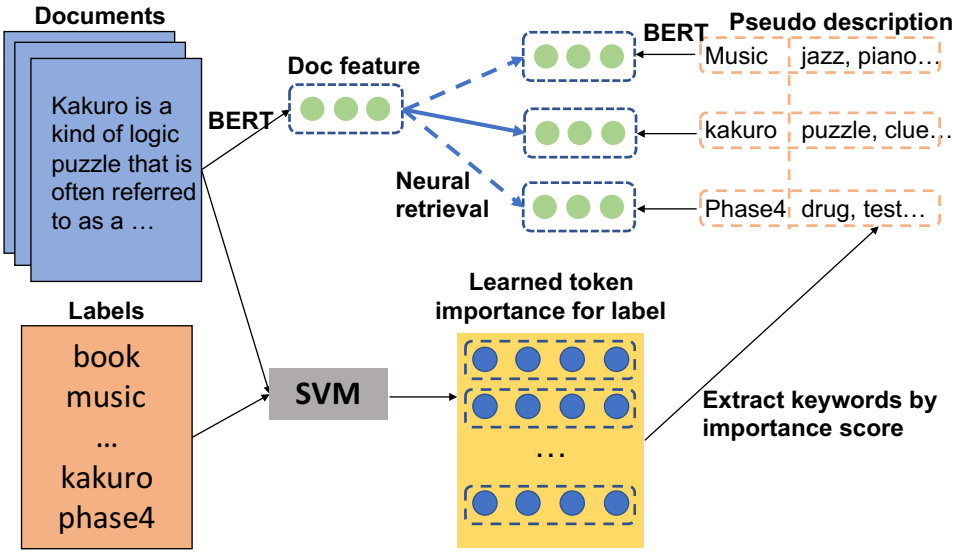


Figure 2.2: The proposed DEPL framework. First, we train a BoW classifier (SVM) and extract the top keywords from the label embeddings according to the learned token importance. Then, we concatenate the keywords with the original label names to form pseudo descriptions. Finally, we leverage the neural retrieval model to rank the labels according to semantic matching between document text and label descriptions.

2. We provide theoretical analyses to motivate the usage of BoW feature for classification under scarce data setting, and prove a performance lower bound of the neural model.
3. We did extensive experiments with different tail label evaluation metrics to show that our method significantly and consistently outperforms strong baselines on multiple challenging benchmark datasets.

2.2 Related Work

XMTC Classifier Traditional BoW classifiers rely on the bag-of-words features such as one-hot vector with tf-idf weights, which capture the word importance in a document. Examples include one-vs-all SVM models such as DiSMEC [5], ProXML [7], PPDSparse [155], tree-based models such as Parabel [112] and Bonsai [71].

To compensate for the lack of semantics in BoW features, deep learning models were proposed for XMTC. Examples include CNN-based models such as XML-CNN [92] and SLICE [60], RNN-based models such as AttentionXML [157] and Transformer-based models such as

X-Transformer [16], LightXML [64] and APLC-XLNet [153].

Label Description The SiameseXML [30] for XMTC encodes both input documents and label descriptions with pretrained word embeddings with shallow networks and leverages the embedding matching. The SOTA pretrained Transformer-based models [16, 64] leverage the label descriptions to build label clusters. To generate label descriptions, Chai et al. [14] adopt reinforcement learning to produce extended label descriptions from predefined label descriptions. However, the algorithm can not scale to the extreme label space and relies on the availability of sufficient training data.

2.3 Proposed Method

2.3.1 Preliminaries

Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{N_{\text{train}}}\}$ be the training data where $\mathbf{x}_i$ is the input text and $\mathbf{y}_i \in \{0,1\}^L$ are the binary ground truth labels of size L. Given an instance $\boldsymbol{x}$ and a label l, a classification system produces a matching score of the text and label:

$$f(\boldsymbol{x}, l) = \langle \phi(\boldsymbol{x}), \boldsymbol{w}_l \rangle$$

where $\phi(\boldsymbol{x})$ represent the document feature vector and $\boldsymbol{w}_l$ represents the label embedding of l. The dot product $\langle \cdot, \cdot \rangle$ is used as the similarity function.

Typically, the label embedding $\boldsymbol{w}_l$ is randomly initialized and trained from the supervised signal. While learning the embedding as free parameters is expressive when data is abundant, it could be difficult to be optimized under the scarce data situation.

Sketch of Method DEPL tackles the long-tailed XMTC by neural retrieval with generated pseudo label descriptions, as shown in figure 2.2. Instead of learning the label embedding from scratch, the retrieval module directly leverages the **semantic matching** between the document and label text, providing a strong inductive bias on tail label prediction. Next, we introduce the components of our system in details.

2.3.2 Generated Pseudo Label Description

As the provided label names are usually short and noisy, we augment it with generated pseudo label description from a SVM model. As the tf-idf features $\phi_t(\boldsymbol{x})$ used by SVM are sparse, we

also call the statistical model a *sparse* model:

$$f_{\text{sparse}}(\boldsymbol{x}, l) = \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_l^{\text{svm}} \rangle$$

The label embedding weight $\boldsymbol{w}_l^{\text{svm}}$ is optimized with the hinge loss:

$$\mathcal{L}_{\text{hinge}} = \frac{1}{LB} \sum_{i=1}^{B} \sum_{l=1}^{L} \max(0, 1 - \tilde{y}_l \cdot f_{\text{sparse}}(\boldsymbol{x}_i, l))$$

where $\tilde{y}_l = 2y_l - 1 \in \{-1, 1\}$, B is the batch size.

For a trained SVM model, $\boldsymbol{w}_l^{\text{svm}}$ has the dimension equal to the vocabulary size and each value w_{li}^{svm} of the label embedding denotes the learned importance of the token i w.r.t label l. We select the top k most important tokens (ranked according to the importance score) as keywords, which are appended to the original label name to form the pseudo label description:

$$\text{pseudo\_label}(l) = \text{label\_name}(l) \oplus \text{keywords}(l)$$

where $\oplus$ is the append operation.

2.3.3 Retrieval Model with Label Text

DEPL leverages the semantic matching of document and label texts via a dual encoder model [42, 68, 99, 143]. We use the BERT [34] model as the backbone of our neural encoder, which is shared for both the document and label text encoding. Since a neural model encodes textual inputs into condensed vector representations, we call them *dense* models.

The similarity between text and label representation is measured by:

$$f_{\text{dual}}(\boldsymbol{x}, l) = \langle \phi_{\text{doc}}(\boldsymbol{x}), \phi_{\text{label}}(\text{text}(l)) \rangle$$

where $\text{text}(l)$ is the textual information of the label l. When the textual information only includes the label name given in the dataset, we call the model **DE-ret**. Otherwise, when the textual information includes the generated pseudo label description, we call the model **DEPL**.

The document embedding $\phi_{\text{doc}}(\boldsymbol{x})$ is obtained from the CLS embedding of the BERT model followed by a linear pooling layer:

$$\phi_{\text{doc}}(\boldsymbol{x}) = \boldsymbol{W}_{doc} \cdot \text{BERT}(\boldsymbol{x}, \text{CLS}) + \boldsymbol{b}_{doc}$$

where $\text{BERT}(\boldsymbol{x}, \text{CLS})$ represents the contextualized embedding of the special CLS token. $\boldsymbol{W}_{doc}$ and $\boldsymbol{b}_{doc}$ are the weights and biases for the document pooler layer.

For the label embedding $\phi_{\text{label}}(\text{text}(l))$, we take an average of the last hidden layer of BERT followed by a linear pooler layer:

$$\phi_{\text{label}}(\text{text}(l)) = \boldsymbol{W}_{label} \cdot \psi_{bert}(\text{text}(l)) + \boldsymbol{b}_{label} \tag{2.1}$$

$$\psi_{bert}(\text{text}(l)) = \frac{1}{|\,\text{text}(l)|} \sum_{j=1}^{|\,\text{text}(l)|} \text{BERT}(\text{text}(l), j) \tag{2.2}$$

where $\text{BERT}(\text{text}(l), j)$ represents the contextualized embedding of the j-th token in $\text{text}(l)$ obtained from the last hidden layer of the BERT model. $\boldsymbol{W}_{label}$ and $\boldsymbol{b}_{label}$ are the weights and biases for the label pooler layer. In the equation 2.2, the average embedding of label tokens yields better performance empirically than the CLS embedding possibly because the keywords are not natural language, and BERT may not effectively aggregate such type of information into CLS.

Learning with Negative Sampling Since calculating all the label embeddings for each batch is both expensive and prohibitive by the memory limit, we resort to negative sampling strategies for in-batch optimization. Specifically, we sample a fixed-sized subset of labels for each batch containing: 1) all the positive labels of the instances in the batch, 2) the top negative predictions by the sparse classifier as the hard negatives, and 3) the rest of the batch is filled with uniformly random sampled negatives labels.

Let $\mathbb{S}_b$ be the subset of labels sampled for a batch. The objective for the dual encoder is:

$$\mathcal{L}_{\text{dual}} = -\frac{1}{B|\mathbb{S}_b|} \sum_{i=1}^{B} \left(\sum_{p \in \boldsymbol{y}_i^+} \log \sigma(f_{\text{dual}}(\boldsymbol{x}_i, p)) + \sum_{n \in \mathbb{S}_b \setminus \boldsymbol{y}_i^+} \log \sigma((1 - f_{\text{dual}}(\boldsymbol{x}_i, n))) \right)$$

where B is the batch size, $\boldsymbol{y}_i^+$ is the positive labels for instance i, and σ is the sigmoid function.

2.3.4 Connection of Sparse and Dense Model

Complementary features: the sparse model uses the tf-idf feature based on corpus-level token statistics, while the dense model relies on the knowledge of the language learned during pre-training. The two types of features focus on different aspects of the text corpus and the combination of the two brings gains in performance.

Difference from ensemble: utilizing the augmented text for retrieval is better than a pure ensemble of sparse and dense methods such as in X-Transformer. In the ensemble method, the semantic meaning of important tokens in a label embedding learned from sparse classifier is not

leveraged. By extracting the keywords from the sparse label embedding and presenting them as pseudo label descriptions, our model can additionally exploit the value of those key token semantics.

2.3.5 Enhance Classification with Retrieval

Our introduced retrieval model can be combined with a neural classifier for a performance boost on overall label classification (since our retrieval model is primarily targeted on improving tail label performance). In a neural classification system, the label embedding is treated as free parameters to be learned from supervised data, which is more expressive for labels with abundant training instances. The neural classifier learns the function:

$$f_{\text{cls}}(\boldsymbol{x}, l) = \langle \phi_{\text{doc}}(\boldsymbol{x}), \boldsymbol{w}_l^{\text{cls}} \rangle \tag{2.3}$$

We propose to enhance the classification model with the retrieval mechanism by jointly fine-tuning:

$$f_{\text{dual-cls}}(\boldsymbol{x}, l) = \frac{\sigma(f_{\text{dual}}(\boldsymbol{x}, l)) + \sigma(f_{\text{cls}}(\boldsymbol{x}, l))}{2} \tag{2.4}$$

The classification and retrieval modules share the same BERT encoder. We refer to the system as **DEPL+cls**. The object function $\mathcal{L}_{\text{dual-cls}}$ is similar to $\mathcal{L}_{\text{dual}}$ except for replacing f_{dual} with $f_{\text{dual-cls}}$.

The **DEPL+cls** model looks like an ensemble of the two systems at first sight, but there are two major differences: 1) As the BERT encoder is shared between the classification and retrieval modules, it doesn't significantly increase the number of parameters as in [16, 64]; and 2) when the two modules are optimized together, the system can take advantages of both units according to the situation of head or tail label predictions.

2.4 Theoretical Analyses of DEPL

2.4.1 Rethinking Dense and Sparse Model for Imbalanced Text Classification

We analyze dense and sparse models from a gradient perspective for classification problems with skewed label distribution.

Preliminary: The predicted probability optimized by the binary cross entropy (BCE) loss is:

$$\mathcal{L}_{\text{BCE}} = -\sum_{l=1}^{L} y_l \log p_l + (1 - y_l) \log(1 - p_l)$$

The derivative of $\mathcal{L}_{\text{BCE}}$ w.r.t the logits s_l is:

$$\frac{\partial \mathcal{L}_{\text{BCE}}}{\partial s_l} = \begin{cases} p_l - 1 & \text{if } y_l = 1 \\ p_l & \text{otherwise} \end{cases} \tag{2.5}$$

Q1: *Why would sparse model with BOW feature benefit tail label prediction?*

Applying the chain rule to equation 2.5, the gradient of $\mathcal{L}_{\text{BCE}}$ w.r.t the document feature $\phi_n(\boldsymbol{x})$ is:

$$\frac{\partial \mathcal{L}_{\text{BCE}}(y_l, p_l)}{\partial \phi_n(\boldsymbol{x})} = \begin{cases} (p_l - 1)\boldsymbol{w}_l & \text{if } y_l = 1 \\ p_l \boldsymbol{w}_l & \text{otherwise} \end{cases}$$

By optimizing parameters θ of feature extractor, the document representation is encourage to move away from the negative label representation, that is:

$$\phi_n(\boldsymbol{x}; \theta') \leftarrow \phi_n(\boldsymbol{x}; \theta) - \eta p_l \boldsymbol{w}_l$$

where η is the learning rate.

For a dense model, the parameter θ of the feature extractor (such as BERT) is shared for all the data, so the optimization of the feature extractor is affected by the distribution of labels in the training data. Since a tail label appears more often as a negative target, the feature extractor is likely to under-represent the tail label information, making a tail label more difficult to be predicted. In comparison, the sparse feature like tf-idf is derived in an unsupervised manner from corpus statistics, which is independent of training label distribution. Therefore, the sparse feature may maintain better representation power to separate the tail labels.

Q2: *What is the advantage of a retrieval system on tail label prediction?*

In a typical classification system, labels are treated as indices whose embeddings are randomly initialized and learned from supervised signals. The gradients of $\mathcal{L}_{\text{BCE}}$ w.r.t the label feature is:

$$\frac{\partial \mathcal{L}_{\text{BCE}}(y_l, p_l)}{\partial \boldsymbol{w}_l} = \begin{cases} (p_l - 1)\phi_n(\boldsymbol{x}) & \text{if } y_l = 1 \\ p_l \phi_n(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

The label embedding is updated by:

$$\boldsymbol{w}_l' = \boldsymbol{w}_l + \frac{\eta}{N_{\text{train}}} \sum_{i:y_{il}=1} (1 - p_{il})\phi_n(\boldsymbol{x}_i) - \frac{\eta}{N_{\text{train}}} \sum_{i:y_{il}=0} p_{il}\phi_n(\boldsymbol{x}_i)$$

As most of the instances are negative for a tail label, the update of tail label embedding is inundated with the aggregation of negative features, making it hard to encode distinctive feature reflecting its identity. Therefore, learning the tail label embedding from supervised signals alone can be distracting. Although previous works leverage negative sampling to alleviate the problem [16, 64], we argue that a fundamental solution is to inject the label information into the embedding. Our proposed retrieval system presents a natural way to incorporate label text for enhanced performance of tail label prediction.

2.4.2 Analysis on Performance Lower Bound

We will show the connection between DEPL and a sparse SVM classifier (for pseudo label extraction) by a performance lower bound. Specifically, DEPL outperforms a sparse model with high probability given that the selected keywords are important and the sparse classifier can separate the positive from the negative instances with non-trivial margin.

Notation: Let $\phi_t(\boldsymbol{x})$ be the normalized tf-idf feature vector of text with $\|\phi_t(\boldsymbol{x})\|_2 = 1$. The sparse label embeddings $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L\}$ satisfies $\|\boldsymbol{w}_l\|_2 \leq 1, w_{li} > 0$. In fact, label embeddings can be transformed to satisfy the condition without affecting the prediction rank. Let $\boldsymbol{z}_l$ be the top selected keywords from the sparse classifier, which is treated as the pseudo label. Define the sparse keyword embedding $\boldsymbol{v}_l$ with $v_{li} = w_{li}$ if i is an index of selected keywords and 0 otherwise.

In the following, we define the keyword importance and the classification error margin.

Definition 1. *For label l and $\delta \geq 0$, the sparse keyword embedding $\boldsymbol{v}_l$ is δ-bounded if $\langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_l \rangle \geq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_l \rangle - \delta$.*

Definition 2. *For two labels p and n, the error margin μ is the difference between the predicted scores $\mu(\phi(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_n) = \langle \phi(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n \rangle$.*

The main theorem is stated as below:

Theorem 3. *Let $\phi_t(\boldsymbol{x})$ and $\phi_n(\boldsymbol{x})$ be the sparse and dense (dimension d) document feature, $\boldsymbol{w}_l$ be the label embedding and $\boldsymbol{z}_l$ be the δ-bounded keywords. For a positive label p, let $\mathbb{N}_p = \{n_1, \ldots, n_{M_p}\}$ be a set of negative labels ranked lower than p. The error margin $\epsilon_i = \mu(\phi_t(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_{n_i})$ and $\epsilon = \min(\{\epsilon_1, \ldots, \epsilon_{M_p}\})$. An error $\mathcal{E}_i$ of the neural classifier occurs when*

$$\mu(\phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_p), \phi_n(\boldsymbol{z}_{n_i})) \leq 0 \tag{2.6}$$

The probability of any such error happening satisfies

$$P(\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{M_p}) \leq 4M_p \exp(-\frac{(\epsilon - \delta)^2 d}{50})$$

When $(\epsilon - \delta) \geq 10\sqrt{\frac{\log M_p}{d}}$*, the probability is bounded by* $\frac{1}{M_p}$.

Discussion: An error event occurs when the sparse model makes a correct prediction but the neural model doesn't. If the neural model avoids all such errors, the performance should be at least as good as the sparse model, and Theorem 3 gives a bound of that probability.

The term δ measures the importance of selected keywords (smaller the more important), the error margin ϵ measures the difficulty the correctly predicted positive and negative pairs by the sparse model. The theorem states that the model achieves a lower bound performance as sparse classifier if the keywords are informative and error margin is non-trivial. Proofs and discussions are presented in section 2.4.3.

2.4.3 Proof

We include the assumptions and proofs of Theorem 3.

Assumptions Similar to Luan et al. [99], we treat neural embedding as fixed dense vector $\boldsymbol{E} \in \mathbb{R}^{d \times v}$ with each entry sampled from a random Gaussian $N(0, d^{-1/2})$. $\phi_n(\boldsymbol{x}) = \boldsymbol{E}\phi_t(\boldsymbol{x})$ is weighted average of word embeddings by the sparse vector representation of text. According to the *Johnson-Lindenstrauss (JL) Lemma* [12, 67], even if the entries of $\boldsymbol{E}$ are sampled from a random normal distribution, with large probability, $\langle \phi_t(\boldsymbol{x}), \boldsymbol{v} \rangle$ and $\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}\boldsymbol{v} \rangle$ are close.

Lemma 4. *Let* $\boldsymbol{v}$ *be the* δ*-bounded keyword-selected label embedding of* $\boldsymbol{w}$*. For two labels* p, n*, the error margins satisfy:*

$$|\mu(\phi_t(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_n) - \mu(\phi_t(\boldsymbol{x}), \boldsymbol{v}_p, \boldsymbol{v}_n)| \leq \delta$$

Proof. By the definition of δ-bounded keywords,

$$\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p \rangle - \delta \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p \rangle \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p \rangle \tag{2.7}$$

$$-\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_n \rangle \leq -\langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_n \rangle \leq -\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_n \rangle + \delta \tag{2.8}$$

Adding equation 2.7 and equation 2.8 finishes the proof:

$$\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n \rangle - \delta \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n \rangle \leq \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n \rangle + \delta \tag{2.9}$$

□

Lemma 5. *Let $\phi_t(\boldsymbol{x})$ and $\phi_n(\boldsymbol{x})$ be the sparse and dense (dimension d) document feature, $\boldsymbol{w}_l$ be the label embedding and $\boldsymbol{z}_l$ be the δ-bounded keywords. Let p be a positive label and n be a negative label ranked below p be the sparse classifier. The error margin is $\epsilon = \mu(\phi_t(\boldsymbol{x}), \boldsymbol{w}_p, \boldsymbol{w}_n)$. An error $\mathcal{E}$ of neural classification occurs when $\mu(\phi_n(\boldsymbol{x}), \phi_n(\boldsymbol{z}_p), \phi_n(\boldsymbol{z}_n)) \leq 0$. The probability $P(\mathcal{E}) \leq 4\exp(-\frac{(\epsilon-\delta)^2 d}{50})$.*

Proof. By the JL Lemma [12]: For any two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^v$, let $\boldsymbol{E} \in \mathbb{R}^{d\times v}$ be a random matrix such that the entries are sampled from a random Gaussian. Then for every constant $\gamma > 0$:

$$P\left(|\langle \boldsymbol{E}\boldsymbol{a}, \boldsymbol{E}\boldsymbol{b}\rangle - \langle \boldsymbol{a}, \boldsymbol{b}\rangle| \geq \frac{\gamma}{2}\left(\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2\right)\right) \leq 4\exp\left(-\frac{\gamma^2 d}{8}\right) \tag{2.10}$$

Let $\gamma = \frac{2}{5}(\epsilon-\delta)$, $\boldsymbol{a} = \phi_t(\boldsymbol{x})$ and $\boldsymbol{b} = \boldsymbol{v}_p - \boldsymbol{v}_n$. Since $\|\boldsymbol{a}\|_2 = 1$ and $\|\boldsymbol{b}\|_2 \leq (\|\boldsymbol{v}_p\|_2 + \|\boldsymbol{v}_n\|_2)^2 \leq 4$, the JL Lemma gives

$$P\left(|\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \geq \epsilon - \delta\right) \tag{2.11}$$

$$\leq 4\exp(-\frac{(\epsilon-\delta)^2 d}{50}) \tag{2.12}$$

To complete the proof, we need to show $P(\mathcal{E}) \leq Eq.2.11$:

$$\mathcal{E} \implies |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle| \geq \epsilon \tag{2.13}$$

$$\implies |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \geq \epsilon - \delta \tag{2.14}$$

where the equation 2.14 is derived by Lemma 4:

$$|\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \tag{2.15}$$

$$\geq |\langle \boldsymbol{E}\phi_t(\boldsymbol{x}), \boldsymbol{E}(\boldsymbol{v}_p - \boldsymbol{v}_n)\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle| - \tag{2.16}$$

$$|\langle \phi_t(\boldsymbol{x}), \boldsymbol{w}_p - \boldsymbol{w}_n\rangle - \langle \phi_t(\boldsymbol{x}), \boldsymbol{v}_p - \boldsymbol{v}_n\rangle| \tag{2.17}$$

$$\geq \epsilon - \delta \tag{2.18}$$

Therefore $P(\mathcal{E}) \leq Eq.2.11$, which completes the proof. □

Proof of **Theorem** 3

Proof. The Lemma 2 shows that

$$P(\mathcal{E}_i) \leq 4\exp(-\frac{(\epsilon_i-\delta)^2 d}{50}) \leq 4\exp(-\frac{(\epsilon-\delta)^2 d}{50}) \tag{2.19}$$

By an union bound on the error events $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_{M_p}\}$,

$$P(\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{M_p}) \leq \sum_{i=1}^{M_p} 4\exp(-\frac{(\epsilon_i - \delta)^2 d}{50}) \tag{2.20}$$

$$= 4M_p \exp(-\frac{(\epsilon - \delta)^2 d}{50}) \tag{2.21}$$

□

When $(\epsilon - \delta)^2 \geq 10\sqrt{\frac{\log M_p}{d}}$, we have $\exp(-\frac{(\epsilon-\delta)^2 d}{50}) \leq \frac{1}{4{M_p}^2}$ and therefore $P(\mathcal{E}_1 \cup \ldots \cup \mathcal{E}_{M_p}) \leq \frac{1}{M_p}$.

Caveat: The performance bound analysis adopts a strong assumption that the neural embeddings are random matrices. This could be very different in real application because the random matrices do not encode any semantic information. We acknowledge this limitation and provide more references on that. We rely on the mathematical tool based on random matrix theory, namely the *Johnson-Lindenstrauss* (JL) lemma. This tool was also adopted by Luan et al. [99] under information retrieval setting, which provides the connection between dense and sparse retrievers. The bound is on its loose end because embeddings from BERT are more meaningful than random matrices (also verified from their empirical study). In our work, we study use the JL lemma to connect sparse and dense classifiers. The bound is reasonable considering that it is on its loose end, but, still, there is no guarantee when applied with real BERT embeddings.

2.5 Experiments

Dataset	N_{train}	N_{test}	$\bar{L}_d$	L	$\lvert\mathbb{L}_{\text{tail}}\rvert$
EURLex-4K	15,539	3,809	5.30	3,956	2,413
AmazonCat-13K	1,186,239	306,782	5.04	13,330	3,936
Wiki10-31K	14,146	6,616	18.64	30,938	26,545
Wiki-500K	1,779,881	769,421	4.75	501,070	338,719

Table 2.1: Corpus Statistics: N_{train} and N_{test} are the number of training and testing instances respectively; $\bar{L}_d$ is the average number of labels per document, and L is the number of unique labels. $\lvert\mathbb{L}_{\text{tail}}\rvert$ is the number of tail labels with $1 \sim 9$ positive training instances.

2.5.1 Datasets

We conduct our experiments on 4 benchmark datasets: EURLex-4K, AmazonCat-13K, Wiki10-31K and Wiki-500K. The statistics of the datasets are shown in Table 2.1. An unstemmed version of EURLex-4K is obtained from the APLC-XLNet github[1] and the rest are from the Extreme classification Repository[2].

For comparative evaluation of methods in tail label prediction, we consider the subset of labels with $1 \sim 9$ positive training instances. Those tail-label subsets correspond to 63.48%, 29.53%, 88.65% and 67.60% of the total labels in the 4 datasets respectively. With mostly more than half of the labels as tail labels, the distributions are indeed highly skewed.

2.5.2 Tail Label Evaluation Metrics

Micro-averaged PSP@k: The PSP [59] metric re-weights the score of each instance according to the label frequency:

$$PSP@k = \frac{1}{k} \sum_{l=1}^{k} \frac{\mathbb{1}_{\mathbf{y}}(\mathbf{p}_l)}{\text{prop}(\mathbf{p}_l)}$$

where the propensity score $\text{prop}(\mathbf{p}_l)$ in the denominator gives higher weights to tail labels.

Since the micro-averaged metric gives an equal weight to the per-instance scores, it can still be dominated by the system's performance on the head labels but not the tail labels. As an alternative, we adopt a macro-averaged metric to evaluate tail label performance.

Macro-averaged F1@k: The macro-averaged metric [150] gives an equal weight to all the labels (we apply it to tail labels specifically). It is defined as the average of the label-specific $F1@k$ values, calculated based on a contingency table for each label, as shown in table 2.2. The precision, recall and $F1$ for a predicted ranked list of length k are computed as $\text{P} = \frac{\text{TP}}{\text{TP+FP}}$, $\text{R} = \frac{\text{TP}}{\text{TP+FN}}$, and $\text{F1} = 2\frac{P \cdot R}{P+R}$.

Table 2.2: Contingency table for label l.

	l is true label	l is not true label
l predicted	True Positive (TP_l)	False Positive (FP_l)
l not predicted	False Negative (FN_l)	True Negative (TN_l)

[1] https://github.com/huiyegit/APLC_XLNet.git
[2] http://manikvarma.org/downloads/XC/XMLRepository.html

For micro-averaged PSP@k, we choose $k = 1, 3, 5$ as in previous works. For macro-averaged F1@k, we choose $k = 19$ for Wiki10-31K because it has an average of 18.64 labels and $k = 5$ for the rest datasets.

2.5.3 Baselines

For the tail label evaluation, our method is compared with the SOTA deep learning models including X-Transformer [16], XLNet-APLC [153], LightXML [64], and AttentionXML [157]. X-Transformer, LightXML, and XLNet-APLC employ pre-trained Transformers for document representation. We reproduced the results of single model (given in their implementation) predictions with BERT as the base model for LightXML, BERT-large for X-Transformer, XLNet for XLNet-APLC, and LSTM for AttentionXML. The AttentionXML utilizes label-word attention to generate label-aware document embeddings, while the other models generate fixed document embedding.

We use the SVM model with tf-idf feature as our choice of sparse classifier and BERT-base as our dense model for neural retrieval and classification.

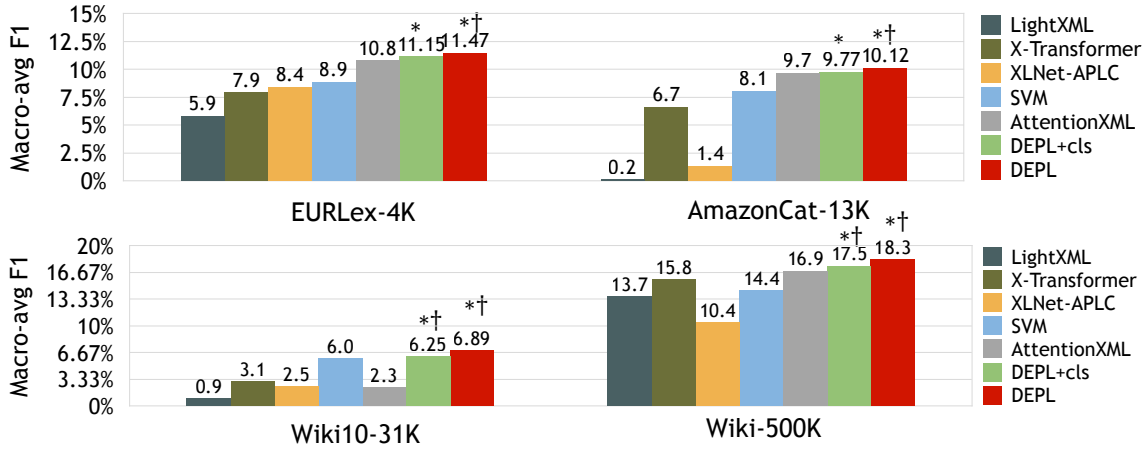


Figure 2.3: Tail-label prediction results in $F1@k$ on the labels with $1 \sim 9$ positive training instances, with $k = 19$ for the Wiki10-31K dataset and $k = 5$ for the rest. $*$ and † indicates the macro t-test is significant ($p < 0.05$) over SVM and previous best neural model respectively.

2.5.4 Results in Tail Label Prediction

SVM on Tail Label Prediction The results evaluated with the F1 metric averaged on the tail labels are shown in figure 2.3. Surprisingly, a simple statistical SVM baseline achieves com-

Table 2.3: Tail label prediction results of methods in $PSP@k$, with $*$ indicating significant improvement ($p < 0.05$) over the previous best model on the micro sign test.

	EURLex-4K			Wiki10-31K			AmazonCat-13K			Wiki-500K		
Methods	PSP@1	PSP@3	PSP@5	PSP@1	PSP@3	PSP@5	PSP@1	PSP@3	PSP@5	PSP@1	PSP@3	PSP@5
X-Transformer	37.85	47.05	51.81	13.52	14.62	15.63	51.42	66.14	75.57	31.20	36.78	40.21
XLNet-APLC	42.21	49.83	52.88	14.43	15.38	16.47	52.55	65.11	71.36	29.73	30.26	30.59
LightXML	40.54	47.56	50.50	14.09	14.87	15.52	50.70	63.14	70.13	31.01	37.10	39.28
AttentionXML	44.20	50.85	53.87	14.49	15.65	16.54	53.94	68.48	76.43	30.05	37.31	41.74
SVM	39.18	48.31	53.37	11.84	14.00	15.81	51.83	65.41	72.82	32.12	32.75	35.20
DEPL	**45.60***	52.28*	53.52	**17.20***	**16.90***	**16.95**	**55.94***	**70.01***	**76.87***	32.07	**40.60***	**43.74***
DEPL+cls	44.60	**52.74***	**54.64**	16.73*	16.84*	16.67	55.21*	69.73*	75.94	**32.18**	39.89*	41.46

petitive results on the tail label predictions. We observe that SVM model can outperform most of the pretrained Transformer-based models on the tail label prediction, and outperform the AttentionXML on the Wiki10-31K dataset. This provides an empirical evidence for the robust performance of a sparse model on tail label prediction. As we analyzed in section 2.4, the SVM model utilizes the unsupervised statistical feature as document representation, which potentially suffers less from the data scarcity issue. The empirical result serves as an evidence for our theoretical analysis that the joint optimization of feature extractor and label embedding is difficult when data is limited.

Table 2.4: Examples of SVM generated keywords from Wiki10-31K. The classifier is trained with only 1 positive training instance per label. The top 20 keywords are shown. with meaningful words highlighted in red manually.

Label Text	**#training instance**	**Top Keywords**
phase4	1	trials clinical protection personal directive processed data trial drug phase eu processing patients sponsor controller legislation regulation art investigator study
ensemble	1	boosting kurtz ferrell weak algorithms learners misclassified learner kearns ensemble charges bioterrorism indictment doj indict cae correlated 2004 reweighted boost
kakuro	1	nikoli kakuro puzzles crossword clues entries entry values sums cells cross digits dell solvers racehorse guineas aa3aa digit clue kaji

Neural Classifier on Tail Label Prediction The neural classifiers include LightXML, X-Transformer, XLNet-APLC and AttentionXML. Specifically, the AttentionXML model leverages a label-word attention to calculate a label specific document representation. As we observe in

figure 2.3, among the baseline models, the AttentionXML performs the best on the tail label predictions, beating the other baselines on 3 out of the 4 benchmark datasets. The superior performance could come from the local word and label matching which benefits the tail label prediction.

As mentioned in section 2.3, X-Transformer model ensembles a neural classifier and a SVM model by directly summing the prediction scores. Although X-Transformer outperforms SVM on the overall label prediction, it underperforms SVM on 3 out of 4 benchmark datasets. This shows that model performance on tail label is dragged down by the neural model prediction, and a simple ensemble does fully exploit the advantage of the sparse model. Compared with the X-Transformer, our model achieves better performance on both macro-F1 and micro-PSP metrics, showing the advantage of leveraging the retrieval of augmented label descriptions rather than a pure ensemble.

DEPL Performance On the 3 smaller scale benchmark datasets, EURLex-4K, AmazonCat-13K and Wiki10-31K, our model directly ranks all the labels. On the large Wiki-500K dataset, our model leverages the prediction of cluster-based algorithm in X-Transformer and replaces the reranker with our retrieval model.

Our proposed models perform the best on the Macro-F1 metric with the DEPL model consistently and significantly showing the best performance on all the benchmark datasets. A macro t-test [150] is conducted to justify the significance of improvement over the SVM and previous best neural model. The significant performance gains over the SVM model shows that our retrieval framework can outperform the sparse model which serves as label keywords extractor. We attribute the success of model on tail label prediction to the retrieval module that focuses on the semantic matching between the document and label text. The DEPL performs better than the DEPL +cls as it is less affected by the large amount of training instances for head labels and thus more biased on the tail label prediction.

According to the evaluation with the PSP metric shown in table 2.3, it also confirms that our proposed models DEPL and DEPL +cls improves over the previous SOTA neural models on all the benchmark datasets, with $*$ indicates significant improvement ($p < 0.05$) over the previous best model on the micro sign test [150]. The Wiki10-31K dataset has the most skewed distribution as the most frequent label covers more than 85% of the training instances, resulting in a low PSP score. Since DEPL relies on the semantic matching between the document and label text, it is less affected by the dominating training pairs, and thus the PSP@1, PSP@3 beats the SOTA models by a larger margin. The DEPL +cls achieves worse performance on this

dataset, because the classification counterpart of the model would benefit more on the head label predictions and tend to rank the head labels at the top.

Metric Comparison Although the PSP metric gives higher weight to the tail labels, it is a micro-averaged metric over the scores of each instance, which can still be affected by the performance on the more common categories that cover most of the instances. For example, SVM model doesn't stand out under the PSP metric, which has lower overall label performance. Since the F1 metric is calculated specifically on the set of tail labels, we argue that it provides a more accurate and fine-grained evaluation on tail label prediction, which better reveals the success of XMTC models on predicting rare categories.

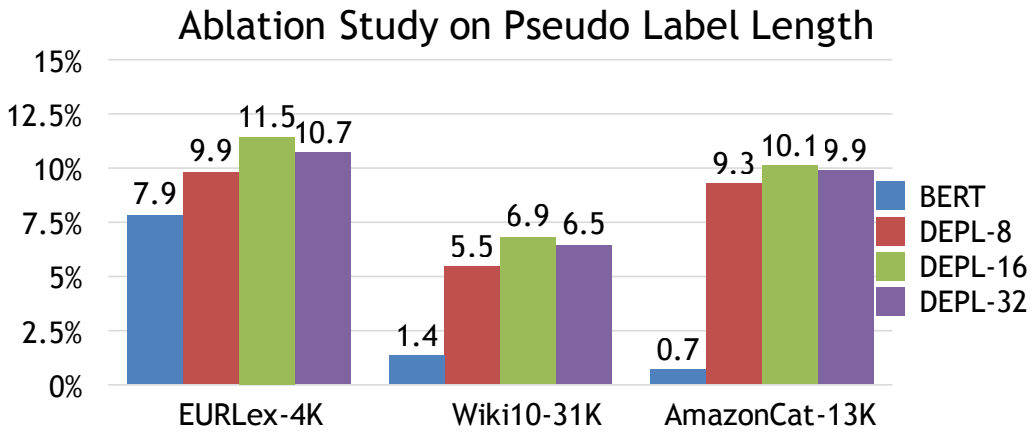


Figure 2.4: The ablation-test results of DEPL in Macro-averaged F1@k metric with varying length of pseudo label descriptions.

2.5.5 Ablation on Generated Pseudo Label

Table 2.4 shows examples of the SVM generated keywords trained on the Wiki10-31K dataset for labels with only 1 training example. We manually highlight the meaningful terms related to the label meaning. For example, the label name *phase4* is ambiguous, whose meaning needs to be inferred from the corresponding document. From the keywords *trial, clinical, drug, etc*, we deduce that the topic is about medical testing phase. In another example, *kakuro* is a Japanese logic puzzle known as a mathematical crossword and the game play involves in adding number in the cells. Generating a description for *kakuro* requires the background knowledge, but the keywords automatically learned from the sparse classifier provide the key concepts. Although

not all the keywords can provide rich semantics to complement the original label name, they may serve as a context for the label to make it more distinguishable from others.

In figure 2.4, we conduct an ablation test on the length of the pseudo label and the performance is measured by Macro-avg F1@k. The BERT classifier is included as a baseline with no label text information. As we observe that the longer description of length 16 performs the better, but when length is 32, the performance doesn't increase as the text may become noisy with more unrelated keywords.

The DE-ret model is a pure retrieval baseline (avg length 3) with only the label name. While it achieves good performance on the EURLex-4K and AmazonCat-13K datasets, it still performs poorly on the Wiki10-31K dataset. This shows that generating the keywords from the sparse classifier can enhance the text quality. Furthermore, the generated text allows DEPL to use the semantic information of the label keywords, which is ignored in the SVM model. This could be another reason why our model performs better than the SVM baseline on the Wiki10-31K dataset.

2.6 Conclusion

In this chapter, we propose a neural retrieval framework (DEPL) that addresses the challenge of tail-label prediction in XMTC by leveraging model-generated training signals. Instead of relying solely on manually annotated text-label pairs, DEPL formulates the problem as a semantic matching task between input documents and system-enhanced label descriptions. By integrating neural embedding-based retrieval with a large-margin bag-of-words (BoW) classifier for generating informative pseudo label descriptions, our approach effectively mitigates data scarcity issues. Extensive experiments on large-scale benchmark datasets demonstrate that DEPL significantly outperforms strong baselines, particularly in improving tail-label predictions, highlighting the effectiveness of model-driven supervision in extreme classification settings.

Chapter 3

Synthetic Training Data for Zero-shot Classification

In this chapter, we address zero-shot text classification by leveraging LLM-generated training signals to replace reliance on labeled data. Since no document-label pairs are available in this setting, traditional methods cannot exploit their co-occurrence. To overcome this, we introduce GenCo, which utilizes an LLM to generate synthetic data that enhances a smaller classifier's training. GenCo leverages LLMs to generate two key signals: (1) synthetic document content for enriched representation and (2) synthetic document-label pairs as new training examples to mitigate the lack of labeled data. This approach not only reduces reliance on human annotations but also lessens dependence on large unlabeled corpora while improving model robustness. Experiments show that GenCo outperforms state-of-the-art methods with minimal (<5%) in-domain data and even surpasses Alpaca-7B guided by human prompts, demonstrating the power of LLM-generated signals for self-training in zero-shot classification.

Highlights We are among the first to leverage instruction-following LLMs to generate training pairs for zero-shot text classification, enabling a smaller model to surpass the performance of its teacher LLM, Alpaca-7B.

Model-generated Signals We utilized instruction-following LLM, Alpaca-7B, to generate synthetic training data for zeroshot text classification, including synthetic document content and new document-label training pairs.

3.1 Introduction

Zero-shot text classification poses a challenge in predicting class labels for text instances without requiring labeled instances for supervised training. Recent research in zero-shot text classification primarily falls into two distinct groups. The first approach applies LLM (with billions of parameters) in label prediction with the help of human instructions or prompts [25, 111]. However, even a relatively smaller LLM such as Alpaca-7B [130] necessitate considerable computational power and time for large-scale inference and model fine-tuning. Without domain-specific fine-tuning, LLMs struggle to discern between classes characterized by unclear decision boundaries. The second approach to zero-shot classification involves the self-training of smaller language models, often comparable in size to BERT [46, 104, 119, 138]. In these methods, the models predict "pseudo labels" for unlabeled instances, and then use these instances alongside their assigned pseudo labels as supervised data for model fine-tuning. This process is iterated for the model to incrementally adapt to the target domain. However, these techniques hinge on accessing a substantial volume of unlabeled texts from the intended domain, sometimes reaching the magnitude of millions as indicated in table 3.1, a volume that may not always be feasible in many practical contexts. Furthermore, due to the capacity limitation of small language models, the pseudo label predictions are prone to error potentially jeopardizing the efficacy of the self-training loops.

In this chapter, we introduce a novel approach called **Gen**eration-driven **Co**ntrastive Self-Training (GenCo). This approach adeptly combines the language understanding ability of LLMs with the adaptability and efficiency of smaller models. Drawing inspiration from PESCO [138], we treat zero-shot classification as a sentence alignment task and employ contrastive self-training with smaller models. We provide a theoretical analysis of how self-training can bolster classification generalization. Crucially, we sidestep the dependency on extensive unlabeled texts by capitalizing on the generative strengths of LLMs.

Our approach exploits the LLM generation power in two ways. Firstly, to enhance pseudo label prediction, we employ an LLM to generate multiple variations or extensions of an input text. This augmentation strategy enriches the available information for the classifier, enabling it to make better predictions based on a more comprehensive understanding of the input. Secondly, we employ the LLM to craft new training instances conditioned on the pseudo labels, ensuring the generated content is closely aligned with its assigned pseudo label. This tackles the prevalent issue of mislabeling in self-training. In summary, this chapter makes three key contributions:

- We propose a novel approach that enables smaller models to acquire knowledge from LLMs within the self-training loop. Our method is compatible with any new LLMs to effectively train better classifier on target domains. In our experiments, our small model outperforms Alpaca with human instructions.
- We explore the more challenging setting of zero-shot classification where only a limited number of unlabeled texts are available. In this setting, we improve the performance over strong baselines.
- We provide theoretical proof to support the effectiveness of the proposed contrastive loss for self-training.

3.2 Related Work

Knowledge Distillation from GPT: To leverage the language modeling power of large model, previous works distills LLM [26, 55], generate text and label pairs [105, 154, 156] to train a classifier for downstream tasks. However, generating training data from scratch can lead to low-quality data with unrelated or ambiguous examples analyzed in [40]. Our generation is grounded in the context of the corpus with enrichment in semantic and diversity, providing a practical alternative to generation-based methods for zero-shot text classification and knowledge distillation.

Zeroshot Text Classification: Zeroshot text classification predicts class labels without labeled instances [27, 39] and can be formulated as sentence alignment [43, 52, 123, 138, 168] between document and labels. Sentence encoders are typically trained with contrastive learning, which optimizes representations by pulling inputs with similar semantics closer in the embedding space and pushing inputs with different semantics further apart. Our model applies LLM to generate training pairs for contrastive learning to train robust classification with limited instances available.

3.3 Preliminary: Zero-shot Text Classification as Sentence Alignment

Given a set of N unlabeled documents $X = \{x_1, x_2, \cdots, x_N\}$ and a set of L category descriptions $C = \{c_1, c_2, \cdots, c_L\}$, the goal is to learn a scoring function $g(x, c_i)$ that takes document

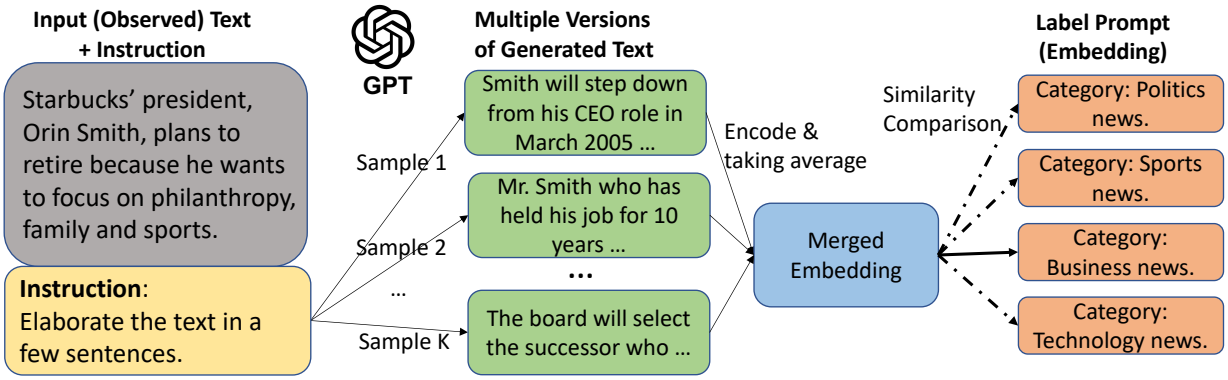


Figure 3.1: Enriching textual semantics through LLM Generation: The input text and an instruction are fed into the LLM to generate multiple pieces of elaborated texts, each of which is concatenated to the original input to obtain an augmented text. The embeddings of the augmented texts are then averaged to obtain a merged embedding, which is used for label prediction and contrastive loss in the self-training process.

x and label description c_i as input and produces a similarity score as the measure of how well the document and the label match to each other.

In the zero-shot setting, text classification can be formulated as a sentence alignment problem [138], where both the input sentence and the label descriptions are encoded using a pre-trained sentence encoder like SimCSE [43]. The similarity scores between the sentence and label embeddings are used to predict related labels. The performance can be further improved by converting a short label description into a full sentence via prompts [52, 138]. For example, the label "sports" can be converted to "This is an article about sports." Subsequently, we represent the label prompt for a label description c_i as p_i. The scoring function can be implemented as follows:

$$g(x, c_i) = \text{sim}\left(f_\theta(x), f_\theta(p_i)\right) \tag{3.1}$$

where $f_\theta(\cdot)$ is the sentence encoder parameterized by θ and $\text{sim}(\cdot, \cdot)$ is a similarity function such as dot product or cosine similarity.

Given an input text at inference time, the predicted label is the one with the highest similarity score:

$$\hat{y} = \underset{j}{\text{argmax}}\, g\left(x, c_j\right) \tag{3.2}$$

3.4 Our Method: GenCo

GenCo is a self-training framework [104, 120, 138] that harnesses the generative power of LLMs to train a smaller pre-trained sentence encoder in an iterative manner. Each self-training step consists of two parts. First, we apply equation 3.2 to predict pseudo labels for unlabeled instances. Second, we fine-tune model on pseudo-labeled data with a proposed contrastive self-training objective. In section 3.4.2 and 3.4.3, we will introduce two types of augmentation with LLM to enhance the self-training process.

3.4.1 Contrastive Self-Training Objective

One well-known challenge of self-training is its tendency to exhibit overconfidence in certain labels due to the model inductive bias [142]. Extensive research has shown that soft labeling [104, 142], label smoothing [108], and entropy regularization [49] can effectively tackle this issue. Motivated by these, we propose to incorporate soft-labeling and entropy regularization into a contrastive loss.

Given an input text x, the distribution of the predicted label space is:

$$P(\hat{y}_i|x;\theta) = \frac{\exp(\text{sim}(f_\theta(x), f_\theta(p_i)))}{\sum_{c\in C}\exp(\text{sim}(f_\theta(x), f_\theta(p_c)))} \tag{3.3}$$

Here, $\hat{y}_i$ is the predicted label and p_i is a label prompt for the predicted label. To prevent the model from being overconfident, we define the weights of the labels as:

$$Q(\hat{y}_i|x;\theta) = \frac{\exp(\text{sim}(f_\theta(x), f_\theta(p_i))/\tau)}{\sum_{c\in C}\exp(\text{sim}(f_\theta(x), f_\theta(p_c))/\tau)} \tag{3.4}$$

, where $\tau \leq 1$ is the temperature. A lower temperature implies a sharper distribution and thus greater weights in the predicted label. We drop the notation of θ for convenience.

Combining the above $P(\hat{y}_i|x)$ and $Q(\hat{y}_i|x)$, we propose a text to label ($t2l$) contrastive loss:

$$\mathcal{L}_{t2l} = -\sum_{i=1}^{N}\sum_{j=1}^{L} Q(\hat{y}_j|x_i)\log P(\hat{y}_j|x_i) \tag{3.5}$$

When $\tau \to 0$, $Q(\hat{y}|x)$ becomes categorical distribution and the loss reduces to a supervised contrastive learning loss [72] with pseudo label $\hat{y}$ as the target:

$$\mathcal{L}_{t2l}^{\tau\to 0} = -\sum_{i=1}^{N}\log P(\hat{y}|x_i) \tag{3.6}$$

It encourages the model to predict label $\hat{y}$ given x with more confident. On the other hand, when $\tau = 1$, the loss reduces to a minimization of conditional entropy function H:

$$\mathcal{L}_{t2l}^{\tau=1} = H\left(C \mid X\right) \tag{3.7}$$

$$= -\sum_{i=1}^{N}\sum_{j=1}^{L} P(\hat{y}_j|x_i)\log P(\hat{y}_j|x_i) \tag{3.8}$$

We show a theorem such that minimizing the loss function equation 3.5 can achieve similar effects Entropy Regularization [49, 50], which is a means to enforce the cluster assumption such that the decision boundary should lie in low-density regions to improve generalization performance [17].

Theorem 6. *Consider a binary classification problem with linearly separable labeled examples. When* $0 < \tau < 1$*, optimizing equation 3.5 with gradient descend will enforce the larger margin between classes and achieves max margin classifier under certain constraint.*

We place our formal theorems and proofs in Appendix 3.6. Theorem 7 suggests that self-training is an in-domain fine-tuning that maximizes class separation, which serves as an explanation of why training on pseudo labels can enhance performance even if no extra labeling information is provided. In our experiment, we show that self-training of a smaller model can outperform LLM (Alpaca-7B) prediction, justifying the claim empirically. We set $\tau = 0.1$ (refer to Appendix 3.5.6) to balance supervised classification and low density separation between classes.

While self-training can potentially improve model generalization, the limitations are obvious: 1) pseudo labels are prone to error and may negatively affect model training. 2) self-learning requires a significant load of unlabeled data, which may not always be available. Next, we introduce generation-driven approaches to improve self-training with LLM, such as an instruction-tuned GPT (Alpaca-7B).

3.4.2 Semantic Enrichment using LLM

In this section, we propose a way to enrich the semantic information of an input text with multiple LLM-generated pieces of text. When the input text is relatively short, such as consisting of only one or a few sentences, the information may not be sufficient for alignment-based method to match relevant labels.

A remedy is to query an LLM to elaborate the input and generate multiple pieces of extended texts. As shown in figure 3.1, the instruction, "Elaborate the text with a few sentences," steers

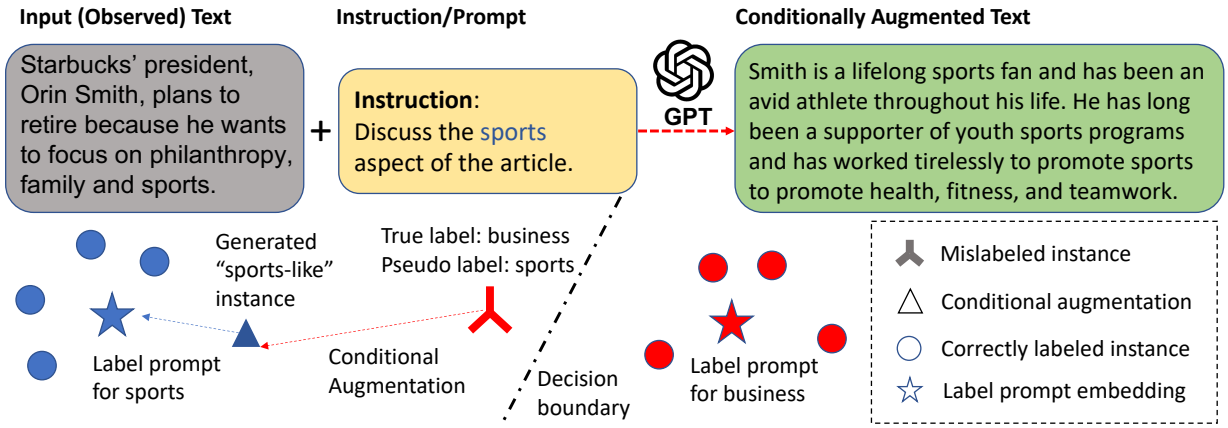


Figure 3.2: Conditional text augmentation to address mislabeling in self-training: When a pseudo label is incorrect, it can mislead the training process and decrease classification performance. We generate augmented text conditioned on the pseudo label, aiming to make the generated text closer to the majority members in the category of the pseudo label. This approach aims to improve the quality of the generated instances for self-training.

the LLM towards creating relevant expansions and continuations for the input text x. These augmented texts, denoted as x^{aug}, serve for two purposes: 1) improving the quality of pseudo label, and 2) forming the positive pair in contrastive learning, as detailed below:

Enhancing pseudo label quality. We enhance pseudo label prediction by enriching the input embedding of equation 3.2 by:

$$\frac{1}{K}\sum_{i=1}^{K} f_\theta(x \oplus x_i^{\text{aug}}), \tag{3.9}$$

where $\oplus$ is the concatenation operator for text and x_i^{aug} is the i-th sample from $P_g(\cdot|x)$. The mean of the embeddings summarize the information induced by LLM.

Constructing positive training pairs. We propose a contrastive loss between input text and generated text as another training objective. Let I be a training batch and $A(i)$ be the set of augmented texts with the same pseudo-label as input x_i. Our objective encourages proximity between x and x^{aug} (sampled from $A(i)$) in the embedding space:

$$\mathcal{L}_{t2g} = \sum_{i\in I} \frac{-1}{|A(i)|} \sum_{x^{\text{aug}}\in A(i)} \log \frac{\exp(sim(f_\theta(x_i), f_\theta(x^{\text{aug}})))}{\sum_{j\in I}\exp(sim(f_\theta(x_i), f_\theta(x_j)))}. \tag{3.10}$$

Algorithm 1: Self-training with GPT assisted in the loop

Require: Unlabeled texts X, label descriptions C, instruction-tuned GPT model $g(\cdot)$.
Initialization: Classifier $f_\theta(\cdot)$ initialized with pre-trained sentence encoder. Empty dictionary $\mathrm{GenDict}$ to cache conditional generated text.
Input augmentation: For each observed text, generate K samples of augmented text from $P_g(\cdot|x)$.
for $t : 1 \rightarrow T$ *self-training iterations* **do**
    Use $f_\theta(\cdot)$ to generate pseudo-labels $\hat{y}$ (eq.3.2) and soft-target Q (eq.3.4) for texts with input augmentation in Section.3.4.2. Sample a balanced subset of pseudo-labeled training pairs of size S_t according to prediction confidence;
    for *each training sample* $(x, \hat{y})$ **do**
        if *key* $(x, \hat{y}) \in \mathrm{GenDict}$ **then**
            Fetch generated texts from $\mathrm{GenDict}$ ▷ Use cached generated text;
        else
            Generate M samples from $P_g(\cdot|x, \hat{y})$ ▷ Conditional augmentation in Section 3.4.3;
            Add generated texts to $\mathrm{GenDict}$ ▷ Cached generated text;
    Use sampled training pairs and the conditionally generated text to update the parameters θ of $f_\theta(\cdot)$ with the objective function $\mathcal{L} = \mathcal{L}_{g2l} + \mathcal{L}_{t2g}$ from equation 3.10 and 3.11.

3.4.3 Crafting Training Pairs with LLM

Self-training can introduce bias into a classifier due to mislabeling instances. To address this issue, we propose to generate high quality pseudo-labeled data pairs, as shown in figure 3.2. Consider an instance where an article about the retirement of Starbucks' president, whose true label is "business", is mistakenly labeled as "sports". Training the model with this incorrect label blurs the distinction between the business and sports categories.

To mitigate this issue, we employ the LLM to conditionally augment the input text based on the sports category. This is achieved by framing instructions like, "Discuss the sports aspects of the article". Consequently, the produced text mirrors typical articles within the sports category. By optimizing this newly generated text, instead of the original mislabeled instance, we correct

its placement relative to the decision boundary separating "sports" and "business". Essentially, by creating texts based on pseudo labels, we synthesize training pairs that enhance the separation of class labels in the embedding space, thereby addressing the challenges of mislabeling inherent to self-training.

Let x^{cond} be the conditionally augmented text, the modified equation 3.5 is:

$$\mathcal{L}_{g2l} = -\sum_{i=1}^{N}\sum_{j=1}^{L} Q(\hat{y}_j|x_i^{\text{cond}}) \log P(\hat{y}_j|x_i^{\text{cond}}) \tag{3.11}$$

3.4.4 Algorithm for Self-training

We apply self-training with equation 3.10 and 3.11 in an iterative way as shown in Algorithm 1 with LLM assisting in the loop. During training, we found that a balanced sampling that keeps the same number (S_t for iteration t) of training for each category is important for the stability of self-training. Additionally, we use a dictionary $\mathrm{GenDict}$ to cache the conditional generated text to avoid repeated generation for better efficiency.

3.5 Experiments

3.5.1 Datasets and Experimental Settings

Dataset	Classification Type	#Classes	#Train	#Test	Avg Length
AG News	News Topic	4	120,000	7,600	38
DBPedia	Wikipedia Topic	14	560,000	70,000	50
Yahoo Answers	Question Answering	10	1,400,000	60,000	70
Amazon	Product Review Sentiment	2	3,600,000	400,000	78

Table 3.1: Statistics of datasets for multi-class text classification.

We conduct experiments on 4 benchmark text classification datasets: AG News, DBpedia, Yahoo Answers and Amazon, with the statistics shown in table 3.1. In the experiments, we initialize our sentence encoder with supervised SimCSE Roberta-base model (110M parameters) [43]. For the generative model, we use the Alpaca-7B [130] as our choice of LLM, which is a GPT model fine-tuned with human instructions [133]. The label prompts and the instruction template are illustrated in table 3.3 in Appendix. Please refer to section 3.5.4 in Appendix for implementation details.

ID	Self-train	Methods	AG News	DBpedia	Yahoo Answers	Amazon
1	–	Supervised	94.2	99.3	77.3	97.1
2	No	SimCSE (Sentence-enc)	74.5	73.8	55.6	88.8
3	No	Alpaca-7B (LLM)	77.4	60.6	52.1	86.6
4	Yes	iPET	86.0	85.2	68.2	95.2
5	Yes	LOTClass	86.4	91.1	–	91.6
6	–	Supervised-downsample*	93.8	98.7	76.5	97.0
7	Yes	PESCO*	85.0	96.6	65.8	92.4
8	Yes	GenCo *	**89.2**	**98.3**	**68.7**	**95.4**
9	Yes	GenCo * - CA	87.5	97.6	65.1	94.3
10	Yes	GenCo * - IA	86.2	97.1	63.5	93.6
11	Yes	SimCSE + Self-training (Eq 3.5)	83.2	94.3	62.7	91.5

Table 3.2: Comparison of classification methods on benchmark datasets. The test accuracy of best performing zero-shot method is highlighted in bold phase. Row 7-11 (with *) use a down-sampled dataset with 4k (3.4%), 11.2k (2%), 15k (<1%), 20k (<1%) unlabeled training instances respectively. Rows 9-11 are ablation tests with input augmentation (IA) or conditional augmentation (CA) removed.

3.5.2 Baseline Methods

Alpaca-7B is a LLM baseline for zero-shot classification. We solicit the LLM for zero-shot classification with the instruction "Classify the text by outputting a single category from [label categories]".

iPET [119] formulates zero-shot text classification as a cloze test, where a pre-trained BERT [34] model is used to predict the output label(s) by completing a prompt such as "This article is about _", which is concatenated right after an input document. An iterative self-training algorithm is used in iPET to improve the model for better generalization.

LOTClass [104] applies the BERT model to extract keywords related to the label names from unlabeled texts and then create pseudo labels based on the extracted keywords. LOTClass also applies a self-training algorithm to further improve the classification performance.

PESCO [138] formulates zero-shot classification as sentence alignment and uses contrastive self-training to improve the model performance. As an augmentation, it selects salient sentences from documents to create additional positive training pairs.

3.5.3 Experimental Results

In table 3.2, we present a comparison of the test accuracy of our model with other baselines on four benchmark classification datasets. Specifically, rows 1-5 are experiments using the entire (unlabeled) training set and rows 6-11 use a down-sampled dataset with 4k (3.4%), 11.2k (2%), 15k (<1%), 20k (<1%) unlabeled training instances from the original datasets respectively.

Comparison with Alpaca-7B: While Alpaca-7B (row 3) has demonstrated strong instruction following ability to solve problems without any training, it exhibits lower performance compared to GenCo (row 8) and other self-training methods on classification task. The reason could be attributed to the domain adaptation effect of self-training. Classification tasks involve comparing instances, such as an article being more likely to belong to the "sports" category when compared to articles in the "business" category. In our analysis in section 3.4.1, self-training enforces the separation between classes to improve the generalization ability. This can be further supported when the number of classes increases in DBpedia and Yahoo Answers dataset, the performance of Alpaca gets worse. Furthermore, Alpaca-7B takes 9 minutes per 10k instances on one A6000 gpu while GenCo takes 10 seconds, which is roughly x50 speed up.

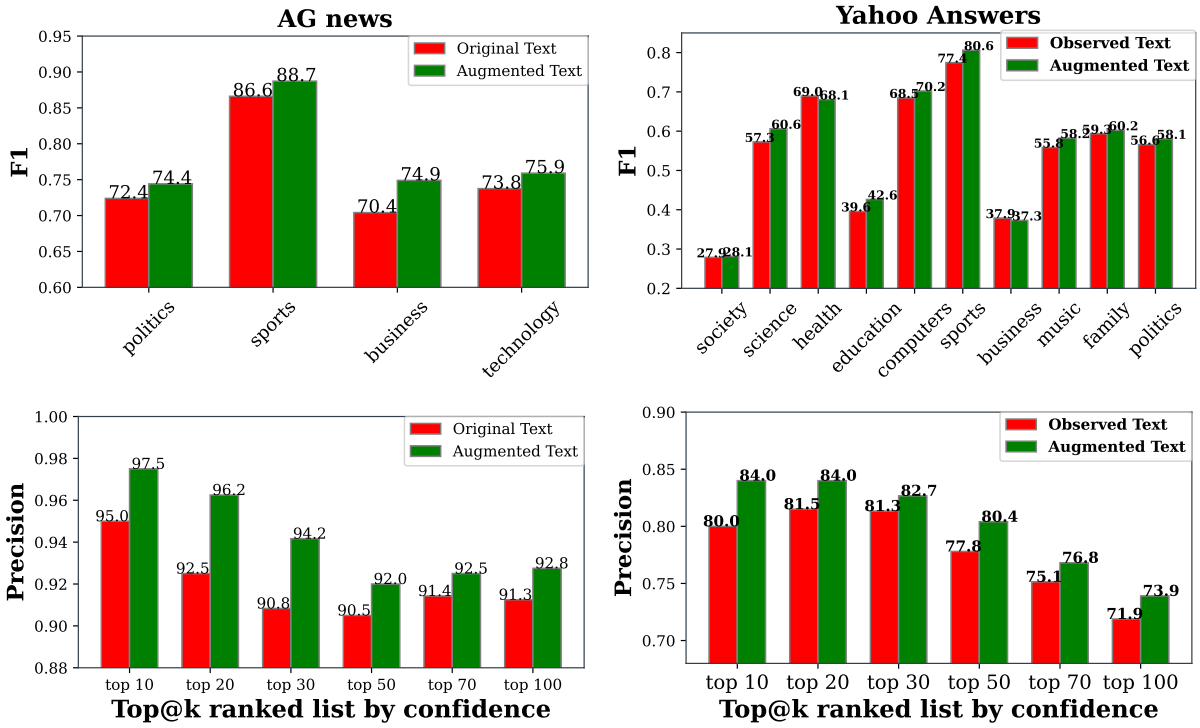


Figure 3.3: Per class F1 (upper) and ranking-based precision (lower) for classification performance with input augmentation.

Comparison with SOTA Methods: Both iPET (row 4) and LOTClass (row 5) use self-training algorithm for zero-shot classification, but GenCo outperforms the previous self-training methods even with significantly fewer instances ($< 5\%$ of original size). The iPET model improves pseudo label prediction with an ensembling about 15 models to reduce prediction variance. In comparison, our approach improves pseudo label prediction by ensembling augmented text embedding during self-training, leading to improved performance and a more memory efficient alternative. While LOTClass uses a BERT model to extract keywords for each category as an augmentation, it is less expressive than using an LLM to generate coherent human language as augmentation. PESCO (row 7) is the most recent SOTA with contrastive self-training and introduced an augmentation technique by learning on salient sentences. However, the method still requires a large amount of data to be effective. In scenarios where only a limited number of unlabeled texts are available, PESCO still underperforms our model.

Effectiveness of Contrastive Self-training: Row 2 represents the sentence encoder baseline with SimCSE, whereas row 11 represents SimCSE + contrastive self-training algorithm as per equation 3.5. The result shows that incorporating contrastive self-training leads to significant gains. Compare row 3 (Alpaca-7B) with row 11. Despite being a larger model in scale, Alpaca-7B still outperforms the self-training approach across all benchmark datasets, underscoring the effectiveness of class separation with self-training for classification task.

3.5.4 Analysis of LLM Augmentation

In this section, we denote the input augmentation in section 3.4.2 as IA and the conditional augmentation based on pseudo label in section 3.4.3 as CA. Rows 9 and 10 in table 3.2 shows ablation tests with CA and IA removed. Overall, our LLM data augmentation, with and without conditioning on pseudo label, both lead to improved performance, due to their ability to provide more accuracy pseudo label and high quality synthetic training pairs.

Effectiveness of IA: In this evaluation, we investigate the effectiveness of input augmentation for first round pseudo-labeling *without training*. We evaluate the performance of our model on two datasets, namely AG News and Yahoo Answers, using two evaluation metrics: per class F1 metric and ranking-based precision metric according to prediction confidence. The per class F1 metric provides an insight into how well the model performs on each individual class by balancing precision and recall. In the upper part of figure 3.3, our findings indicate that LLM augmented data leads to improved performance across all categories for AG News and in eight out of ten classes for Yahoo Answers.

In the lower part of figure 3.3, we employ a ranking-based precision metric to assess the quality of the most confident cases. Our results demonstrate that using augmented data yields better precision for the most confident cases. Notably, our study on the Yahoo Answers dataset indicates that the predictions are better calibrated with the use of augmented data, implying that highly confident samples exhibit better precision. Conversely, such a trend was not observed in unaugmented data, where the top 30 had higher accuracy than the top 10. Better calibration justifies the sampling from the most confident pools for self-training, making it a more reliable method for improving model performance.

Effectiveness of CA: To study the quality of conditional generation based on class labels, we first present examples of generated texts from an sample in AG News dataset, shown in table 3.6 in Appendix. Each example is a cherry-picked sample out of five random samples. The generated text expands on a specific aspect regarding the label while retaining the original meaning of the observed text.

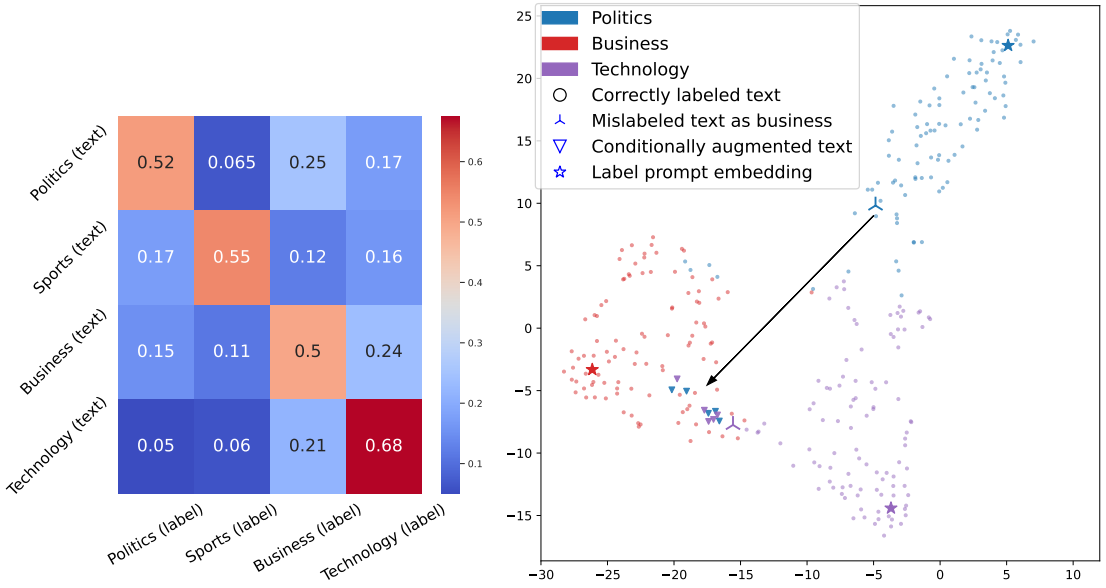


Figure 3.4: The left figure shows a heatmap of the probability when a conditionally generated text based on pseudo label aligns with each of the label prompts. The right figure shows the distribution of the generated text plotted using T-SNE (sports category is out of scope).

In the left of figure 3.4, we show a heatmap of the probability when a conditionally generated text (vertical) aligns with the corresponding label class (horizontal). The highest probability occurs along the diagonal, indicating that the conditionally augmented text based on pseudo label has a closer meaning to the corresponding label class. In the right of figure 3.4, we plot

the distribution of the generated text plotted using T-SNE. The embeddings were obtained by our sentence encoder trained on the 100-th (out of 1000) iteration. We selected two instances that were misclassified as business and located close to the decision boundary. The augmented text, conditioned on the business category, was found to be closer to the label prompt embedding of the business category. This demonstrates the effectiveness of our method to generate less confusing training pairs away from the decision boundary and closer to the pseudo label centroid.

3.5.5 Implementation Details

The label prompts are shown in the upper part of table 3.3. The label prompts are similar to the ones used in in PESCO [138]. We solicit LLM for text augmentation with the instruction template in the lower part of table 3.3, which is the same ones used for Alpaca fine-tuning.

For the generation parameters, we used $temperature$=0.8, top_p=0.95, and sample K=5 augmented texts for each instance with $min\_length = 64$ and $max\_length = 128$. For the self-training of sentence encoder model, we used $batch\_size$=$3 * |C|$ ($|C|$ is the number of categories), lr=1e-5, the max length is 128 for AG News and DBPedia and 192 for Yahoo Answers and Amazon. All the experiments are performed on NVIDIA RTX A6000 gpus. Please refer to our code for details.

Label Prompt
(1)Category: [label]. (2)It is about [label].
Instruction-based (Conditional) Augmentation
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. ### Instruction: Elaborate the text in a few sentences. (Discuss the [pseudo label] aspects of the article.) ### Input: [text] ### Response:

Table 3.3: The designed prompts for enhanced label description and conditional augmentation based on pseudo label.

3.5.6 Selection of Temperature in Eq 3.5

As shown in table 3.4, we include the results with over 5 runs on each dataset. We found $\tau = 0.1$ to be a reasonble choice with slightly better performance, but we acknowledge that the difference is rather small, sometimes fall within std. The choice of τ may serve more of a theoretical motivation rather than practically concerns (as acknowledged in limitation). The theoretical framework unifies previous soft labeling approaches in [104, 138] and is easier for the proof of theorem.

	Agnews	DBpedia	Yahoo Answers	Amazon
τ=1.0	82.75 $\pm$ 0.06	93.77 $\pm$ 0.07	62.66 $\pm$ 0.06	91.39 $\pm$ 0.06
τ=0.5	83.04 $\pm$ 0.05	94.19 $\pm$ 0.05	62.70 $\pm$ 0.10	91.44 $\pm$ 0.06
τ=0.1	**83.18 $\pm$ 0.05**	94.29 $\pm$ 0.05	62.74 $\pm$ 0.08	**91.48 $\pm$ 0.05**
τ=0.05	83.03 $\pm$ 0.05	**94.34 $\pm$ 0.03**	**62.77 $\pm$ 0.10**	91.42 $\pm$ 0.04
τ=0.01	83.02 $\pm$ 0.05	94.33 $\pm$ 0.03	62.76 $\pm$ 0.11	91.42 $\pm$ 0.04

Table 3.4: For the choice of temperature τ in equation 3.5, we include the results with over 5 runs on each dataset. We found $\tau = 0.1$ to be a reasonble choice with slightly better performance, but we acknowledge that the difference is rather small, sometimes fall within std.

3.5.7 Inference Time Augmentation

While GenCo doesn't require LLMs during inference, in our ablation test in table 3.5, we study the impact of inference time augmentation (assuming GPT is available at test time) and self-training on the performance metric. To test inference time augmentation, we performed experiments on a downsampling of both training and testing instances.

Our results show that inference time augmentation (rows with "IA") leads to a performance gain of 1-2%, with a more substantial improvement observed for AG News and Yahoo Answers. This may be attributed to the fact that AG News has an average text length of only 38 words, and the Yahoo Answers dataset includes many answers with only one phrase. Inference time augmentation effectively enhances the quality of shorter text inputs.

ID	Self-train	Methods	AG News	DBpedia	Yahoo Answers	Amazon
		# unlabeled train	4k (3.4%)	11.2k (2%)	15k (< 1%)	20k (< 1%)
		# test	7.6k	28k	20k	20k
1	No	Sentence-enc	75.6	73.4	55.5	89.6
2	No	Sentence-enc + Inf-Aug	78.2	74.7	57.4	90.2
3	Yes	Self-train	83.3	96.3	62.5	91.1
4	Yes	Self-train + Inf-Aug	83.9	96.8	64.3	91.3
5	Yes	GenCo	89.2	98.4	68.6	95.3
6	Yes	GenCo + Inf-Aug	89.7	98.5	70.2	95.4

Table 3.5: Evaluation of inference time augmentation. "Inf-Aug" represents input augmentation added during inference.

Observed Text	Starbucks Corp's president and chief executive, Orin Smith, said Tuesday he plans to retire early next year because he wants to slow down and focus on philanthropy, family and sports.
Politics	The announcement of Orin Smith's retirement is likely to have political implications for Starbucks Corporation... His replacement will have big shoes to fill, and the decision could have long-term effects on the company's policies and direction.
Sports	...Sports have always been a major part of Smith's life, as he was a college athlete and later went on to become the CEO of Starbucks. It is clear that sports have had a major influence on his life and he wants to make time for them in his retirement.
Business	Starbucks Corp has seen great success under the leadership of Orin Smith, with the company's stock price more than tripling since he became CEO in 2005. This success has allowed him to retire early and ...
Technology	Orin Smith's plan to retire early next year is an example of how technology has changed the way we work and live. By utilizing technology, Smith is able to take advantage of the increasingly popular trend of "work-life balance" ...

Table 3.6: Examples of generated text conditioned on pseudo labels in the left column.

3.5.8 Qualitative Examples for Conditionally Generated Examples on Pseudo-label

In table 3.6, we show generated examples of a sample text from the Agnews dataset. We generate 5 examples conditioned on each of the 4 labels, and cherry-pick one for each label in the table presentation. The example shows that the topic of a generated text is related to the label which is conditioned on, while pertains the original meaning. This opens a path to leverage the language understanding ability of LLM for data augmentation, especially during self-training.

3.6 Proof of Theorems

Theorem 7. *Consider a binary classification problem with linearly separable labeled examples, when* $0 < \tau < 1$*, optimizing* $\mathcal{L}_{t2l} = -\sum_{i=1}^{N}\sum_{j=1}^{L} Q(\hat{y}_j|x_i)\log P(\hat{y}_j|x_i)$ *with gradient descend will enforce the larger margin between classes.*

Proof. We use dot product $\langle\cdot,\cdot\rangle$ as implementation of similarity function. Let the embedding of instance i be $\boldsymbol{x}_i = f_\theta(x_i)$ and the embedding of label prompt j be $\boldsymbol{e}_c = f_\theta(p_c), c \in \{1, 2\}$ for binary classification. Then,

$$P(\hat{y}_1|x_i;\theta) = \frac{\exp(\langle \boldsymbol{x}_i, \boldsymbol{e}_1\rangle)}{\exp(\langle \boldsymbol{x}_i, \boldsymbol{e}_1\rangle) + \exp(\langle \boldsymbol{x}_i, \boldsymbol{e}_2\rangle)} = \frac{1}{1+\exp(-\langle \boldsymbol{x}_i, \boldsymbol{e}_1 - \boldsymbol{e}_2\rangle)} \tag{3.12}$$

$$P(\hat{y}_2|x_i;\theta) = 1 - P(\hat{y}_1|x_i;\theta) \tag{3.13}$$

Notation-wise, define $d_i = \langle \boldsymbol{x}_i, \boldsymbol{e}_1 - \boldsymbol{e}_2\rangle$, then

$$P(\hat{y}_1|x_i;\theta) = \frac{1}{1+e^{-d_i}} \tag{3.14}$$

$$P(\hat{y}_2|x_i;\theta) = 1 - \frac{1}{1+e^{-d_i}} \tag{3.15}$$

$$\tag{3.16}$$

In binary classification, the margin is simply

$$\text{margin} = \begin{cases} d_i & x_i \text{ is class 1} \\ -d_i & x_i \text{ is class 2} \end{cases}$$

For soft-label distribution Q,

$$Q(\hat{y}_1|x_i;\theta) = \frac{1}{1+e^{-d_i/\tau}} \tag{3.17}$$

$$Q(\hat{y}_2|x_i;\theta) = 1 - \frac{1}{1+e^{-d_i/\tau}} \tag{3.18}$$

$$\tag{3.19}$$

Then $\mathcal{L}_{t2l}$ is derived as

$$\mathcal{L}_{t2l} = \sum_{i=1}^{N} \log(1+e^{-d_i}) + \frac{d_i e^{-d_i/\tau}}{1+e^{-d_i/\tau}} \tag{3.20}$$

Calculate the derivative of $\mathcal{L}_{t2l}$ w.r.t d_i,

$$\frac{\partial \mathcal{L}_{t2l}}{\partial d_i} = \frac{-d_i e^{-d_i/\tau}}{\tau(e^{-d_i/\tau}+1)^2} + \frac{e^{-d_i/\tau} - e^{-d_i}}{(e^{-d_i/\tau}+1)(e^{-d_i}+1)} \tag{3.21}$$

For the first part of equation 3.21, the sign depends on $-d_i$. For the second part, the sign depends on $e^{-d_i/\tau} - e^{-d_i}$. When $0 < \tau < 1$,

$$\begin{cases} e^{-d_i/\tau} - e^{-d_i} < 0 & \text{when } d_i > 0 \\ e^{-d_i/\tau} - e^{-d_i} > 0 & \text{when } d_i < 0 \end{cases}$$

Therefore,

$$\begin{cases} \frac{\partial \mathcal{L}_{t2l}}{\partial d_i} < 0 & \text{when } d_i > 0 \\ \frac{\partial \mathcal{L}_{t2l}}{\partial d_i} > 0 & \text{when } d_i < 0 \end{cases} \tag{3.22}$$

One step of gradient descend optimizes d by $d_i' = d_i - \eta \frac{\partial \mathcal{L}_{t2l}}{\partial d_i}$. From equation 3.22, we get the conclusion that $|d_i'| > |d_i|$. In other words, the margin becomes larger after optimization, which finishes the proof. □

Theorem 8. *Under the setting in Theorem 7, let m_i be the margin of instance i and consider the constraint $m_i \leq B$ for all i, the classifier converges to a max margin classifier, as the bound B goes to infinity.*

Proof. Using the definition from Theorem 7,

$$\mathcal{L}_{t2l} = \sum_{i=1}^{N} \log(1+e^{-d_i}) + \frac{d_i e^{-d_i/\tau}}{1+e^{-d_i/\tau}} \tag{3.23}$$

The margin m_i for instance i can be written as $m_i = \begin{cases} d_i & x_i \text{ is class 1} \\ -d_i & x_i \text{ is class 2} \end{cases}$.

The equation 3.23 can be written as

$$\mathcal{L}_{t2l} = \sum_{y_i=0} \log(1 + e^{-m_i}) + \frac{m_i e^{-m_i/\tau}}{1 + e^{-m_i/\tau}} + \sum_{y_j=1} \log(1 + e^{m_j}) - \frac{m_j e^{m_j/\tau}}{1 + e^{m_j/\tau}} \tag{3.24}$$

Let $m^* = \min(m_i)$ be the minimal margin, let N_1 and N_2 be the number of instances in class 1 and class 2 respectively which reaches the minimal margin. From the gradient analysis in equation 3.22, the examples with $m_i > m^*$ has loss lower bounded by that with minimal margin. Then

$$\begin{aligned} \mathcal{L}_{t2l} = N_1(\log(1 + e^{-m^*}) + \frac{m^* e^{-m^*/\tau}}{1 + e^{-m^*/\tau}}) + N_2(\log(1 + e^{m^*}) - \frac{m^* e^{m^*/\tau}}{1 + e^{m^*/\tau}}) \\ + O(\log(1 + e^{-m^*}) + \frac{m^* e^{-m^*/\tau}}{1 + e^{-m^*/\tau}}) + O(\log(1 + e^{m^*}) - \frac{m^* e^{m^*/\tau}}{1 + e^{m^*/\tau}}) \end{aligned} \tag{3.25}$$

When B approaches ∞, for N_1 part in equation 3.25,

$$\log(1 + e^{-m^*}) + \frac{m^* e^{-m^*/\tau}}{1 + e^{-m^*/\tau}} \sim e^{-m^*} + m^* e^{-m^*/\tau} \tag{3.26}$$

When $m \to B$, $\lim_{m\to B} e^{-m^*} \to 0$, and $\lim_{m\to B} m^* e^{-m^*/\tau} = \lim_{m\to B} \frac{1}{1/\tau e^{m^*/\tau}} = 0$ by L'Hopital's rule.

For N_2 part in equation 3.25,

$$\log(1 + e^{m^*}) - \frac{m^* e^{m^*/\tau}}{1 + e^{m^*/\tau}} \sim \log(1 + e^{m^*}) - m^* \tag{3.27}$$

When $m \to B$, $\lim_{m\to B} \log(1 + e^{m^*}) - m^* = \lim_{m\to B} \log(1 + \frac{1}{e^{m^*}}) = 0$.

Therefore, the loss is minimized when the minimal margin is maximized and thus the classifier converges to a max margin classifier when B goes to infinity. □

3.7 Conclusion

Our proposed approach, GenCo, demonstrates how LLM-generated training signals can replace or supplement manual labels, enabling robust neural network optimization in zero-shot text classification. By integrating an LLM into the self-training loop of a smaller sentence encoder classifier with contrastive learning, GenCo achieves state-of-the-art performance across four benchmark datasets, even with minimal in-domain text data. This work highlights the potential of leveraging LLM-generated synthetic training signals to improve the efficiency and effectiveness of smaller classifiers, reducing reliance on both human annotations and large unlabeled corpora. We hope this approach inspires further research into model-driven supervision for data-efficient NLP.

Part II

Auxiliary Graph Generation for Time Series Analysis

Chapter 4

Change-point Detection with Correlation Graph Generation

In this chapter, we address the problem of change point detection (CPD) in time series analysis, which involves identifying abrupt shifts in data. A key challenge in CPD is the scarcity of labeled training data, as changes occur infrequently and rely on manually annotated change points. To address this, unsupervised CPD methods leverage model-generated future predictions to compare against observed time series. Building on this idea, we propose a **Correlation-aware Dynamics Model** for CPD, which enhances detection by incorporating **model-generated correlation structures** through graph neural networks within an encoder-decoder framework. Our approach dynamically infers relationships among variables at each time step, reducing dependence on manual feature engineering. Comprehensive experiments on synthetic and real-world datasets demonstrate that our model not only outperforms existing baselines but also differentiates between changes in correlation structures and independent shifts. These results highlight the effectiveness of model-generated signals in improving the robustness and accuracy of CPD.

Highlights We are the first to explore dynamic correlation modeling in multivariate time series, achieving superior performance in CPD tasks.

Model-Generated Signals Our approach employs a spatiotemporal Transformer to dynamically generate correlation structures at each time step, allowing the model to adaptively learn evolving dependencies without human supervision.

4.1 Introduction

Change-point detection (CPD) aims to detect abrupt property changes over time series data. In this study, change-points are detected through the changes of *dynamics* and *correlation* of variables. Dynamics refers to the physical property that determines a variable's modus operandi and correlation describes the interactions between variables. Previous CPD methods [107, 161] model dynamics by parametric distributions like Hidden Markov Models (HMM), but they don't explicitly capture the correlation information. Other works capture static correlation structures in the multivariate time series [74], but they can't detect any correlation changes.

We propose a **Cor**relation-aware **D**ynamics Model for **C**hange-**p**oint **D**etection (CORD_CPD) which incorporates graph neural networks into an encoder-decoder framework to explicitly model both changeable correlation structure and variable dynamics. We refer to the changes of correlation structure as **correlation changes** and the changes of variable dynamics as **independent changes**, as shown in Fig. 4.1.

Our model is capable of distinguishing the two types of changes, which could have a broader impact on decision-making. In financial markets, traders use pair trading strategy to profit from correlated stocks, such as Apple and Samsung (both are phone sellers), which share similar dips and highs. News about Apple expanding markets may independently raise its price without breaking its correlation with Samsung. However, news about Apple building self-driving cars will break its correlation with Samsung, and establish new correlations with automobile companies. While both of them are change-points, the former is an independent change of variables and the latter is a correlation change between variables. Knowing the type of change can guide financial experts to choose trading strategies properly.

Our contributions can be summarized as follows:

- We propose CORD_CPD to capture both changeable correlation structure and variable dynamics.
- Our CORD_CPD classifies the change-points as correlation changes or independent changes, and ensembles them for robust CPD.
- Experiment on synthetic and real datasets demonstrates that our model can bring enhanced interpretability and improved performance in CPD tasks.

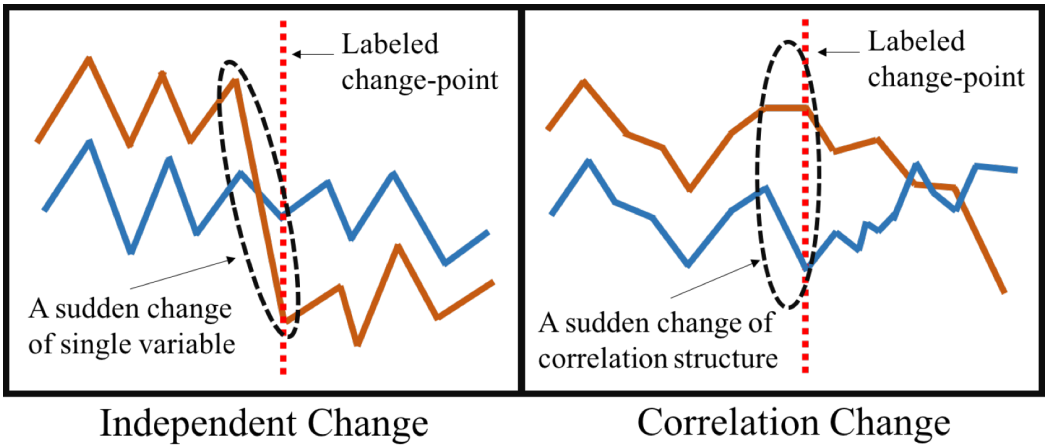


Figure 4.1: (Left) an independent change of one variable and (Right) a correlation change between two variables. The red vertical line is the labeled change-point.

4.2 Method for CPD

A multivariate time series is denoted by $\mathbf{x} \in \mathbf{R}^{T \times N \times M}$, where T is the time steps, N is the number of variables and M is the number of features for each variable. We study the CPD problem in a retrospective setting and assume there is one change-point per $\mathbf{x} = \{\mathbf{x}^j\}_{j=1}^{T}$. The change-point at time step t satisfies:

$$\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{t-1}\} \sim \mathbb{P}$$
$$\{\mathbf{x}^t, \mathbf{x}^{t+1}, \ldots, \mathbf{x}^{T}\} \sim \mathbb{Q}$$

Where $\mathbb{P}$ and $\mathbb{Q}$ denotes two different distributions. We attribute this difference to a correlation change (of the correlation structure), an independent change (of variable dynamics), or a mixture of both.

Correlation Change corresponds to the change of the correlation structure of multivariate time series, which is modeled by correlation matrices $\mathbf{A} \in \mathbb{R}^{T \times N \times N}$. At each time step, the pairwise interaction between variables (A_{ij}^t) is represented as a continuous value between 0 and 1, indicating how much they are correlated. The correlation change score $\mathbf{s}_r$ is calculated by the L_1 distance between two neighboring correlation matrices:

$$s_r^t = \|\mathbf{A}^t - \mathbf{A}^{t-1}\|_1, t > 1 \tag{4.1}$$

Independent Change corresponds to the change of the variable dynamics. Given the current values of time series (and the extracted correlation matrices), if the dynamics rule is followed,

the expected values of the future time steps predicted by our model will be close to the observed values; Otherwise, the difference will be large. This difference is used as the independent change score $\mathbf{s}_d$. Formally, we use the Mean Squared Error (MSE) as a metric to compare the expected values $\hat{\mathbf{w}}^{t+1} = \{\hat{\mathbf{x}}^i\}_{i=t+1}^{t+k}$ with the observed values $\mathbf{w}^{t+1} = \{\mathbf{x}^i\}_{i=t+1}^{t+k}$ over a window of size k.

$$s_d^t = \text{MSE}(\hat{\mathbf{w}}^t, \mathbf{w}^t), t > 1 \tag{4.2}$$

Note that if only a correlation change takes place, the expected value $\hat{\mathbf{w}}^t$ should not be different from the observed value $\mathbf{w}^t$, since we model a conditional probability $P(\mathbf{x}^t|\mathbf{x}^{<t}, \mathbf{A})$ and any correlation change will be factored in.

Ensemble of Change-point Scores aims to combine the correlation change with the independent change, because in real world applications, change-points could be resulted from a mixture of both. A simple way to ensemble them (for $\mathbf{s}_{en}$) is to sum the normalized scores of $\mathbf{s}_r$ and $\mathbf{s}_d$:

$$\mathbf{s}_{en} = \text{Norm}(\mathbf{s}_r) + \text{Norm}(\mathbf{s}_d) \tag{4.3}$$

$$\text{Norm}(\mathbf{s}) = \frac{\mathbf{s} - u_s}{\sigma_s} \tag{4.4}$$

where u_s and σ_s are mean and standard deviation of score $\mathbf{s}$.

In order to use our CPD methods above, we need to model correlation matrices and to be able to predict a future window of time steps based on the extracted correlation. We will introduce our CORD_CPD in the next section.

4.3 Correlation-aware Dynamics Model

The CORD_CPD has an encoder for correlation extraction and a decoder for variable dynamics. Given a time series $\mathbf{x}$, the encoder models a distribution of correlation matrix $q_\phi(\mathbf{A}^t|\mathbf{x})$ for each time step t, and by factorization,

$$q_\phi(\mathbf{A}|\mathbf{x}) = \prod_{t=1}^{T} q_\phi(\mathbf{A}^t|\mathbf{x}) \tag{4.5}$$

The decoder models a distribution of time steps $p_\theta(\mathbf{x}|\mathbf{A})$ auto-regressively,

$$p_\theta(\mathbf{x}|\mathbf{A}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}^t|\mathbf{x}^{<t}, \mathbf{A}^{t-1}) \tag{4.6}$$

The objective function maximizes the log likelihood,

$$\mathcal{L}_{obj} = \mathbb{E}_{q_\phi(\mathbf{A}|\mathbf{X})}[\log p_\theta(\mathbf{x}|\mathbf{A})] \tag{4.7}$$

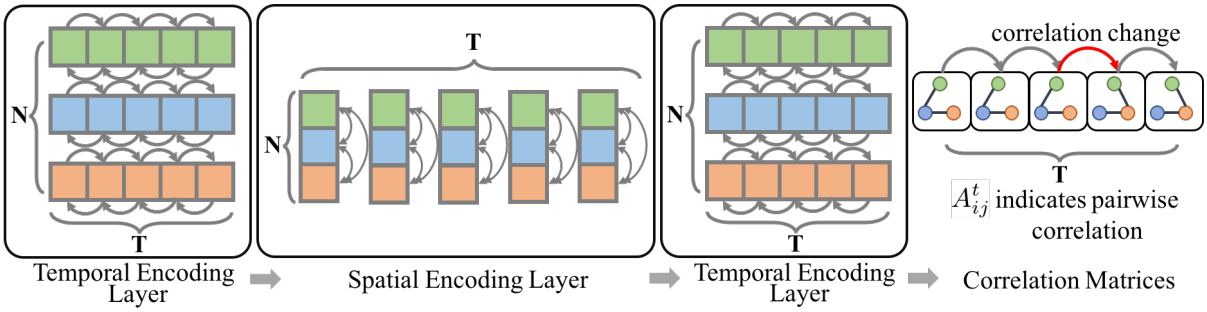


Figure 4.2: CORD_CPD Encoder: the encoder extracts correlation matrices from multivariate time series. The temporal encoding layer captures time dependent features, and the spatial encoding layer models relational features between variables.

4.3.1 Correlation Encoder

The encoder infers a correlation matrix $\mathbf{A}^t$ at each time step, which depends on both temporal features and variable interactions. To leverage both sources, we propose Temporal Encoding Layers (TEL) to extract features across time steps and Spatial Encoding Layers (SEL) to extract features from variable interactions. As shown in Fig 4.2, the two types of layers are alternatively applied to progressively incorporate temporal and correlation features into latent embeddings. Practically, we found 2 TEL and 1 SEL is enough for our tasks.

For each layer, let $\mathbf{h} \in \mathbb{R}^{T\times N\times K}$ denote the input and let $\tilde{\mathbf{h}} \in \mathbb{R}^{T\times N\times K'}$ denote the output, where T is the time steps, N is the number of variables, and K, K' are the number of input and output features respectively. The input to the initial layer is the multivariate time series data $\mathbf{x} \in \mathbb{R}^{T\times N\times M}$. The posterior distribution of the correlation matrix is modeled by

$$q_\phi(A_{ij}^t|\mathbf{x}) = \text{Softmax}(\text{Linear}([\tilde{\mathbf{h}}_{(f)i}^t; \tilde{\mathbf{h}}_{(f)j}^t]) \tag{4.8}$$

$$\tilde{\mathbf{h}}_{(f)} = \text{TEL}_2(\text{SEL}(\text{TEL}_1(\mathbf{x}))) \tag{4.9}$$

where $[\cdot;\cdot]$ is the concatenation operator and $\tilde{\mathbf{h}}_{(f)}$ is the embedding of the final layer. As an additional trick, we apply Gumbel-Softmax [61] to enforce sparse connections in correlation matrices in order to reduce noise.

Temporal Encoding Layer (TEL) leverages information across T time steps (independently for each variable). For a fixed variable i, let $\mathbf{h}_i = \{\mathbf{h}_i^t\}_{t=1}^{t=N}$ denote the embeddings of that variable at all time steps. We offer two implementations of TEL with different neural architectures: RNN_{TEL} and $\text{Trans}_{\text{TEL}}$.

RNN_{TEL} is a bidirectional GRU network [28]:

$$\overrightarrow{\mathbf{h}_i}^t = \overrightarrow{\text{GRU}}(\overrightarrow{\mathbf{h}_i}^{t-1}, \mathbf{h}_i^t) \tag{4.10}$$

$$\overleftarrow{\mathbf{h}_i}^t = \overleftarrow{\text{GRU}}(\overleftarrow{\mathbf{h}_i}^{t+1}, \mathbf{h}_i^t) \tag{4.11}$$

$$\tilde{\mathbf{h}}_i^t = [\overrightarrow{\mathbf{h}_i}^t, \overleftarrow{\mathbf{h}_i}^t] \tag{4.12}$$

where $\overrightarrow{\mathbf{h}_i}^t, \overleftarrow{\mathbf{h}_i}^t$ are intermediate representation from forward and backward GRU. The output $\tilde{\mathbf{h}}^t$ is a concatenation of embeddings from both directions.

$\text{Trans}_{\text{TEL}}$ uses the Transformer model [134] with self-attention to capture temporal dependencies. For the self-attention layer, the input is transformed into query matrices $\mathbf{Q}_i^t = \mathbf{h}_i^t \mathbf{W}_Q$, key matrices $\mathbf{K}_i^t = \mathbf{h}_i^t \mathbf{W}_K$ and value matrices $\mathbf{V}_i^t = \mathbf{h}_i^t \mathbf{W}_V$. Here $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ are learnable parameters. Finally, the dot-product attention is a weighted sum of value vectors:

$$\tilde{\mathbf{h}}_i^t = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \cdot \mathbf{V} \tag{4.13}$$

where d_k is the size of hidden dimension. Similar to [134], we use residual connection, layer normalization and positional encoding for $\text{Trans}_{\text{TEL}}$.

Spatial Encoding Layer (SEL) leverages the information between the N variables (independently at each time step) via graph neural networks (GNN) [75]. For a fixed time step t, let $\mathbf{h}^t = \{\mathbf{h}_i^t\}_{i=1}^N$ denote the embeddings all variables at time t. The output is obtained by

$$\tilde{\mathbf{h}}^t = \text{GNN}(\{\mathbf{h}_i^t\}_{i=1}^N) \tag{4.14}$$

where a GNN module is implemented by the feature aggregation and combination operations:

$$\mathbf{e}_{ij} = f_e([\mathbf{h}_i^t; \mathbf{h}_j^t]) \tag{4.15}$$

$$\tilde{\mathbf{h}}_j = f_v(\mathbf{h}_j + \sum_{i \neq j} \mathbf{e}_{ij}) \tag{4.16}$$

where Eq. 4.15 aggregates features between neighboring nodes and Eq. 4.16 combines those features by a summation. $f_e(\cdot)$ and $f_v(\cdot)$ are non-linear neural networks for which we provide two implementations: GNN_{SEL} and $\text{Trans}_{\text{SEL}}$.

GNN_{SEL} is implemented by a multilayer perceptron (MLP) and $\text{Trans}_{\text{SEL}}$ is implemented by the Transformer model. Compared with MLP, Transformer has could be advantageous for spatial encoding because of well-designed self-attention, residual connection and layer normalization. The positional encoding layer is removed from Transformer because the variables are order invariant.

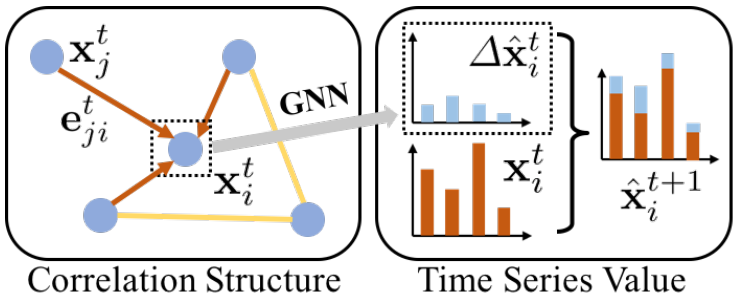


Figure 4.3: CORD_CPD Decoder: Given a correlation matrix, the decoder predicts the change of future steps.

4.3.2 Dynamics Decoder

At a high level, the decoder learns the dynamics of variables by predicting the future time steps to be as close as the observed values. Instead of predicting the value of $\hat{\mathbf{x}}_i^{t+1}$ directly, we predict the change $\Delta\hat{\mathbf{x}}_i^t = \hat{\mathbf{x}}_i^{t+1} - \mathbf{x}_i^t$ as shown in Fig. 4.3.

Since the prediction has to factor in the correlation between variables, we also need GNN to incorporate correlation matrices into feature embeddings. Again, the feature aggregation and combination operations are performed on the input $\mathbf{x}^t$,

$$\mathbf{e}_{ji}^t = \mathbf{A}_{ji}^t g_e([\mathbf{x}_j^t; \mathbf{x}_i^t]) \tag{4.17}$$

$$\tilde{\mathbf{h}}_i^t = g_v(\mathbf{x}_i^t + \sum_{j \neq i} \mathbf{e}_{ji}^t) \tag{4.18}$$

where the functions $g_e(.)$ and $g_v(.)$ are MLPs. We model $\Delta\hat{\mathbf{x}}_i^t = g_{\text{out}}(\tilde{\mathbf{h}}_i^{\leq t})$, where $g_{\text{out}}(\tilde{\mathbf{h}}_i^{\leq t})$ can be $\text{MLP}(\tilde{\mathbf{h}}_i^t)$ or $\text{RNN}(\tilde{\mathbf{h}}_i^{\leq t})$ depending on the application. Together, $\hat{\mathbf{x}}_i^{t+1} = \mathbf{x}_i^t + \Delta\hat{\mathbf{x}}_i^t = \mathbf{x}_i^t + g_{\text{out}}(\tilde{\mathbf{h}}_i^{\leq t})$.

The log likelihood of density $p_\theta(\mathbf{x}|\mathbf{A})$ can be expressed as:

$$\log p_\theta(\mathbf{x}|\mathbf{A}) = \sum_{t=1}^{T} \log p_\theta(\mathbf{x}^t|\mathbf{x}^{<t}, \mathbf{A}^{t-1}) \tag{4.19}$$

$$= \sum_i \sum_{t=1}^{T} \log \mathcal{N}(\mathbf{x}_i^t|\hat{\mathbf{x}}_i^t, \sigma^2\mathbf{I}) \tag{4.20}$$

$$\propto -\sum_i \sum_{t=2}^{T} \frac{\|\mathbf{x}_i^t - \hat{\mathbf{x}}_i^t\|_2^2}{2\sigma^2} \tag{4.21}$$

Maximizing Eq. 4.21 is equivalent to minimizing $\mathcal{L}_{obj} = \sum_i \sum_{t=2}^{T} \frac{\|\mathbf{x}_i^t - \hat{\mathbf{x}}_i^t\|_2^2}{2\sigma^2}$.

Since change-points are sparse in time series data, we introduce an additional regularization

to ensure the smoothness of correlation matrix:

$$\mathcal{L}_{smooth} = \frac{1}{T-1} \sum_{t=2}^{T} \|\mathbf{A}^t - \mathbf{A}^{t-1}\|_2^2 \tag{4.22}$$

Finally, the loss function is $\mathcal{L} = \mathcal{L}_{obj} + \lambda \mathcal{L}_{smooth}$, where λ controls the relative strength of smoothness regularization.

4.4 Experiment with Physics Simulations

4.4.1 Particle-spring Change-point Dataset

We developed a dataset with a simulated physical particle-spring system. The system contains $N = 5$ particles that move in a rectangular space. Some randomly selected pairs (out of the 10 pairs in total) of particles are connected by invisible springs. The motion of particles are determined by the laws of physics such as Newton's law, Hooke's law, and Markov property. The trajectories of length $T = 100$ of the particles are recorded as the multivariate time series data. Each variable has $M = 4$ features: location l_x, l_y and speed v_x, v_y.

While the physical system is similar to the one in [74], we additionally design 3 types of change-points by perturbing the location, speed, and connection at a random time step between $[25, 75]$:

- **location**: A perturbation to the current location sampled from $\mathcal{N}(0, 0.1)$, where the range of the location is $[-5, 5]$.
- **speed**: A perturbation to the current speed by sampled from $\mathcal{N}(0, 0.02)$, where range of the speed is $[-1, 1]$.
- **connection**: re-sample connections and ensure that at least 5 out of 10 pairs of connections are changed.

The change of location or speed (both are dynamics) belongs to the independent change, and the change of connection (a type of correlation) belongs to the correlation change. Since the change-point is either a correlation change or an independent change, we are able to test the ability of our model to classify them.

We generate 500 time series for each type of change and mix them together (totally 1500 time series) as training data. For validation and testing data, we generate 100 time series for each type of change and evaluate on them separately. Our model is unsupervised, so the validation set is only used for hyperparameter tuning. In real world datasets, human labeled change-points are scarce in quantity, which usually results in large variance in evaluation. As a remedy, our

synthetic data can be generated in a large amount to reduce such a variance in testing.

4.4.2 Evaluation Metric and Baselines

For quantitative evaluation of CPD performance, we consider two metrics:

Area-Under-the-Curve (AUC) of the receiver operating characteristic (ROC) is a metric commonly used in the CPD literature [15].

Triangle Utility (TRI) is a hinge-loss-based metric: $\max(0, 1 - \frac{\|y-l\|}{w})$, where $w = 15$ is the margin, l and y are the labeled and predicted change-points.
Both of the metrics range from $[0, 1]$ and higher values indicate better predictions. However, AUC treats the change-point scores at each time step independently, without considering any temporal patterns. TRI considers the distance between the label and the predicted change-point (the one with highest change-point score), but it doesn't measure the quality of predictions at the other time steps. We use both metrics because they complement with each other.

Next, we introduce 6 baselines of the state-of-the-art statistical and deep learning models:

- **ARGP-BODPD** [118] is Bayesian change-point model that uses auto-regressive Gaussian Process as underlying predictive model.
- **RDR-KCPD** [94] uses relative density ratio technique that considers f-divergence as the dissimilarity measure.
- **Mstats-KCPD** [84] uses kernel maximum mean discrepancy (MMD) as dissimilarity measure on data space.
- **KL-CPD** [15] uses deep neural models for kernel learning and generative method to learn pseudo anomaly distribution.
- **RNN** [28] is a recurrent neural network baseline to learn variable dynamics from multivariate time series (without modeling correlations).
- **LSTNet** [77] combines CNN and RNN to learn variable dynamics from long and short-term temporal data (without modeling correlations).

4.4.3 Main Results

Table 4.1 shows the performance of the statistical baselines (first panel), the deep learning baselines (second panel) and our proposed CORD_CPD (third panel).

Statistical Baselines are not as competitive as the other deep learning models among all the types of changes. One explanation is that those models have strong assumption on the pa-

model	location		speed		connection	
	AUC	TRI	AUC	TRI	AUC	TRI
ARGP-BOCPD	0.5244	0.0880	0.5231	0.0660	0.5442	0.1287
RDR-KCPD	0.5095	0.0680	0.5279	0.1093	0.5234	0.0860
Mstats-KCPD	0.5380	0.0730	0.5369	0.0727	0.5508	0.0833
RNN	0.5413	0.2567	0.5381	0.2660	0.5446	0.3047
LSTNet	0.5817	0.3487	0.5817	0.3460	0.5337	0.2193
KL-CPD	0.5247	0.1053	0.5378	0.1352	0.5574	0.3127
GNN_{SEL}+RNN_{TEL}	0.9864	0.9740	0.9700	**0.9320**	**0.9681**	**0.9153**
$\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$	**0.9885**	**0.9773**	**0.9755**	0.9080	0.9469	0.9040
GNN_{SEL}+$\text{Trans}_{\text{TEL}}$	0.9692	0.9333	0.9609	0.8473	0.8840	0.8527

Table 4.1: AUC and TRI metrics on synthetic datasets for the prediction of location, speed and connection change. Our CORD_CPD (evaluated with $\mathbf{s}_{en}$) has the best performance on both metrics among all the baselines.

rameterization of probability distributions, which may hurt the performance on datasets that demonstrate complicated interactions of variables. The dynamics rule of the physics system can be hardly captured by those methods.

Deep Learning Baselines are slightly better than the statistical models, in which the LSTNet has the best performance on location and speed changes. Since LSTNet has a powerful feature extractor for long and short-term temporal data, it is better at learning variable dynamics. However, as correlation plays an important role in the synthetic data, ignoring it will hurt performance in general.

CORD_CPD is evaluated on the test data by the ensemble score $\mathbf{s}_{en}$. It has the best performance on both metrics among all the baselines. We didn't include the result of $\text{Trans}_{\text{SEL}}$+$\text{Trans}_{\text{TEL}}$, because empirically it is harder to converge. $\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$ is the best at detecting the independent changes, while GNN_{SEL}+RNN_{TEL} is the best at detecting the correlation changes. The reason could be that the Transformer models are better at identifying local patterns, while RNNs are more stable at combining features with long term dependencies. Among the three types of changes, the score of connection change is lower than that of the other two, indicating the detection of correlation changes is harder than independent changes.

model	type	location		speed		connection	
		AUC	TRI	AUC	TRI	AUC	TRI
$\mathbf{GNN_{SEL}}$+$\mathbf{RNN_{TEL}}$	cor	0.5145	0.3153	0.5590	0.3553	**0.9649**	**0.9073**
	ind	**0.9835**	**0.9727**	**0.9587**	**0.9493**	0.8093	0.7320
$\mathbf{Trans_{SEL}}$+$\mathbf{RNN_{TEL}}$	cor	0.4944	0.2626	0.5463	0.3266	**0.9755**	**0.9273**
	ind	**0.9859**	**0.9720**	**0.9685**	**0.9233**	0.7774	0.6460
$\mathbf{GNN_{SEL}}$+$\mathbf{Trans_{TEL}}$	cor	0.5544	0.3467	0.5832	0.4266	**0.9098**	**0.8787**
	ind	**0.9855**	**0.9693**	**0.9623**	**0.9133**	0.7912	0.7620

Table 4.2: Our CORD_CPD separately computes the scores for correlation change (cor) and independent change (ind). The correlation change score is high on the connection data, while the correlation change score is high on the location and speed data.

4.4.4 Change-point Type Classification

Our CORD_CPD separately computes change-point scores for correlation change ($\mathbf{s}_r$) and independent change ($\mathbf{s}_d$). We show the ability of our model to separate the two types of changes based on the scores.

Correlation Vs. Independent Change.

In Table 4.2, the correlation change (cor) and independent change (ind) are separately evaluated. The correlation change scores ($\mathbf{s}_r$) are high on the connection data, while the independent change scores ($\mathbf{s}_d$) are high on location and speed change. This result shows that our system can indeed distinguish the two types of changes.

Location&speed The independent changes can be successfully distinguished. For location and speed data, AUC of the independent changes is over 0.97, close to a perfect detection; AUC of the correlation change is close to 0.5, nearly a random guess. Therefore, our system doesn't signal a correlation change for location and speed data, but it gives a strong signal of an independent change.

Connection The correlation changes are harder to be detected, but CORD_CPD gives a good estimation. In the connection data, AUC of correlation change are higher than independent change, but the gap was smaller than that in location and speed data. The reason could be that

the errors made by encoder are propagated into the decoder, and thus made the forecasting of time series values inaccurate.

Classification Method.

While our model shows a potential to distinguish the two types of changes, we want it to be able to classify them. We propose to use the difference between normalized correlation change score $\mathbf{s}_r$ and independent change score $\mathbf{s}_d$ as an indicator of change-point type, at time t:

$$\mathrm{Norm}(\mathbf{s}_r)^t - \alpha\,\mathrm{Norm}(\mathbf{s}_d)^t \begin{cases} \geq \tau, & \text{correlation change} \\ < \tau, & \text{independent change} \end{cases}$$

where $\alpha = 0.75$ is our design choice, and τ is a threshold to separate the correlation change and the independent change. Moving the value of τ controls the type I error (False Positive) and the type II error (False Negative). To measure the classification quality by leveraging the error, ROC AUC is a typical solution.

We classify the change-point types under two settings: with label and without label, according to whether the labeled change-point is provided.

With Label: When a labeled change-point is provided by human experts, our model classifies it as either a correlation change or an independent change, whichever dominates.

Without Label: When the label information is unavailable, our model performs classification from the predicted change-point with the highest $\mathbf{s}_{en}$ score.

The results are shown in Table 4.3. Our best model $\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$ achieves an ROC AUC of 0.979 (with label) and 0.973 (without label). This indicates that our model has a strong ability to discriminate the two types of change-points under both settings. GNN_{SEL}+$\text{Trans}_{\text{TEL}}$ has the worst classification performance, which is consistent to the observation in Table 4.2 that it is not good at capturing correlation changes.

In the next experiment, we set $\tau = 0$ and report the classification accuracy on the three data types. As shown in right part of Table 4.3, a high accuracy of 98% on identifying the location and speed change demonstrates that our model can predict the independent changes well. For correlation changes, the $\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$ shows the best performance by achieving 93% on supervised setting and 84% on unsupervised setting.

When labeled change-points are not provided, the classification task could be more difficult, because the it relies on the predicted change-points. If a predicted change-point is far from the

ground truth, the classification is prone to errors.

model	ROC AUC	location	speed	connection
With Label				
GNN_{SEL}+RNN_{TEL}	0.972	**98%**	**97%**	68%
$\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$	**0.979**	**98%**	96%	**93%**
GNN_{SEL}+$\text{Trans}_{\text{TEL}}$	0.916	**98%**	96%	87%
Without Label				
GNN_{SEL}+RNN_{TEL}	0.969	**98%**	**96%**	73%
$\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$	**0.973**	96%	92%	**84%**
GNN_{SEL}+$\text{Trans}_{\text{TEL}}$	0.929	91%	83%	75%

Table 4.3: ROC AUC metric demonstrates the ability of our model to separate the two types of change-points. When $\tau = 0$, we report the change-point classification accuracy on the 3 types of data.

model	AUC	TRI
ARGP-BOCPD	0.5079	0.1773
RDR-KCPD	0.5633	0.1933
Mstats-KCPD	0.5112	0.1480
RNN	0.5540	0.2393
LSTNet	0.5688	0.3145
KL-CPD	0.5326	0.2102
GNN_{SEL}+RNN_{TEL}	0.7868	0.7574
$\text{Trans}_{\text{SEL}}$+$\text{RNN}_{\text{TEL}}$	0.7903	0.7750
GNN_{SEL}+$\text{Trans}_{\text{TEL}}$	**0.8277**	**0.8020**

Table 4.4: We report the performance of our CORD_CPD on a real-world multivariate time series dataset (PAMAP2). The variables are sensors and the features includes temporatures and 3-D motions. The change-points are transitions between activities.

4.5 Experiments with Physical Activity Monitoring

In addition to our synthetic dataset, we test our CORD_CPD on real-world data: the PAMAP2 Physical Activity Monitoring dataset [117]. The dataset contains sensor data collected from 9 subjects performing 18 different physical activities, such as walking, cycling, playing soccer,

etc. Specifically, the variables we consider are $N = 3$ Inertial Measurement Units (IMU) on wrist, chest and ankle respectively, measuring $M = 10$ features including temperature, 3D acceleration, gyroscope and magnetometer. The change-points are labeled as the transitions between activities.

To account for the transitions between activities, the independent changes could possibly include the rising of temperature and the correlation changes could be from the switch of different moving patterns between wrist, chest and ankle.

The data was sample every 0.01 second over totally 10 hours. In the pre-processing, we down-sample the time-series by 20 time steps and then slice them into windows of a fixed length $T = 100$ steps. Each window contains exactly one transition from range $[25, 75]$. There are totally 184 multivariate time series with change-points: 150 of them are used as training, 14 are used as validation and 20 are used as testing.

The results are shown in Table 4.4. Our CORD_CPD achieves the best performance among the 6 statistical and deep learning baselines. We attribute the enhanced performance to the ability of CORD_CPD to better model the two types of changes and to successfully ensemble them. In real life scenarios, a change-point could arise from a mixture of independent change and correlation change. The experiment results show that explicitly modeling both types of changes injects a positive inductive bias during learning, and thus enhances the performance of CPD tasks.

4.6 Conclusion

In this chapter, we study the problem of change point detection (CPD) in multivariate time series. We propose CORD_CPD, which explicitly leverages **model-generated correlation structures** by integrating graph neural networks within an encoder-decoder framework. As a result, CORD_CPD dynamically infers evolving relationships among variables, enabling to detect change-points more accurately without any labeled data. Extensive experiments on a physics simulation dataset and real-world PAMAP2 demonstrate that CORD_CPD effectively detects change-point in multivariate time series, and outperforms competitive statistical and deep learning baselines. This highlights the advantage of leveraging model-generated signals for robust CPD.

Part III

Self-enhancement Methods for Large Language Model

Chapter 5

Domain-specific Chatbot Training via Knowledge Mining and Digest

In this chapter, we explore the use of Large Language Models (LLMs) to autonomously synthesize conversational data from textual documents, such as those from Wikipedia, for domain-specific chatbot training. While vast amounts of domain-specific documents exist, they cannot be directly used for chatbot training, which requires structured conversational data. Effectively transforming this unstructured knowledge into an instructional format remains a challenge. To address this, we propose a methodology that leverages **model-generated training signals** to extract relevant domain-specific knowledge and reformat it into structured dialogue. This approach enables chatbot training with minimal human supervision, bridging the gap between raw textual data and structured conversational training sets. Experimental results demonstrate that our chatbot, trained on LLM-mined instructional data, significantly outperforms baseline models that do not benefit from such model-generated augmentation.

Highlights We pioneer an alignment technique that autonomously generates instructional data from domain-specific corpora, improving chatbot performance across four domains while reducing reliance on manual annotation.

Model-Generated Signals By employing a self-aligned LLM to generate structured training data and further fine-tuning the chatbot on this augmented knowledge base, we demonstrate the power of model-generated synthetic conversation in enhancing chatbot in specific domains.

5.1 Introduction

Large Language Models (LLMs) have significantly advanced the fields of natural language understanding and generation. Despite these advancements, training LLMs as dynamic AI agents capable of answering complex, domain-specific, and knowledge-intensive questions remains a significant challenge [1, 45]. Current methods to integrate knowledge into LLMs primarily involve either modifying model weights with manually curated knowledge [31, 51, 105, 106] or continuous pre-training with a language modeling objective on domain-specific texts [62, 66, 113]. However, these methods often fall short in addressing complex queries that extend beyond simple fact retrieval [173], and they tend to rely on the mere memorization [135, 176] of factual content, which proves insufficient for delivering insightful responses to specialized questions.

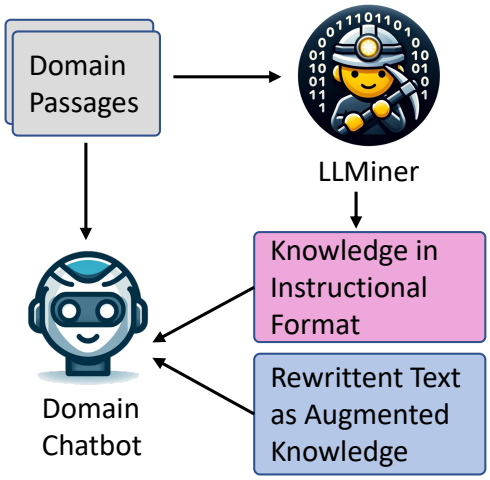


Figure 5.1: Illustration of the LLMiner approach for enhancing Large Language Models (LLMs) as domain-specific chatbots.

This chapter introduces a novel approach for empowering LLMs with domain-specific expertise through an autonomous knowledge mining process (Fig. 5.1). We propose LLMiner, an LLM agent that autonomously extracts knowledge directly from domain-specific texts and structures it in conversational format. During inference, LLMiner first analyzes the importance of the sentence within a given context, and then employs this analysis in a chain-of-thought reasoning process to generate pertinent questions. Subsequently, it structures answers by synthesizing information from both the original text and the formulated questions. This procedure is repeated for each sentence in the document, thereby creating a multifaceted knowledge in instructional formats from the raw text. Unlike previous work largely depends on the LLM's inherent ability to structure knowledge from raw text, LLMiner offloads this knowledge structuring process to the training phase. We feed in multifaceted knowledge as training examples to enable LLMs to develop a more comprehensive understanding of domain-specific content.

Our contributions are highlighted in the following:

1. We develop an LLM-based system for automatic knowledge mining, harnessing the model's

inherent natural language processing capabilities to extract and organize domain-specific knowledge.

2. The data mined by LLMiner offers a multi-perspective view of document content, facilitating a deeper comprehension of the original content.
3. Our method demonstrates a pathway for LLM self-improvement using model-synthesized training data, enabling continuous adaptation and growth in line with the expanding landscape of domain knowledge.

5.2 Related Work

5.2.1 Question Generation

Automatic question generation has been introduced to synthesize data for tasks such as question answering [4, 53, 54, 79, 109], retrieval tasks [33] or dialogue systems [32, 137]. However, previous works use documents or specific spans within them as answer without re-writing. Furthermore, they require labeled QA data to train question generation. Models such PAQ requires training each model for each step in the generation pipeline, while our methodology harnesses a single LLM throughout the generation procedure.

5.2.2 Instruction-tuning

Recent endeavors to enhance LLMs have revolved around instruction tuning in QA [128, 130, 174] or conversation style [25, 76]. These efforts predominantly aim at aligning models with human preferences rather than bolstering knowledge acquisition, which requires large amount of knowledge-intensive quality QA data. Our chapter presents such an effort in extracting high quality QA data from raw text as supervised data to encode knowledge into LLMs.

5.2.3 Chain-of-thought Reasoning

Chain-of-thought reasoning methods are proposed to improve the generation quality in various tasks [139]. Techniques like ReciteLM [125] have been introduced, enabling LLMs to recite pertinent information during question answering (QA). However, these methodologies are tethered to the necessity of finely crafted in-context examples and exhibit shortcomings in incorporating external knowledge efficiently [125].

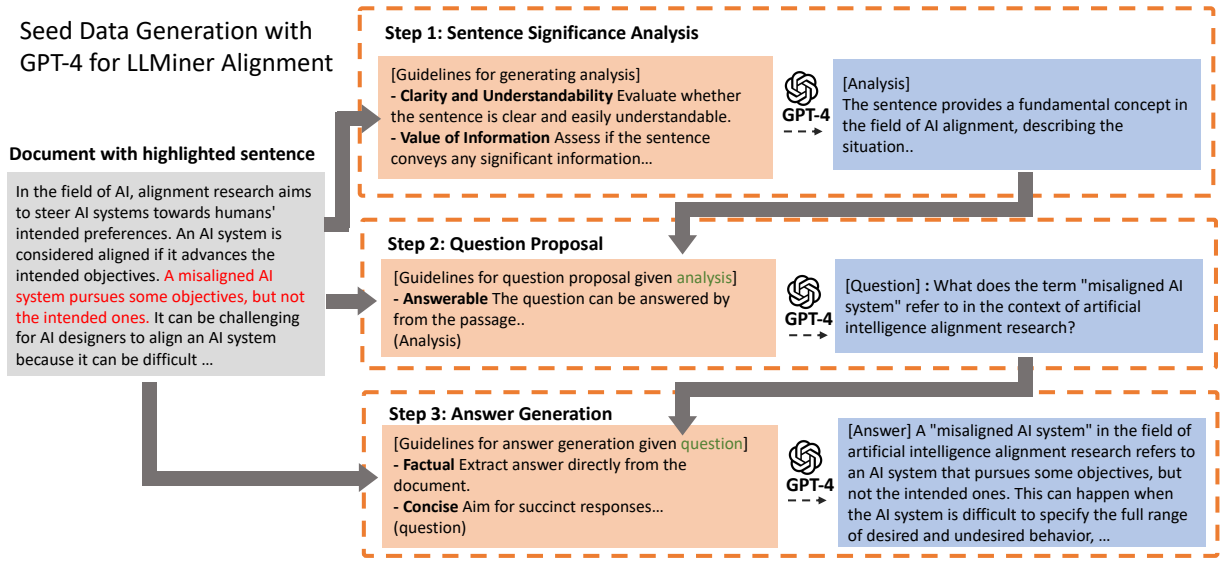


Figure 5.2: LLMiner Training Data Collection: Given a passage with a randomly selected sentence within it, we prompt GPT-4 with specific generation guidelines to analyze, question, and answer about the emphasized sentence in relation to the passage. The seed data is then applied for behavior cloning, fine-tuning a smaller LLM to mimic GPT-4's responses with the efficiency of shorter prompts.

5.3 Methodology

In this section, we outline our approach for training a Large Language Model (LLM) into a specialized knowledge miner, which we refer to as LLMiner.

5.3.1 Preparing Seed Data with GPT-4

Our aim in training LLMiner is to enable it for versatile knowledge mining scenarios, capitalizing on the robust language understanding abilities inherent in pre-trained LLMs. To this end, we compile a diverse corpus that includes 300 randomly sampled passages from Wikipedia, along with 50 each from TREC-COVID, NFCorpus, ArguAna, FEVER, DBPedia, and SCIDOCS [131], making for a total of 600 instances.

As illustrated in Figure 5.2, for each passage, we start by randomly selecting a sentence to serve as the focal point. We then employ three prompts to generate an analysis, a question, and an answer, respectively. In each generation process, we adhere to a set of guidelines, along with a specific formatting approach as input for GPT-4 [110]. The full list of prompts is included in

Section 10.1.

Significance Analysis We prompt GPT-4 to assess the significance of the selected sentence within the broader context of the entire passage. The guidelines for this analysis include factors such as clarity, importance, knowledge addition, and relevance to the overall content. The analysis is a chain-of-thought reasoning for organizing knowledge in the paragraph with the picked sentence as a focal point, serving as the initial step for question proposal. If the sentence is deemed unclear or uninformative, GPT-4 is required to output the specific guideline that has not been met, and we then bypass the subsequent steps for such sentences. These exceptions are used as training data to refine the model's filtering capabilities.

Question Proposal If the sentence is determined to be significant, GPT-4 will be subsequently prompted to formulate questions aimed at encapsulating the insights obtained from the analysis. The guidelines specify that questions should be self-contained, answerable, and insightful. Given that the final deliverable from LLMiner consists solely of QA pairs without accompanying passage context, it is crucial that the questions are formulated to be self-contained. To ensure this, our prompts direct the model to avoid using context-dependent phrases such as "from the document" or "in the report."

Answer Generation Lastly, GPT-4 proceeds to generate an answer to the previously formulated question. This phase engages the model in a reading comprehension task, adhering to guidelines that ensure the answer is factual, concise, and self-contained. By rephrasing or summarizing the information present in the original sentence, the model effectively reorganizes the information in the passage that aligns with the perspective of the generated question.

By following this three-step procedure, we gather a set of 600 training instances from GPT-4 including the analysis, question and answer, which will be used to align an open-source LLM for the knowledge mining task. GPT-4 substitutes for human labeling efforts in this context.

5.3.2 Training for Model Alignment

Utilizing GPT-4 for the above knowledge mining comes with two primary drawbacks:

1. The computational overhead for large-scale inference is significant.
2. Each generative step necessitates detailed prompts that outline the guidelines for generating appropriate output.

To address these issues, we employ a secondary, smaller LLM (LLaMa-7b) that we fine-tune to align with the GPT-4-generated seed data. The fine-tuning employs a simplified prompt (detailed in Section 10.1), with a document and a highlighted sentence as input. The model outputs either an analysis or an analysis followed by a QA pair, depending on whether the sentence is important. This process effectively clones GPT-4's responses using more straightforward prompts. The fine-tuning serves as a behavior cloning process to mimic the responses generated by GPT-4 using more concise and straightforward prompts.

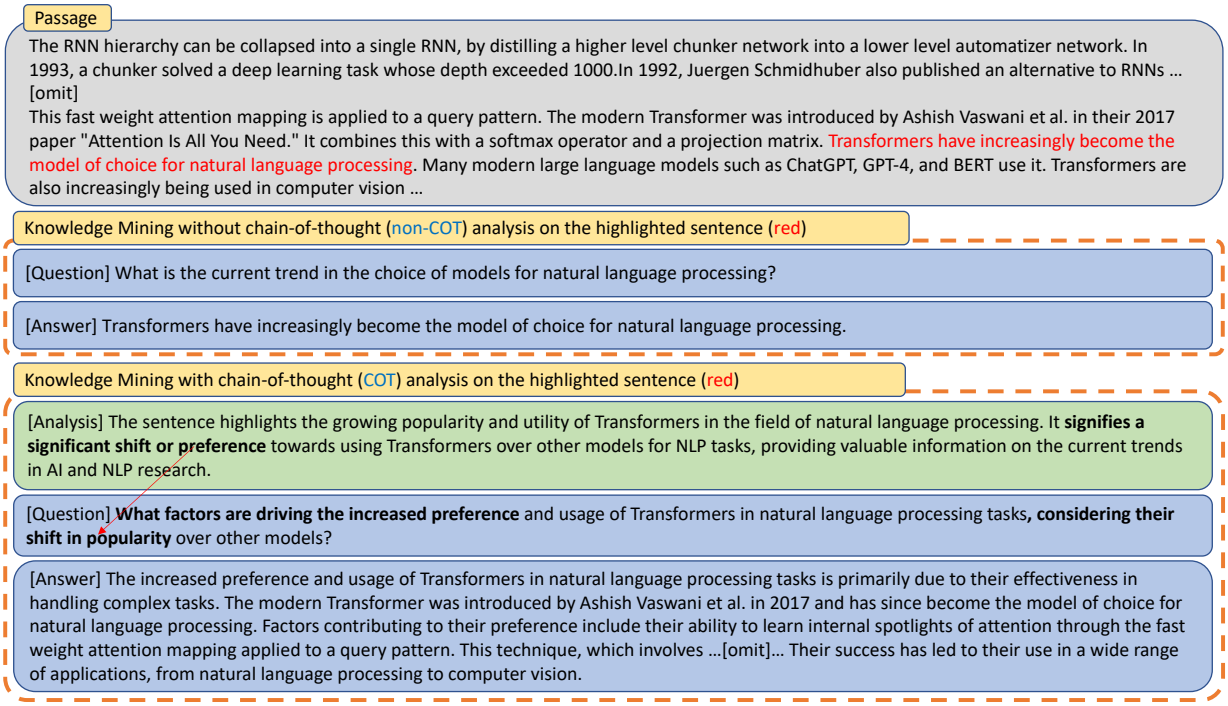


Figure 5.3: Example illustrating the utility of chain-of-thought (COT) analysis in extracting comprehensive knowledge from a passage. Without COT, the question can be directly answered from highlighted sentence, COT facilitates a deeper understanding by formulating context-rich questions referring to the analysis and synthesizing answers that requires an understanding the entire passage.

5.3.3 Inference with LLMiner

During the inference stage, we process each incoming passage by iterating through all its sentences, treating each one as a "highlighted sentence" for the model to focus on. We use the same prompt as in the training phase of LLMiner for inference with each highlighted sen-

tence within the passage. We found (Fig. 5.3) generating analysis as a chain-of-thought step improves the quality of proposed question, and thus gives better in-domain instructional data for subsequent model fine-tuning.

5.4 Methodology

In this section, we outline our approach for training a Large Language Model (LLM) into a specialized knowledge miner, which we refer to as LLMiner.

5.4.1 Preparing Seed Data with GPT-4

Our aim in training LLMiner is to enable it for versatile knowledge mining scenarios, capitalizing on the robust language understanding abilities inherent in pre-trained LLMs. To this end, we compile a diverse corpus that includes 300 randomly sampled passages from Wikipedia, along with 50 each from TREC-COVID, NFCorpus, ArguAna, FEVER, DBPedia, and SCIDOCS [131], making for a total of 600 instances.

As illustrated in Fig. 5.2, for each passage, we start by randomly selecting a sentence to serve as the focal point. We then employ three prompts to generate an analysis, a question, and an answer, respectively. In each generation process, we adhere to a set of guidelines, along with a specific formatting approach as input for GPT-4 [110]. The full list of prompts is included in **??**.

5.4.2 Training for Model Alignment

Utilizing GPT-4 for the above knowledge mining comes with two primary drawbacks:

1. The computational overhead for large-scale inference is significant.
2. Each generative step necessitates detailed prompts that outline the guidelines for generating appropriate output.

To address these issues, we employ a secondary, smaller LLM (LLaMa-7b) that we fine-tune to align with the GPT-4-generated seed data. The fine-tuning employs a simplified prompt (detailed in Section 10.1), with a document and a highlighted sentence as input. The model outputs either an analysis or an analysis followed by a QA pair, depending on whether the sentence is important. This process effectively clones GPT-4's responses using more straightforward prompts. The fine-tuning serves as a behavior cloning process to mimic the responses generated by GPT-4 using more concise and straightforward prompts.

5.4.3 Inference with LLMiner

During the inference stage, we process each incoming passage by iterating through all its sentences, treating each one as a "highlighted sentence" for the model to focus on. We use the same prompt as in the training phase of LLMiner for inference with each highlighted sentence within the passage. We found (Fig. 5.3) generating analysis as a chain-of-thought step improves the quality of proposed question, and thus gives better in-domain instructional data for subsequent model fine-tuning.

5.5 Domain-Specific Text Collection

This section outlines the methodologies for gathering domain-specific text and crafting test data. The objective is to facilitate the development of a specialized chatbot, tailored to user-defined topics such as "Artificial General Intelligence" or "Traditional Medicine in Southeast Asia." We delineate the processes for constructing a domain-specific corpus and formulating testing datasets.

5.5.1 Construction of a Domain-Specific Corpus

We employ two open-source tools, Wikipedia-API and GPT-4, to assemble a domain-specific corpus based on user-defined topics.

Initial Topic Expansion As a first step, we utilize GPT-4 to generate hypothetical Wikipedia titles related to the user-specified topic, as detailed in Fig. 10.5. These titles, potentially non-existent on Wikipedia, act as initial search queries for the Wikipedia-API. Users may also contribute additional keywords to enhance these seed titles.

Wikipedia-Based Retrieval and Selection With the generated hypothetical titles and user-provided keywords in hand, we utilize the Wikipedia-API to retrieve actual Wikipedia titles. Subsequently, GPT-4 is used to assess relevance, following the prompt specified in Fig. 10.6. Articles corresponding to the relevant titles are incorporated into our domain-specific text corpus.

	Artificial General Intelligence	Traditional Medicine in Southeast Asia	History of Steam Engine	Advances in Robotics Surgery
OASST	77.14	66.65	70.6	70.96
OASST + passage	78.00	68.22	73.2	71.56
OASST + passage + QA	80.67	68.92	79.8	**75.63**
OASST + passage + QA + Aug	**82.38**	**70.47**	**83.2**	74.96

Table 5.1: Performance comparison of different training setting for chatbot on the domain-specific texts. The numbers are normalized scores from GPT-4 evaluation.

5.5.2 Testing Dataset Creation

This subsection elaborates on the approach to simulate real-world user interactions with the chatbot and the process of testing dataset creation, focusing on a scenario where the trainer customizes a chatbot for a specific topic, and users interact with it for inquiries within that domain.

Simulating User Queries To mimic user interactions, we engage annotators to propose questions that reflect potential user queries within the chosen topic. We aim to compile around 30 human-crafted questions per topic. These questions are designed to be factually based, targeting the chatbot's knowledge in the domain text. In addition, annotators are responsible for creating concise reference answers, typically one or two sentences long, for each question.

Independence from Training Corpus Annotators are allowed to utilize external resources like Google or ChatGPT to ensure the accuracy and relevance of their questions and answers. However, it's crucial to note that they do not have access to the actual text corpus or augmented data compiled by the chatbot trainer. This constraint ensures that the testing questions and responses are independent of the chatbot's training data, thereby allowing for an unbiased evaluation of the chatbot's performance in responding to unanticipated queries.

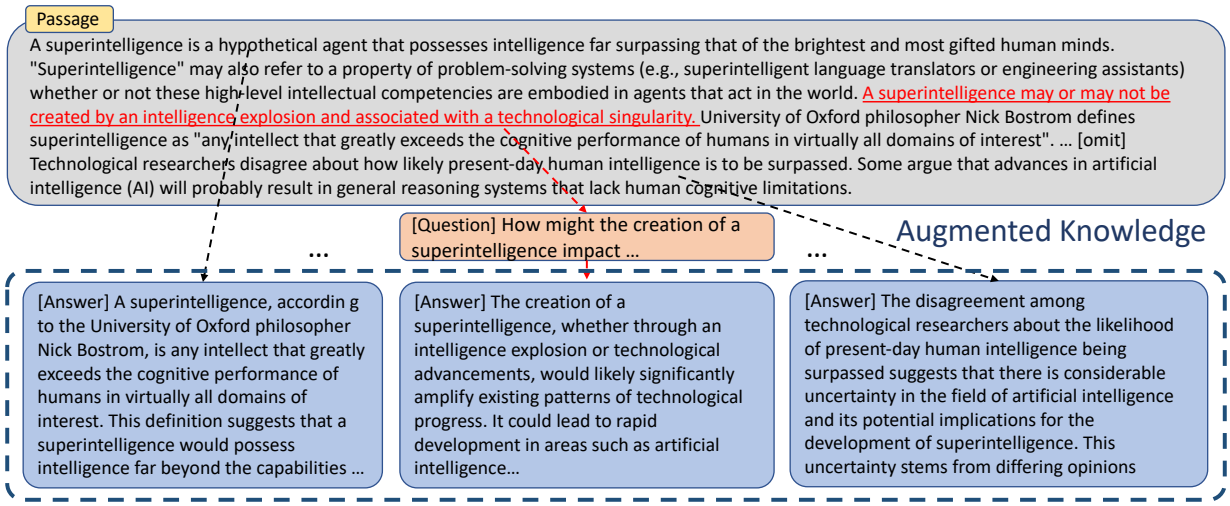


Figure 5.4: Knowledge Augmentation of the Original Passage: The collection of minded answers serves as augmented knowledge with each emphasizes specific facets determined by the highlighted sentence, enriching the understanding of the original passage.

5.6 Chatbot Evaluations

5.6.1 Training and Baselines

We consider a mixed training of 4 different types of data:

Domain Passage (Passage): This consists of the domain-specific text that we have gathered from Wikipedia, serving as a foundational source for domain expertise.

Conversation Data (OASST): We leverage the OpenAssistant Conversations Dataset (OASST) [76] as a source of dialog interactions to train model for instruction-following ability. To ensure quality, we only include entries with an overall score greater than 0.5, resulting in approximately 17k conversations.

Mined QA (QA): These are the mined data in conversational format using the LLMiner for in-domain instruction finetuning.

Augmented Knowledge (Aug): This consists of the list of mined answers as standalone elements to serve as augmented knowledge providing variations of original text (Fig. 5.4).

Since the mixing ratio of these different data types is not the primary focus of the chapter, we employ uniform sampling for the data types as our design choice.

5.6.2 Evaluation

We employ GPT-4 as a judge to rate the quality of the generated answers. The evaluations are conducted using a Likert scale ranging from 1 to 5, with prompts specified in Appendix **??**. We report the normalized score from GPT-4 eval in Table 5.1.

In the evaluation, we observe that the incorporation of QA data from LLMiner can help learning domain knowledge from raw text corpus, which enhances the quality of chatbot to answer factual-based questions. Furthermore, mixing additional the training data with augmented knowledge (rewritten version of original passage by the list of answers) offers more perspectives of the original text, which improves the model's performance across three of the four topics assessed.

5.7 Conclusion

In this chapter, we address the challenge of chatbot training in domains where no structured conversational data is available. We propose leveraging a LLM to mine conversational data from domain-specific text corpora. By autonomously generating question-answer pairs and enriching domain knowledge, LLMiner enhances training data for chatbot adaptation across multiple topics. Experimental evaluations demonstrate significant improvements in chatbot performance. Our findings highlight the potential of LLMs as effective domain-specific knowledge miners and showcase the benefits of autonomous data mining for self-improving LLM training.

Chapter 6

Video Large Language Model Training with Synthetic Data

In this chapter, we explore the use of large language models (LLMs) to generate conversation data for video-based large language models (Video LLMs), extending LLMiner to multimodal scenarios. Prior to our work on LLaVA-Hound-DPO, Video LLMs were primarily pre-trained with short video captions and human-annotated conversation data. However, these sources present limitations: video captions are often noisy, while human-labeled conversations are scarce. To address this, we propose leveraging model-generated training signals to synthesize large-scale video captions and instruction-following dialogue data. Specifically, we generate detailed video descriptions using GPT-4V and multimodal conversations using LLMs, providing richer and more scalable supervision.

Highlights

- We introduce large-scale synthetic video captions and conversations to improve Video LLM alignment.
- We are the first to apply direct preference optimization (DPO) as a reinforcement learning (RL) approach for alignment in multimodal context.

Model-Generated Signals Our model-generated signals include:

- GPT-4V-generated detailed video captions, enhancing visual grounding.
- LLM-generated multimodal conversations, improving instruction-following and coherence.

6.1 Introduction

This section addresses the challenge of aligning LMMs, particularly in tasks that involve video instruction following. Despite recent advancements in reinforcement learning (RL) [9, 78, 111, 127] and DPO [24, 56, 115], which have been effective in guiding LLMs towards generating more honest, helpful, and harmless content, their effectiveness in video domain remains limited. The critical obstacle lies in developing a robust reward system capable of distinguishing preferred responses from less preferred ones based on video inputs. The challenge is further complicated by the coverage and potential inaccuracies in generated content, stemming from the scarcity of alignment data across different modalities [90, 126].

While human preference data is valuable, it is challenging to scale due to its cost and labor-intensive nature, as highlighted by the LLaVA-RLHF [126] paper, which collected 10k human-evaluated instances at a considerable cost of $3000. Existing approaches for distilling preferences, such as those for image data using GPT-4V [83], encounter scalability issues, especially for video inputs that require analyzing multiple frames. While [3] leverage a supervised fine-tuning (SFT) model for self-evaluation, the efficacy of the SFT model remains uncertain, particularly in accurately assessing the factuality of responses in relation to their corresponding videos.

To tackle the aforementioned challenges, we introduce a cost-effective reward mechanism that is both computationally and financially efficient for evaluating the quality of responses generated by video LLMs, serving as a basis for further on-policy preference optimization. We propose the use of detailed video captions as a proxy for video content, enabling a language model analyze the content and assess the quality of an LMM's response to related questions. The language model generates natural language feedback as a chain-of-thought step, and produces a numerical score as the reward, thereby creating an efficient feedback system.

However, high-quality video captions are essential for this process. To mitigate the shortage of high-quality video captions, we have developed a comprehensive video caption dataset, ShareGPTVideo, using a simple prompting technique with the GPT-4V model, comprising 900k captions that encompass a wide range of video content, including temporal dynamics, world knowledge, object attributes, and spatial relationships. With this video caption dataset available, we verify that our reward mechanism, which utilizes video captions as a proxy, is well-aligned with evaluations derived from the more powerful, albeit costlier, GPT-4V model-generated rewards. Employing this reward mechanism as the basis for DPO algorithm, we train LLaVA-Hound-DPO that achieves an 8.1% accuracy improvement over the SFT counter-

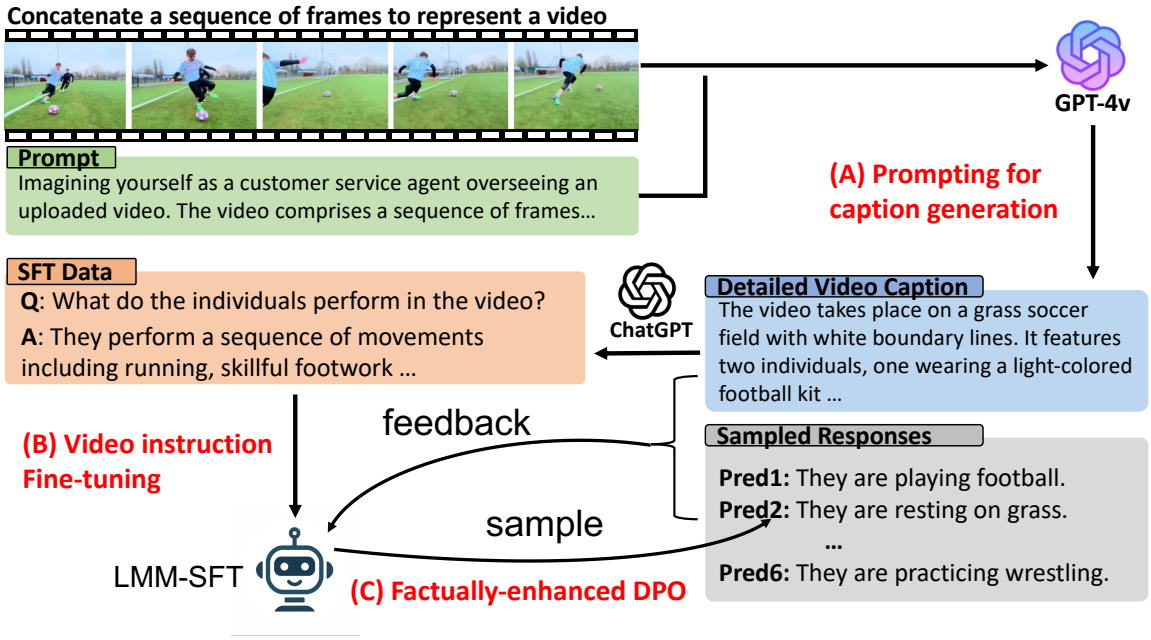


Figure 6.1: Workflow diagram showing: a) the use of GPT-4V for creating a detailed caption dataset for videos; b) generating video instruction data for SFT; c) integrating captions into a feedback loop for DPO, improving the model's performance on video instruction-following tasks.

part. This marks a significant advancement in video LMM alignment and represents the first successful application of a DPO method in this domain.

6.2 Method

As shown in Fig. 6.1, our methodology enhances video LMM alignment through DPO method using rewards from a language model. We elaborate on constructing a video caption dataset in Section 6.2.1. Subsequently, in Section 6.2.2, we discuss the generation of video instruction data and the fine-tuning process of our model. Lastly, Section 6.2.3 details the incorporation of generated captions as a feedback mechanism for DPO method to refine our model's factual alignment in video instruction-following tasks.

6.2.1 Prompting GPT-4V Model for Detailed Video Caption Distillation

The selection of dataset includes videos from three sources: WebVid (400k) and VIDAL (450k) ActivityNet (50k) datasets. WebVid and VIDAL videos are in the general domain sourced from YouTube, and ActivityNet videos focus on human activities. The three datasets together result in a comprehensive collection of 900k videos. To accommodate the requirement that GPT-4V only takes images as input, we preprocess videos by uniformly extracting ten frames per video content. These frames are then concatenated into a sequence to serve as a proxy for the video. We use GPT-4V to generate a coherent caption for the represented video based on the frame sequence. The prompt (Fig. 11.12) adheres to guidelines covering temporal dynamics, world knowledge, object attributes, spatial relationships, aesthetic assessments, etc., with the goal of comprehensively understanding the video contents (examples in ??).

6.2.2 SFT with Generated Video Instruction Data from Detailed Caption

To generate video instruction-following data for SFT, we adopt a similar methodology outlined in Video-ChatGPT [81]. Specifically, we first randomly sample 300k video captions and then employ ChatGPT to generate 3 question-answer pairs conditioned on each caption (prompt in Fig. 11.13). We release the 900k instruction-following data to public, but we only use a random subset of 240k for our training. This approach ensures that the instructional data remains factually consistent with the content of the detailed captions.

6.2.3 DPO with Language Model Reward

Acquiring high-quality on-policy preference data can be costly and labor-intensive. Although GPT-4V can be used for reward distillation, for video data, its high computation cost[1], slow response, and limited accessibility hinder scalability. We propose a cost-efficient method to generate reward data for DPO using detailed video captions as supporting evidence, as shown in Fig. 6.2.

Initially, we randomly select a subset of 20k instruction pairs from the dataset described in Section 6.2.2. The SFT model generates six responses per input at a temperature of 1.0. This

[1]Video representation is typically encoded with 2048 tokens, while our captions only uses roughly 140 tokens.

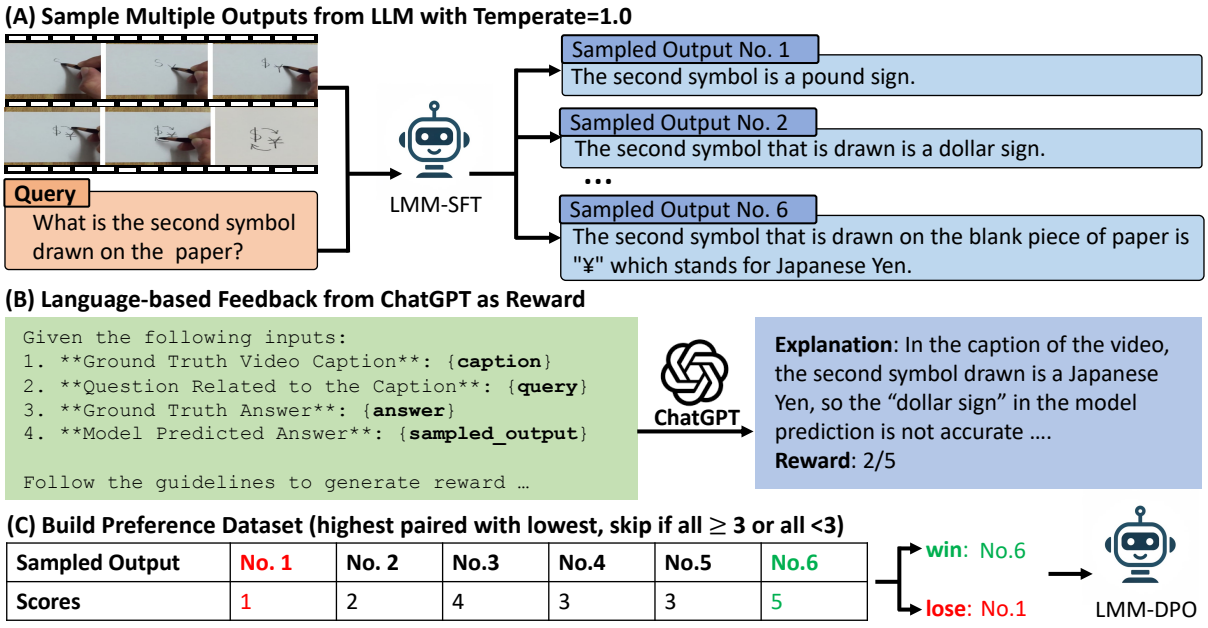


Figure 6.2: Detailed illustration of the proposed factually-enhanced DPO method.

procedure results in 120k question-answer pairs. Subsequently, we employ ChatGPT to evaluate the model responses based on the ground truth answer and detailed description (prompt in Fig. 11.15). ChatGPT generates an output that includes a natural language explanation as chain-of-thought step, followed by a numerical reward score on a scale from 1 to 5, indicating the overall quality.

For each video and question pair, we randomly select an answer with a score ≥ 3 as positive example, and an answer with a score below 3 as negative. Cases where all responses are uniformly scored above or below 3 are excluded from the dataset. After the selection process, approximately 17k training instances are compiled for DPO training. Formally, the dataset is denoted as $\mathcal{D}_{DPO} = \{(\mathcal{V}, x, y_w, y_l)\}$, where $\mathcal{V}$ is the video, x is the question, y_w and y_l are the positive and negative responses. The DPO objective is defined as below:

$$\mathcal{L}_{\text{DPO}}\left(\pi_\theta; \pi_{\text{ref}}\right) = -\mathbb{E}_{(\mathcal{V},x,y_w,y_l)\sim\mathcal{D}_{DPO}}\Bigg[\log\sigma\left(\beta\log\frac{\pi_\theta\left(y_w \mid x,\mathcal{V}\right)}{\pi_{\text{ref}}\left(y_w \mid x,\mathcal{V}\right)} - \beta\log\frac{\pi_\theta\left(y_l \mid x,\mathcal{V}\right)}{\pi_{\text{ref}}\left(y_l \mid x,\mathcal{V}\right)}\right)\Bigg],$$

where π_θ is the policy model to be optimized and π_{ref} is the base reference model, both models are initialized with SFT weights. σ is the logistic function and β is set to 0.1.

For on-policy reward generation, our method incurs a cost of less than $20, under a pricing

model of $1.5 per million tokens. In comparison, previous methods of preference data collection, such as in [126], required an expenditure of $3,000 to gather 10k human preference data points. Additionally, the method proposed by [83], which employs GPT-4V for reward data labeling, incurs a significantly higher cost—$30 per million tokens—and demonstrates considerably slower inference speeds.

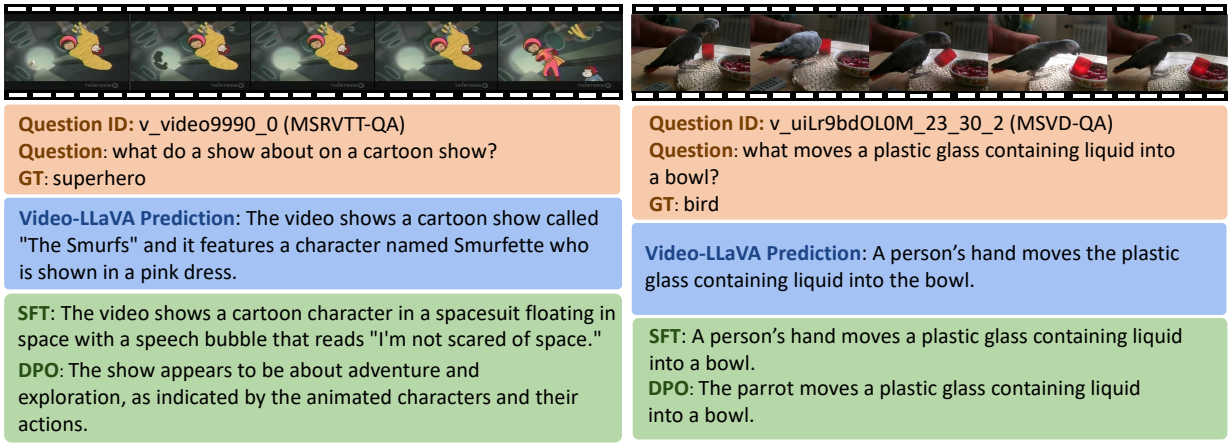


Figure 6.3: Examples from MSRVTT-QA and MSVD-QA showcase that our LLaVA-Hound-DPO generates better responses, and reveal key limitations of the existing benchmark evaluation.

6.3 Experiments

We adopt Video-LLaVA [87] as the backbone of our video LMM, but our method can be applied to any other architectures as well.

Caption Pre-training Stage (LLaVA-Hound-PT): We use captioning data including 650k image caption data from ALLaVA [19] and our distilled 900k video caption. We freeze the visual encoder and fine-tune the MLP projector and LLM, with learning rate 2e-5 and batch size 128.

SFT Stage (LLaVA-Hound-SFT): We use 600k image instruction data from ALLaVA and our generated 240k video instruction data, with learning rate 5e-6 and batch size 128.

DPO training Stage (LLaVA-Hound-DPO): We use the 17k preference data introduced in Section 6.2.3 for DPO training. Following [58], we train our policy model with full model training for 3 epochs with learning rate 5e-7, and a batch size of 128. All the experiments are performed on 8 A100 gpus.

Methods	LLM Size	Existing Video QA Benchmark from [100]					
		MSVD-QA		MSRVTT-QA		TGIF-QA	
		Acc.	Score	Acc.	Score	Acc.	Score
FrozenBiLM [149]∗	1B	32.2	-	16.8	-	41.0	-
VideoLLaMA [160]∗	7B	51.6	2.5	29.6	1.8	-	-
LLaMA-Adapter [163]∗	7B	54.9	3.1	43.8	2.7	-	-
VideoChat [81]∗	7B	56.3	2.8	45.0	2.5	34.4	2.3
BT-Adapter [93]∗	7B	67.5	3.7	57.0	3.2	-	-
Video-ChatGPT [100]	7B	68.6	3.8	58.9	3.4	47.8	3.2
Chat-UniVi [65]	7B	70.0	3.8	53.1	3.1	46.1	3.1
VideoChat2 [82]	7B	70.0	3.9	54.1	3.3	-	-
Video-LLaVA [88]	7B	71.8	3.9	59.0	3.4	48.4	3.2
LLaMA-VID [86]	7B	72.6	3.9	58.7	3.4	49.2	3.3
LLaMA-VID [86]	13B	74.3	4.0	59.8	3.4	50.8	3.3
VLM-RLAIF [3]∗	7B	76.4	4.0	63.0	3.4	-	-
LLaVA-Hound-SFT	7B	75.7	3.9	58.7	3.3	53.5	3.3
LLaVA-Hound-DPO	7B	**80.7**	**4.1**	**70.2**	**3.7**	**61.4**	**3.5**

Table 6.1: **Evaluation of Model Performance on Zero-Shot Video Question Answering Benchmarks Using gpt-3.5-turbo-0613.** Models denoted with ∗ have their results directly sourced from their original publications. Caution is advised when interpreting these results; see Appendix 11.1 for an in-depth analysis of evaluation challenges. All other baseline models were reproduced by our team.

6.3.1 Benchmark Evaluation

Dataset and Testing Environment We evaluate model performance on four benchmark datasets: MSVD-QA [18], MSRVTT-QA [144], TGIF-QA [63], and Next-QA [141] using ChatGPT with version gpt-3.5-turbo-0611 to assess model predictions. The evaluation prompts follow [100]. In our experiment, we found that different ChatGPT versions have high impact on absolute score of metric, but the overall ranking of models is relatively stable. We select gpt-3.5-turbo-0613 due to its closeness to the reported score in Video-LLaVA paper. Further details on the selection rationale and evaluation pitfalls are discussed in Appendix 11.1.

Baseline Selection We select video LMM models that have demonstrated SOTA performance with with accessible code and checkpoints at the time of paper writing, specifically including Video-LLaVA, which is also our choice of architecture. We replicate results including Video-ChatGPT [100], LLaMA-VID [86] (7B and 13B), Chat-UniVi [65], and Video-LLaVA [88]. We copy the results from additional baselines including FrozenBiLM [149], VideoChat [81] and VideoLLaMA [160], sourced from their original publication.

Results In Table 6.1, our analysis shows that within the SFT models, LLaMA-VID-7B and Video-LLaVA exhibit comparable performance, with LLaMA-VID-13B performing the best. Our LLaVA-Hound-SFT model achieves comparable performance to LLaMA-VID-13B. Incorporating preference modeling, LLaVA-Hound-DPO achieves an average accuracy of 70.75%, surpassing LLaVA-Hound-SFT, which has an average accuracy of 62.65%, by 8.1%. Furthermore, LLaVA-Hound-DPO exhibits superior accuracy compared to other RL methods such as VLM-RLAIF. Our model demonstrated consistent result on a relative new benchmark Next-QA.

Error Analysis Figure 6.3 illustrates two examples. In the left example, LLaVA-Hound-SFT provides an accurate description of the video's first half but introduces a hallucination with the phrase "I'm not scared of space," absent in the video content. LLaVA-Hound-DPO yields a more accurate inference. In the right example, both LLaVA-Hound-SFT and Video-LLaVA models produce incorrect inferences, whereas LLaVA-Hound-DPO successfully correctly identifies the subject in the video.

6.3.2 Open-ended QA Analysis

In this section, we conduct analysis on open-ended long-form QA with a proposed development benchmark. Specifically, we select 2,000 videos from each source: WebVid [10], VIDAL [175], ActivityNet [38], MSRVTT [144], MSVD [18], TGIF [63], and Something-something V2 (SSV2) [48]. For each video, ChatGPT was utilized to generate three QA pairs based on the detailed captions, and we evaluate model predictions with our reward mechanism. WebVid, VIDAL, ActivityNet are classified as in-domain, which are involved in the model's training pipeline. MSRVTT, MSVD, TGIF, SSV2 are classified as out-of-domain.

The evaluation reveals insights into (1) the quality of long-form open-ended QA, (2) in-domain and out-of-domain generalization, and (3) Ablations on SFT and DPO experiments. Additionally, we select our best performing model on the development bench before evaluating

No.	Methods	Proposed Video QA Benchmark (In-domain)					
		ActivityNet-QA		VIDAL-QA		WebVid-QA	
		Acc.	Score	Acc.	Score	Acc.	Score
[1]	Video-ChatGPT [100]	34.17	2.19	29.35	2.10	38.88	2.27
[2]	LLaMA-VID-7B [86]	36.54	2.27	30.58	2.15	36.99	2.24
[3]	LLaMA-VID-13B [86]	37.33	2.29	32.50	2.18	39.73	2.30
[4]	Chat-UniVi [65]	39.35	2.32	31.40	2.16	40.05	2.31
[5]	Video-LLaVA [88]	41.35	2.38	34.30	2.24	42.47	2.39
[6]	LLaVA-Hound-SFT	66.62	3.05	60.50	2.88	71.07	3.17
[7]	LLaVA-Hound-DPO	**76.62**	**3.18**	**70.06**	**3.04**	**79.82**	**3.29**
[8]	LLaVA-Hound-PT + Image Inst.	69.31	3.09	60.57	2.85	68.03	3.02
[9]	LLaVA-Hound-PT + VChat	67.34	3.02	62.33	2.89	68.98	3.00
[10]	LLaVA-Hound-DPO + training MLP	71.89	3.10	65.57	2.95	75.37	3.21
[11]	LLaVA-Hound-SFT + Self-play	64.11	2.85	56.28	2.68	67.89	2.95
[12]	LLaVA-Hound-DPO w/ lr3e-7	71.13	3.08	64.90	2.92	73.25	3.17

Table 6.2: Our proposed video QA benchmark evaluation on in-domain dataset using gpt-3.5-turbo-0301, with detailed captions as supporting evidence.

on public benchmarks, which avoids tuning hyperparameters on test data. Comparisons are shown in Section 11.4.

Domain Generalization: Table 6.2 and Table 6.3 shows the in-domain and out-of-domain evaluation. SFT with our data tends to perform better both in- and out-of-domain, and DPO further enhances the model performance, showing the effectiveness of preference modeling.

Video LMM without Video Instruction: [8] in Table 6.2 is baseline trained with only image instruction fine-tuned on LLaVA-Hound-PT, which achieves an average accuracy of 65.97%, comparable to the LLaVA-Hound-SFT model's 66.06% in in-domain QA scenarios. However, its performance significantly drops in out-of-domain QA contexts (49.32% vs. 56.50%), suggesting that Video QA training could potentially enhance generalization capabilities.

Quality of Generated SFT: [9] substitutes our generated video QA with the Video-ChatGPT dataset for Video-LLaVA fine-tuning. A comparison between the findings of [9] and [6] reveals a marginal performance disparity of 0.2% in average accuracy, indicating that the quality of our generated QA closely parallels that of the existing video QA datasets. Given the similar quality in SFT data, the large gain of [6] over [5] can be reasonably concluded from large-scale

Methods	Proposed Video QA Benchmark (Out-of-domain)							
	MSVD-QA		MSRVTT-QA		TGIF-QA		SSV2-QA	
	Acc.	Score	Acc.	Score	Acc.	Score	Acc.	Score
Video-ChatGPT [100]	34.06	2.20	25.65	1.98	31.35	2.09	19.36	1.75
LLaMA-VID-7B [86]	34.14	2.21	25.02	1.99	27.18	2.00	22.16	1.84
LLaMA-VID-13B [86]	35.81	2.25	26.34	2.02	27.58	2.01	21.98	1.83
Chat-UniVi [65]	35.61	2.23	25.89	2.01	33.23	2.13	20.59	1.79
Video-LLaVA [88]	39.46	2.37	30.78	2.15	32.95	2.18	24.31	1.90
LLaVA-Hound-SFT	66.99	3.09	57.82	2.85	66.13	3.07	35.07	2.23
LLaVA-Hound-DPO	**73.64**	**3.12**	**68.29**	**2.98**	**74.00**	**3.12**	**48.89**	**2.53**
LLaVA-Hound-PT + Image Inst.	65.19	2.96	48.66	2.52	53.83	2.62	29.60	2.04

Table 6.3: Our proposed video QA benchmark evaluation on out-of-domain dataset using gpt-3.5-turbo-0301, with detailed captions as supporting evidence.

pre-training on video captions.

Unfreeze MLP: The comparison between [10] and [7] reveals a significant decrease in performance when the MLP is unfrozen during DPO training. Despite this drop, however, the performance remains superior to that of the SFT baseline.

Smaller Learning Rate: The comparison between [12] and [7] reveals that using a smaller learning rate of 3e-7 (vs. 5e-7) results in a decreasing of model performance. This highlights the future improvements by finding better hyperparameters.

Self-Play vs. DPO: [24] introduced a self-play methodology for DPO training, which designates ground truth answers as preferred and model-generated responses as dispreferred. When comparing the results of [11] with those in [6], a notable decrease in accuracy by 3% from the SFT model is observed, suggesting that self-play may be less effective for video LMM alignment, and introducing reward model is helpful.

DPO Accuracy vs. Training Epochs. The left of Fig. 6.4 depicts the generalization performance of the model on out-of-domain video QA tasks with respect to the number of training epochs. We observe a consistent enhancement in model performance among datasets during the initial 0 to 2 epochs, with peak performance materializing at around 2.5 epochs, which corresponds to 350 training steps.

DPO as Ranker vs. Generator. Following [56], we compare the performance of employing the DPO model as a ranker for candidate answers produced by the SFT model, operating at a

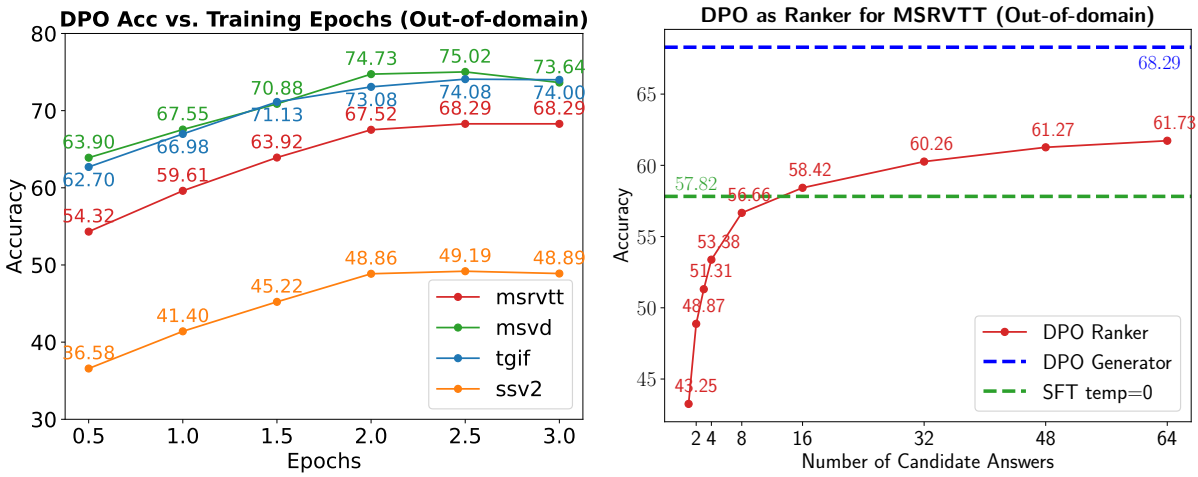


Figure 6.4: The left figure shows the test set accuracy of the DPO model w.r.t the number of training epochs. The right figure shows a comparison of DPO model performance as generator vs. ranker.

temperature setting of 1.0. As depicted on the right in Fig. 6.4, we illustrate the test accuracy progression through the selection of the best among N candidates by the DPO ranker. Initial observations indicate that the SFT model, when set to a temperature of 1.0, demonstrates a reduced accuracy (43.3%) compared to that achieved through greedy decoding (57.8%). A steady enhancement in performance is noted as the number of candidates increases, plateauing at an accuracy of approximately 62% with 64 candidates. This performance, however, falls short when compared with the direct application of the DPO model for answer generation, which yields an accuracy of 68.29%. This difference suggests the stronger generalization of DPO model in answer generation, despite it is trained on a reward classification loss. The contradictory results to [56] may be due to the difference of tasks, i.e. Math vs. Video QA. Refer to Section 11.5 for more results.

6.4 Conclusion

In this chapter, we explored effective methods for leveraging model-generated signals to enhance video LLM alignment. Specifically, we proposed distilling a large amount of high-quality video caption data for pretraining and using LLMs to automatically generate instruction-following data for SFT. Our results demonstrate that scaling up synthetic data leads to improved model performance.

Chapter 7

Improve Vision Language Reasoning with Generated Traces

In this chapter, we study the use of model-generated (self-generated) reasoning traces to enhance the chain-of-thought (CoT) reasoning capabilities of vision-language models (VLMs). While CoT reasoning has been highly effective in mathematical and code-related tasks, its application in visual reasoning remains under-explored. Key challenges include the scarcity of CoT reasoning data and the difficulty of calibrating reasoning quality. To address these issues, we leverage model-generated reasoning traces for reinforcement learning. Our experiments demonstrate that this approach achieves state-of-the-art performance compared to models of similar sizes and, in certain cases, performs on par with proprietary models on specific datasets.

Highlights

- We use large-scale model-generated reasoning traces to improve VLM reasoning.
- We are the first to demonstrate the effectiveness of outcome-based rewards for vision-language reasoning.

Model-Generated Signals Our model-generated signals include VLM-generated reasoning traces, which enhance reasoning capabilities in vision-language tasks.

7.1 Introduction

Chain-of-thought (CoT) reasoning in vision language models (VLMs) is crucial for improving interpretability and trustworthiness. However, current training recipes often relying on datasets

dominated by short annotations with minimal rationales. In this work, we show that training VLM on short answers leads to poor generalization on reasoning tasks that require more detailed explanations. To address this limitation, we propose a two-stage post-training strategy that extends the usage of short answer data for enhanced CoT reasoning. First, we augment short answers with CoT reasoning generated by GPT-4o, enhancing the VLM's CoT capabilities through fine-tuning. Second, we leverage short answers as outcome rewards for reinforcement learning. Specifically, short answers are used as correctness indicators to construct positive (correct) and negative (incorrect) pairs from model-generated reasoning chains. These pairs are then used to calibrate the model's reasoning via Direct Preference Optimization. Our experiments show significant improvements in CoT reasoning on benchmark datasets, along with enhanced generalization to direct answer prediction. This work provides a critical data resource for VLM CoT training and demonstrates the effectiveness of outcome rewards for multimodal models post-training.

7.2 Background and Goal

As VLMs are increasingly applied to more complex tasks, the ability to generate robust CoT reasoning becomes essential for improving interpretability and trustworthiness [8, 20, 80, 89, 90, 91]. However, current training recipes often rely on datasets dominated by short answers with limited rationales, potentially hindering the models' ability to generalize to tasks requiring comprehensive reasoning. In this work, we critically examine the effectiveness of short-answer data for reasoning capabilities and propose augmenting it to enhance CoT reasoning during supervised fine-tuning (SFT) and reinforcement learning (RL).

An example in Fig. 7.1 asks for the number of food items in a bar graph. A human would typically enumerate the bars and then calculate the total. However, writing out this enumeration process is far more cumbersome than simply providing the short answer of "14." Consequently, the annotated training data is predominantly composed of short answers, with minimal rationale provided. This raises a critical research question: *Does training on direct prediction implicitly teach the model to perform chain-of-thought reasoning to derive correct answers?* Our findings indicate that after training on 26k direct predictions from ChartQA, the accuracy of direct predictions increased by 2.9 (70.2 to 73.1), while CoT prediction accuracy improved by only 0.6 points (71.2 to 71.8), with CoT under-performing direct prediction as a result. This suggests that current training approaches have limited effectiveness in enhancing CoT reasoning.

We hypothesize that developing CoT reasoning capabilities requires explicit training on data

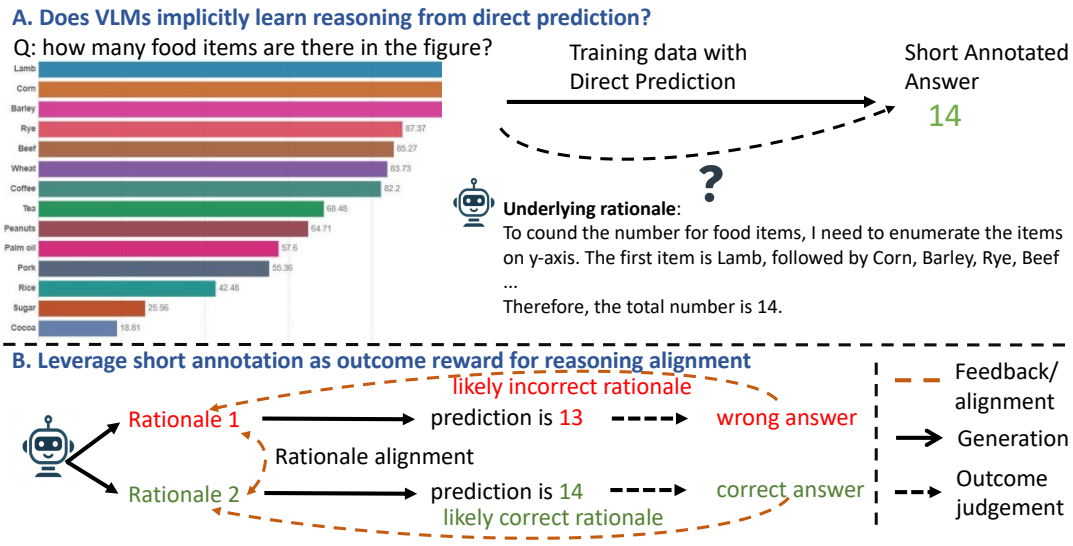


Figure 7.1: The upper figure questions whether training exclusively on direct-answer prediction can effectively teach CoT prediction. In the lower figure, we leverage short annotation as outcome reward for reasoning alignment, allowing the model to improve with self-generated data.

that includes detailed reasoning steps. To address the scarcity of high quality CoT reasoning data, we propose leveraging datasets with short ground truth annotations and employing the GPT-4o model to generate reasoning paths that lead to the correct answer. Our approach encompasses a diverse range of tasks, utilizing 9 datasets that demand different reasoning skills, including common world knowledge (A-OKVQA [121]), chart interpretation (ChartQA [121]), document information localization (DocVQA [102], InfoVQA [103]), real-world text extraction (TextVQA [124]), scientific reasoning (AI2D [69], SQA [96]), and mathematical reasoning (MathVision [136], G-LLaVA [41]). We distilled a total of 193k CoT examples for SFT and the model, LLaVA-Reasoner-SFT, demonstrates significant improvements in VLM chain-of-thought reasoning performance.

In the lower part of Fig. 7.1, we propose further calibrating SFT model reasoning with short answer for outcome rewards [122, 129]. Specifically, the model generates multiple CoT steps to to arrive at a final prediction, which is then compared against a provided short annotation. Rationales leading to correct predictions are more likely to be accurate, while those leading to incorrect predictions are less so. By optimizing positive (likely correct) and negative (likely incorrect) rationale pairs using DPO, we align the VLM towards a more accurate reasoning process. The aligned model, LLaVA-Reasoner-DPO, demonstrates consistent performance improvements across all domains compared to its SFT counterpart. Additionally, we find that the DPO model can act as a strong verifier to assign appropriate rewards for CoT reasoning, en-

abling more effective credit assignment [98, 114].

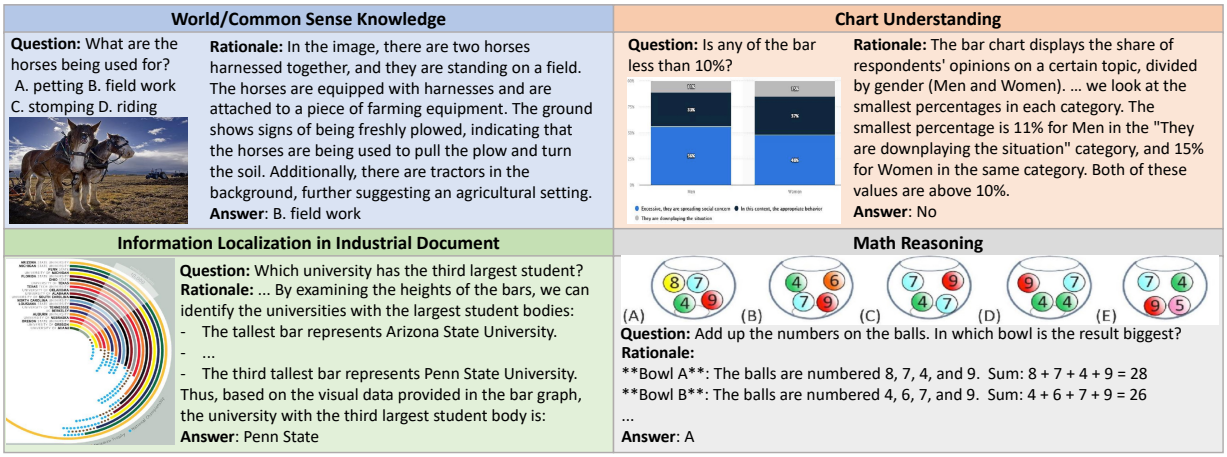


Figure 7.2: Distillation of examples from various VLM task domains, highlighting the specific reasoning capabilities required.

7.3 Method

Our pipeline consists of three stages: (A) CoT data distillation from GPT-4o (Section 7.3.1), (B) SFT with CoT (and direct) data to enable VLM CoT reasoning (Section 7.3.2), and (C) Outcome reward RL for CoT reasoning enhancement (Section 7.3.3).

7.3.1 Reasoning Data Distillation

We leverage VQA datasets with short annotations to augment them with rationales generated by the GPT-4o model. We collect 193k visual CoT instances to create the ShareGPT-4o-Reasoning dataset for community usage. We focus on the following reasoning types as demonstrated in Fig. 7.2:

Real-World Knowledge includes A-OKVQA, which covers a broad range of commonsense reasoning and real-world knowledge for answering questions.

Chart Understanding includes ChartQA, which involves tasks like item comparison, counting, and numerical computation.

Textual Reasoning includes DocVQA, InfoVQA, and TextVQA, focusing on information localization and extraction in industrial documents and real-world image comprehension.

Dataset	Size
A-OKVQA	16.9k
ChartQA	26.0k
SQA	6.1k
AI2D	11.9k
InfoVQA	22.4k
DocVQA	37.3k
TextVQA	29.7k
MathVision	11.0k

Math and Science includes MathVision, G-LLaVA, SQA, and AI2D, focusing on scientific knowledge and mathematical reasoning.

After distillation, we filtered out examples whose answer predicted by GPT-4o is different from ground truth. The data statistics are presented in Section 7.3.1, and a comparison of answer lengths is shown in Fig. 7.3, highlighting that CoT responses peak around 100 tokens, while direct answers are typically under 5 tokens.

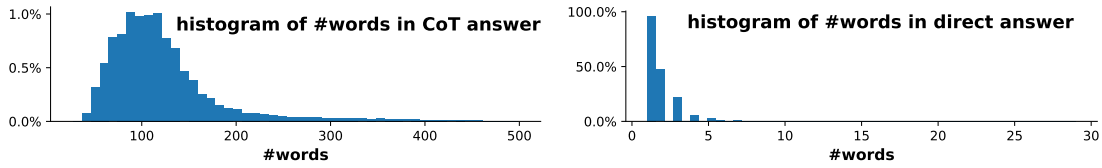


Figure 7.3: The distribution of word counts for CoT and direct answer.

7.3.2 SFT for CoT Prediction

We choose LLaMA3-LLaVA-NeXT-8B as our base architecture, whose weight is initialized with the Open-LLaVA-NeXT weights [21]. To ensure the model handles both direct and chain-of-thought (CoT) predictions, we implement two types of prompts during training.

Direct Prediction: For direct prediction tasks, we use the prompt "Answer the question with a short answer" for short-answer questions, and "Answer with the option's letter from the given choices directly" for multiple-choice questions.

CoT Prediction: For CoT prediction tasks, we use the prompt "Generate a reason first and then output a letter answer" for multiple-choice questions, and "Generate a reason first and then output a short answer" for short-answer questions. In the model's response, the rationale is followed by the answer, which is formatted as "### Answer: " to enable answer extraction during evaluation.

7.3.3 RL for Enhanced Reasoning

To further improve the quality of reasoning chains, we apply RL using the DPO algorithm to better align the model's reasoning process toward more accurate predictions. The DPO algo-

rithm requires both positive and negative responses. To generate these, we use the SFT model as the policy model (i.e., generator), producing 32 candidate predictions per question (temperature 1.0 for short answer and 1.2 for multiple-choice questions). Each prediction is compared with the ground truth to determine its correctness. Following the approach in [36], we select instances with an accuracy between 0.25 and 0.85. From these, we randomly pair positive and negative responses, creating up to three pairs per question.

Formally, the dataset is denoted as $\mathcal{D}_{DPO} = \{(\mathcal{V}, x, y_w, y_l)\}$, where $\mathcal{V}$ is the image, x is the question, y_w and y_l are the positive and negative responses. The DPO objective is defined as below:

$$\mathcal{L}_{\text{DPO}}\left(\pi_\theta; \pi_{\text{ref}}\right) = -\mathbb{E}_{(\mathcal{V},x,y_w,y_l)\sim\mathcal{D}_{DPO}}\Bigg[\log\sigma\left(\beta\log\frac{\pi_\theta\left(y_w \mid x,\mathcal{V}\right)}{\pi_{\text{ref}}\left(y_w \mid x,\mathcal{V}\right)} - \beta\log\frac{\pi_\theta\left(y_l \mid x,\mathcal{V}\right)}{\pi_{\text{ref}}\left(y_l \mid x,\mathcal{V}\right)}\right)\Bigg],$$

where π_θ is the policy model to be optimized and π_{ref} is the base reference model, both models are initialized with SFT weights. σ is the logistic function and β is set to 0.1.

7.4 SFT Experiments for Chain-of-thought Learning

In this section, we explore how SFT can enhance VLM reasoning by addressing two key research questions: (1) *Can CoT reasoning be implicitly learned from short responses?* and (2) *How effectively can CoT be learned from GPT-4o distilled data?* Additionally, we analyze the composition of CoT data across various reasoning capabilities and compare the performance of SOTA models with GPT-4o.

7.4.1 Training Setting

As shown in the upper part of Fig. 7.4, we present the data composition for SFT. The training data includes CoT distillation (193k instances) from Section 7.3.1 and corresponding short answers (193k). Additionally, for CoT data, we incorporate 16k visual math examples from G-LLaVA. To maintain general instruction-following capability as the base model, we include 2k randomly sampled instruction data from LLaVA pretraining [91]. To ensure the SFT models can handle both direct and CoT prompts during inference, we sample a small set of format-aligned data—50 examples from each of the 9 datasets—resulting in 450 instances.

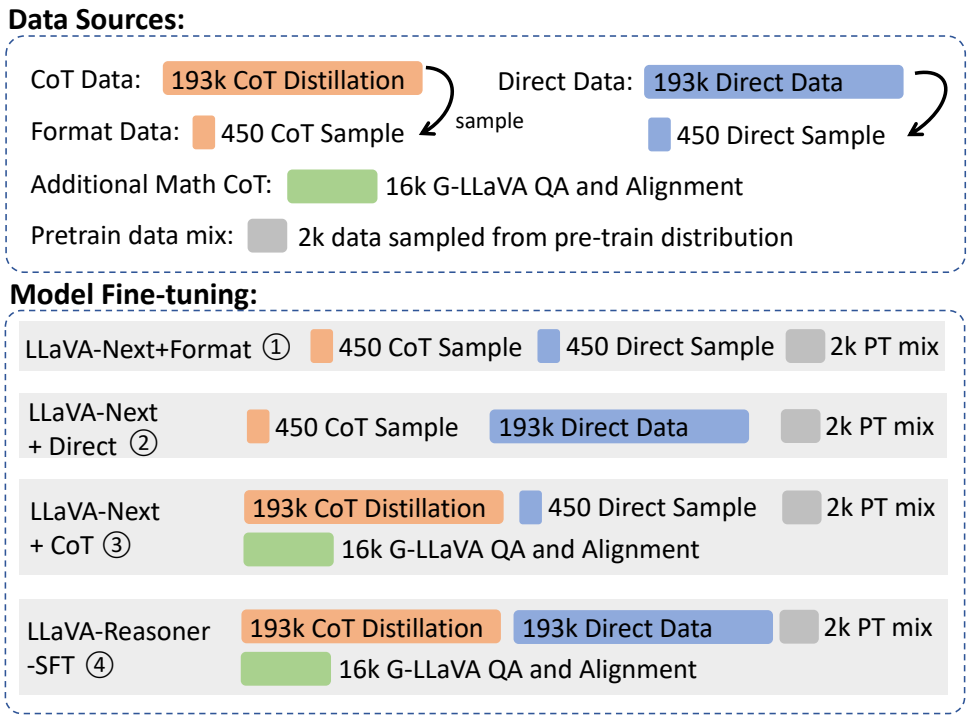


Figure 7.4: The upper section displays the data sources used for the SFT experiments, while the lower section illustrates the data composition for model training.

In the lower part of Fig. 7.4, we outline the data composition for model training. Specifically, LLaVA-NeXT-Format (Fig. 7.4 ①) serves as the baseline model, trained exclusively on format-aligned data to enforce the desired output format without learning any task-specific reasoning skills. In contrast, models in Fig. 7.4 ② and ③ incorporate either direct or CoT datasets, enabling the model to be expert in one type of skill as well as following the both direct and CoT prompt styles. Finally, LLaVA-Reasoner-SFT (Fig. 7.4 ④) represents the SFT model trained on both CoT and direct data, making it to be expert in both types of reasoning.

We use the LLaMA3-LLaVA-NeXT-8B architecture, initializing the weights with Open-LLaVA-NeXT. All Supervised Fine-Tuning (SFT) experiments are trained for 1 epoch with a learning rate of 5e-6 and a batch size of 32. The experiments are conducted on 8 H100 GPUs.

7.4.2 Evaluation Setting

We evaluate our method using a range of benchmark datasets, including A-OKVQA [121], ChartQA [101], DocVQA [102], InfoVQA [103], TextVQA [102], AI2D [69], ScienceQA [96], and MathVista [97]. We also conduct more evaluation on general datasets OCRBench [95], MM-

Table 7.2: SFT experiments with data composition in Fig. 7.4: ① format alignment only, ② direct responses only, ③ CoT responses only and ④ both direct and CoT responses. Inference is performed using both direct and CoT templates. The best CoT prediction result is highlighted in orange, while the best direct prediction result is marked in blue. The results demonstrate that combining CoT and direct responses during training leads to the best performance across both types of prompts. Refer to Section 7.4 for detailed analysis.

Methods	**Prompting**	A-OK	ChartQA	DocVQA	InfoVQA	TextVQA	AI2D	SQA	MathVista	Avg
LLaVA-Next	direct	85.8	70.2	75.7	37.7	68.2	71.5	75.4	39.3	65.5
+ Format ①	CoT	84.3	71.2	67	34.9	62.2	67.4	74.4	40.3	62.7
LLaVA-Next	direct	86.4	73.7	78	45.4	71.9	78.8	91.5	43.2	71.1
+ Direct ②	CoT	85.7	71.8	68.8	38.6	63.6	72.5	85.4	38.6	65.6
LLaVA-Next	direct	84.9	71.8	81.2	45.7	72.1	75.3	85	41.9	69.7
+ Cot ③	CoT	85.1	82.2	81.2	49.7	69.9	77	91.3	49.2	73.2
LLaVA-Reasoner	direct	85.4	76.1	82.9	50.6	73.1	79.4	90.4	44.3	72.8
-SFT ④	CoT	86.2	83.0	81.8	51.6	71.1	78.5	92.7	50.6	74.4

Star [22], and MMMU [159] in later sections. The evaluation for A-OKVQA was implemented by us, while for the other datasets, we follow the evaluation protocols outlined in VLMEval [35].

For CoT evaluation, answers are extracted after the pattern "###Answer: " before sent to evaluation. More comparison with LLaMA3-LLaVA-NeXT-8B model is shown Section 12.2.

7.4.3 Can reasoning be implicitly learnt from direct prediction?

Table 7.2 presents the performance of the models introduced in Fig. 7.4. Since LLaVA-NeXT-8B training data contains very few CoT reasoning examples, CoT performance of ① lags behind direct prediction across most tasks. The only improvement is observed in ChartQA and MathVista with a modest gain of +1.0 in CoT performance, showing CoT is helpful for calculation related tasks.

When comparing model trained on direct only data (②) to that trained on format-aligned data (①), we observe an average gain of +5.6 in direct prediction accuracy (65.5 $\rightarrow$ 71.1) and a +2.9 improvement in CoT performance (62.7 $\rightarrow$ 65.6). Surprisingly, closer inspection of CoT performance in calculation-involved tasks, such as ChartQA and MathVista, reveals only marginal gains (+0.6 for ChartQA CoT) or even a performance drop (-1.7 on MathVista), which contrasts with the improvements seen on the two tasks in ①. On text-rich tasks, positive gains (>1) are observed, with the most improvement seen in InfoVQA (+3.7). Significant gains are also evident

in science-related tasks like AI2D (+5.1) and SQA (+11.0). Despite these improvements, CoT performance still trails behind direct prediction overall (CoT: 65.6 vs. direct: 71.1). This result suggests that training on direct only prediction may not effectively help with CoT prediction.

7.4.4 How Effective is CoT Reasoning Data?

When comparing the model trained on CoT-only data (③) with the one trained on format-aligned data (①), we observe improvements in both direct and CoT predictions. Direct prediction performance increases by an average of +4.2 (65.5 → 69.7), while CoT prediction improves significantly by +10.5 (62.7 → 73.2). Notably, the CoT performance of the model ③ surpasses its direct prediction (73.2 CoT vs. 69.7 direct). Significant gains are observed in calculation-intensive tasks like ChartQA and MathVista, with increases of +11.0 and +8.9 in CoT performance, respectively. Interestingly, for text-rich tasks such as DocVQA, InfoVQA, and TextVQA, the direct performance of model ③ (trained on CoT-only data) outperforms that of model ② (trained on direct-only data). This suggests that even for text-heavy tasks, reasoning processes, such as localizing information in documents or recognizing text in real-world scenarios, may benefit from CoT training. The skills learned from CoT training appear to generalize to direct prediction as well.

When both CoT and direct data are combined (④), performance is further enhanced for both prediction types, with an average gain of +7.3 in direct prediction (65.5 → 72.8) and +11.7 in CoT prediction (62.7 → 74.4). This demonstrates that combining direct and CoT data yields the best overall performance. Interestingly, in model ④, for 3 out of 8 datasets (TextVQA, DocVQA, AI2D), direct prediction outperforms CoT prediction. We hypothesize that these tasks involve a significant proportion of concise fact extraction, where generating long-form CoT responses may not provide additional benefits or even hurts. Further validation of this hypothesis will be explored in future work.

7.4.5 Comparing with SOTA model and GPT-4o

In Table 7.3, we compare the performance of GPT-4o and a recent state-of-the-art model, Cambrian [132]. For GPT-4o, we include both direct and CoT predictions, following the prompt optimization steps outlined in [13], with the prompts detailed in **??**. For Cambrian, we report the numbers from [132] and replicated the results using the official checkpoint on MMStar, InfoVQA, and A-OKVQA. Specifically for Cambrian, CoT predictions were used for the MathVista dataset, while direct predictions were applied for the remaining datasets.

Table 7.3: Performance Comparison of GPT-4o, Cambrian-7b, and our SFT Model. For Cambrian, * indicates our replicated results, while others are adapted from [132], † indicate CoT prompt used for evaluation. 'Our-SFT' refers to LLaVA-Reasoner-SFT.

Dataset	GPT-4o direct/cot	Cambrian official	Our-SFT direct/cot
A-OK	89.6/90.1	83.1*	85.4/86.2
ChartQA	79.6/84.7	73.3	76.1/83.0
DocVQA	90.3/90.8	77.8	82.9/81.8
InfoVQA	72.4/72.8	45.7*	50.6/51.6
TextVQA	78.1/75.4	71.7	73.1/71.1
AI2D	80.7/81.5	73.0	79.4/78.5
SQA	85.9/87.2	80.4	90.4/92.7
MathVista	54.8/63.4	49.0†	44.3/50.6
OCRBench	80.2/79.2	62.4	61.6/62.0
MMStar	55.1/64.7	50.3*	51.6/54.0
MMMU	57.8/63.6	42.7	41.6/40.0
Avg (of best)	77.9	64.5	68.8

When compared to open-source models, GPT-4o outperforms on nearly all benchmark datasets, with the exception of SQA. Notably, significant improvements from CoT predictions are observed on tasks involving calculation or complex reasoning, such as ChartQA, MathVista, MMMU, and MMStar.

Cambrian-7B is trained on a dataset of 7 million open-source instruction-following examples. In contrast, our model, fine-tuned on fewer than 400k instruction examples, outperforms Cambrian-7B on most benchmark datasets, underscoring the effectiveness of incorporating CoT data. While we recognize the challenge of comparing against other models, such as One-Vision [80], MiniCPM-V [152], X-Composer [162], and InternVL [23], due to differences in model architecture, training datasets, and evaluation pipelines, our primary focus is on studying the effectiveness of CoT learning rather than competing for state-of-the-art performance on visual-language tasks.

7.5 Nearly Zero Data Learning for CoT Reasoning

Table 7.4: We study a self-taught reasoner with minimal CoT data (only 450 format-aligned examples). LLaVA-NeXT-Direct is used as the baseline, and our LLaVA-Next-STaR is trained with a rejection sampling method. The best CoT predictions are highlighted in orange, and the best direct predictions are highlighted in blue. Our rejection sampling method outperforms both CoT and direct prediction, with the exception of two data points.

Methods	**Prompting**	A-OK	ChartQA	DocVQA	InfoVQA	TextVQA	AI2D	SQA	MathVista
LLaVA-Next	direct	86.4	73.7	78	45.4	71.9	78.8	91.5	43.2
+ Direct ①	CoT	85.7	71.8	68.8	38.6	63.6	72.5	85.4	38.6
LLaVA-Next	direct	85.9	74.6	79.2	47.4	72.1	79.5	92.2	44.4
-STaR ④	CoT	85.9	77.9	75.8	44.0	25.1	76.6	86.8	42.0

In this section, we demonstrate how minimal CoT training data can enhance CoT reasoning capabilities. Specifically, we use only 450 CoT format-aligned examples alongside all available direct prediction data, with LLaVA-NeXT-Direct as the baseline. We apply rejection sampling fine-tuning (RFT) following [122, 129] to train a self-taught chain-of-thought reasoner, denoted as LLaVA-Next-STaR. From LLaVA-NeXT-Direct, we sample 32 CoT examples for each training instance and select those whose final predictions match the ground truth. Up to three positive examples are selected per question, resulting in a dataset of 260k RFT examples.

As shown in Table 7.4, RFT training improves both CoT reasoning and direct predictions overall, with the exception of two data points. Notably, TextVQA shows a significant drop in CoT performance, which we will explore further in future work. Notable (>3%) gain is observed on ChartQA, DocVQA, InfoVQA, AI2D and MathVista, and roughly 1% gain is observed on direct prediction on those datasets as well.

DPO Experiments Prior to the RFT experiments, we conducted DPO experiments on the ChartQA dataset under the same conditions as described in Section 7.4. However, the improvements were modest, with a 72.3 (+0.5) gain in CoT prediction and a 74.2 (+0.5) gain in direct prediction. In contrast, RFT yielded a significant improvement, with 77.9 (+6.1) on CoT prediction and 74.6 (+0.9) on direct prediction. We hypothesize that for models with relatively weak CoT reasoning capabilities, RFT may be more effective in enhancing model performance, whereas DPO with preference modeling may be less impactful. We leave further analysis for

future work.

7.6 RL for Enhanced CoT Reasoning

In this section, we demonstrate the effectiveness of RL in further enhancing CoT reasoning. By leveraging short-answer feedback (Section 7.3.3), we construct preference pairs across three domains: A-OKVQA (real-world knowledge reasoning), ChartQA (chart interpretation), and math (MathVision and G-LLaVA). Although additional DPO data from other datasets could be incorporated, data scaling and balancing will be addressed in future work.

For the DPO dataset, we include 24.5k examples from ChartQA, 18.3k from A-OKVQA, and 22.0k from math domain, totaling 64.8k preference data pairs. We train LLaVA-Reasoner-SFT on this dataset using a learning rate of 5e-7, a batch size of 32, and for 1 epoch. We found an additional trick to truncate the responses up to 90 tokens to be crucial for DPO training (details in Section 12.3). To compare the effectiveness of different DPO datasets, we include RLAIF-V [158], which contains 80k DPO pairs representing the state-of-the-art dataset for aligning VLMs for reducing hallucinations.

7.6.1 Can DPO Calibrate Reasoning?

In **??**, we present the results of the DPO model optimized on top of LLaVA-Reasoner-SFT (④). Model ⑤ uses the SOTA RLAIF-V [158] data, while model ⑥ uses our dataset. We observe that Model ⑤ shows a slight improvement in both direct prediction (+0.2) and CoT prediction (+0.2), whereas model ⑥ demonstrates a greater improvement in CoT prediction (+1.1) with equal gains on direct prediction. Interestingly, though only 3 out of 8 datasets are selected to construct DPO pairs, gains are observed across 7 out of 8 datasets except for SQA with a slight decrease (92.9 $\rightarrow$ 92.6). These results suggest that DPO dataset constructed from model-generated rationales can effectively enhance reasoning accuracy and show generalization across tasks.

7.6.2 DPO as Verifier for Re-ranking CoT

In Fig. 7.5, we present the re-ranking results using the DPO model as a verifier, following the approach of [57, 98, 169]. The DPO reward score is calculated as $\log \frac{\pi_{\text{dpo}}(y|x,\mathcal{V})}{\pi_{\text{sft}}(y|x,\mathcal{V})}$, where $\mathcal{V}$ represents the image, x the question, and y the candidate answer. We explore two re-ranking strategies: Best-of-N and Weighted Voting. A Majority Voting (or self-consistency) baseline is also included for comparison.

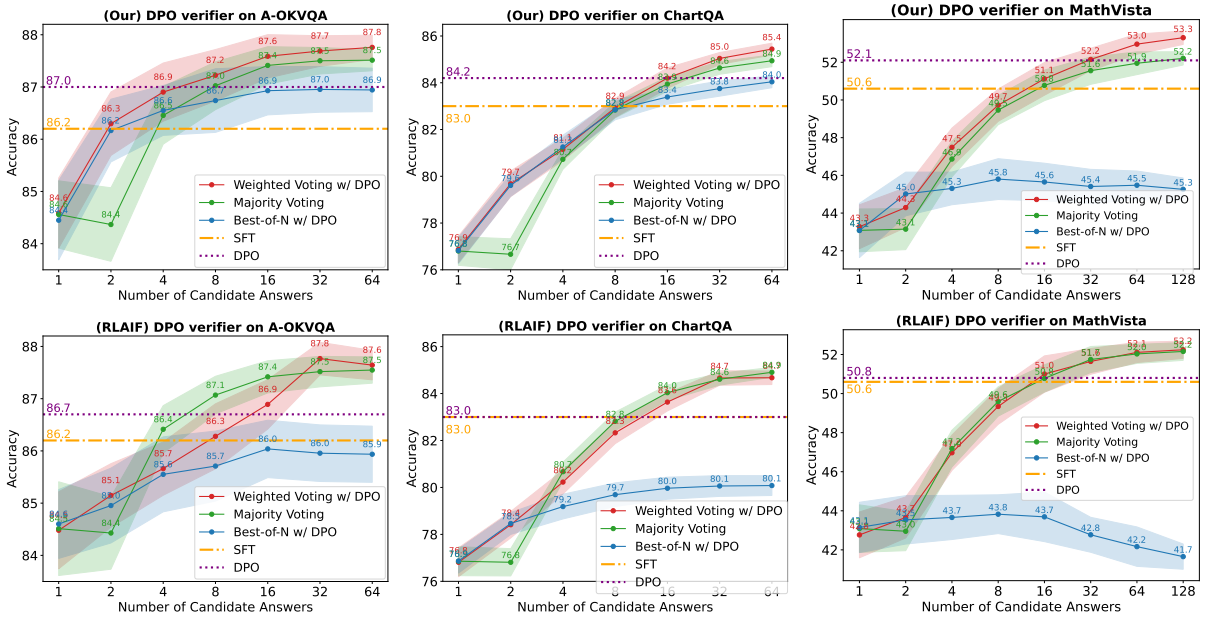


Figure 7.5: The figures illustrate the performance of the DPO model as a verifier on ChartQA, A-OKVQA, and MathVista. Compared to the model trained with RLAIF-V, the model trained on our reasoning data pairs consistently shows improvement in both best-of-N selection and weighted voting.

When trained with RLAIF-V data (⑤), the DPO model demonstrates improvements as both a generator and verifier on A-OKVQA, likely due to the dataset's alignment with real-world images, which matches the nature of A-OKVQA. Interestingly, while model ⑤ does not show improvements as a generator on ChartQA, it still produces positive results in best-of-N re-ranking, indicating that the learned preferences can generalize across domains. However, weighted voting does not lead to any improvements, and no significant gains are observed in re-ranking for MathVision. In contrast, when trained with reasoning data pairs, LLaVA-Reasoner-DPO (⑥) shows improvements across both re-ranking metrics, underscoring the effectiveness of DPO on

Table 7.5: More DPO results on general evaluation benchmark datasets.

Methods	OCRBench	MMStar	MMMU	Avg
SFT ④	62.0	54.0	40.1	52.0
SFT+RLAIF ⑤	**63.7**	53.5	42.3	53.2
SFT+DPO-ours ⑥	**63.7**	**54.1**	**42.6**	**53.5**

reasoning data pairs.

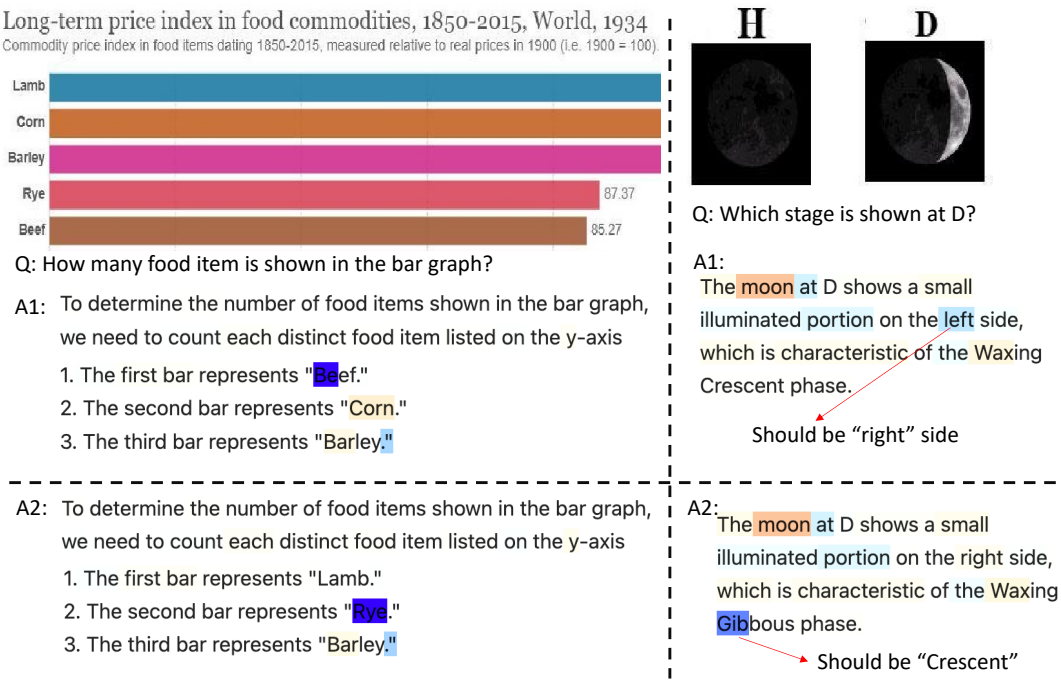


Figure 7.6: Credit assignment of the DPO model on a portion of the responses from the ChartQA and AI2D datasets. The DPO token-level reward is computed for each token, with the rewards normalized to have a mean of 0. Negative scores are highlighted in cool colors (blue), while positive scores are highlighted in warm colors (orange). We observe that the DPO model is particularly sensitive to the first mistakes or hallucinations introduced in the response.

7.6.3 DPO CoT Prediction and Re-ranking Performance Generalization

In Table 7.5, we present the DPO CoT performance on OCRBench, MMStar, and MMMU. We observe that DPO models trained on both RLAIF and our datasets outperform the SFT baseline, with our DPO model trained on CoT reasoning pairs achieving slightly better results.

While the CoT prediction performance across DPO models is similar, **??** highlights the pronounced effectiveness of our DPO verifier's generalization on the MMMU dataset, which contains challenging college-level subject questions. We provide re-ranking results for multiple-choice problems from the Dev+Val split (988/1050). The SFT model with self-consistency shows steady improvements, reaching 45.5% with 64 candidate votes. LLaVA-Reasoner-DPO, trained

on reasoning data pairs, demonstrates strong generalization on MMMU by excelling in both weighted voting and best-of-N voting during candidate re-ranking.

In contrast, the DPO model trained on RLAIF-V (⑤) improves CoT predictions but fails to achieve gains in re-ranking metrics, indicating its limitations in distinguishing correct from incorrect reasoning on more complex data. We hypothesize that, compared to ChartQA, the reasoning questions in MMMU are more challenging and span a broader range of subjects. The RLAIF-V dataset, being primarily focused on the COCO image domain, may lack sufficient coverage of this diversity, leading to weaker performance in re-ranking. These results underscore the potential of our approach for generalizing visual language reward models to reasoning tasks.

7.6.4 DPO Credit Assignment

While the DPO model is trained on pairwise data, prior works [98, 114] have shown that DPO policies can learn to predict *token-level rewards* from binary preference data. These experiments primarily focused on math reasoning with LLMs. In this work, we provide examples of credit assignment learned by the VLM DPO, as shown in Fig. 7.6. The token-level DPO reward can be expressed as $\log \frac{\pi_{\text{dpo}}(y_i|x,\mathcal{V})}{\pi_{\text{sft}}(y_i|x,\mathcal{V})}$, where $\mathcal{V}$ represents the image, x the question, and y_i the i-th token in the generated response. This reward reflects the relative confidence of the DPO model compared to the SFT model for a given token in a candidate response.

In Fig. 7.6, negative scores are shown in cool colors, while positive scores are shown in warm colors, with rewards normalized to a mean of 0. On the left, we observe that the DPO model is particularly sensitive to errors during chart interpretation from the ChartQA dataset. For instance, when the response incorrectly lists "Lamb" as "Beef" in a chart reading task, the DPO model assigns a highly negative score to this mistake.

On the right, we present examples from the AI2D dataset. Here, a hallucination in the response, such as incorrectly stating that the left side of the moon is illuminated (the correct answer is the right side), receives a low score. Additionally, when external knowledge is required to correctly identify the moon's phase as "Crescent" instead of "Gibbous," the DPO model penalizes the incorrect "Gibbous" answer with a negative score. This indicates that the DPO model is more sensitive to knowledge-based errors than the SFT model, explaining its superior performance on CoT reasoning tasks in datasets such as AI2D.

7.7 Conclusion

In this chapter, we explored the use of model-generated reasoning traces to enhance the chain-of-thought reasoning capabilities of vision-language models. By leveraging reinforcement learning with outcome-based rewards, we demonstrated that self-generated reasoning traces can improve reasoning quality despite the scarcity of human-annotated data. Our approach achieves state-of-the-art performance among models of similar sizes and, in some cases, matches the performance of proprietary models. These findings highlight the potential of model-generated signals in advancing vision-language reasoning and suggest promising directions for future research in automated reasoning calibration and data-efficient learning.

Part IV

Conclusion Remarks

Chapter 8

Conclusion

This thesis aims to effectively leverage model-generated signals to enhance neural network training, particularly in scenarios with limited or no human-labeled data. The proposed methodologies are successfully applied across the following domains:

- **Few-/Zero-Shot Text Classification:** This part addresses the low-resource problem in NLP by augmenting text-based training signals using model-generated data. We began our exploration with few-shot learning for extreme text classification, where the majority of labels receive fewer than 10 training instances due to the natural low occurance of such label phrases. In Chapter 2, we demonstrated that our **DEPL** model significantly improves performance by incorporating SVM-generated labels. Extending this approach to zero-shot classification, Chapter 3 investigates the use of instruction-following LLMs to generate document content and novel document-label training pairs. Together, these studies provide a comprehensive answer to the research question of how model-generated signals can enhance text classification in data-scarce settings.
- **Unsupervised Change-Point Detection:** We next explore zero-shot learning for time-series change-point detection (CPD), where labeled data is scarce due to the infrequent nature of change points and the need for domain expertise. In Chapter 4, we propose CorD-CPD, a method that utilizes graph neural networks to generate correlation graphs, improving both forecasting and CPD performance in multivariate time-series data. This work highlights the critical role of model-generated auxiliary data in enabling effective learning for tasks with minimal labeled supervision.
- **Large Language Model Self-Enhancement:** With the advancement of large language models (LLMs), we focus on improving their alignment, particularly in conversational

and reasoning abilities. In Chapter 5, we introduce LLMiner, a conversation mining model that automatically generates dialogue data for chatbot training. This approach is later extended in Chapter 6 to enhance video-language models, where LLMiner generates conversation data from video captions. Finally, we propose a reinforcement learning framework leveraging self-generated reasoning traces to improve vision-language reasoning. Our findings demonstrate that LLMs can benefit from model-synthesized data, particularly for instruction-following and reasoning tasks. This underscores the importance of synthetic data in advancing AI capabilities.

Each part of this thesis provides an in-depth examination of its respective topic, collectively contributing to the broader methodology of advancing neural network optimization through model-generated signals. These signals are closely related to model **self-play**, where a model generates data to improve itself, potentially leading to stronger intelligence with minimal human intervention. Below, we summarize the key types of signals that facilitate self-play:

1. **Pseudo Labels:** When no labeled data is available for zero-shot classification, the model first generates labels for unlabeled instances, referred to as pseudo-labels. Training on pseudo-labels can enhance classification performance by reducing the entropy of decision boundaries, as demonstrated in Algorithm 1 within Chapter 3. Our findings highlight the importance of pseudo-labeling as a mechanism for model self-improvement.
2. **Language Model Responses:** In Chapter 6, we show that for a given question, a model can generate diverse responses by sampling at a higher temperature, leading to variations in response quality. We leverage reinforcement learning with the Direct Preference Optimization (DPO) algorithm on paired model responses, which proves to be both effective and data-efficient. This demonstrates that model-generated responses serve as a valuable resource for self-improvement.
3. **Reasoning Traces:** Chain-of-thought (CoT) reasoning can be viewed as intermediate states leading to a final answer in problems requiring multi-step reasoning. In Chapter 7, we show that by comparing model predictions with labeled answers using an outcome-based reward, the model can calibrate its reasoning through self-generated traces. This finding sheds light on a pathway toward stronger intelligence, where static data, supplemented with self-generated signals, drives continual model enhancement.

The studies in this thesis present successful applications of model-generated signals across various tasks, demonstrating their effectiveness in enhancing neural network training. Looking ahead, we envision a broader paradigm shift in artificial intelligence driven by model-generated

data.

Over the past five years, pre-training has largely relied on vast amounts of human-created data sourced from the internet. However, as those data becomes increasingly exhausted, and as models continue to grow in power, we anticipate a transition toward self-generated data as a primary driver of model improvement. This shift could revolutionize AI training, enabling continuous learning beyond the limitations of static human-annotated datasets.

For future work, we foresee the expansion of model-generated signals into a wider range of tasks, including those involving audio, video, and other modalities. Additionally, we anticipate further advancements in model training through the use of evolving synthetic data at different training stages, continuously refining model capabilities in an autonomous and scalable manner.

Part V

Appendices

Chapter 9

Appendix of Chapter 4

9.1 Synthetic Data Demo

We plot an example of the synthetic change-point data in Figure Figure 9.5. We show the trajectories of 5 particles before and after a change-point, where the dashed lines represent the expected trajectory if no change-point happened, and the solid lines are the observed trajectory.

9.1.1 Location Change

The location change example is shown in Figure 9.5. In this case, we treat the location as multivariate time series, and we observe a small shift of location at and after the change-point. The gap between the expected value and observed value is maintained during the particle movement, but it may vary due to the complicated interactions between those variables.

9.1.2 Velocity Change

The velocity change example is shown in Figure 9.6. Compared with the location change, there is no immediate shift in the time series value observed. The gap between the expected value and the observed value tend to become more obvious over time. This is due to the nature of speed such that a small perturbation can cause large difference in location over long time. In order to detect such kind of changes, a window based comparison (of expected values and observed values) introduced in Section 3 is preferable to using only a single predicted time step.

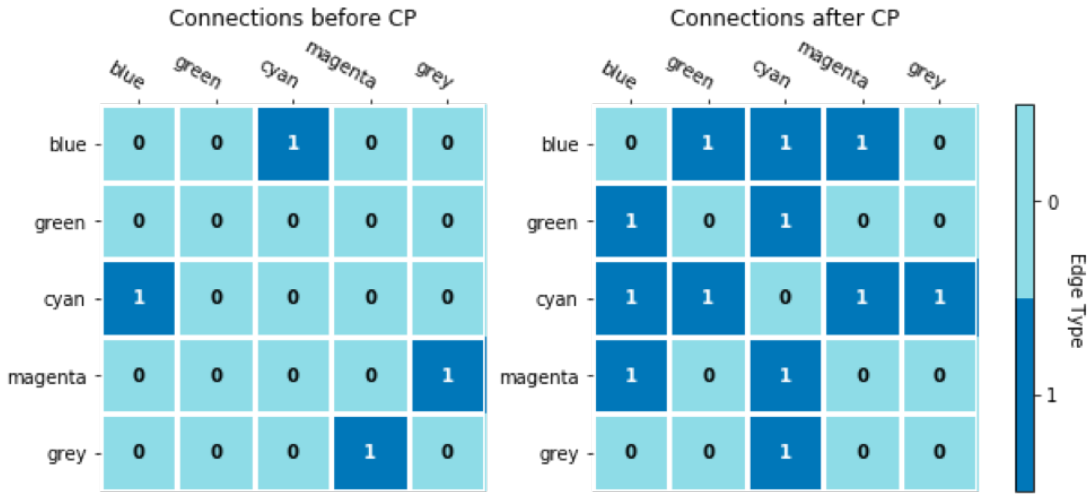


Figure 9.1: The connections before and after the change-point for the correlation change example.

9.1.3 Connection Change

The connection change example is shown in Figure 9.7. Similar to the velocity change, the difference between the expected value and the observed value becomes more obvious over time. The Figure Figure 9.1 shows the underlying spring connections before and after the change-point. In particular, the *green* particle was not connected with any other particles before the change-point, and it was connected with *blue* and *cyan* particles after the change-point. This altered the trajectory of the *green* particle from a straight line to curved path. Detecting the spring re-connections change-points requires the modelling of dynamic correlations in multivariate time series, as our model did.

9.2 Further Analysis on Synthetic Experiments

In this section, we further describe our model training, baselines, and some other metrics on the synthetic dataset.

9.2.1 Implementation Detail

We perform a grid search for hyperparameters of the following values: the learning rate l_r in $\{0.001, 0.005, 0.01, 0.05\}$, the hidden dimension size d for the time series feature embeddings in $\{64, 128, 256\}$, and the number of levels of GNN or spatial transformer in $\{2, 3, 4\}$. We finally

selected $l_r = 0.001$ using Adam optimizer, $d = 64$ for transformers and $d = 256$ for GNN. The level of GNN or spatial transformer is set as 2, which is sufficient for our experiments. We used batch size of 32 for temporal and spatial transformers and batch size of 128 for GNN.

We report the results for three encoder models: **GNN$_{\text{SE}}$+RNN$_{\text{TE}}$**, **Trans$_{\text{SE}}$+RNN$_{\text{TE}}$** and **GNN$_{\text{SE}}$+Trans$_{\text{TE}}$**. Using both temporal and spatial transformer modules was hard to optimize and resulted in degraded performance, so we didn't include it as our model.

9.2.2 Correlation prediction Accuracy

Since the ground truth connections of springs ($\mathbf{A}$) are known in the synthetic dataset, we can calculate the accuracy of learnt correlation ($\hat{\mathbf{A}}$). The accuracy p_{acc} is calculated by

$$p_{acc} = \frac{1}{T \times N \times (N-1)} \sum_{t=1, i<j}^{t \leq T} \mathbb{1}_{\{\hat{\mathbf{S}}_{i,j}^t = \mathbf{A}_{i,j}^t\}}$$

$$\hat{\mathbf{S}}_{i,j}^t = \begin{cases} 1, & \text{if } \hat{\mathbf{A}}_{i,j}^t \geq 0.5 \\ 0, & \text{otherwise} \end{cases}$$

Where $\hat{\mathbf{S}}$ is the sampled categorical relation. T is the number of time step and N is the number of variables. For every pair of variables i, j, the function $\mathbb{1}$ is an indicator function which outputs 1 if $\hat{\mathbf{S}}_{i,j}^t = \mathbf{A}_{i,j}^t$, and 0 otherwise.

model	location	speed	connection
GNN$_{\text{SE}}$+RNN$_{\text{TE}}$	96.07%	96.04%	90.45%
Trans$_{\text{SE}}$+RNN$_{\text{TE}}$	**97.79%**	97.36%	**93.11%**
GNN$_{\text{SE}}$+Trans$_{\text{TE}}$	97.49%	**97.47%**	92.53%

Table 9.1: The accuracy of predicted connections compared with ground truth connections in synthetic dataset.

We observe that spatial and temporal transformers achieve better performance in the accuracy metrics, with **Trans$_{\text{SE}}$+RNN$_{\text{TE}}$** the being best on location and connection change, and nearly competitive as **GNN$_{\text{SE}}$+Trans$_{\text{TE}}$** on velocity changes. This result is consistent in the CPD task, such that **Trans$_{\text{SE}}$+RNN$_{\text{TE}}$** has the best score for separated predictions of independent changes and correlation changes.

9.2.3 Correlation vs. Independent Change

In the experiment, our model separately predicts the correlation change-point score by the correlation encoder, and the independent change-point score by the dynamics decoder. We also report the results when the two scores are separately evaluated. In Figure Figure 9.4, we plot the two types of scores predicted by our model in the three types of changes, and the ground truth change-point label as the red dashed line.

We observe that for location and velocity changes, the independent scores are peaked at the labeled change-point; For connection changes, the correlation scores are peaked at the labeled change-point. We conclude that model has the ability to separate the two types of change-points.

9.2.4 Change-point Type Classification

In the change-type classification experiment in Section 5, we propose to evaluate our model in both supervised setting and unsupervised setting. In the supervised setting, the time step of change-point is provided, and our goal is to predict the whether the change-point is resulted from an independent change or a correlation change. In the unsupervised setting, the time step of change-point not given, and we use the predicted change-point by our ensemble model instead.

Our model separately predicts s_r, the correlation change-point score, and s_d, the independent change-point score. The change-point type is determined by $\mathrm{Norm}(s_r) - \alpha\,\mathrm{Norm}(s_d)$, such that

$$\mathrm{Norm}(s_r) - \alpha\,\mathrm{Norm}(s_d) \begin{cases} \geq \tau, & \text{correlation change} \\ < \tau, & \text{independent change} \end{cases}$$

Where α is a hyperparameter and τ is a threshold. Norm is the mean-std normalization function.

In our study, we set $\tau = 0$ and visualize (in Figure Figure 9.2) how α value affects the change-point type classification accuracy. The model we choose is $\textbf{GNN}_{\textbf{SE}}$+$\textbf{Trans}_{\textbf{TE}}$. We observe that if α is small, the correlation change-point score dominates, and connection changes are more accurately predicted. When α is large, the independent change-point score dominates, and the location and velocity changes are more accurately predicted. As a trade off between the two, we choose $\alpha = 0.75$ in the experiment section.

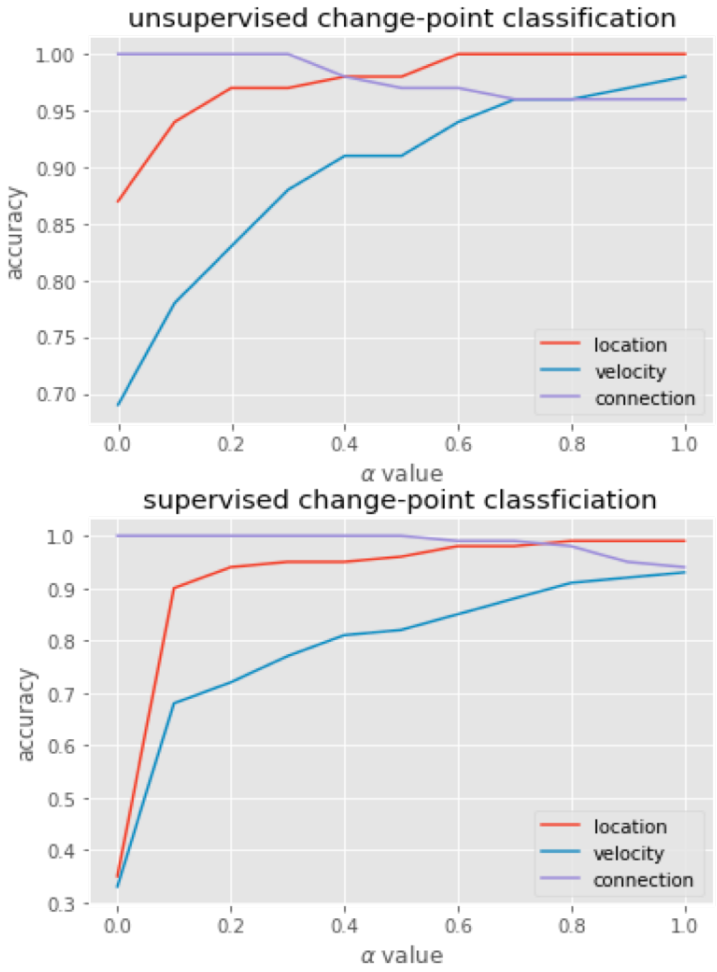


Figure 9.2: α value vs. the change-point type classification accuracy. α is in range $[0, 1]$.

9.2.5 Discussion on CPD

In the experiments, we use three metrics: *AUC-ROC*, *DIST* and *TRI*. The AUC-ROC score was widely used in previous literatures [15, 84, 94, 147], but DIST and TRI are what we proposed in this paper. The reason is that AUC-ROC treats each instance independently, but time-series data has a strong locality dependence. We do observe cases where the peak of the prediction is close to but not aligned with the labels, as shown in Figure Figure 9.3.

We observe that our model has the best performance in all the three metrics. For statistical and other deep learning baselines, we observe that they have similar AUC-ROC, but the statistical models are worse at DIST and TRI metrics. The reason is that statistical model concatenates a window sampled from training data at the start and end of each test case, to avoid the cold-start problem. However, this may lead the algorithm to give higher scores at the start

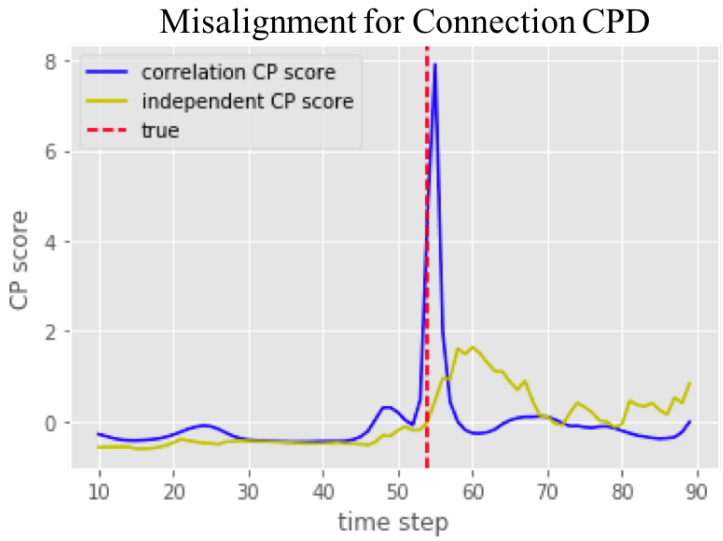


Figure 9.3: The peak of prediction is not aligned with the label, but very close due to a 1-step delay.

or end of the test cases, resulting a larger gap from the labeled change-point.

9.3 Robust CPD on Real Data

In Table Table 9.2, we report the performance of CORD_CPD on the real-world PAMAP2 dataset. The multivariate time series includes 3 variables and 10 features. The variables are sensors on wrist, chest and ankle, and the features are temperature, 3D acceleration, gyroscope and magnetometer. The change-points are transitions between activities, such as walking, cycling, playing soccer.

In the real-world scenario, the change-points are often resulted from a mixture of correlation change and independent change. We show the evaluated scores for the predicted correlation changes and independent changes of each model in Table Table 9.2. We have two observations: 1) Each model capture similar trends for correlation changes and independent changes. There are more independent changes than correlation changes involved. 2) The ensemble of two reasons of change-points further boosts the performance, as the true reason of the change-point could include both of them.

model	type	AUC	DIST	TRI
$\mathbf{GNN_{SE}}$+$\mathbf{RNN_{TE}}$	rel	0.6882	10.40	0.5972
	ind	0.7850	7.73	0.7088
	ens	0.7868	7.16	0.7574
$\mathbf{Trans_{SE}}$+$\mathbf{RNN_{TE}}$	rel	0.6538	13.95	0.4118
	ind	0.7722	7.73	0.7360
	ens	0.7903	6.54	0.7750
$\mathbf{GNN_{SE}}$+$\mathbf{Trans_{TE}}$	rel	0.6715	15.04	0.4013
	ind	0.7787	8.10	0.7102
	ens	**0.8277**	**4.20**	**0.8020**

Table 9.2: The performance of our CORD_CPD on a real-world PAMAP2 dataset for CPD. We include the scores of independent changes and correlation changes as well.

9.4 Desiderata and Related Work

In this section, we conclude our project with the settings of our model and related work to emphasize our contribution and difference from the previous methods.

9.4.1 Settings

Unsupervised In real life, training labels for change-points are hard to obtain, so we want our model to learn the patterns in an unsupervised settings.

Multivariate Time Series Multivariate time series is ubiquitous in our life. We focus on change-point detection by modelling multivariate time series.

Interpretability Previous work in CPD literature mostly focuses on detecting change-points instead of providing explanations to them. We give an attempt to reason the causes behind change-points as correlation changes and independent changes.

Neural Architecture We give our effort to explore the application of deep learning models in CPD.

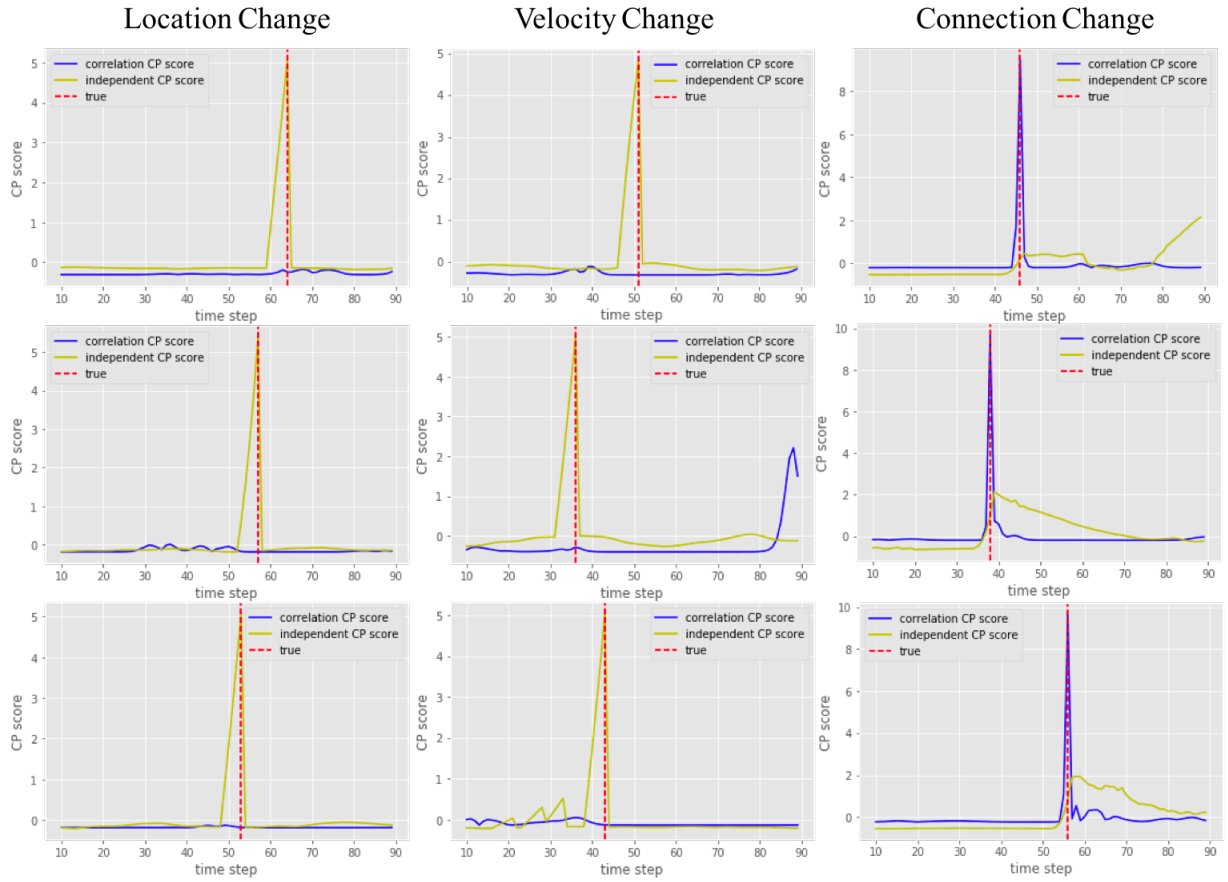


Figure 9.4: We show the correlation change-point score and the independent change-point score in the three types of change-points. The ground-truth change-point is labeled as red dashed line.

Graph Neural Networks Previous neural models mostly rely on CNNs and RNNs to extract local and long term dependencies [77]. For relational learning, GNNs may have better inductive bias and we incorporate it into our encoder and decoder.

9.4.2 More on Related Work

Time series Forecasting

Time series forecasting aims at predicting future time steps based on historical observations on time series signals, with a wide range of applications including forecasting new trends or generating alert for potential hazardous events. Traditional time series forecasting models uses State Space Models (SSMs) and Autoregressive (AR) models. Recently, deep learning models

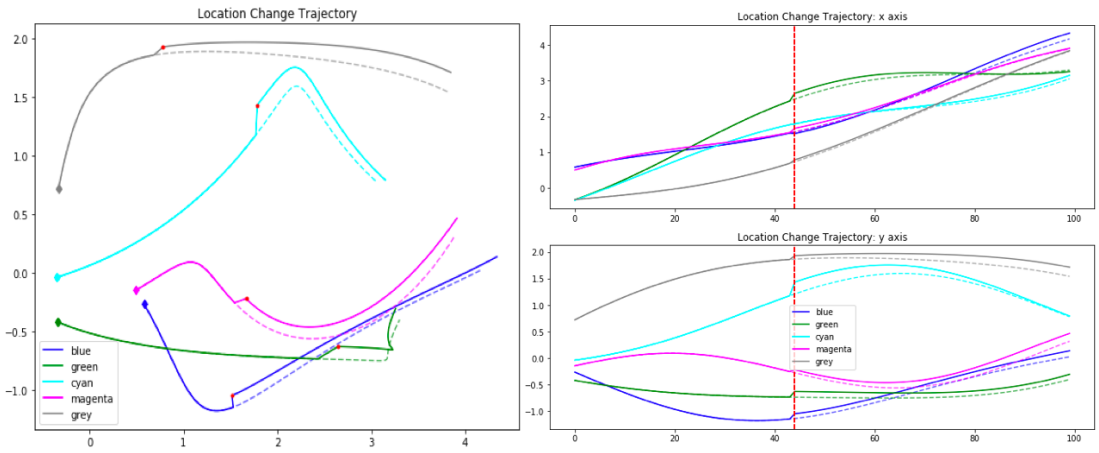


Figure 9.5: Trajectory of Location Change . The figures show the trajectories of different types of change-points of 5 particles (in 5 colors) connected by underlying springs. The figures on the left show the 2-D trajectories of the particles, and the figures on the right show the x and y axis of the trajectories separately. The dashed lines represent the expected trajectory if no change-point happened, and the solid lines are the actual observed trajectory.

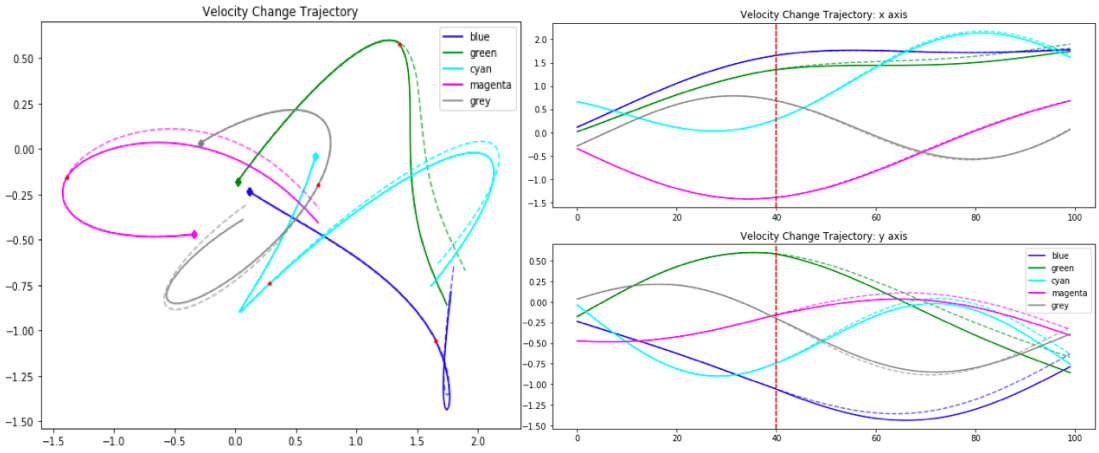


Figure 9.6: Trajectory of Velocity Change.

uses CNN, RNN and Transformer models to extract features automatically and predict the future steps.

Time series forecasting can be applied to CPD by comparing the expected future steps with the observed value. In our work, we use the forecasting models as baselines to calculate the change-point scores.

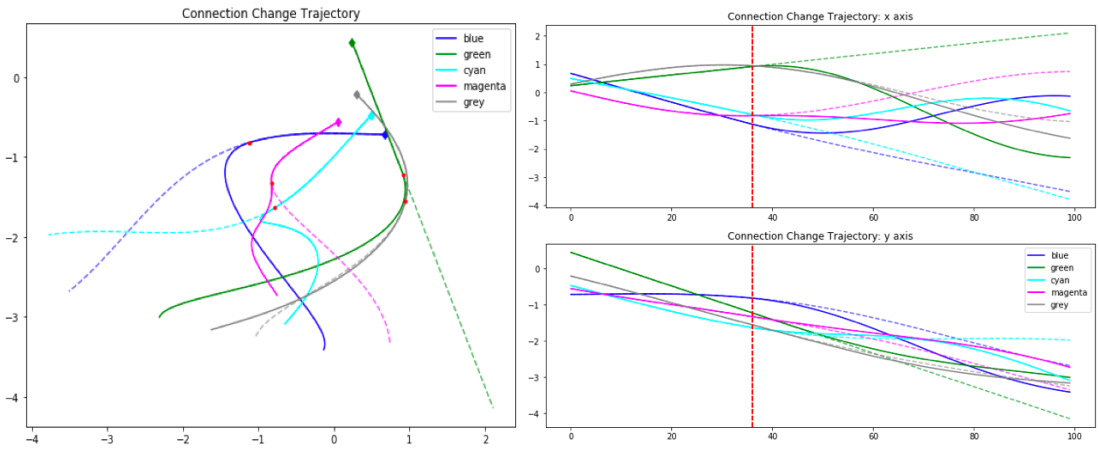


Figure 9.7: Trajectory of Correlation Change.

Graph Neural Networks

Graph Neural Networks have attracted considerable amount of attention in deep learning community. As graph is a natural way to represent underlying interactions between objects, learning a relational graph in a data-driven approach is interesting. Recently, GloMo [151] proved that the relations between units learned by GNN from text and image in one dataset can be generalized to other datasets in transfer learning setting. Neural relational inference [74] extracts non-changing relations from time series data, and structure-informed graph auto-encoder [85] incorporated prior knowledge as regularization to disentangle different relations from time series.

Besides the success of graph networks in applications, researchers are devoted to a theoretical understanding of the capacity of GNN. Graph Neural Networks recursively gather messages from local neighborhood and combine them by an aggregation scheme to form new node features. This recursive aggregation scheme grants GNN the power to be better aligned with relation learning tasks [146], and different ways of combining messages results in different capacity in those tasks [145]. In our work, we study how to use GNN networks to extract dynamic correlation from time-series data and its applications to CPD.

Change-point Detection

In supervised setting, various methods have been applied to model the phases in time series, including decision tree [172], Baysian net [171], support vector machine [116], Gaussian mixture model [29] etc. Recently, deep learning models using wave-net style architecture for super-

vised CPD [37]. Supervised methods such as decision trees could possibly give explanations on change-points, but the labels are usually hard to obtain in large quantities.

In the unsupervised setting, most statistical CPD models are based on statistical inference, or hypothesis tests. Bayesian CPD models [11, 148] computes the probability of change-points using the Bayesian framework. BOCPD algorithms [2, 44, 73, 118] detect change-points by sequentially considering the correlated intervals between them. Recent deep learning models improves the state-of-the-art statistical methods by approximating density ratio [70], or learning kernel functions [15] using neural networks.

Our work belongs to the unsupervised deep learning CPD methods, but we propose a novel encoder-decoder architecture to learn correlation changes and independent changes, and ensemble them for final predictions.

Chapter 10

Appendix of Chapter 5

10.1 Prompts for LLMiner

The sentence analysis prompt is shown in Fig. 10.1, the question proposal prompt is shown in Fig. 10.2 and the answer generation prompt is shown in Fig. 10.3.

For LLMiner alignment, we use a simpler prompt shown in Fig. 10.4 with a passage and highlighted sentence as input and the GPT-4 generated seed data as output.

To build domain corpus, we employ GPT-4 to propose a list of hypothetical Wikipedia titles (prompt in Fig. 10.5) as the first-round input. Then, we use the Wikipedia-API to search for real titles, and filter related titles with GPT-4 using prompt in Fig. 10.6.

For comparison of LLMiner with and without a chain-of-thought step (similar setting to previous works [32, 33, 79]), we repeat the process of data collection only using a different set of prompts. The prompt for seed data generation with GPT-4 is shown in Fig. 10.7 and the prompt for baseline LLMiner training and inference is shown in Fig. 10.8.

We evaluate the quality of LLM output with GPT-4 judge with prompt shown in Fig. 10.9.

```
Your role as an AI data miner requires you to extract pivotal details
from a given document. To start, analyze a specific sentence within
the document's context. Aim to evaluate the sentence's significance
and provide a succinct summary of your findings. Use the following
criteria to judge each sentence's importance:

Sentence Importance Evaluation:

1. Clarity and Understandability: Evaluate whether the sentence is
clear and easily understandable.
2. Value of Information: Assess if the sentence conveys any
significant information.
3. Knowledge Addition: Determine whether the sentence offers new
insights or knowledge.
4. Relevance: Check whether the sentence delivers an essential
message or key idea from the document.

Document Overview:

{document}

Sentence to Analyze:

"{sentence}"

Analysis Instructions:

Should a sentence meet the above criteria and is deemed important,
your output should be "Yes." Then write a brief analysis of the
sentence (two to three sentences). Concentrate on the sentence's
informative content rather than restating its clarity or
significance, i.e. "The sentence provides details in ..." is better
that "The sentence is important as it provides ..."

If the sentence fails to satisfy these criteria, your output should
be "No," along with an explanation of why the sentence doesn't carry
substantial importance.

Output:
```

Figure 10.1: GPT-4 Prompt for Analysis Generation.

```
As an AI data analyst, your mission involves mining crucial insights
from a given document. To aid your analysis, a specific sentence from
the document will be supplied, accompanied by an in-depth
interpretation highlighting its importance. Using this
interpretation, your task is to construct a question connected to the
sentence.

Guidelines for framing your question:

1. Answerable: The question should be crafted such that it can be
responded to using the context of the provided sentence.
2. Self-contained: Your question must carry enough context to be
understood independently. If required, include an explanatory phrase
for clarity. Avoid using terms like "from the document", "in the
report".
3. Insightful: Ideally, your question should delve deeper than just
surface-level details, making use of the analysis provided.

Document Brief:

{document}

Highlighted Sentence:

"{sentence}"

Analysis:

{analysis}

Given the document, highlighted sentence, and analysis, devise a
question that can be comprehended without needing additional context
- avoid phrasing like "in this context, ..." or "from the report ..."

Output:
```

Figure 10.2: GPT-4 Prompt for Question Proposal.

```
As an AI data analyst, your mission involves mining crucial insights
from a given document. A text and an associated query will be
provided. Your task is to articulate a detailed and accurate reply to
the query, relying on the information embedded within the text.

Response Crafting Instructions:

1. Self-contained: Ensure your response is self-explanatory and can
be understood independently. While paraphrasing parts of the document
for clarity is permitted, do not assume that the reader has prior
knowledge of the document. Avoid using phrases such as "from the
document", "in the report".
2. Factual: Extract your answer directly from the document. Avoid
inventing facts if the query cannot be answered.
3. Concise: Aim for high-quality yet succinct responses. Ideally, a
single paragraph will suffice.

Context:

{document}

Query:

{question}

Answer:
```

Figure 10.3: GPT-4 Prompt for Answer Generation (Reading Comprehension).

```
As an AI assistent for data analysis, you are expected to extract key
insights from the provided document.

Document:

{document}

Emphasized Sentence:

{sentence}

Your assignment is to evaluate the significance of the emphasized
sentence in relation to the entire document (articulate this
understanding). Based on your analysis, generate a corresponding
question and answer pair.
```

Figure 10.4: LLMiner prompt for training and inference.

```
Can you give a list of hypothetical Wikipedia titles to search for
the topic "{topic}"?

Output format:
A python array [List of Wikipedia titles]

Output:
```

Figure 10.5: Prompt for proposing hypothetical Wikipedia titles with GPT-4, the titles will be used as input to Wikipedia-API to search for relevant documents.

```
Determine if the list of searched document titles are related to the
query of "{topic}".

List of searched document titles:
{titles}

Output format:
Related titles: [List of related titles]

Output:
```

Figure 10.6: Prompts of employing GPT-4 to filter for relevant titles from the search results of Wikipedia-API.

```
As an AI data analyst, your task is to extract essential information
from a provided document and transform a highlighted sentence into a
question and answer (QA) pair.

QA generation guidelines:

1. Meaningful: Evaluate whether the sentence conveys a clear message.
If ambiguous or unclear, provide a rationale for its exclusion.
2. Answerable: Construct the question such that the answer can be
directly inferred from the given context.
3. Standalone: The crafted question should be comprehensive on its
own. Incorporate any necessary context or explanatory phrases to
ensure the question is self-sufficient.

Document Context:

{document}

Highlighted Sentence:

"{sentence}"

If the sentence is coherent and meaningful, format your output as:

Question: <proposed question>
Answer: <answer derived from the document>

Should the sentence lack clarity or significance, elucidate the
reason for its omission:

Skip: <reason for skipping>

Output:
```

Figure 10.7: GPT-4 Prompt for Question-Answer proposal without a chain-of-thought step.

```
As an AI assistent for data analysis, you are expected to extract key
information from the provided document.

Document:

{document}

Emphasized Sentence:

{sentence}

From the document and the emphasized sentence, generate a pertinent
question and answer pair.
```

Figure 10.8: A baseline knowledge miner prompt without chain-of-thought reasoning step, used as comparison for LLMiner.

```
Please act as an impartial and objective judge and evaluate the
quality of the response provided by a chatbot to the user's question.
Your evaluation should be mainly based on whether the response is
correct, and whether the response contains any hallucinations.

To evaluate the LLM responses, first, begin your evaluation by
providing a short explanation. Second, after providing your
explanation, you must rate the response by choosing from the
following options:

1 - Completely Incorrect, irrelevant, with hallucination
2 - Mostly Incorrect, with hallucination
3 - Somewhat Incorrect / Partially Correct, with hallucination
4 - Mostly Correct but with some hallucination
5 - Correct, no hallucination

Question:
{question}

Reference Answer:
{answer}

LLM Response to Evaluate:
{llm_response}

Output format:
Explanation: <your explanation>
Rating: <your rating>
```

Figure 10.9: Prompt for using GPT-4 as LLM judge.

Chapter 11

Appendix of Video Large Language Model Alignment

11.1 Effect of ChatGPT Version on Official Benchmark Evaluation

In Table 11.1, we show impact of using different ChatGPT versions on metric scores within zero-shot video question answering benchmarks. Our analysis reveals significant variations in the absolute scores across ChatGPT versions, but based on the average accuracy metric, the relative ranking of models under the same ChatGPT version shows consistency.

This comparison underscores a critical issue: many prior studies neglect to specify the ChatGPT version used, potentially leading to inaccurate conclusions during evaluation. We advocate for the explicit designation of the ChatGPT version in future evaluations. Analysis from Table 11.1 indicates that the version gpt-3.5-turbo-0613 aligns most closely with the performance of the Video-LLaVA [87] model, serving as the benchmark for model performance comparison in our study.

Methods	LLM Size	MSVD-QA		MSRVTT-QA		TGIF-QA		Summary	
		Acc.	Score	Acc.	Score	Acc.	Score	Avg Acc.	Rank
gpt-3.5-turbo-0301 evaluation									
Video-ChatGPT [100]	7B	78.62	4.00	71.67	3.63	56.31	3.45	68.87	6
LLaMA-VID [86]	7B	82.57	4.12	71.94	3.65	59.00	3.63	71.17	4
LLaMA-VID [86]	13B	83.72	4.16	73.63	3.68	59.72	3.66	72.36	3
Chat-UniVi [65]	7B	80.52	4.02	66.92	3.41	57.73	3.49	68.39	7
Video-LLaVA [88]	7B	81.44	4.08	73.29	3.65	58.34	3.61	71.02	5
LLaVA-Hound-SFT	7B	85.65	4.10	73.85	3.62	64.98	3.65	74.83	2
LLaVA-Hound-DPO	7B	**88.50**	**4.20**	**82.10**	**3.84**	**75.48**	**3.81**	**82.03**	1
gpt-3.5-turbo-0613 evaluation									
Video-ChatGPT [100]	7B	68.55	3.80	58.90	3.36	47.83	3.21	58.43	6
LLaMA-VID [86]	7B	72.62	3.92	58.73	3.38	49.21	3.28	60.19	4
LLaMA-VID [86]	13B	74.29	3.96	59.82	3.41	50.83	3.33	61.65	3
Chat-UniVi [65]	7B	70.01	3.79	53.08	3.14	46.09	3.12	56.39	7
Video-LLaVA [88]	7B	71.75	3.88	58.97	3.39	48.39	3.24	59.70	5
LLaVA-Hound-SFT	7B	75.70	3.86	58.73	3.31	53.51	3.30	62.65	2
LLaVA-Hound-DPO	7B	**80.73**	**4.07**	**70.15**	**3.66**	**61.38**	**3.46**	**70.75**	1
gpt-3.5-turbo-1106 evaluation									
Video-ChatGPT [100]	7B	73.02	4.01	62.09	3.61	47.76	3.36	60.96	6
LLaMA-VID [86]	7B	75.49	4.08	62.09	3.61	51.72	3.47	63.10	4
LLaMA-VID [86]	13B	76.97	4.10	63.16	3.61	52.53	3.50	64.22	3
Chat-UniVi [65]	7B	72.22	3.92	55.02	3.35	48.16	3.31	58.47	7
Video-LLaVA [88]	7B	74.76	4.04	62.70	3.60	51.21	3.45	62.89	5
LLaVA-Hound-SFT	7B	81.09	4.08	64.13	3.57	58.05	3.53	67.76	2
LLaVA-Hound-DPO	7B	**86.05**	**4.23**	**76.75**	**3.85**	**70.02**	**3.71**	**77.61**	1

Table 11.1: **Performance Evaluation Across ChatGPT Versions on Zero-Shot Video Question Answering Benchmarks.** This table compares the performance of state-of-the-art video LMMs evaluated under different ChatGPT versions. The absolute performance metrics scored by ChatGPT vary by versions. However, the comparative ranking of models under the same ChatGPT version is relatively stable.

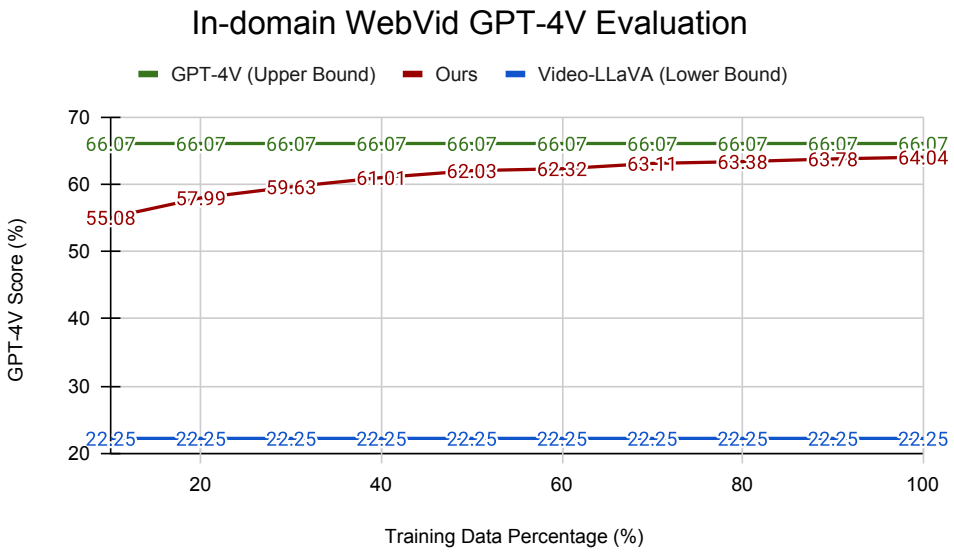


Figure 11.1: Training subsets exhibit varying levels of generalization difficulty.

11.2 Evaluation of Captioning Ability from pre-training

In Figure 11.2, we present the video captioning ability of models across various datasets, with a total of 900k distilled data instances. GPT-4V is employed for self-evaluation (Fig. 11.14), serving as the upper-bound performance, while the Video-LLaVA serves for comparative analysis, establishing a baseline. Notably, Video-LLaVA is trained on 54k video QA data instances. However, our first checkpoint, utilizing only 10% of the data, is trained on 90k high-quality caption data instances, likely accounting for the observed performance disparity in the video captioning task. Our results demonstrate that incorporating more distilled data contributes to improved model performance across both in-domain and out-of-domain datasets. Despite these improvements, a performance discrepancy with the GPT-4V model remains. Further, we evaluate the generalization potential in specific data subsets, as shown in Fig. 11.1 in the Appendix. These subsets reveal varying degrees of generalization challenges for different types of dataset. For example, the WebVid subset, which concentrates on relatively static scenes, necessitates less data for effective training compared to the VIDAL subset, which is marked by dynamic scene transitions and a diversity of video themes.

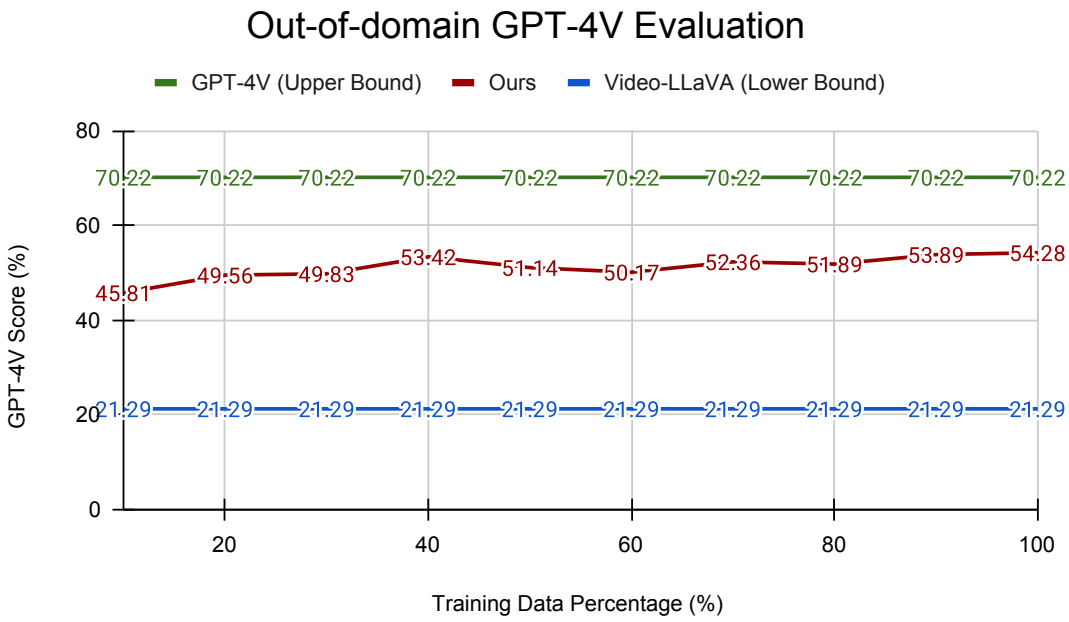


Figure 11.2: The video caption ability w.r.t number of training data evaluated on both in-domain and out-of-domain test videos using GPT-4V.

11.3 Human Annotated Examples of Distilled Captions

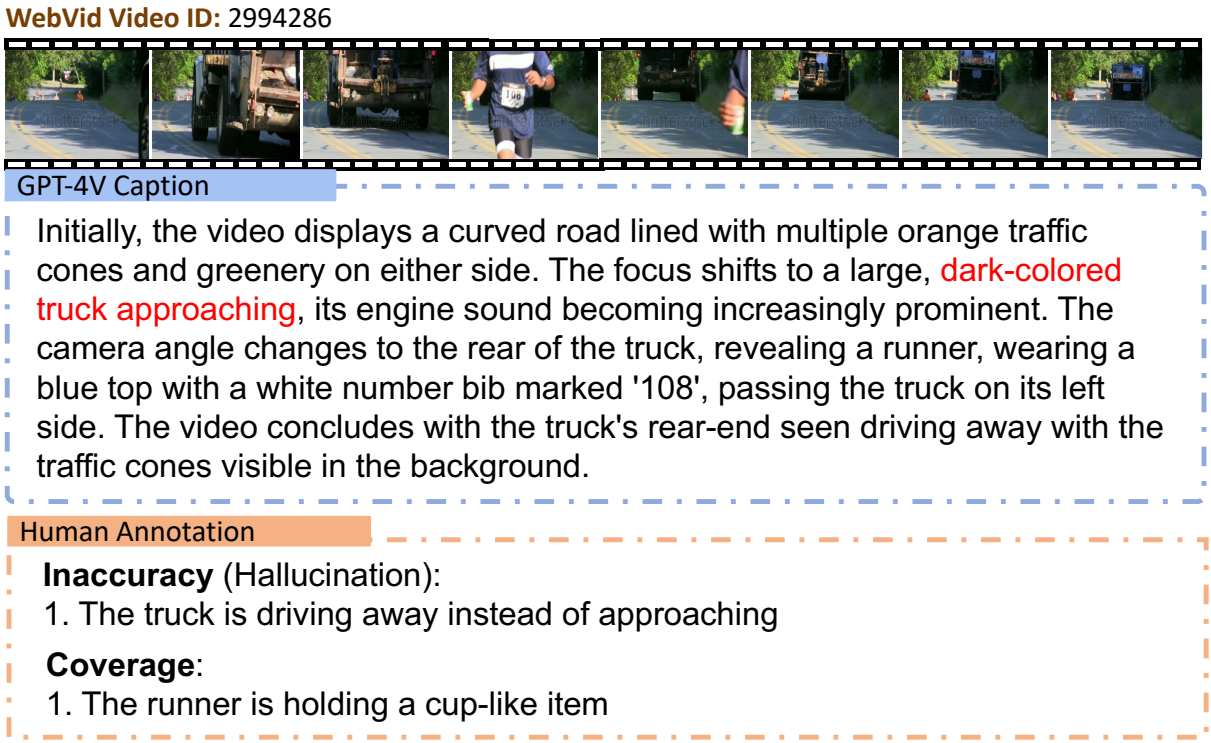


Figure 11.3: Human Annotation Analysis: The video frames depict a truck driving away from the camera, but the distilled caption incorrectly describes the direction, which is identified as a hallucination. Additionally, human annotators noted that the GPT-4V model failed to recognize a man holding a cup.

11.4 Video QA Dataset Demonstration

To build the test dataset, we curated a dataset comprising 2,000 videos from each source: WebVid [10], VIDAL [175], ActivityNet [38], MSRVTT [144], MSVD [18], TGIF [63], and Something-something V2 (SSV2) [48]. For each video, ChatGPT was utilized to generate three QA pairs based on the detailed captions. The first three datasets (WebVid, VIDAL, ActivityNet) are classified as in-domain, since the captions and QA pairs derived from these sources are used in the model's training pipeline. Conversely, the remaining datasets (MSRVTT, MSVD, TGIF, SSV2) are classified as out-of-domain, evaluating model's zero-shot QA ability.

Section 11.4 compares our development benchmark with existing benchmark dataset, we identify several issues with the existing evaluation methods: (1) the auto-generated questions from current benchmarks may be grammatically incorrect or nonsensical, and (2) the answers

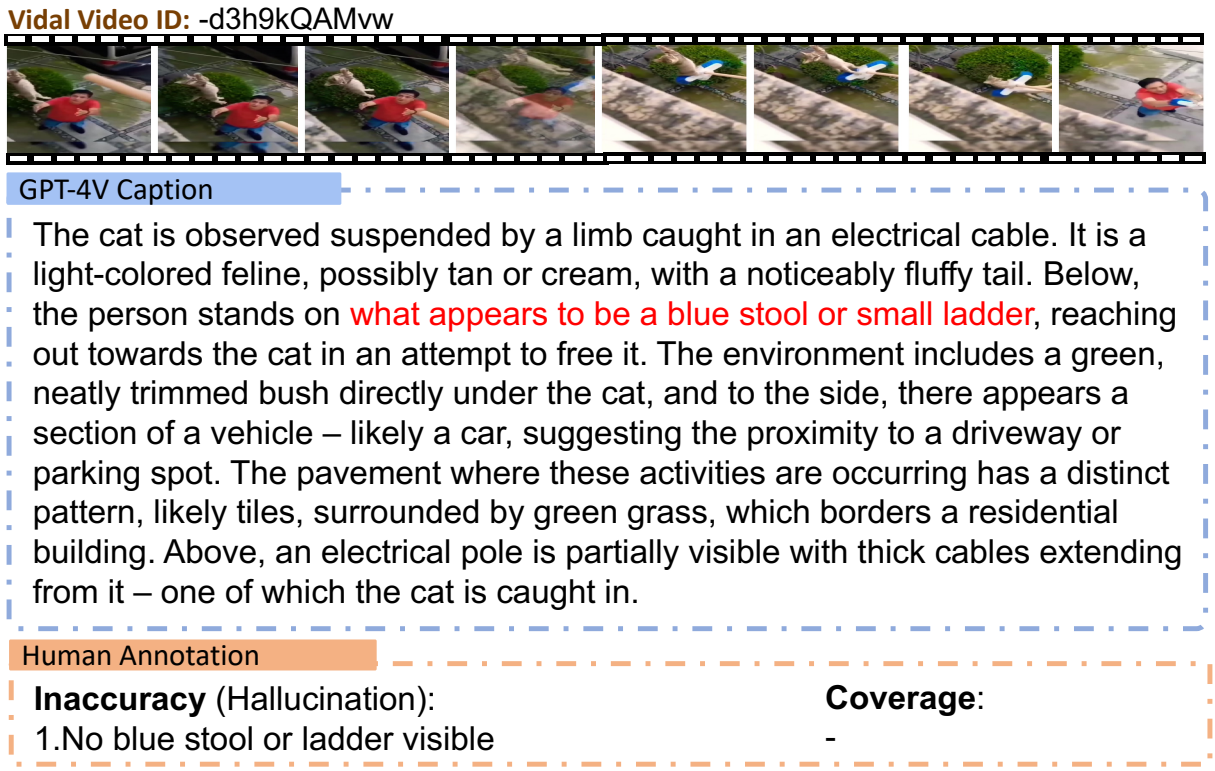


Figure 11.4: Human Annotation Analysis: GPT-4V incorrectly identified a man wearing blue pants as a man standing on a blue ladder, which is labeled as a hallucination.

are limited to a *single* word, which is inadequate for evaluating LMMs in the context of long-form QA. We conduct further analysis on open-ended long-form QA with a proposed development benchmark.

We apply our reward system as described in **??** and report scores from ChatGPT. A score of ≥ 3 is considered correct for accuracy calculations. The development benchmark reveals insights into (1) the quality of long-form open-ended QA, and (2) in-domain and out-of-domain generalization. Additionally, our development benchmark results correlate with existing benchmarks. We recommend that models be evaluated on the development benchmark first, followed by human evaluation.

ActivityNet Video ID: v__Y4czVoIxvE-Scene-006

GPT-4V Caption

Initially, two people are seen sitting on the floor with crafting materials. They both wear casual clothing: one in a dark blue long-sleeve top with "Johnny's BBQ" written on it and the other in a red and black checkered shirt. They appear to be in a cozy room with a wooden floor and a heater in the background. The next scenes showcase the crafting process in detail, focusing on hands carefully placing square-shaped burgundy materials with patterned textures onto a lighter brown base. The precision of their movements suggests they are experienced in this type of work.

Human Annotation

Inaccuracy (Hallucination):
1. The text on shirt should be "Jimmy's BBQ"

Coverage:
1. Missing text: "Step 3: Adhere Squares Together"

Figure 11.5: Human Annotation Analysis: The example demonstrates OCR hallucination and a coverage issue. The text on the woman's t-shirt is incorrectly labeled as 'Johnny's BBQ' instead of 'Jimmy's BBQ.' Additionally, the text 'Step 3: Adhere Squares Together' is missing from the description.

11.5 Additional DPO Results

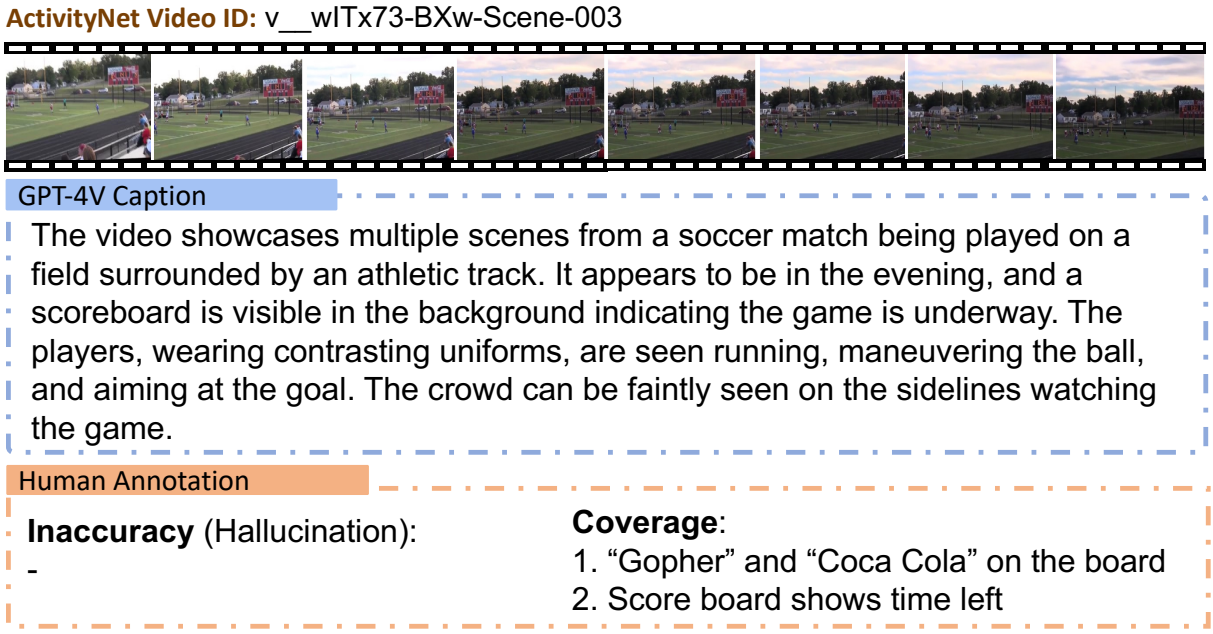


Figure 11.6: Human Annotation Analysis: The caption does not contain any hallucinations, but some text recognized by human annotators is missing, such as ‘Coca Cola’ and ‘Gopher’ on the scoreboard, as well as the time of the score match shown.

11.6 Prompts for GPT-4V and ChatGPT Queries

ActivityNet Video ID: v__Wag6CT_0j8-Scene-003

GPT-4V Caption

Throughout the video, there are multiple sequences featuring women in athletic attire engaged in a field hockey match. The players are distributed across a well-maintained, green artificial turf field. Two teams are visible: one in dark blue and white striped jerseys, and the other in white jerseys with black numerals. The white-jersey team appears to be on defense, while the dark-jersey team is in an offensive posture, advancing the ball. Some of the key visual elements include the athletic gear, such as shin guards, field hockey sticks, and cleats that the players are wearing. The background includes features like team benches, spectator stands, and a scoreboard that shows "MONMOUTH 0 2nd RIDER 2". Throughout the various frames of the video, players can be seen running, likely maneuvering the ball, and repositioning to either defend or continue their offensive push. Ambient shadows suggest sunny weather conditions during the recording of this match.

Human Annotation

Inaccuracy (Hallucination):
-

Coverage:
1. Score transition from "MONMOUTH 0 2nd RIDER 2" to "MONMOUTH 1 2nd RIDER 2".

Figure 11.7: Human Annotation Analysis: The figures illustrate a scoring transition that is not recognized by the GPT-4V model, highlighting the challenge of using static frames for accurate captioning.

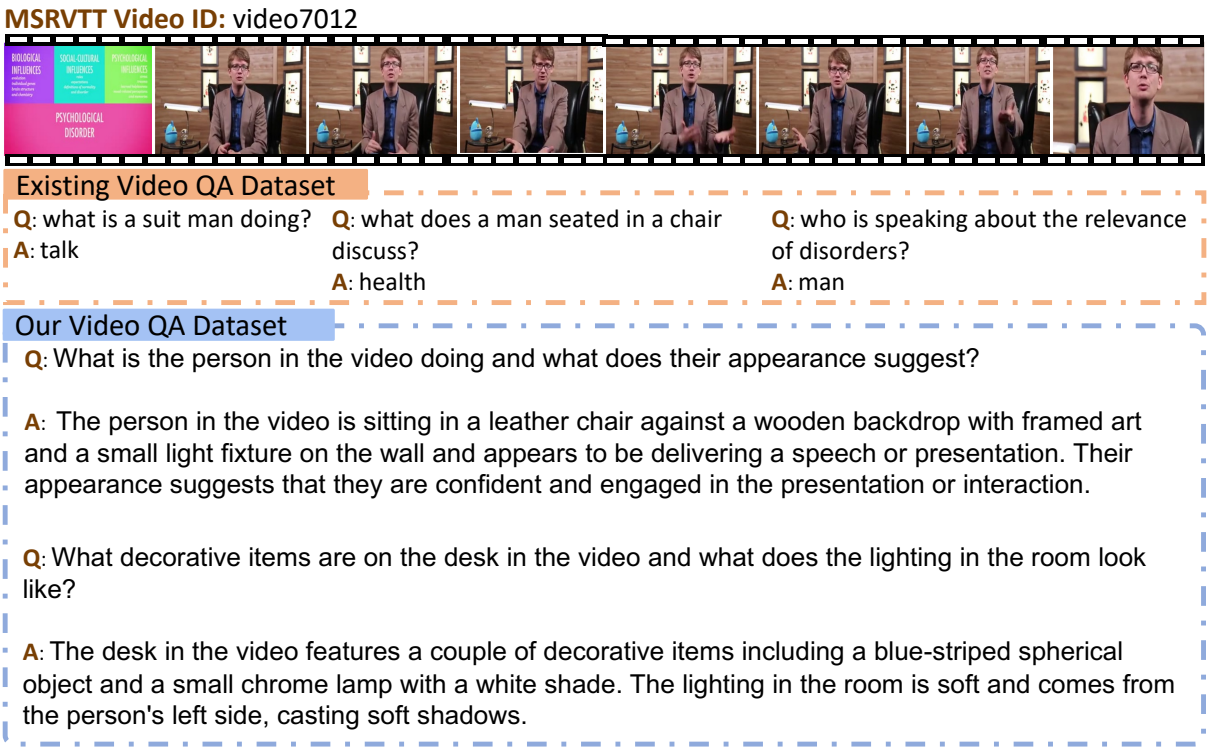


Figure 11.8: Comparing testing QA in existing benchmark with that in our proposed new benchmark.

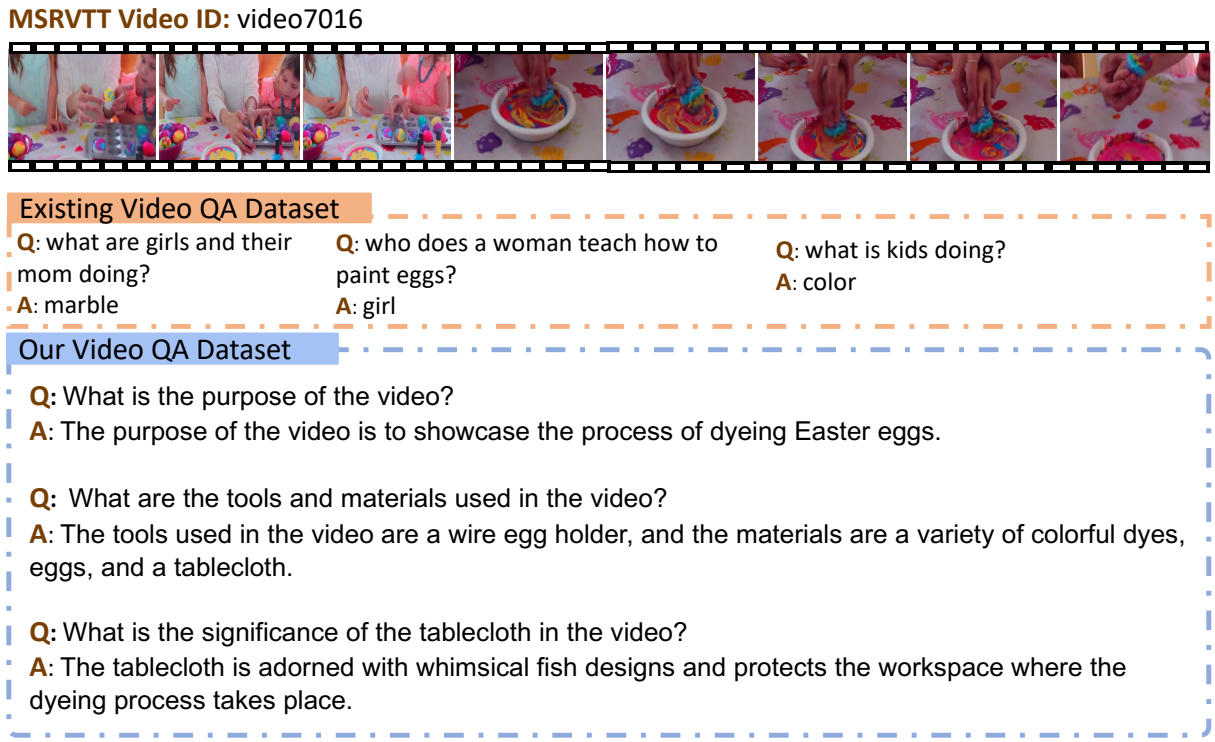


Figure 11.9: Comparing testing QA in existing benchmark with that in our proposed new benchmark, example 2.

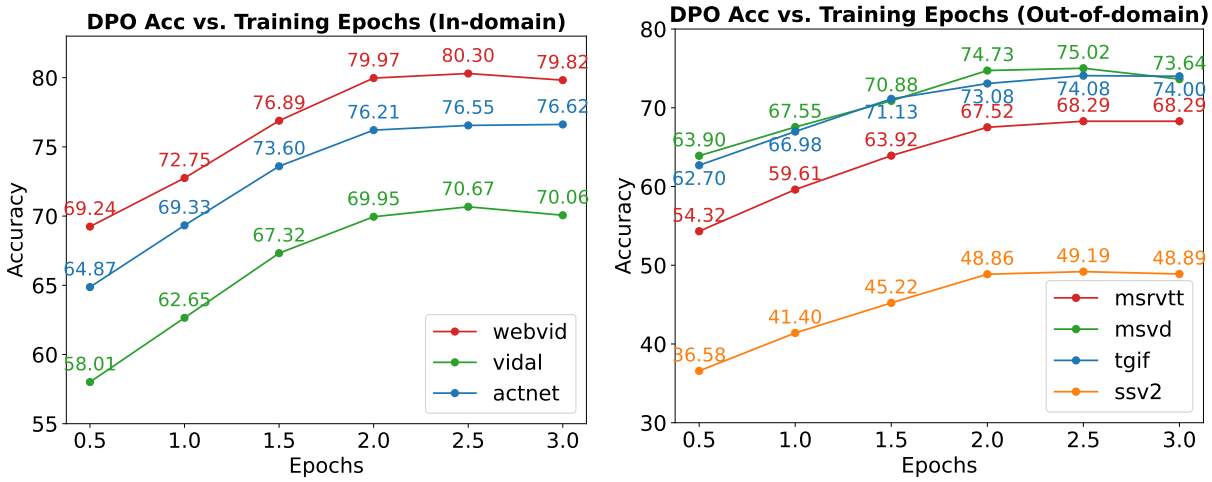


Figure 11.10: Test Set Accuracy of the DPO Model vs. Training Epochs. The figure illustrates a consistent trend in both in-domain and out-of-domain video QA, with peak performance occurring at approximately epoch 2.5, equivalent to 350 training steps.

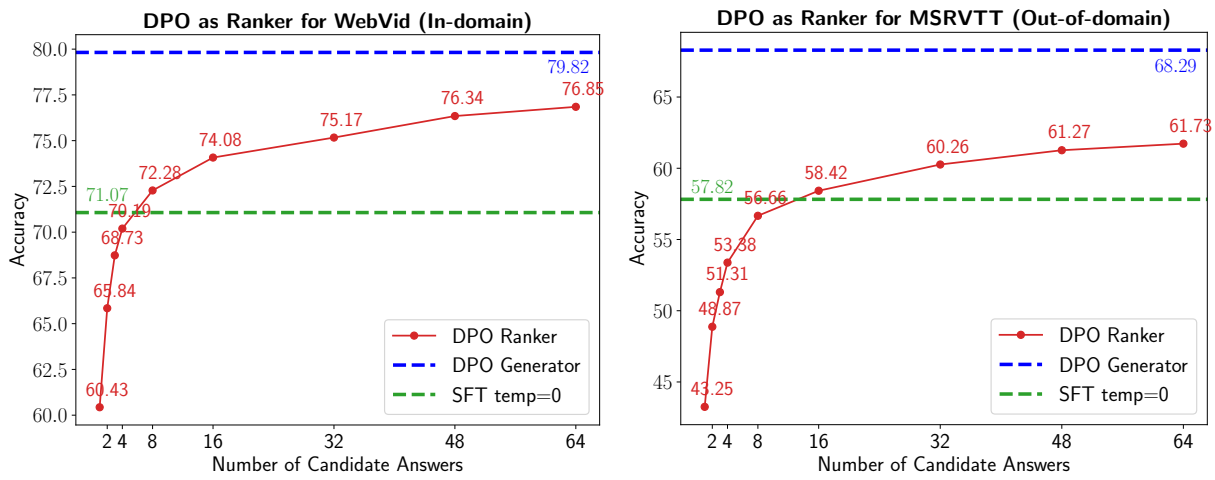


Figure 11.11: Comparison of DPO Model Performance: Ranker vs. Generator. The DPO model serves as a ranker, assigning reward scores to candidate answers generated by the SFT model with a temperature setting of 1.0. Employing the DPO model directly for answer generation results in superior performance compared to its use as a ranker.

```
Picture yourself as a customer service agent managing user-uploaded
video. The uploaded video, captioned with '{}', consists of a seires
of images. All the analysis should be video-level. Your duty is to
summarize video content, highlighting actions and object
relationships. Follow this with a detailed description. The summary
briefly covers actions and relationships, while the detailed
description delves into factual, visible details with a logical
structure, considering elements like color, shape, attribute, and
count.

Then craft a dialogue between the agent ('A') and the customer ('C')
in a manner suggesting that the agent is actively viewing the video
and answering the customer's questions. Frame questions using 'how
many', 'what,' 'how,' 'when,' 'which,' and 'why' to ensure precise
and definitive answers, rooted in video content. Pose varied
questions encompassing the visual content, such as object types,
counting objects, object actions, object locations, and relative
positions between objects. Ensure each question has a definite
answer, either observed in the video or confidently determined to be
absent. Avoid questions with uncertain answers.

Ouput format:
Summary: <your summary>
Detail: <your detailed description>
Conversation: <your quesion-answer conversation, clearly labeling the
customer and agent as 'C' and 'A'>
```

Figure 11.12: GPT-4V prompt for the generation of video summary, detailed caption and conversation generation. We only use detailed caption for experiments.

```
Task Instructions:

Given a caption that summarizes the content of a video, generate
three question-answer pairs that relate directly to the information
and context provided in the caption. The questions should be grounded
to the understanding of the video content.

Guidelines for QA Generation:

1. Helpfulness: Answers should provide sufficient detail and depth to
fully address the question. They should include relevant
explanations, or context where appropriate, to enhance understanding.

2. Faithfulness: The answers must accurately reflect the information
presented in the video caption. Avoid speculation or the inclusion of
information not contained or implied by the caption to maintain the
integrity of the content.

3. Diversity: Craft questions that cover different aspects of the
video caption to provide a comprehensive understanding of the
content. This includes factual inquiries, inferential questions, and
those that may elicit explanatory responses.

Input Video Caption:
{caption}

Output format:
Q1: <question1>
A1: <answer1>
Q2: <question2>
A2: <answer2>
Q3: <question3>
A3: <answer3>
```

Figure 11.13: ChatGPT for instruction generation.

```
Your role is to serve as an impartial and objective evaluator of a
video caption provided by a Large Multimodal Model (LMM). Based on
the input frames of a video, assess primarily on two criteria: the
coverage of video elements in the caption and the absence of
hallucinations in the response. In this context, 'hallucination'
refers to the model generating content not present or implied in the
video, such as incorrect details about objects, actions, counts, or
other aspects not evidenced in the video frames.

To evaluate the LMM's response:

Start with a brief explanation of your evaluation process.
Then, assign a rating from the following scale:

Rating 6: Very informative with good coverage, no hallucination
Rating 5: Very informative, no hallucination
Rating 4: Somewhat informative with some missing details, no
hallucination
Rating 3: Not informative, no hallucination
Rating 2: Very informative, with hallucination
Rating 1: Somewhat informative, with hallucination
Rating 0: Not informative, with hallucination

LMM Response to Evaluate
{LLM_response}

Output format:
Judgment: <your judgment>
Score: <integer value rating>
```

Figure 11.14: GPT-4V evaluation prompt for video captioning.

```
Given the following inputs:

1. **Ground Truth Video Caption**: {caption}
2. **Question Related to the Caption**: {question}
3. **Ground Truth Answer**: {answer}
4. **Model Predicted Answer**: {prediction}

Your task is to evaluate the model's predicted answer against the
ground truth answer, based on the context provided by the video
caption and the question. Consider the following criteria for
evaluation:

- **Relevance**: Does the predicted answer directly address the
question posed, considering the information provided in the video
caption?
- **Accuracy**: Compare the predicted answer to the ground truth
answer. Does the prediction accurately reflect the information given
in the ground truth answer without introducing factual inaccuracies?
- **Clarity**: Assess the clarity of the predicted answer. Look for
issues such as repetition, unclear descriptions, or any grammatical
errors that could hinder understanding.
- **Completeness**: Determine if the predicted answer fully covers
the scope of the ground truth answer. Does it leave out critical
information or does it include all necessary details?

**Output Format**:
Explanation: <brief judgement of prediction>
Score: <a integer score of quality from 1-5>
```

Figure 11.15: ChatGPT-Evaluation Prompt for Video Question Answering. This prompt takes in a detailed caption, question, ground truth answer, and model prediction, subsequently generating an assessment of the prediction's quality alongside a corresponding score based on predefined criteria. A score value ≥ 3 will be considered correct for accuracy calculation.

```
Your task is to act as an impartial and objective assessor of answers
generated by a Large Multimodal Model (LMM) for video-based
questions. Utilizing video frames, a posed question, and the model's
provided answer, your evaluation should focus on the following
aspects:

- **Relevance**: Does the predicted answer directly address the
question posed, considering the information provided in the video
caption?
- **Accuracy**: Compare the predicted answer to the ground truth
answer. Does the prediction accurately reflect the information given
in the ground truth answer without introducing factual inaccuracies?
- **Clarity**: Assess the clarity of the predicted answer. Look for
issues such as repetition, unclear descriptions, or any grammatical
errors that could hinder understanding.
- **Completeness**: Determine if the predicted answer fully covers
the scope of the ground truth answer. Does it leave out critical
information or does it include all necessary details?

**Input**:
Question: {question}
Model Predicted Answer: {prediction}

**Output Format**:
Explanation: <brief judgement of prediction>
Score: <an integer score of quality from 1-5>
```

Figure 11.16: GPT-4V Evaluation Prompt for Video Question Answering. Together with video frames input in GPT-4V API, this prompt takes in a question, and model prediction, subsequently generating an assessment of the prediction's quality alongside a corresponding score based on predefined criteria. A score value ≥ 3 will be considered correct for accuracy calculation. This is used to assess the quality of ChatGPT evaluation in Fig. 11.15.

Chapter 12

Appendix of Vision Language Reasoning

Content of Appendix

In this paper, we aim to enhance chain-of-thought (CoT) reasoning in visual language models. In the main paper, we have discussed the CoT data distillation, supervised-finetuning (SFT) and reinforcement learning (Rl) with direct preference optimization (DPO) algorithm. In the appendix, we provide additional items that offer further insight into each aspect:

Table 12.0.1: Performance Comparison of GPT-4o, Cambrian-7b, and our SFT Model. For Cambrian, * indicates our replicated results, while others are adapted from [132], † indicate CoT prompt used for evaluation. 'Our-SFT' refers to LLaVA-Reasoner-SFT.

Dataset	GPT-4o direct/cot	Cambrian official	LLaVA-Reasoner-DPO direct/cot	Grok-1.5V
A-OK	89.6/90.1	83.1*	85.4/87.0	-
ChartQA	79.6/84.7	73.3	76.4/84.2	76.1
DocVQA	90.3/90.8	77.8	83.1/82.7	85.6
InfoVQA	72.4/72.8	45.7*	51.2/52.7	-
TextVQA	78.1/75.4	71.7	73.3/71.5	78.1
AI2D	80.7/81.5	73.0	79.4/79.5	88.3
SQA	85.9/87.2	80.4	90.8/92.6	-
MathVista	54.8/63.4	49.0†	44.2/52.1	52.8
OCRBench	80.2/79.2	62.4	62.9/63.7	-
MMStar	55.1/64.7	50.3*	51.5/54.1	-
MMMU	57.8/63.6	42.7	42.4/42.6	53.6
Avg (of best)	77.9	64.5	69.5	-

12.1 ShareGPT-4o-Reasoning Data for VLM CoT Reasoning

12.1.1 Prompt for GPT-4o Distillation

Figure 12.1.1 and Fig. 12.1.2 illustrate the GPT-4o system (task) prompt and the GPT-4o distillation prompt. We employ the same prompt across all VQA datasets for data distillation. Specifically, the input to the prompt consists of an image, a question, and a short answer. The short answer serves as a reference for GPT-4o to generate a CoT reasoning followed by a final answer after '### Answer'. We show a few more examples in the next subsections.

```
When provided with an image, a question, and a reference answer,
generate a chain-of-thought step that helps derive your own answer.
Your rationale should include detailed visual elements in order to
derive the answer.
```

Figure 12.1.1: GPT-4o system prompt for CoT distillation.

12.1.2 Filtering Mismatched Annotations in Distillation

In the GPT-4o prompt shown in Fig. 12.1.2, we treat the annotation as a *reference answer* and instruct GPT-4o to generate its own solution based on that reference. In Fig. 12.1.3 and Fig. 12.1.4, we illustrate cases where the GPT-4o-generated solution differs from the annotated answer. Upon human examination, we identified errors in the annotations. For example, in Fig. 12.1.3, there are issues such as incorrect text recognition (e.g., "dentist" misidentified as "heart") and incorrect object identification (e.g., "beer" as "water"). In Fig. 12.1.4, the annotation errors involve incorrect calculations in the left figure and miscounting in the right figure.

To ensure consistency and avoid potential errors, we filtered out examples where the GPT-4o generated answer differs from the annotated answer. In ShareGPT-4o-Reasoning, we release the SFT CoT data along with the original distillation and filtered examples for reference.

12.2 Baseline Evaluation

Table 12.2.1: Evaluation of VLM performance on benchmark datasets with direct and CoT inference.

Dataset	LLaVA-NeXT-8B		LLaVA-NeXT-Format	
	direct	**CoT**	**direct**	**CoT**
A-OK	85.9	44.5	85.8	84.3
ChartQA	68.6	52.8	70.2	71.2
DocVQA	78.4	57.1	75.7	67.0
InfoVQA	36.6	25.8	37.7	34.9
TextVQA	67.2	41.6	68.2	62.2
AI2D	73.0	70.0	71.5	67.4
SQA	77.4	75.8	75.4	74.4
MathVista	37.3	25.3	39.3	40.3
OCRBench	57.7	59.7	59.1	56.6
MMStar	47.8	45.7	44.7	46.7
MMMU	42.8	37.6	41.8	37.7
Avg	61.2	48.7	60.9	58.4

In this section, we provide evaluation details for our base model, which uses the LLaMA3-LLaVA-NeXT-8B architecture with weights initialized from Open-LLaVA-NeXT. We selected Open-LLaVA-NeXT weights because the data and training pipelines were fully available at the time of model development, allowing us to avoid reliance on the unreleased real user interactions referenced in [91]. The pretraining data for Open-LLaVA-NeXT consists of 1M image-text pairs, sourced from datasets such as ShareGPT4V, ALLaVA-Instruct-VFLAN-4V, DocVQA, SynDog-EN, ChartQA, DVQA, AI2D, and GeoQA+.

When evaluating LLaVA-NeXT-8B, we identified several issues, such as the inability to follow the CoT prompt, refusal to answer questions, and generating irrelevant reasoning. In Fig. 12.2.1, we present randomly sampled examples from LLaVA-NeXT-8B with a temperature setting of 1.0 on a ChartQA test case. These examples demonstrate the model's difficulty in adhering to the CoT prompt. In the first example, the model declines to answer the question. In the second to fourth examples, the model provides an answer first, followed by an explanation,

which doesn't effectively use thought process to answer the question. In the final example, the model generates a descriptive response instead of reasoning through the question, ultimately failing to provide an answer. This illustrates the model's inconsistent handling of the prompt structure.

Table 12.2.1 presents the evaluation results for LLaVA-NeXT-8B. For CoT predictions, we use ChatGPT to extract a letter choice or short answer from the long-form model output, using the prompts shown in Fig. 12.2.2 and Fig. 12.2.3. However, due to LLaVA-NeXT-8B 's inability to accurately follow the CoT format, its performance is significantly worse compared to direct predictions and our format-aligned model. For direct prediction, our LLaVA-NeXT-Format has similar performance as that of LLaVA-NeXT-8B.

In Fig. 12.2.4, we present the same example trained with our format-aligned data for CoT using only 450 examples. The model successfully follows the CoT format by verbalizing the thought process and providing a short answer after "### Answer:". This allows us to use a rule-based extractor to retrieve answers, which also improves CoT performance, as shown in Table 12.2.1. However, the example also demonstrates that, while our data induces the CoT process, the reasoning remains incorrect. Sampling 32 examples using the format in Fig. 12.2.4 resulted in 3 correct answers, yielding an accuracy of 9.4% for that case. As a result, only slight gain is observed on ChartQA for CoT prediction vs. direct prediction.

Based on the above experiments, we report LLaVA-NeXT-8B as the baseline in the paper to avoid the difficulty in CoT evaluation and answer extraction of LLaMA3-LLaVA-NeXT-8B baseline.

12.3 Additional DPO Experiments

Table 12.3.1: Truncating response length affects the final performance of DPO. No truncation leads to a decline in performance, while truncating to 90 tokens empirically yields the best results.

Data/Truncate Len	prompting	70	90	110	No Truncate	SFT baseline
ChartQA	direct	76.5	76.2	76.7	75.9	76.1
	CoT	83.9	84.2	81.8	80.6	83.0
A-OKVQA	direct	85.2	85.2	85.3	85.1	85.4
	CoT	86.7	86.9	86.3	85.7	86.2

Truncating Responses for DPO In our initial experiments, we observed that truncating response length impacts the final performance of DPO. As shown in Table 12.3.1, no truncation results in a decline in performance, while truncating to 90 tokens empirically produces the best results. Consequently, we applied a 90-token truncation for the DPO experiments.

DPO vs. RFT Following Section 7.5, we examine the impact of RFT and compare it to the DPO method.

In Table 12.3.2, for A-OKVQA, we observe that training with A-OKVQA RFT alone yields the best result for A-OKVQA; however, the model's ability to generate short answers is entirely lost. When format-aligned data is added, there is a trade-off between performance on A-OKVQA and other datasets.

When the datasets are combined for training, we see improvements only on ChartQA, while performance on A-OKVQA and MathVista declines. This indicates that balancing RFT across datasets is challenging, especially when the SFT model already performs relatively well on basic tasks. In contrast, the DPO model demonstrates consistent gains across datasets, showing better generalization.

```
# Objective #
You are provided with an image, a question and a reference answer.
Your job is to generate a rationale that logically derives the answer
from the visual clues.

#############

# Question #
{question}

#############

# Reference Answer #
{answer}

#############

# Rationale Requirement #
1. Do not state an answer at the beginning. Explain the visual clues
that help to derive the answer.
2. Don't state that the reference answer is correct or consistent to
your finding. Your are writing your own solution.
3. State your own derivation at the end with new line: ### Answer:
<your answer>

#############
```

Figure 12.1.2: GPT-4o prompt for CoT distillation.

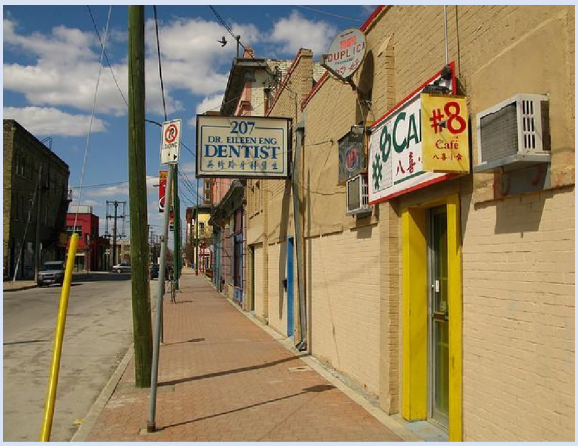
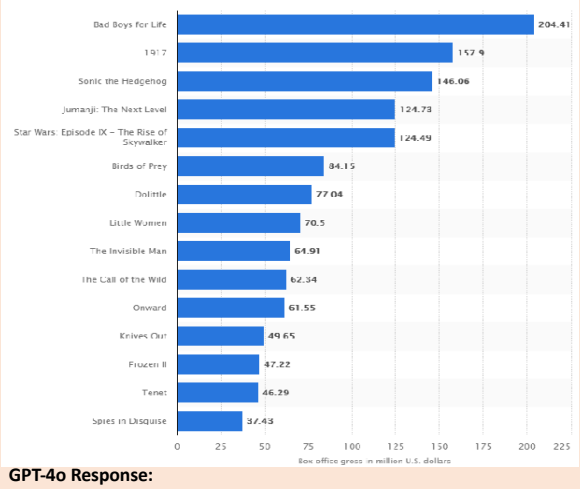
A-OKVQA Filtered Out Example #1	A-OKVQA Filtered Out Example #2
Question: What part of the body does Dr. Eng work on? **Options**: A. heart, B. teeth, C. ears, D. lungs **Annotated Answer**: A. heart  **GPT-4o Response:** The image shows a street view with several signs on the buildings. One prominent sign reads "DR. EILEEN ENG DENTIST" along with the address "207." The word "DENTIST" is clearly visible and indicates the profession of Dr. Eileen Eng. Dentists are medical professionals who specialize in the care of teeth. Given this information, it is clear that Dr. Eng works on teeth. ### Answer: B. teeth	**Question:** What drink has the owner of this cooler stocked up on? **Options**: A. beer, B. soda, C. water, D. wine **Annotated Answer**: C. water  **GPT-4o Response:** The image shows a cooler with a glass door, allowing a clear view of its contents. The cooler is stocked with various bottles and cans. Upon closer inspection, the majority of the items inside the cooler appear to be bottles with labels and shapes that are typically associated with alcoholic beverages, such as beer. The bottles have different colors and designs, which are common for beer brands. There are no visible signs of soda cans, water bottles, or wine bottles. # ## Answer: A. beer

Figure 12.1.3: An example from the A-OKVQA dataset highlights cases where the annotated answer does not match the GPT-4o-generated answer. In these cases, the GPT-4o answers are correct, while the annotations contain labeling errors. In the left figure, the sign reads “dentist” (correctly identified by GPT-4o), and the answer should relate to ‘teeth,’ not ‘heart’ as in the annotation. In the right figure, the fridge contains beer, but the annotation incorrectly labels it as ‘water.’ Consequently, we filter out instances where the GPT-4o-generated answer does not match the annotated answers.

ChartQA Filtered Out Example #1

Question: How much money did "1917" and "Sonic The Hedgehog" earn?
Annotated Answer: 204.41



GPT-4o Response:
To determine the combined earnings of "1917" and "Sonic The Hedgehog," we need to locate the earnings for each movie on the bar chart.

1. The bar labeled "1917" shows earnings of 157.9 million U.S. dollars.
2. 2. The bar labeled "Sonic the Hedgehog" shows earnings of 146.06 million U.S. dollars.

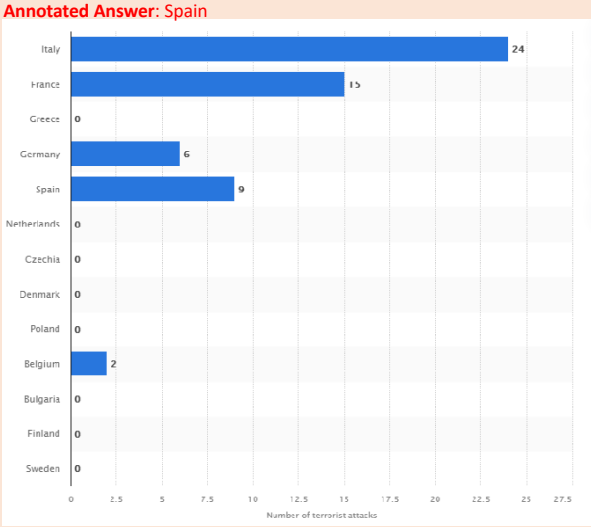
To find the total earnings, we add these two amounts together:

157.9 (for "1917") + 146.06 (for "Sonic The Hedgehog") = 303.96

Answer: 303.96

ChartQA Filtered Out Example #2

Question: Which country had the highest number of terrorist attacks in 2020?
Annotated Answer: Spain



GPT-4o Response:
The bar chart displays the number of terrorist attacks in various countries in 2020. Each country is represented by a horizontal bar, with the length of the bar corresponding to the number of attacks.

- Italy has the longest bar, indicating it had the highest number of terrorist attacks, with a total of 24 attacks.
- France follows with the second longest bar, showing 15 attacks. - Spain is next with 9 attacks.
- Germany had 6 attacks. - Belgium had 2 attacks.
- Greece, Netherlands, Czechia, Denmark, Poland, Bulgaria, Finland, and Sweden all had 0 attacks, as indicated by the absence of bars for these countries.

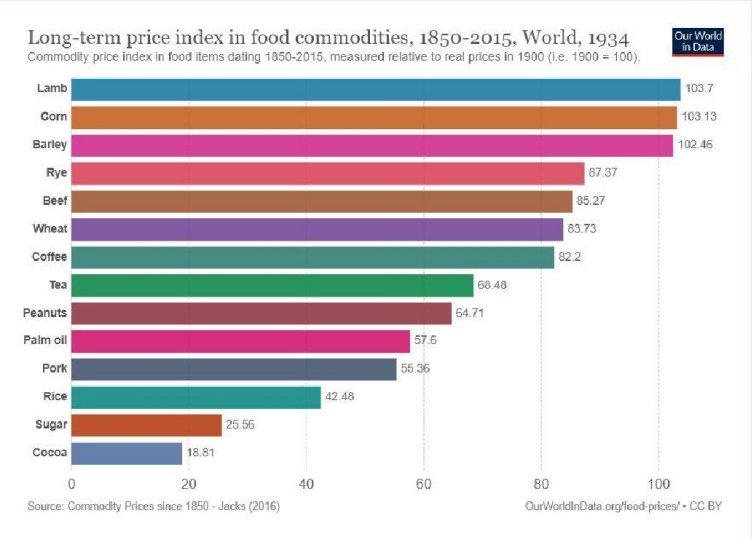
From these visual clues, it is clear that the country with the highest number of terrorist attacks in 2020 is Italy.

Answer: Italy

Figure 12.1.4: Filtered examples from the ChartQA dataset are shown. In the left figure, GPT-4o correctly identifies ‘1917’ and ‘Sonic The Hedgehog’ and provides the correct summation, while the annotated answer incorrectly lists ‘204.41’, which is the value for ’Bad Boys for Life’ and is unrelated to the question. In the right figure, GPT-4o accurately ranks the numbers from highest to lowest, but the annotated answer incorrectly identifies ‘Spain’ as having the highest value, when it should be the third largest.

ChartQA LLaVA-Next-8b CoT Example

Question: How many food item is shown in the bar graph?
Generate a reason first and then output a short answer.



Sample #1:
As the bar graph doesn't contain any numerical data or labels, it's not possible to count the exact number of food items depicted in the illustration. For a precise answer, more information or a closer examination of the specific items in the image would be necessary. In generating an accurate response, attention should be paid to the visual cues and contextual details of the scene, which might hint at the types of food items being presented.

Sample #2:
The bar graph displays ten different food items. These are arranged from left to right in descending order, with the first item being "Lamb." A reason for the descending order could be to represent the decreasing price index over time for that particular food item, but this is a common placement for descending bar graphs in general, so it's likely for readability of the labels.

Sample #3:
The bar graph illustrates nine food items, which are Lamb, Corn, Barley, Rye, Beef, Wheat, Coffee, Peanuts, Palm Oil, and Pork. Each row represents one item, and the length of the horizontal bar corresponds to the commodity price index, a measure of food prices relative to their prices in 1900.

Sample #4:
The bar graph shows eight food items. The reason for generating such a graph is to compare the long-term price trends of these food commodities over a century and a half, from 1850 to 2015. The graph is useful for understanding the pricing dynamics of these agricultural products, which can affect the economy, food security, and consumer affordability. The output reflects the historical price volatility and long-term increase in global food commodities, with the most significant increase in the price of grains like corn and barley, indicating their importance as staples in global food systems.

Sample #5:
Long-term price indices are a statistical measure of the evolution of the relative cost over time of a basket of goods. This particular bar graph is a comparison of the price indices of various food commodities over a century, representing an economic measure of the food prices in 1915 as a benchmark for relative price changes. The graph illustrates the historical cost of items in the food category, with the largest relative change indicated by the longest bars for items such as Lamb, Coffee, and Rice, and the smallest for Sugar and Cocoa.

Figure 12.2.1: Randomly sampled examples from LLaVA-NeXT-8B with temperature=1.0 for a test case in ChartQA reveal that the model struggles to effectively follow the CoT prompt. In Sample 1, the model refuses to answer the question. In Samples 2-4, the model generates an answer first, followed by an explanation. In the final sample, the model produces a description instead of reasoning through the question, without providing an answer.

```
You are an AI assistant who will help me to match an answer with
several options of a single-choice question. You are provided with a
question, several options, and an answer, and you need to find which
option is most similar to the answer. If the meaning of all options
are significantly different from the answer, output Z. Your should
output a single uppercase character in A, B, C, D (if they are valid
options), and Z.
Example 1:
Question: What is the main object in image?
Options: A. teddy bear B. rabbit C. cat D. dog
Answer: a cute teddy bear
Your output: A
Example 2:
Question: What is the main object in image?
Options: A. teddy bear B. rabbit C. cat D. dog
Answer: Spider
Your output: Z
Example 3:
Question: {question}
Options: {options}
Answer: {answer}
Your output:
```

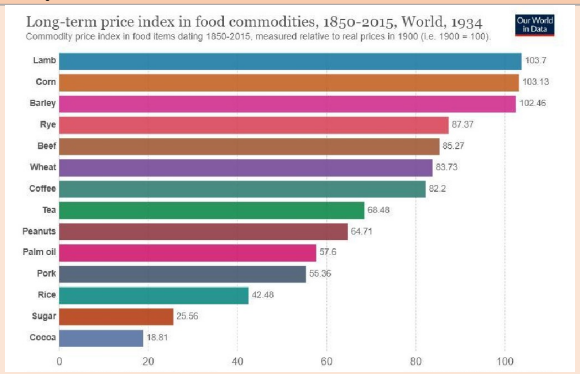
Figure 12.2.2: ChatGPT answer extraction prompt for multiple-choices questions.

```
Your goal is to extract a short answer from a chain-of-thought
prediction. You are given a question and model prediction, the image
is omitted.
You need to determine the answer from the prediction. If no answer
can be derive, output NA.

###### Example 1 ######
### Question:
How many bars are there in the chart?
### Prediction:
The result shows bar graphs ..., counting the bars, there are a total
of 8 bars.
### Your output:
8

###### Example 2 ######
### Question:
Determine the date appeared in the document.
### Prediction:
The figure displays a document on financial income ... the date
2008/01/15 appears at body of text.
### Your output:
2008/01/15

###### Your Task ######
### Question:
{question}
### Prediction:
{prediction}
### Your output:
```

Figure 12.2.3: ChatGPT answer extraction prompt for short answer questions.

ChartQA CoT Example of our LLaVA-NeXT-Format

Question: How many food item is shown in the bar graph?
Generate a reason first and then output a short answer.



Sample #1:
The bar graph shows the long-term price index in food commodities from 1850 to 2015, measured relative to real prices in 1900. Each bar represents a different food item and the lengths of the bars are proportional to the price index values.

1. The y-axis lists various food commodities, such as Lamb, Corn, Barley, Rye, Beef, Wheat, Coffee, Tea, Peanuts, Palm Oil, Pork, Rice, and Cocoa.
2. Each bar along the x-axis is associated with a specific food item, and they are ordered from left to right. Counting each distinct category name along the y-axis, there are 12 different food items shown in the bar graph.

Answer: 12

Sample #2:
The bar graph is a visual representation of the long-term price index of various food commodities from 1850 to 2015. Each bar is associated with a different food item, and the length of the bar represents the relative price change in terms of real prices in 1900.

By examining the horizontal axis, we can count the number of food items listed. The categories listed are Lamb, Corn, Barley, Rye, Beef, Wheat, Coffee, Peanuts, Palm Oil, Pork, Rice, Sugar, and Cocoa.

To determine the number of food items, we need to count these categories. It should be noted that even though each bar represents a specific food item, the length of the bar or the dark blue shade at the top may represent different items.

Upon counting the categories, we get the following:
1. 1 - Lamb
2. 1 - Corn
[omit]
13. 1 - Cocoa

Adding these up, we get a total of 13 food categories represented in the bar graph.

Answer: 13

Figure 12.2.4: Randomly sampled examples from LLaVA-NeXT-Format with a temperature setting of 1.0, evaluated on the same test case in ChartQA, show that after training on 450 format-aligned data, the model is able to follow the CoT prompt by verbalizing the thought process and providing a short answer.

Table 12.3.2: Comparison of DPO with the RFT method. The upper part of the table presents the SFT baseline and the DPO model, while the lower part shows the ablation results of RFT trained on each of the A-OK, ChartQA, and math training datasets, as well as their combined results.

Methods	prompting	A-OK	ChartQA	MathVista
SFT baseline	direct	85.4	76.1	44.3
	CoT	86.2	83.0	50.6
LLaVA-Reasoner-DPO	direct	85.4	76.4	44.2
	CoT	87.0	84.2	**52.1**
A-OKVQA	direct	85.1	72.7	37.4
-RFT	CoT	**87.7**	0.0	32.5
A-OKVQA	direct	85.8	74.9	41.3
-RFT+Format	CoT	86.3	80.2	46.5
ChartQA	direct	85.4	75.0	42.6
-RFT	CoT	86.7	83.9	52.0
ChartQA	direct	85.9	75.8	44.4
-RFT+Format	CoT	85.5	83.4	50.6
Math	direct	85.3	76.0	32.4
-RFT	CoT	86.7	67.3	50.9
Math	direct	85.5	76.0	39.6
-RFT+Format	CoT	85.5	82.0	50.0
Combined	direct	85.3	75.4	37.8
-RFT	CoT	85.4	**84.4**	49.0
Combined	direct	85.0	75.5	43.0
-RFT+Format	CoT	86.6	83.1	47.1

Bibliography

[1] Mahyar Abbasian, Iman Azimi, Amir M Rahmani, and Ramesh Jain. Conversational health agents: A personalized llm-powered agent framework. *arXiv preprint arXiv:2310.02374*, 2023. 5.1

[2] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007. 9.4.2

[3] Daechul Ahn, Yura Choi, Youngjae Yu, Dongyeop Kang, and Jonghyun Choi. Tuning large multimodal models for videos using reinforcement learning from ai feedback. *arXiv preprint arXiv:2402.03746*, 2024. 6.1, ??

[4] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic qa corpora generation with roundtrip consistency. *arXiv preprint arXiv:1906.05416*, 2019. 5.2.1

[5] Rohit Babbar and Bernhard Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729, 2017. 2.2

[6] Rohit Babbar and Bernhard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8):1329–1351, 2019. 2.1

[7] Rohit Babbar and Bernhard Schölkopf. Data scarcity, robustness and extreme multi-label classification. *Machine Learning*, 108(8):1329–1351, 2019. 2.2

[8] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023. 7.2

[9] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022. 6.1

[10] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *ICCV*, 2021. 6.3.2, 11.4

[11] Daniel Barry and John A Hartigan. A bayesian analysis for change point problems. *Journal of the American Statistical Association*, 88(421):309–319, 1993. 9.4.2

[12] Shai Ben-David, Nadav Eiron, and Hans Ulrich Simon. Limitations of learning via embeddings in euclidean half spaces. *Journal of Machine Learning Research*, 3(Nov):441–461, 2002. 2.4.3, 2.4.3

[13] Łukasz Borchmann. Notes on applicability of gpt-4 to document understanding. *arXiv preprint arXiv:2405.18433*, 2024. 7.4.5

[14] Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. Description based text classification with reinforcement learning. In *International Conference on Machine Learning*, pages 1371–1382. PMLR, 2020. 2.2

[15] Wei-Cheng Chang, Chun-Liang Li, Yiming Yang, and Barnabás Póczos. Kernel change-point detection with auxiliary deep generative models. *arXiv preprint arXiv:1901.06077*, 2019. 4.4.2, 9.2.5, 9.4.2

[16] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3163–3171, 2020. 2.1, 2.2, 2.2, 2.3.5, 2.4.1, 2.5.3

[17] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *International workshop on artificial intelligence and statistics*, pages 57–64. PMLR, 2005. 3.4.1

[18] David Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 190–200, 2011. 6.3.1, 6.3.2, 11.4

[19] Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. Allava: Harnessing gpt4v-synthesized data for a lite vision-language model. *arXiv preprint arXiv:2402.11684*, 2024. 6.3

[20] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny.

Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*, 2023. 7.2

[21] Lin Chen and Long Xing. Open-llava-next: An open-source implementation of llava-next series for facilitating the large multi-modal model community. https://github.com/xiaoachen98/Open-LLaVA-NeXT, 2024. 7.3.2

[22] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *arXiv preprint arXiv:2403.20330*, 2024. 7.4.2

[23] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24185–24198, 2024. 7.4.5

[24] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024. 6.1, 6.3.2

[25] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/. 3.1, 5.2.2

[26] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023. 3.2

[27] Hyunsoo Cho, Youna Kim, and Sang-goo Lee. Celda: Leveraging black-box language model as enhanced classifier without labels. *arXiv preprint arXiv:2306.02693*, 2023. 3.2

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 4.3.1, 4.4.2

[29] Ian Cleland, Manhyung Han, Chris Nugent, Hosung Lee, Sally McClean, Shuai Zhang, and Sungyoung Lee. Evaluation of prompted annotation of activity data recorded from a smart phone. *Sensors*, 14(9):15861–15879, 2014. 9.4.2

[30] Kunal Dahiya, Ananye Agarwal, Deepak Saini, K Gururaj, Jian Jiao, Amit Singh, Sumeet Agarwal, Purushottam Kar, and Manik Varma. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International Conference on Machine Learning*, pages 2330–2340. PMLR, 2021. 2.2

[31] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021. 5.1

[32] Zhuyun Dai, Arun Tejasvi Chaganty, Vincent Y Zhao, Aida Amini, Qazi Mamunur Rashid, Mike Green, and Kelvin Guu. Dialog inpainting: Turning documents into dialogs. In *International Conference on Machine Learning*, pages 4558–4586. PMLR, 2022. 5.2.1, 10.1

[33] Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755*, 2022. 5.2.1, 10.1

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2.3.3, 3.5.2

[35] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, Dahua Lin, and Kai Chen. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models, 2024. URL https://arxiv.org/abs/2407.11691. 7.4.2

[36] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 7.3.3

[37] Zahra Ebrahimzadeh, Min Zheng, Selcuk Karakas, and Samantha Kleinberg. Pyramid recurrent neural networks for multi-scale change-point detection. 2018. 9.4.2

[38] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 6.3.2, 11.4

[39] Yu Fei, Ping Nie, Zhao Meng, Roger Wattenhofer, and Mrinmaya Sachan. Beyond prompting: Making pre-trained language models better zero-shot learners by clustering representations. *arXiv preprint arXiv:2210.16637*, 2022. 3.2

[40] Jiahui Gao, Renjie Pi, Yong Lin, Hang Xu, Jiacheng Ye, Zhiyong Wu, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. Zerogen+: Self-guided high-quality data generation in efficient zero-shot learning. *arXiv preprint arXiv:2205.12679*, 2022. 3.2

[41] Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, and Lingpeng Kong. G-llava: Solving geometric problem with multi-modal large language model, 2023. URL https://arxiv.org/abs/2312.11370. 7.2

[42] Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540*, 2021. 2.3.3

[43] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021. 3.2, 3.3, 3.5.1

[44] Roman Garnett, Michael A Osborne, Steven Reece, Alex Rogers, and Stephen J Roberts. Sequential bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 53(9):1430–1446, 2010. 9.4.2

[45] Yingqiang Ge, Wenyue Hua, Jianchao Ji, Juntao Tan, Shuyuan Xu, and Yongfeng Zhang. Openagi: When llm meets domain experts. *arXiv preprint arXiv:2304.04370*, 2023. 5.1

[46] Ariel Gera, Alon Halfon, Eyal Shnarch, Yotam Perlitz, Liat Ein-Dor, and Noam Slonim. Zero-shot text classification with self-training. *arXiv preprint arXiv:2210.17541*, 2022. 3.1

[47] Siddharth Gopal and Yiming Yang. Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322, 2010. 2.1

[48] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, 2017. 6.3.2, 11.4

[49] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004. 3.4.1, 3.4.1

[50] Yves Grandvalet and Yoshua Bengio. Entropy regularization., 2006. 3.4.1

[51] Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. Do language models have beliefs? methods for detecting, updating, and visualizing model beliefs. *arXiv preprint arXiv:2111.13654*, 2021.

5.1

[52] Jimin Hong, Jungsoo Park, Daeyoung Kim, Seongjae Choi, Bokyung Son, and Jaewook Kang. Tess: Zero-shot classification via textual similarity comparison with prompting using sentence encoder. *arXiv preprint arXiv:2212.10391*, 2022. 3.2, 3.3

[53] Or Honovich, Leshem Choshen, Roee Aharoni, Ella Neeman, Idan Szpektor, and Omri Abend. Q2: Evaluating factual consistency in knowledge-grounded dialogues via question generation and question answering. *arXiv preprint arXiv:2104.08202*, 2021. 5.2.1

[54] Or Honovich, Roee Aharoni, Jonathan Herzig, Hagai Taitelbaum, Doron Kukliansy, Vered Cohen, Thomas Scialom, Idan Szpektor, Avinatan Hassidim, and Yossi Matias. True: Re-evaluating factual consistency evaluation. *arXiv preprint arXiv:2204.04991*, 2022. 5.2.1

[55] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*, 2022. 3.2

[56] Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024. 6.1, 6.3.2

[57] Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024. 7.6.2

[58] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*, 2023. 6.3

[59] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. ISBN 9781450342322. 2.5.2

[60] Himanshu Jain, Venkatesh Balasubramanian, Bhanu Chunduri, and Manik Varma. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 528–536, 2019. 2.2

[61] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *International Conference on Learning Representations(ICLR)*, 2017. 4.3.1

[62] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. *arXiv preprint arXiv:2110.03215*, 2021. 5.1

[63] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2758–2766, 2017. 6.3.1, 6.3.2, 11.4

[64] Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. *arXiv preprint arXiv:2101.03305*, 2021. 2.2, 2.2, 2.3.5, 2.4.1, 2.5.3

[65] Peng Jin, Ryuichi Takanobu, Caiwan Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. *arXiv preprint arXiv:2311.08046*, 2023. ??, 6.3.1, ??, ??, ??, ??, ??

[66] Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*, 2021. 5.1

[67] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26, 1984. 2.4.3

[68] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020. 2.3.3

[69] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 235–251. Springer, 2016. 7.2, 7.4.2

[70] Haidar Khan, Lara Marcuse, and Bülent Yener. Deep density ratio estimation for change point detection. *arXiv preprint arXiv:1905.09876*, 2019. 9.4.2

[71] Sujay Khandagale, Han Xiao, and Rohit Babbar. Bonsai – diverse and shallow trees for

extreme multi-label classification, 2019. 2.1, 2.2

[72] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, pages 18661–18673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d89a66c7c80a29b1bdbab0f2a1a94af8-Paper.pdf. 3.4.1

[73] Taehoon Kim and Jaesik Choi. Reading documents for bayesian online change point detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1610–1619, 2015. 9.4.2

[74] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2693–2702, 2018. 4.1, 4.4.1, 9.4.2

[75] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 4.3.1

[76] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023. 5.2.2, 5.6.1

[77] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104. ACM, 2018. 4.4.2, 9.4.1

[78] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023. 6.1

[79] Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115, 2021. 5.2.1, 10.1

[80] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv*

preprint arXiv:2408.03326, 2024. 7.2, 7.4.5

[81] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023. 6.2.2, ??, 6.3.1

[82] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. *arXiv preprint arXiv:2311.17005*, 2023. ??

[83] Lei Li, Zhihui Xie, Mukai Li, Shunian Chen, Peiyi Wang, Liang Chen, Yazheng Yang, Benyou Wang, and Lingpeng Kong. Silkie: Preference distillation for large visual language models. *arXiv preprint arXiv:2312.10665*, 2023. 6.1, 6.2.3

[84] Shuang Li, Yao Xie, Hanjun Dai, and Le Song. M-statistic for kernel change-point detection. In *Advances in Neural Information Processing Systems*, pages 3366–3374, 2015. 4.4.2, 9.2.5

[85] Yaguang Li, Chuizheng Meng, Cyrus Shahabi, and Yan Liu. Structure-informed graph auto-encoder for relational inference and simulation. 9.4.2

[86] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. *arXiv preprint arXiv:2311.17043*, 2023. ??, ??, 6.3.1, ??, ??, ??, ??, ??, ??, ??, ??, ??, ??, ??

[87] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection, 2023. 6.3, 11.1

[88] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023. ??, 6.3.1, ??, ??, ??, ??, ??

[89] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 7.2

[90] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 6.1, 7.2

[91] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/. 7.2, 7.4.1, 12.2

[92] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124, 2017. 2.2

[93] Ruyang Liu, Chen Li, Yixiao Ge, Ying Shan, Thomas H Li, and Ge Li. One for all: Video conversation is feasible without video instruction tuning. *arXiv preprint arXiv:2309.15785*, 2023. ??

[94] Song Liu, Makoto Yamada, Nigel Collier, and Masashi Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013. 4.4.2, 9.2.5

[95] Yuliang Liu, Zhang Li, Biao Yang, Chunyuan Li, Xucheng Yin, Cheng-lin Liu, Lianwen Jin, and Xiang Bai. On the hidden mystery of ocr in large multimodal models. *arXiv preprint arXiv:2305.07895*, 2023. 7.4.2

[96] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022. 7.2, 7.4.2

[97] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023. 7.4.2

[98] Zimu Lu, Aojun Zhou, Ke Wang, Houxing Ren, Weikang Shi, Junting Pan, and Mingjie Zhan. Step-controlled dpo: Leveraging stepwise error for enhanced mathematical reasoning. *arXiv preprint arXiv:2407.00782*, 2024. 7.2, 7.6.2, 7.6.4

[99] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181*, 2020. 2.3.3, 2.4.3, 2.4.3

[100] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. ??, ??, 6.3.1, 6.3.1, ??, ??, ??, ??, ??

[101] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv*

preprint arXiv:2203.10244, 2022. 7.4.2

[102] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2200–2209, 2021. 7.2, 7.4.2

[103] Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1697–1706, 2022. 7.2, 7.4.2

[104] Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. Text classification using label names only: A language model self-training approach. *arXiv preprint arXiv:2010.07245*, 2020. 3.1, 3.4, 3.4.1, 3.5.2, 3.5.6

[105] Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. Generating training data with language models: Towards zero-shot language understanding. *arXiv preprint arXiv:2202.04538*, 2022. 3.2, 5.1

[106] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022. 5.1

[107] George Montanez, Saeed Amizadeh, and Nikolay Laptev. Inertial hidden markov models: Modeling change in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. 4.1

[108] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019. 3.4.1

[109] Shashi Narayan, Joshua Maynez, Reinald Kim Amplayo, Kuzman Ganchev, Annie Louis, Fantine Huot, Anders Sandholm, Dipanjan Das, and Mirella Lapata. Conditional generation with a question-answering blueprint. *Transactions of the Association for Computational Linguistics*, 11:974–996, 2023. 5.2.1

[110] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL https://arxiv.org/abs/2303.08774. 5.3.1, 5.4.1

[111] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 35: 27730–27744, 2022. 1.2, 3.1, 6.1

[112] Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, pages 993–1002, 2018. 2.1, 2.2

[113] Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Elle: Efficient lifelong pre-training for emerging data. *arXiv preprint arXiv:2203.06311*, 2022. 5.1

[114] Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. From r to q* : Your language model is secretly a q-function. *arXiv preprint arXiv:2404.12358*, 2024. 7.2, 7.6.4

[115] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024. 6.1

[116] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Using mobile phones to determine transportation modes. *ACM Trans. Sen. Netw.*, 6(2): 13:1–13:27, March 2010. ISSN 1550-4859. doi: 10.1145/1689239.1689243. URL http://doi.acm.org/10.1145/1689239.1689243. 9.4.2

[117] Attila Reiss and Didier Stricker. Introducing a new benchmarked dataset for activity monitoring. In *2012 16th International Symposium on Wearable Computers*, pages 108–109. IEEE, 2012. 4.5

[118] Yunus Saatçi, Ryan D Turner, and Carl Edward Rasmussen. Gaussian process change point models. In *ICML*, pages 927–934, 2010. 4.4.2, 9.4.2

[119] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*, 2020. 3.1, 3.5.2

[120] Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Transactions of the Association for Computational Linguistics*, 9:1408–1424, 2021. 3.4

[121] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge, 2022. URL https://arxiv.org/abs/2206.01718. 7.2, 7.4.2

[122] Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by

eight-fold. *arXiv preprint arXiv:2406.14532*, 2024. 7.2, 7.5

[123] Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. Nearest neighbor zero-shot inference. *arXiv preprint arXiv:2205.13792*, 2022. 3.2

[124] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019. 7.2

[125] Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. Recitation-augmented language models. *arXiv preprint arXiv:2210.01296*, 2022. 5.2.3

[126] Zhiqing Sun, Sheng Shen, Shengcao Cao, Haotian Liu, Chunyuan Li, Yikang Shen, Chuang Gan, Liang-Yan Gui, Yu-Xiong Wang, Yiming Yang, et al. Aligning large multimodal models with factually augmented rlhf. *arXiv preprint arXiv:2309.14525*, 2023. 6.1, 6.2.3

[127] Zhiqing Sun, Yikang Shen, Hongxin Zhang, Qinhong Zhou, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Salmon: Self-alignment with principle-following reward models. *arXiv preprint arXiv:2310.05910*, 2023. 6.1

[128] Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv preprint arXiv:2305.03047*, 2023. 5.2.2

[129] Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. Easy-to-hard generalization: Scalable alignment beyond human supervision. *arXiv preprint arXiv:2403.09472*, 2024. 7.2, 7.5

[130] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023. 3.1, 3.5.1, 5.2.2

[131] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=wCu6T5xFjeJ`. 5.3.1, 5.4.1

[132] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *arXiv preprint arXiv:2406.16860*, 2024. 7.4.5, 7.3, 12.0.1

[133] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 3.5.1

[134] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 4.3.1, 4.3.1

[135] Cunxiang Wang, Pai Liu, and Yue Zhang. Can generative pre-trained language models serve as knowledge bases for closed-book qa? *arXiv preprint arXiv:2106.01561*, 2021. 5.1

[136] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *arXiv preprint arXiv:2402.14804*, 2024. 7.2

[137] Rui Wang, Jianzhu Bao, Fei Mi, Yi Chen, Hongru Wang, Yasheng Wang, Yitong Li, Lifeng Shang, Kam-Fai Wong, and Ruifeng Xu. Retrieval-free knowledge injection through multi-document traversal for dialogue models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6608–6619, 2023. 5.2.1

[138] Yau-Shian Wang, Ta-Chung Chi, Ruohong Zhang, and Yiming Yang. Pesco: Prompt-enhanced self contrastive learning for zero-shot text classification, 2023. 1.3, 3.1, 3.2, 3.3, 3.4, 3.5.2, 3.5.5, 3.5.6

[139] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022. 1.2, 5.2.3

[140] Tong Wei, Wei-Wei Tu, Yu-Feng Li, and Guo-Ping Yang. Towards robust prediction on tail labels. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1812–1820, 2021. 2.1

[141] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of

question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786, 2021. 6.3.1

[142] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016. 3.4.1

[143] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020. 2.3.3

[144] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016. 6.3.1, 6.3.2, 11.4

[145] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 9.4.2

[146] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *CoRR*, abs/1905.13211, 2019. URL http://arxiv.org/abs/1905.13211. 9.4.2

[147] Zhao Xu, Kristian Kersting, and Lorenzo von Ritter. Stochastic online anomaly analysis for streaming time series. In *IJCAI*, pages 3189–3195, 2017. 9.2.5

[148] Xiang Xuan and Kevin Murphy. Modeling changing dependency structure in multivariate time series. In *Proceedings of the 24th international conference on Machine learning*, pages 1055–1062, 2007. 9.4.2

[149] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Zero-shot video question answering via frozen bidirectional language models. *NeurIPS*, 2022. ??, 6.3.1

[150] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49, 1999. 2.5.2, 2.5.4

[151] Zhilin Yang, Jake Zhao, Bhuwan Dhingra, Kaiming He, William W Cohen, Russ R Salakhutdinov, and Yann LeCun. Glomo: Unsupervised learning of transferable relational graphs. In *Advances in Neural Information Processing Systems*, pages 8950–8961, 2018. 9.4.2

[152] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint arXiv:2408.01800*, 2024. 7.4.5

[153] Hui Ye, Zhiyu Chen, Da-Han Wang, and Brian Davison. Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification. In *International Conference on Machine Learning*, pages 10809–10819. PMLR, 2020. 2.2, 2.5.3

[154] Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*, 2022. 3.2

[155] Ian EH Yen, Xiangru Huang, Wei Dai, Pradeep Ravikumar, Inderjit Dhillon, and Eric Xing. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 545–553, 2017. 2.2

[156] Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. Gpt3mix: Leveraging large-scale language models for text augmentation. *arXiv preprint arXiv:2104.08826*, 2021. 3.2

[157] Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *arXiv preprint arXiv:1811.01727*, 2018. 2.2, 2.5.3

[158] Tianyu Yu, Haoye Zhang, Yuan Yao, Yunkai Dang, Da Chen, Xiaoman Lu, Ganqu Cui, Taiwen He, Zhiyuan Liu, Tat-Seng Chua, et al. Rlaif-v: Aligning mllms through open-source ai feedback for super gpt-4v trustworthiness. *arXiv preprint arXiv:2405.17220*, 2024. 7.6, 7.6.1

[159] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 7.4.2

[160] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023. 1.3, **??**, 6.3.1

[161] Nancy R Zhang, David O Siegmund, Hanlee Ji, and Jun Z Li. Detecting simultaneous changepoints in multiple sequences. *Biometrika*, 97(3):631–645, 2010. 4.1

[162] Pan Zhang, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Rui Qian, Lin Chen, Qipeng Guo, Haodong Duan, Bin Wang, Linke Ouyang, et al. Internlm-xcomposer-2.5: A versatile large vision language model supporting long-contextual input and output. *arXiv preprint arXiv:2407.03320*, 2024. 7.4.5

[163] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023. ??

[164] Ruohong Zhang, Yu Hao, Donghan Yu, Wei-Cheng Chang, Guokun Lai, and Yiming Yang. Correlation-aware unsupervised change-point detection via graph neural networks, 2020. 1.3

[165] Ruohong Zhang, Yau-Shian Wang, Yiming Yang, Tom Vu, and Likun Lei. Exploiting local and global features in transformer-based extreme multi-label text classification. *arXiv preprint arXiv:2204.00933*, 2022. 1.3

[166] Ruohong Zhang, Luyu Gao, Chen Zheng, Zhen Fan, Guokun Lai, Zheng Zhang, Fangzhou Ai, Yiming Yang, and Hongxia Yang. A self-enhancement approach for domain-specific chatbot training via knowledge mining and digest, 2023. 1.3

[167] Ruohong Zhang, Yau-Shian Wang, and Yiming Yang. Generation-driven contrastive self-training for zero-shot text classification with instruction-tuned gpt, 2023. 1.3

[168] Ruohong Zhang, Yau-Shian Wang, Yiming Yang, Donghan Yu, Tom Vu, and Likun Lei. Long-tailed extreme multi-label text classification by the retrieval of generated pseudo label descriptions. In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1092–1106, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.81. URL https://aclanthology.org/2023.findings-eacl.81. 1.3, 3.2

[169] Ruohong Zhang, Liangke Gui, Zhiqing Sun, Yihao Feng, Keyang Xu, Yuanhan Zhang, Di Fu, Chunyuan Li, Alexander Hauptmann, Yonatan Bisk, et al. Direct preference optimization of video large multimodal models from language model reward. *arXiv preprint arXiv:2404.01258*, 2024. 7.6.2

[170] Ruohong Zhang, Bowen Zhang, Yanghao Li, Haotian Zhang, Zhiqing Sun, Zhe Gan, Yin-

fei Yang, Ruoming Pang, and Yiming Yang. Improve vision language model chain-of-thought reasoning. *arXiv preprint arXiv:2410.16198*, 2024. 1.3

[171] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 247–256, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: 10.1145/1367497.1367532. URL http://doi.acm.org/10.1145/1367497.1367532. 9.4.2

[172] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. Understanding transportation modes based on gps data for web applications. *ACM Transaction on the Web*, 4:1–36, January 2010. URL https://www.microsoft.com/en-us/research/publication/understanding-transportation-modes-based-on-gps-data-for-web-appli 9.4.2

[173] Zexuan Zhong, Zhengxuan Wu, Christopher D Manning, Christopher Potts, and Danqi Chen. Mquake: Assessing knowledge editing in language models via multi-hop questions. *arXiv preprint arXiv:2305.14795*, 2023. 5.1

[174] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023. 5.2.2

[175] Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, HongFa Wang, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. *arXiv preprint arXiv:2310.01852*, 2023. 6.3.2, 11.4

[176] Zeyuan Allen Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction, 2023. 5.1