VerifyMatch: A Semi-Supervised Learning Paradigm for Natural Language Inference with Confidence-Aware MixUp

Seo Yeon Park
Computer Science & Engineering
Hanyang University (ERICA)
seoyeonpark@hanyang.ac.kr

Cornelia Caragea
Computer Science
University of Illinois Chicago
cornelia@uic.edu

Abstract

While natural language inference (NLI) has emerged as a prominent task for evaluating a model's capability to perform natural language understanding, creating large benchmarks for training deep learning models imposes a significant challenge since it requires extensive human annotations. To overcome this, we propose to construct pseudo-generated samples (premise-hypothesis pairs) using class-specific fine-tuned large language models (LLMs) thereby reducing the human effort and the costs in annotating large amounts of data. However, despite the impressive performance of LLMs, it is necessary to verify that the pseudo-generated labels are actually correct. Towards this goal, in this paper, we propose ***VerifyMatch***, a semi-supervised learning (SSL) approach in which the LLM pseudo-labels guide the training of the SSL model and, at the same time, the SSL model acts as a *verifier* of the LLM-generated data. In our approach, we retain all pseudo-labeled samples, but to ensure unlabeled data quality, we further propose to use MixUp whenever the verifier does not agree with the LLM-generated label or when they both agree on the label but the verifier has a low confidence—lower than an adaptive confidence threshold. We achieve competitive accuracy compared to strong baselines for NLI datasets in low-resource settings.

1 Introduction

Natural Language Inference (NLI) (Bowman et al., 2015) aims to determine the relation between two sentences (referred as premise and hypothesis)—whether it is *entailment, neutral,* or *contradiction*. NLI plays a pivotal role in assessing a model's ability to perform Natural Language Understanding (NLU) and Reasoning. The advancement of NLI has been fueled, in part, by the creation of large datasets such as SNLI (Bowman et al., 2015), MNLI (Williams et al., 2018), and ANLI (Nie et al., 2020) for training massive deep learning models. However, creating a large-scale NLI benchmark requires a considerable amount of human effort. This is because human annotators have to generate texts that demand logical inferences. For example, in the creation of the SNLI and MNLI datasets (Bowman et al., 2015; Williams et al., 2018), human workers receive unlabeled premises and are prompted to ***generate hypotheses***, one per class, for each class label—entailment, neutral, contradiction. Similarly, in the creation of the ANLI dataset (Nie et al., 2020), human annotators receive an unlabeled premise and a target label, and are asked to ***generate a hypothesis*** that deceives a model into producing a misclassified prediction of the given target label. In this manner, creating new large-scale NLI datasets becomes a burdensome task. Hence, the high cost and difficulty of collecting labeled data for NLI has driven interest in semi-supervised learning (SSL), which effectively utilizes both labeled and unlabeled data. However, the nature of unlabeled data for SSL on NLI is more complex compared to single-sentence classification tasks. This is because one of the sentences in the pair (usually the hypothesis) along with the class label, is missing from the data and requires intensive human annotations as described above. Therefore, in order to leverage unlabeled data for SSL on NLI, the unavailability of both hypotheses and class labels must be tackled.

To overcome this, we propose to leverage Large Language Models (LLMs) to generate missing hypotheses and to assign initial pseudo-labels where we create readily available unlabeled data for SSL on NLI. However, LLMs may not always generate the most relevant or accurate output. Hence, we further propose to leverage pseudo-labeling (Lee, 2013) to ensure the quality of the generated hypotheses and their assigned labels. Pseudo-labeling is a widely used semi-supervised learning method that automatically assigns pseudo-labels to unlabeled data and incorporates them into model

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 19319–19335

training. Prior research on pseudo-labeling generally employs a pre-defined high threshold for all classes, which assumes pseudo-labels with confidence above the threshold are of high quality and hence beneficial for training while others are of low quality so are discarded (Chen et al., 2020; Sohn et al., 2020; Sadat and Caragea, 2022). Thus, this approach results in restricting access to a considerable amount of samples. To address this issue, Zhang et al. (2021) propose to use adaptive thresholds for different classes to encourage a model to learn from more diverse samples and achieve better performance in low-resource settings compared to approaches that use a fixed high confidence threshold (Sohn et al., 2020). Despite their promising results when using flexible thresholds, many pseudo-labeled samples are still discarded. Chen et al. (2023) propose to use all pseudo-labeled samples by assigning lower weights to unconfident pseudo-labeled samples during training. Although the diversity of training data increases substantially compared to previous works, there are still erroneous pseudo-labels that enter with high weights in the training set as training progresses.

To this end, we propose ***VerifyMatch***, a semi-supervised learning approach, which uses all pseudo-labeled samples in model training where unconfident pseudo-labeled samples are incorporated into training instead of being discarded or used with lower weights during training as in previous works. VerifyMatch consists of two components: (1) pseudo-generated data construction using large language models (LLMs), and (2) a verifier that leverages pseudo-labeling to ensure the quality of LLM-generated pseudo-labels. In VerifyMatch, the LLM pseudo-labels guide the training of the verifier and, at the same time, the verifier determines the veracity of the LLM-generated labels. Our pseudo-generated data construction produces readily available unlabeled data for semi-supervised learning (SSL) on Natural Language Inference (NLI). Specifically, given a small amount of labeled data, we first fine-tune LLMs for every class. We then use these class-specific fine-tuned LLMs for generating hypotheses for a given unlabeled premise along with assigning the initial pseudo-label. By leveraging class-specific fine-tuned LLMs, we prevent potential skew or imbalance in the distribution of class labels within pseudo-generated data, thereby ensuring comprehensive coverage of all class labels. For example, given a premise 'A man painting over graffiti', we produce three hypotheses, one for each class, 'entailment,' 'contradiction,' and 'neutral,' by using the corresponding class-specific fine-tuned LLM.

To ensure the quality of LLM-generated hypotheses and their pseudo-labels, our verifier (a task classifier) produces pseudo-labels on sentence pairs and checks them against LLM-assigned pseudo-labels. If there is disagreement between the labels, we call these "*mismatched samples*". Even when there is agreement, the verifier might be unsure of its prediction (i.e., unconfident on a predicted class), because the sample is ambiguous or possibly mislabeled. We consider these samples as "*unconfident samples*" Both types of samples are then "denoised" by interpolating them with human-annotated labels through MixUp (Zhang et al., 2018). Hence, VerifyMatch improves the diversity of training data while ensuring its quality. We show competitive performance on various NLI datasets in low-resource settings compared to strong baseline methods.

Our contributions are as follows:

- We propose a semi-supervised learning framework called VerifyMatch which consists of two components: (1) pseudo-generated data construction using LLMs and (2) a verifier to ensure the quality of pseudo-generated data.
- On the verifier, we propose to identify mismatched and unconfident pseudo-generated samples that are potentially mislabeled hence incorporating them into training after denoising through MixUp between them and human-labeled samples where we denoise a possibly incorrect pseudo-label by mixing it with a correct one, thus exposing a model to a larger diversity of samples during training.
- We conduct comprehensive experiments showing that our method achieves competitive performance compared with strong baselines on SSL for NLI datasets in low-resource settings.

2 Related Work

Large Language Models (LMMs) The emergence of large language models (LLMs) has revolutionized the field of natural language processing (NLP) which have achieved major milestones in the advancement of various tasks including text generation, question answering, and dialogue generation (Zhang et al., 2020; Touvron et al., 2023; Jiang et al., 2023; Team et al., 2024; Achiam et al.,

2024; Chen et al., 2024). To truly leverage LLMs, customization is key which involves fine-tuning LLMs on specialized datasets. Fine-tuning often provides competitive performance mainly because pre-training with language modeling objectives provides a useful starting point for model parameters and allows task-specific objective customization (Zhang et al., 2022; Liu et al., 2022a; Schmidt et al., 2022; Garimella et al., 2022; Do et al., 2023; Wang et al., 2023a). However, full fine-tuning is usually expensive in both computation and memory due to a large number of parameters for recent advanced LLMs (e.g., Llama 2/3[1]). Hence, parameter-efficient fine-tuning methods such as P-tuning (Liu et al., 2022b) and Low-Rank Adaptation (LoRA; Hu et al. (2021a)) have gained attention. In this work, we leverage Llama 3 with LoRA for semi-supervised learning on natural language inference. In addition, prompting, which is a method of a pre-trained LLM to be adapted to different tasks via priming on natural language prompts —pieces of text that are combined with input and then fed to the language model to produce an output for that task (Brown et al., 2020)—has been successful for few-/zero-shot learning at many general-domain tasks (Gao et al., 2021; Agrawal et al., 2022; Li et al., 2024). Hence, we compare our proposed method with various LLMs using prompting to understand the effectiveness of our method.

Semi-supervised Learning (SSL) SSL has produced a diverse collection of approaches including self-training (also called pseudo-labeling) (Chen et al., 2020; Xie et al., 2020; Yu et al., 2021; Lee et al., 2021; Hu et al., 2021b; Sadat and Caragea, 2022; Min et al., 2024). In general, self-training relies on a fixed high threshold value on model confidence in the pseudo-label class to filter out low-confidence pseudo-labeled samples (Li and Yang, 2018; Chen et al., 2020; Lee et al., 2021; Hu et al., 2021b; Yu et al., 2021; Sadat and Caragea, 2022; Wang et al., 2023b) thereby limiting access to a broader range of training samples. To overcome this, Chen et al. (2023) proposed to integrate low-confidence samples by assigning lower weights to them during training. However, this method still may introduce erroneous pseudo-labels with high weights as training iteration progresses. Hence, we propose to integrate low-confidence pseudo-generated samples after denoising.

[1]https://ai.meta.com/blog/meta-llama-3/

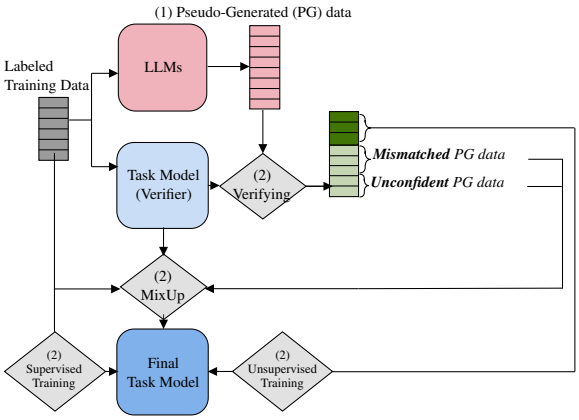

Figure 1: The overview of our proposed approach: (1) LLMs construct Pseudo-Generated (PG) data, and (2) the verifier identifies ***mismatched*** PG data and ***unconfident*** PG data, to denoise them through MixUp while the rest PG data are used via unsupervised training in addition to using labeled data in supervised training to obtain the final classifier.

MixUp MixUp (Zhang et al., 2018) is a regularizer for neural models by training convexly combining random pairs and their associated labels. Many works have empirically noticed regularization effects of MixUp that improve performance on deep neural networks (Verma et al., 2019; Guo et al., 2019; Yun et al., 2019; Kim et al., 2020; Yin et al., 2021; Park and Caragea, 2022; Qiao et al., 2022). MixUp also has shown effectiveness in SSL for NLP tasks (Chen et al., 2020; Sawhney et al., 2021; Yang et al., 2021). Building upon this, we propose to use a MixUp approach for SSL to denoise low-confidence pseudo-generated samples by mixing them with labeled samples.

3 Proposed Approach: VerifyMatch

In this section, we introduce VerifyMatch, our semi-supervised learning (SSL) approach for natural language inference. VerifyMatch seamlessly combines two components: 1. pseudo-hypothesis generation and pseudo-label assignment for the unlabeled data using Large Language Models (LLM) as one component, and 2. SSL model training with pseudo-labeling as another component. In VerifyMatch, the LLM pseudo-labels guide the training of the SSL model and to ensure the quality of pseudo-labels it includes three key elements: a verification step that accounts for the agreement / disagreement of LLM-generated and SSL-generated pseudo-labels; adaptive confidence thresholding that leverages the SSL model's confidence and uncertainty in the predictions; and the use of all training samples—no matter how noisy through a mixup

data augmentation strategy that mixes in-between labeled and unlabeled samples.

3.1 Pseudo-Hypothesis Generation and Label Assignment with LLMs

Let $\mathcal{D}_l = \{(\mathbf{x}_i, y_i)\}_{i=1,\cdots,n}$ be a labeled training set of size n where $\mathbf{x}_i = (p_i, h_i)$ refers to a premise and hypothesis sentence-pair in NLI, and y_i represents one of three NLI classes (i.e., 'contradiction', 'entailment', 'neutral'). Furthermore, let $\mathcal{D}_u = \{p_i^u\}_{i=1,\cdots,N}$ be a set of unlabeled premises of size N, with $N >> n$.

In our approach, we use large language models (LLMs) to generate pseudo-hypotheses for unlabeled premises. That is, we first fine-tune a class-specific LLM ϕ^c for each NLI class c using labeled samples corresponding to that class. Subsequently, we provide an unlabeled premise to each of these class-specific fine-tuned LLMs to generate three hypotheses (one per class with the corresponding LLM). For each pair—premise, generated hypothesis—we assign an LLM pseudo-label according to the class of the LLM that generated the hypothesis. Thus, we ensure comprehensive coverage of all classes within pseudo-generated samples. We formulate pseudo-generated data as follows:

$$\mathcal{D}_{pseudo} = \{\hat{\mathbf{x}}_i = (\hat{p}_i, \hat{h}_i = \phi^c(\hat{p}_i)), \hat{y}_i^{llm} = c\}$$
$$i = 1 \ldots c \cdot N, c \in C$$

where $\hat{p}_i$ is an unlabeled premise, $\hat{h}_i$ is a generated hypothesis by class-specific fine-tuned LLM ϕ^c on class c, and $\hat{y}_i^{llm}$ is the pseudo-label assigned by ϕ^c. We mainly adopt a parameter-efficient fine-tuning named LoRA (Hu et al., 2021a) on Llama-3-8B-Instruct[2]. We provide the details (e.g., prompts, hyper-parameters) in Appendix A.1. To explore the impact of leveraging various LLMs in pseudo-generated data construction, we provide the results of using LoRA Llama-2, fully fine-tuning GPT-2, and zero-shot prompting Llama-2 in Appendix A.2.

3.2 Semi-Supervised Model Training with Pseudo-Labeling

Our SSL model training leverages pseudo-labeling, an approach that uses the model itself to obtain artificial labels for unlabeled data. However, if the artificial labels are incorrect, the model will suffer from error accumulation (Arazo et al., 2020). In contrast, in VerifyMatch, we consider agreement / disagreement between the LLM-generated and the SSL-generated pseudo-labels through a verification step. Moreover, pseudo-labeling (Sohn et al., 2020) exploits a confidence thresholding mechanism to discard samples that are predicted with a low confidence by the model and retains only the labels whose largest class probability fall above a predefined fixed threshold. Thus, a large pool of samples are completely ignored despite containing potentially useful information for model training (Zhang et al., 2021; Chen et al., 2023). In contrast, we use all training samples with an adaptive confidence thresholding to separate samples in high and low confidence samples and with a mixup strategy (Zhang et al., 2018) to handle potential noise in low-confidence and disagreement pseudo-labels.

3.2.1 Label Verification

VerifyMatch verifies the agreement / disagreement between the two labels for each pseudo-generated (PG) sample $\hat{\mathbf{x}}_i$, the LLM label $\hat{y}_i^{llm}$ and the label derived from the SSL model θ, i.e., the verifier (BERT in our case). Specifically, the verifier computes a pseudo-label by itself $\hat{y}_i$ for every sample $\hat{\mathbf{x}}_i \in \mathcal{D}_{pseudo}$. If the verifier's label does not match the LLM label $\hat{y}_i \neq \hat{y}_i^{LLM}$, we consider the PG sample as a "*mismatched PG sample*" which is ambiguous for the model or potentially incorrectly labeled due to failure in agreement on the pseudo-label (either the LLM has generated a wrong hypothesis or the SSL model returns an incorrect prediction). Thus, it is necessary to handle noise in the pseudo-labels. Inspired by Yang et al. (2021), we interpolate these PG samples and labeled samples using MixUp (Zhang et al., 2018) as follows:

$$\tilde{\mathbf{x}_k} = (1-\lambda)\hat{\mathbf{x}}_i + \lambda \mathbf{x}_j$$
$$\tilde{y_k} = (1-\lambda)\hat{y}_i^{llm} + \lambda y_j \quad (1)$$

where $\hat{\mathbf{x}}_i$ and $\mathbf{x}_j$ are inputs' feature representations of pseudo-generated and labeled samples, respectively, $\hat{y}_i^{llm}$ and y_j are their associated one-hot encoded labels, and λ is a mixing ratio sampled from a Beta(α, α) distribution with a hyper-parameter α. In mixing labels, we interpolate pseudo-labels of pseudo-generated samples and gold-standard labels of labeled samples. Accordingly, we not only denoise possibly incorrect pseudo-labels by mixing them with correct ones but also smooth the level of uncertainty of unconfident pseudo-generated samples. After mixing *mismatched PG samples* with human labeled samples, we compute the unlabeled mismatched data loss as the cross-entropy loss on

[2]https://llama.meta.com/llama3/

Algorithm 1 : VerifyMatch

Inputs: Labeled data $\mathcal{D}_l$; unlabeled data $\mathcal{D}_u$; SSL model (i.e., a task classifier, verifier) θ, class-specifically fine-tuned LLMs ϕ^c for every $c \in C$
Construct Pseudo-Generated (PG) data $\mathcal{D}_{pseudo} = \{(\hat{x}_i, \hat{y}_i^{llm})\}_{i=1,\cdots,c\hat{N}}$ where $\hat{x}_i = (\hat{p}_i, \hat{h}_i = \phi^c(\hat{p}_i))$, $p_i \in \mathcal{D}_u$, and $\hat{y}_i^{llm}$ is a initial pseudo-label assigned by ϕ^c, $\hat{y}_i^{llm} = c$
for $t = 1$ to T **do**
  while $\mathcal{D}_{pseudo}$ not exhausted **do**
    Randomly sample labeled batch B_l from $\mathcal{D}_l$, and pseudo-generated batch B_{pseudo} from $\mathcal{D}_{pseudo}$
    Initialize B_m, B_{mm} and B_{unconf} as empty sets, $B_m, B_{mm}, B_{unconf} \leftarrow \emptyset, \emptyset$
    for each $(\hat{x}_i, \hat{y}_i^{llm}) \in B_{pseudo}$ **do**
      Obtain the pseudo-label $\hat{y}_i$ from the task model θ, $\hat{y}_i = \text{argmax}\, P_\theta(y|\hat{\mathbf{x}}_i)$
      if $\hat{y}_i = \hat{y}_i^{llm}$ **then**
        $B_m \leftarrow B_m \cup \{(\hat{\mathbf{x}}_i, \hat{y}_i)\}$
      else # Identify *"mismatched pseudo-generated (PG) data"*
        $(\tilde{\mathbf{x}_k}, \tilde{y}_k)$= $\texttt{MixUp}((\hat{\mathbf{x}}_i, \hat{y}_i^{llm}), (\mathbf{x}_j, y_j))$ using Eq. (1) where $(\mathbf{x}_j, y_j)$ is randomly sampled from B_l
        $B_{mm} \leftarrow B_{mm} \cup \{(\tilde{\mathbf{x}_k}, \tilde{y}_k)\}$
      end if
    end for
    Compute the mean of confidence $\bar{P}$ on B_m using Eq. (3)
    for each $(\hat{x}_i, \hat{y}_i) \in B_m$ **do**
      if $max(P_\theta(y|\hat{\mathbf{x}}_i)) < \bar{P}$ **then** # Identify *"unconfident pseudo-generated (PG) data"*
        $(\tilde{\mathbf{x}}_k, \tilde{y}_k) = \texttt{MixUp}((\hat{\mathbf{x}}_i, \hat{y}_i), (\mathbf{x}_j, y_j))$ using Eq (1) where $(\mathbf{x}_j, y_j)$ is randomly sampled from B_l
        $B_{unconf} \leftarrow B_{unconf} \cup (\tilde{\mathbf{x}}_k, \tilde{y}_k)$
      end if
    end for
    $\mathcal{L}_{sup} = \frac{1}{|B_l|} \sum_{i=1}^{|B_l|} \mathcal{H}(y_i, P_\theta(y|\mathbf{x}_i))$,
    $\mathcal{L}_{unsup} = \frac{1}{|B_m^{\geq \bar{P}}|} \sum_{i=1}^{|B_m^{\geq \bar{P}}|} \mathcal{H}(\hat{y}_i, P_\theta(y|\hat{\mathbf{x}}_i)))$
    $\mathcal{L}_{mm} = \frac{1}{|B_{mm}|} \sum_{k=1}^{|B_{mm}|} \mathcal{H}(\tilde{y}_k, P_\theta(y|\tilde{\mathbf{x}}_k))$
    $\mathcal{L}_{unconf} = \frac{1}{|B_{unconf}|} \sum_{k=1}^{|B_{unconf}|} \mathcal{H}(\tilde{y}_k, P_\theta(y|\tilde{\mathbf{x}}_k))$
    Update the verifier parameter θ using $\mathcal{L}_{sup} + \mathcal{L}_{unsup} + \mathcal{L}_{mm} + \mathcal{L}_{unconf}$
  end while
end for

MixUp samples $(\tilde{\mathbf{x}}_k, \tilde{y_k})$, as follows:

$$\mathcal{L}_{mm} = \frac{1}{|B_{mm}|} \sum_{k=1}^{|B_{mm}|} \mathcal{H}(\tilde{y}_k, P_\theta(y|\tilde{\mathbf{x}}_k)) \quad (2)$$

where B_{mm} is the set of the *mismatched samples* with label disagreement between LLM and verifier.

3.3 Adaptive Confidence Thresholding

If the verifier's label matches the LLM label $\hat{y}_i = \hat{y}_i^{LLM}$, we consider these samples as "*matched samples*" and denote their set as B_m. Even when the verifier agrees with the LLMs assigned pseudo-label, the verifier might be unsure of its prediction (i.e., unconfident on a predicted class), because the PG sample is ambiguous or possibly mislabeled. We consider this PG sample as an "*unconfident PG sample*". To identify unconfident PG samples, instead of using a fixed threshold as in vanilla pseudo-labeling, we derive the mean of the verifier's confidence of the matched samples (as shown below), since it empirically verified better generalization (Chen et al., 2023; Wang et al., 2023b; Zhang et al., 2021):

$$\bar{P} = \frac{1}{|B_m|} \sum_{i=1}^{|B_m|} P_\theta(\hat{y}_i|\hat{\mathbf{x}}_i) \quad (3)$$

The pseudo-labels predicted by the verifier with confidence above $\bar{P}$ are used to compute the unsupervised (matched, high-confidence) loss as:

$$\mathcal{L}_{unsup} = \frac{1}{|B_m^{\geq \bar{P}}|} \sum_{i=1}^{|B_m^{\geq \bar{P}}|} \mathcal{H}(\hat{y}_i, P_\theta(y|\hat{\mathbf{x}}_i)) \quad (4)$$

For the *unconfident PG samples* that fall under the $\bar{P}$ threshold, i.e., $B_m^{<\bar{P}}$ (or B_{unconf}), as these are ambiguous or possibly mislabeled samples, we again use mixup to mix an unconfident PG sample with a human (clean) labeled sample, and compute the unsupervised (matched but unconfident) loss:

$$\mathcal{L}_{unconf} = \frac{1}{|B_{unconf}|} \sum_{k=1}^{|B_{unconf}|} \mathcal{H}(\tilde{y}_k, P_\theta(y|\tilde{\mathbf{x}}_k)) \quad (5)$$

Here, $(\tilde{\mathbf{x}}_k, \tilde{y}_k) = \texttt{MixUp}((\hat{\mathbf{x}}_i, \hat{y}_i), (\mathbf{x}_j, y_j))$, where $(\hat{\mathbf{x}}_i, \hat{y}_i)$ is an unconfident PG sample (i.e., $P_\theta(\hat{y}_i|\hat{\mathbf{x}}_i) < \bar{P}$) and $(\mathbf{x}_j, y_j)$ is a human labeled sample, selected at random from $\mathcal{D}_l$. Consequently, unconfident (i.e., low-confidence under the $\bar{P}$)

Pseudo-Generated (PG) samples are incorporated in training after denoising, hence, increasing the diversity of training data while ensuring the quality.

In addition, we calculate the supervised loss on labeled samples as follows:

$$\mathcal{L}_{sup} = \frac{1}{|B_l|} \sum_{i=1}^{|B_l|} \mathcal{H}(y_i, P_\theta(y|\mathbf{x}_i)) \quad (6)$$

We train the verifier by using the sum of all losses (see Algorithm 1). The final loss is:

$$\mathcal{L} = \mathcal{L}_{sup} + \mathcal{L}_{unsup} + \mathcal{L}_{mm} + \mathcal{L}_{unconf} \quad (7)$$

Note that our implementation uses separate data loaders for labeled and pseudo-generated data to conduct a MixUp operation.

4 Experiments

4.1 Datasets

RTE (Wang et al., 2018) has $\approx 2,500$ sentence pairs with two pre-defined classes, *entailment* and *not_entailment*. We extract unlabeled premises from WikiPedia and CNNDM (Nallapati et al., 2016). Since the test set of RTE is not publicly available, we use its development set as the test set and randomly sample a small subset from the training set to be used as the development set.

SICK (Marelli et al., 2014) has $4,500$ sentence pairs with three pre-defined classes, which are *entailment, contradiction* and *neutral*. We use the 8k ImageFlickr dataset and WikiPedia to extract unlabeled premises.

SNLI-2.5k (Bowman et al., 2015) SNLI is a large dataset of 570k human-written English sentence pairs classified as *entailment, contradiction,* or *neutral* (Bowman et al., 2015). To simulate a low-resource setting, we randomly sampled 2,500 examples from the training set of SNLI to be used as labeled data and considered the premises of the remaining examples as unlabeled data.

MNLI-2.5k (Williams et al., 2018) We create the labeled/unlabeled data in the same manner as SNLI. Similar to RTE, we used the development set of MNLI as the test set and sampled a small subset of examples from the training set to be used as development set.

4.2 Comparison Methods

BERT Fine-tuning We use the labeled data only of each dataset to fine-tune a pre-trained language model BERT (Devlin et al., 2019).

In-context Learning (Brown et al., 2020) is a simple prompting[3] on GPT-2 and Llama 3-8B-Instruct with 10 labeled data.

Zero-shot Learning (Brown et al., 2020) is a simple prompting[3] on Mistral-7B-Instruct-v0.1 (Jiang et al., 2023), Llama 2-7B-chat-hf (Touvron et al., 2023), and Llama 3-8B-Instruct, without labeled data.

LM-BFF (Gao et al., 2021) is a prompt-based fine-tuning method using the manual prompt of Gao et al. (2021) for BERT, using the labeled data only of each downstream task.

Back Translation (Edunov et al., 2018) synthesizes additional data by back-translating labeled data using German-English translation models.

TMix (Chen et al., 2020) synthesizes additional data by interpolating randomly selected labeled data in the hidden space of BERT on transformer layers of 7, 9, 12.

FixMatch (Sohn et al., 2020) generates pseudo-labels using the model's predictions on weakly augmented data and only retains a pseudo-label if the model produces a high-confidence prediction. The model is trained to predict the pseudo-label when fed strongly-augmented data of the same data[4].

FlexMatch (Zhang et al., 2021) extends FixMatch by using flexible confidence thresholds to adjust for the learning difficulty of each class instead of using high fixed confidence thresholds.

FreeMatch (Wang et al., 2023b) extends FlexMatch by leveraging both global/local thresholds to reflect the model's learning status with self-adaptive class fairness regularization penalty.

SoftMatch (Chen et al., 2023) extends FixMatch by deriving a truncated Gaussian function to weight unlabeled samples based on their confidence to leverage all unlabeled data.

Unsupervised Data Augmentation (UDA) (Xie et al., 2020) computes consistency loss to minimize the distance between unlabeled samples' original predictions and predictions on data augmentation[5] along with the supervised loss.

[3]The prompt is constructed by referring to Brown et al. (2020) as shown in A.4. We follow the evaluation protocol provided by Gao et al. (2021).

[4]Weak augmentation is a synonym replacement using WordNet on both premise-hypothesis randomly chosen tokens. Strong augmentation is back-translation pre-trained language models (with German) both on premise-hypothesis pairs.

[5]We use back-translation and tf-idf word replacement data augmentation methods. We use pre-trained back translation models (with German) released by FairSeq and set the random sampling temperature as 0.9.

	RTE	SICK	SNLI-2.5K	MNLI-2.5k$_m$	MNLI-2.5k$_{mm}$
Fine-tuning (FT) BERT (Devlin et al., 2019)	$60.90_{1.6}$	$84.63_{0.7}$	$79.03_{0.1}$	$69.26_{0.9}$	$70.26_{0.7}$
GPT-2 ICL (Brown et al., 2020)	$54.94_{2.2}$	$59.38_{3.2}$	$33.37_{0.3}$	$33.51_{1.3}$	$33.09_{0.4}$
Llama 3-8B-Instruct ICL	$68.22_{0.0}$	$55.31_{0.0}$	$59.67_{0.0}$	$59.74_{0.0}$	$58.72_{0.0}$
Mistral-7B ZSL (Jiang et al., 2023)	$60.41_{0.0}$	$48.82_{0.0}$	$45.34_{0.0}$	$47.27_{0.0}$	$49.69_{0.0}$
Llama 2-7B ZSL (Touvron et al., 2023)	$67.30_{0.0}$	$49.06_{0.0}$	$56.70_{0.0}$	$55.04_{0.0}$	$57.23_{0.0}$
Llama 3-8B-Instruct ZSL	$68.88_{0.0}$	$55.47_{0.0}$	$60.19_{0.0}$	$58.87_{0.0}$	$59.61_{0.0}$
LM-BFF (Gao et al., 2021)	$60.64_{0.9}$	$81.59_{0.8}$	$73.91_{0.6}$	$62.89_{1.2}$	$65.54_{0.8}$
LM-BFF + Demo	$61.26_{1.8}$	$82.22_{0.5}$	$74.56_{0.9}$	$62.55_{1.2}$	$64.09_{0.5}$
Back Translation (Edunov et al., 2018)	$61.22_{1.3}$	$84.38_{1.1}$	$79.15_{1.2}$	$72.01_{1.0}$	$73.38_{0.9}$
TMix (Chen et al., 2020)	$61.59_{1.5}$	$83.23_{1.9}$	$79.13_{1.0}$	$71.86_{0.6}$	$73.21_{0.8}$
UDA (Xie et al., 2020)	$65.53_{0.9}$	$85.46_{0.8}$	$80.06_{0.4}$	$\underline{72.97}_{0.5}$	$\underline{73.82}_{0.5}$
MixText (Chen et al., 2020)	$\underline{68.49}_{2.1}$	$85.44_{0.6}$	$80.11_{0.2}$	$72.45_{0.8}$	$73.42_{1.0}$
SSL for NLI (Sadat and Caragea, 2022)	$68.32_{2.3}$	$\underline{85.77}_{0.7}$	$80.26_{1.1}$	$72.56_{0.3}$	$73.48_{0.1}$
FixMatch (Sohn et al., 2020)	$67.69_{2.8}$	$85.01_{0.6}$	$80.65_{0.9}$	$71.76_{0.5}$	$72.31_{0.6}$
FlexMatch (Zhang et al., 2021)	$67.87_{0.5}$	$84.87_{1.1}$	$79.91_{0.2}$	$72.21_{0.3}$	$73.59_{0.4}$
FreeMatch (Wang et al., 2023b)	$67.75_{1.8}$	$84.65_{0.6}$	$80.52_{1.2}$	$72.59_{0.8}$	$73.21_{1.1}$
SoftMatch (Chen et al., 2023)	$68.11_{1.3}$	$84.36_{0.7}$	$\underline{80.83}_{1.2}$	$72.35_{0.5}$	$73.11_{0.6}$
VerifyMatch (Ours)	$\mathbf{71.03}^{\dagger}_{2.1}$	$\mathbf{86.96}^{\dagger}_{0.8}$	$\mathbf{82.06}_{0.3}$	$\mathbf{74.20}^{\dagger}_{0.5}$	$\mathbf{74.10}^{\dagger}_{0.3}$

Table 1: The comparison of test accuracy (%) of our method and baselines. The underlined text shows the best performance baseline methods. We report the mean and standard deviation across three training runs with random restarts. †: VerifyMatch improves the the best baseline at $p < 0.05$ with paired t-test.

MixText (Chen et al., 2020) uses MixUp to interpolate labeled and unlabeled data in the hidden space of BERT on transformer layers 7, 9, and 12. The pseudo-label of unlabeled data is generated by multiple back-translations combined with a weighted average of their predictions.

SSL for NLI (Sadat and Caragea, 2022) is a self-debiasing method on unlabeled samples that are generated by fine-tuning conditional pre-trained language models, but only employs unlabeled samples whose model confidence in pseudo-label class is above a pre-defined fixed high-threshold value.

4.3 Implementation Details

We use Llama-3-8B-Instruct as LLMs and use BERT-base as a task classifier from HuggingFace Transformers library. The hyper-parameters settings are shown in Appendix A.1.

5 Results and Analysis

Main Results We observe our method improves over all baseline methods as shown in Table 1. We can also observe that in-context learning (ICL) and zero-shot learning (ZSL) on LLMs generally perform significantly worse compared to fine-tuning (FT) BERT. While LM-BFF and LM-BFF+Demo achieve better performance compared to ICL/ZSL, it still generally performs worse compared to FT BERT, even though we use the same number of labeled data on both settings. We conclude fine-tuning is still a robust method. We observe SSL baselines utilizing the same pseudo-generated data as unlabeled data as our approach (i.e., *UDA, MixText, SSL for NLI, FixMatch, FlexMatch, SoftMatch, FreeMatch*) outperform data augmentation baselines (i.e., *Back Translation, TMix*), and FT BERT. We conclude leveraging pseudo-generated data boosts performance more than when we only utilize labeled data. Still, our method achieves better performance than the best SSL baseline. In particular, our method outperforms SoftMatch, that also leverages all samples from the unlabeled data, supporting that our denoising strategy through MixUp to incorporate unconfident samples is effective.

In addition to this, to understand the effect of utilizing different LLMs in the pseudo-generated data construction of VerifyMatch instead of using LoRA Llama 3, we explore zero-shot prompting Llama 2, LoRA Llama 2, and fully fine-tune GPT-2, and report results in Appendix A.2. We find that using LoRA Llama 3 in VerifyMatch achieves the best result compared to using other LLMs. To understand this, we provide the comparison of pseudo-generated samples on MNLI using various LLMs in Appendix A.3. We conclude that using LoRA Llama 3 in VerifyMatch is the reasonable design choice to generate hypotheses for each class label so that we achieve the best performance by leveraging Llama 3 LoRA.

Various low-resource settings We evaluate VerifyMatch by lowering the number of labeled samples per class to 500 and 1,000 and show results in Table 3. The size of the pseudo-generated data re-

	RTE	SICK	SNLI-2.5k	MNLI-2.5k$_m$	MNLI-2.5k$_{mm}$
VerifyMatch (Ours)	**71.03**$_{2.1}$	**86.96**$_{0.8}$	**82.06**$_{0.3}$	**74.20**$_{0.5}$	**74.10**$_{0.3}$
w/o mismatched PG data	68.55$_{1.5}$	85.37$_{0.8}$	80.85$_{0.6}$	71.87$_{0.4}$	72.12$_{0.8}$
w/o unconfident PG data	68.34$_{1.3}$	85.88$_{0.5}$	79.21$_{0.4}$	71.71$_{0.6}$	72.13$_{0.2}$
w/ Lower Weights	68.49$_{2.1}$	85.59$_{1.2}$	78.92$_{0.8}$	72.12$_{0.5}$	72.36$_{0.4}$
w/ Single Llama 3	69.54$_{1.0}$	86.11$_{0.6}$	81.45$_{0.5}$	72.03$_{0.8}$	72.31$_{0.5}$
w/ Fixed Threshold	65.92$_{2.0}$	86.00$_{0.4}$	80.36$_{0.8}$	72.46$_{0.7}$	73.06$_{0.4}$
w/ Median-Conf	68.88$_{1.5}$	86.24$_{0.5}$	80.91$_{0.2}$	72.91$_{0.4}$	73.57$_{0.4}$

Table 2: The results comparisons of ablation study.

	RTE	SICK	SNLI	MNLI$_m$	MNLI$_{mm}$
FT BERT, 500 labeled data	58.16	81.48	63.35	55.79	56.88
SoftMatch, 500 labeled data	65.38	82.22	73.72	62.21	62.81
VerifyMatch, 500 labeled data	66.43	83.99	76.62	68.73	69.26
FT BERT, 1,000 labeled data	60.90	84.63	71.89	64.85	65.37
SoftMatch, 1,000 labeled data	68.11	84.36	77.35	66.78	66.63
VerifyMatch, 1,000 labeled data	71.03	86.96	78.57	69.17	69.81
FT BERT, 2,500 labeled data	-	-	79.03	69.26	70.26
SoftMatch, 2,500 labeled data	-	-	80.83	72.35	73.11
VerifyMatch, 2,500 labeled data	-	-	82.06	74.20	74.10

Table 3: The comparison on various low-resource settings. The maximum number of samples in each class for RTE and SICK is 1,000 since these datasets are small in size.

	RTE	SICK	SNLI-2.5k	MNLI-2.5k$_m$	MNLI-2.5k$_{mm}$
7,500 PG samples	69.55	85.56	80.44	72.55	73.01
15,000 PG samples	71.03	86.96	82.06	74.20	74.10
30,000 PG samples	70.11	86.85	81.19	73.34	73.48
45,000 PG samples	68.47	85.76	81.01	72.15	72.71
60,000 PG samples	68.19	85.87	80.58	72.11	72.61

Table 4: The comparison of our method varying the number of pseudo-generated (PG) samples.

Train Data	SNLI-2.5k			MNLI-2.5k		
Test Data	SNLI-hard	HANS	DNLI	SNLI-hard	HANS	DNLI
Fine-tuning BERT	65.33	49.97	43.57	56.50	49.82	68.83
VerifyMatch	**67.61**†	50.16	43.99	**59.05**†	**50.31**†	**76.15**†

Table 5: The comparison between our method and the BERT fine-tuning baseline method on challenging out-of-distribution data. †: VerifyMatch improves the Fine-tuning BERT baseline at $p < 0.05$ with paired t-test.

mained unchanged (i.e., 15,000 samples per class). VerifyMatch achieves the best performance compared to baselines on all settings.

Varying the number of pseudo-generated samples We use different amounts of pseudo-generated (PG) samples (from 7,500 to 60,000 samples per class) in VerifyMatch while maintaining the size of labeled samples (e.g., 2,500 samples per class) and show results in Table 4. We observe the performance becomes worse in general when using more than 15,000 PG samples per class. Interestingly, we observe that performance may degrade when increasing the amount of pseudo-generated data. This is because the larger number of pseudo-generated samples possibly contains erroneously labeled samples that can significantly hurt the performance.

Out-of-domain results To test the robustness of VerifyMatch, we use in-domain trained models to predict out-of-distribution test samples. Specifically, we train the model on SNLI-2.5k and MNLI-2.5k (using 2,500 labeled samples per class) respectively, and test it on SNLI-hard (Gururangan et al., 2018), HANS (McCoy et al., 2019), and DNLI (Welleck et al., 2019). We report the results in Table 5. We observe improvements in VerifyMatch compared to the baseline and conclude our approach is also robust.

6 Ablation Study

Mismatched & unconfident Pseudo-Generated (PG) data To explore the impact of identifying mismatched and unconfident PG data and incorporating them after denoising through MixUp, we show the results of VerifyMatch without leveraging them. These results are obtained by removing $\mathcal{L}_{mm}$ and $\mathcal{L}_{unconf}$ one at a time in the final training loss Eq. (7). That is, we simply discard mismatched PG data and unconfident PG data, and show results in Table 2, under the line "w/o mismatched PG data (w/o $\mathcal{L}_{mm}$)" and "w/o unconfident PG data (w/o $\mathcal{L}_{unconf}$)", respectively. We observe a drop in performance which shows the effectiveness of using both PG samples through MixUp.

MixUp as a Denoising Technique To understand the use of MixUp as a denoising technique, we compare it with VerifyMatch employing a different denoising method. Specifically, we lower the weights of both mismatched and unconfident pseudo-generated samples during training as in Chen et al. (2023). We report the results in Table 2 under the line "w/ Lower Weights". Note that this result is different from SoftMatch reported in Table 1 since we specifically identify both mismatched and unconfident PG data, whereas SoftMatch in Table 1 only identifies unconfident data. We observe that this results in performance degradation compared to using MixUp as denoising, demonstrating that MixUp offers a more effective approach.

	FEVER-2.5k			QQP-2.5k	
	Dev	Symm-v1	Symm-v2	Test	TwitterPPDB
FT BERT (Devlin et al., 2019)	79.75	49.37	57.31	76.08	84.68
Back Translation (Edunov et al., 2018)	80.54	48.66	57.04	76.60	85.48
FixMatch (Sohn et al., 2020)	82.21	51.16	58.37	78.59	84.69
UDA (Xie et al., 2020)	82.89	52.38	60.57	79.28	84.09
FreeMatch (Wang et al., 2023b)	81.07	49.23	57.58	79.76	84.71
SoftMatch (Chen et al., 2023)	83.72	53.08	61.37	79.85	84.83
VerifyMatch (Ours)	**85.68**†	**54.95**	**62.12**†	**80.89**†	**86.03**†

Table 6: Evaluation on fact verification (FEVER) and paraphrase detection (QQP) on in- and out-of-domain test data. †: VerifyMatch improves the best baseline at $p < 0.05$ with paired t-test.

Single Llama 3 We use class-specific fine-tuned Llama 3 models to ensure the coverage of all classes in pseudo-generated samples. To explore the effect of this, we use a single fine-tuned Llama 3 model, and show results in Table 2 (i.e., w/ Single Llama 3). We observe performance degradation in all cases, which proves the effectiveness of using class-specific fine-tuned Llama 3 models.

The average confidence Instead of calculating the average confidence as in Eq. (3), we use (1) a fixed threshold (i.e., 0.9), and (2) the median of the model's confidence on a predicted pseudo-label class in a PG batch , and show the results in Table 2 under the lines "w/ Fixed Threshold" and "w/ Median-Conf", respectively. We observe that neither case outperforms our method, supporting our design choice is reasonable.

Evaluation on other NLU tasks We evaluate our method for other sentence pairs classification tasks, which are fact verification, and paraphrase detection, and show results in Table 6. We follow the similar settings as SNLI-2.5k/MNLI-2.5k for the evaluation (i.e., randomly sampled 2,500 labeled data per class and generate pseudo-generated data).We evaluate both in-domain (ID) and challenging out-of-domain (OOD) test data. For fact verification, we evaluate FEVER-dev and Symmetric-v1/v2 (Schuster et al., 2019) test data. For paraphrase detection, we evaluate QQP and TwitterPPDB (Lan et al., 2017) test data. VerifyMatch outperforms competitive baselines on both ID and OOD test data, proving its effectiveness.

7 Conclusion

We proposed VerifyMatch, which constructs pseudo-generated samples using large language models (LLMs), and introduced semi-supervised learning (SSL) that acts as a verifier to ensure the quality of pseudo-generated samples for natural language inference. For SSL, we further proposed to identify and incorporate mismatched and unconfident pseudo-generated samples after denoising through MixUp, which allows a model to have access to a broader range of training samples. We empirically validate that VerifyMatch achieves competitive performance compared to strong baselines.

8 Limitations

Our approach, like any other semi-supervised approach, is computational more expensive than standard supervised learning. Nonetheless, our empirical results consistently demonstrate significant performance improvements. We believe that our method provides an important step forward for semi-supervised learning on NLI datasets, providing valuable insights.

Acknowledgements

We thank the National Science Foundation for support from grants IIS-2107518 which supported the research and the computation in this study. We also thank our reviewers for their insightful feedback and comments.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2024. Gpt-4 technical report.

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1998–2022, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. 2020. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. 2023. Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning. volume abs/2301.10921.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157.

Yuhan Chen, Shuqi Li, and Rui Yan. 2024. FlexiQA: Leveraging LLM's evaluation capabilities for flexible knowledge selection in open-domain question answering. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 56–66, St. Julian's, Malta. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Truong Do, Phuong Nguyen, and Le-Minh Nguyen. 2023. Structsp: Efficient fine-tuning of task-oriented dialog system by using structure-aware boosting and grammar constraints. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10206–10220. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Aparna Garimella, Rada Mihalcea, and Akhash Amarnath. 2022. Demographic-aware language model fine-tuning as a bias mitigation technique. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 311–319, Online only. Association for Computational Linguistics.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021a. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xuming Hu, Chenwei Zhang, Fukun Ma, Chenyao Liu, Lijie Wen, and S Yu Philip. 2021b. Semi-supervised relation extraction via incremental meta self-training. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 487–496.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. Quora question pairs. In *First Quora Dataset Release: Question Pairs*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Dong-Hyun Lee. 2013. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*.

Ju Hyoung Lee, Sang-Ki Ko, and Yo-Sub Han. 2021. Salnet: Semi-supervised few-shot text classification with attention-based lexicon construction. In *AAAI*.

Ximing Li and Bo Yang. 2018. A pseudo label based dataless naive Bayes algorithm for text classification with seed words. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1908–1917, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18608–18616. AAAI Press.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022a. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. European Language Resources Association (ELRA).

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Zeping Min, Jinfeng Bai, and Chengfei Li. 2024. Leveraging local variance for pseudo-label selection in semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14370–14378.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Seo Yeon Park and Cornelia Caragea. 2022. A data cartography based MixUp for pre-trained language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4244–4250, Seattle, United States. Association for Computational Linguistics.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems*, 34:11054–11070.

Dan Qiao, Chenchen Dai, Yuyang Ding, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. 2022. SelfMix: Robust learning against textual label noise with self-mixup training. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 960–970, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Mobashir Sadat and Cornelia Caragea. 2022. Learning to infer from unlabeled data: A semi-supervised learning approach for robust natural language inference. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4763–4776, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ramit Sawhney, Megh Thakkar, Shivam Agarwal, Di Jin, Diyi Yang, and Lucie Flek. 2021. HypMix: Hyperbolic interpolative data augmentation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9858–9868, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fabian David Schmidt, Ivan Vulić, and Goran Glavaš. 2022. Don't stop fine-tuning: On training regimes for few-shot cross-lingual transfer with multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10725–10742, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. 2019. Towards debiasing fact verification models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3419–3425, Hong Kong, China. Association for Computational Linguistics.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608.

Anirudh Som, Sujeong Kim, Bladimir Lopez-Prado, Svati Dhamija, Nonye Alozie, and Amir Tamrakar. 2020. A machine learning approach to assess student group collaboration using individual level behavioral cues. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 79–94. Springer.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in neural information processing systems*, 32.

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

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Lijing Wang, Yingya Li, Timothy A. Miller, Steven Bethard, and Guergana Savova. 2023a. Two-stage fine-tuning for improved bias and variance for large pretrained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 15746–15761. Association for Computational Linguistics.

Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. 2023b. Freematch: Self-adaptive thresholding for semi-supervised learning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33:6256–6268.

Luyu Yang, Yan Wang, Mingfei Gao, Abhinav Shrivastava, Kilian Q Weinberger, Wei-Lun Chao, and Ser-Nam Lim. 2021. Deep co-training with task decomposition for semi-supervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8906–8916.

Wenpeng Yin, Huan Wang, Jin Qu, and Caiming Xiong. 2021. Batchmixup: Improving training by interpolating hidden states of the entire mini-batch. In *FINDINGS*.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077, Online. Association for Computational Linguistics.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34.

Haode Zhang, Haowen Liang, Yuwei Zhang, Li-Ming Zhan, Xiao-Ming Wu, Xiaolei Lu, and Albert Lam. 2022. Fine-tuning pre-trained language models for few-shot intent detection: Supervised pre-training and isotropization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 532–542, Seattle, United States. Association for Computational Linguistics.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

A Appendix

A.1 Training Details

Pseudo-generated data construction We mainly use `Llama-3-8B-Instruct` as large language models (LLMs) in pseudo-generated data construction from HuggingFace Transformers library[6]. For LoRA-tuned Llama 3 (Low-Rank Adaptation; Hu et al. (2021a)), we set hyperparameters as follows: learning rate as 2e-3, training epoch as 3, LoRA alpha as 8, LoRA dropout as 0.05, train batch size as 1, gradient accumulation steps as 64. We set the LoRA rank value as 4 for RTE, 16 for SICK, and 8 for both SNLI and MNLI datasets. We use the system prompt as follows: "`<s>[INST] «SYS»\nYou are a helpful, respectful, and honest assistant. Always follow the instructions provided and answer honestly.\n«/SYS»\n\n`" and provide customized prompts depending on target labels as follows: (1) entailment: "`We will give you the sentence. Using only the given sentence and what you know about the world. Write one alternate sentence that is definitely a` **`true`** `description of the given sentence. Sentence: {premise}`", (2) contradiction: "`We will give you the sentence. Using only the given sentence and what you know about the world. Write one alternate sentence that is definitely a` **`false`** `description of the given sentence. Sentence: {premise}`" (3) neutral: "`We will give you the sentence. Using only the given sentence and what you know about the world. Write one alternate sentence that` **`might be a true`** `description of the given sentence. Sentence: {premise}`". We construct the system prompt as suggested by the Llama 3 pre-training step while constructing task-dependent prompts by referring to the instructions provided when generating a large-scale Natural Language Inference (NLI) benchmark as in Bowman et al. (2015).

We additionally compare the results of our method using other various Large Language Models (LLMs) on different settings in pseudo-generated data construction, which are zero-shot prompting/LoRA Llama 2 and full fine-tuning GPT-2, and show results in Table 7. For this, we use `Llama-2-7b-chat-hf`, and use `GPT2` from the huggingface transformers library. For zero-shot prompting Llama 2 in pseudo-generated data construction, we observe that using the same prompts as LoRA Llama 3 results in poor genera-

[6]https://huggingface.co/docs/ transformers/index

tion quality. This is because prompting can be a brittle process due to LLMs being overly sensitive to the surface form of the instruction (Perez et al., 2021; Lu et al., 2022). Hence, we devise the following prompt for zero-shot prompting Llama 2 for pseudo-generated data construction: "`Please generate a hypothesis that has the {target label} relationship with the given sentence: {premise} Hypothesis:`", where "`target label`" is every possible class label. For LoRA Llama 2, we follow the same prompts and hyper-parameters as LoRA Llama 3. To fully fine-tune GPT-2, we set the training epoch as 30, training batch size as 16, eval batch size as 1, weight decay 0.01, and use the AdamW optimizer (Loshchilov and Hutter, 2018) with learning rate 3e-5. To produce class-specifically fine-tuned GPT-2, we provide training samples to the model by concatenating the premise and hypothesis with special tokens that belong to a specific class. We use the following special tokens: `[BOS]`, `[EOS]`, `[UNK]`, `[PAD]`, `[SEP]`. We then generate hypotheses by giving unlabeled premises as the inputs using class-specifically tuned GPT-2 models. For this, we employ beam search with beam size 5 and penalize repeated bi-grams to introduce diversity to the generated target sentences where the minimum length is 30 and the maximum length is 100. The hypotheses generation models leveraging fully fine-tuned GPT-2 are trained in $\approx$ 1 hour using a single NVIDIA RTX A5000 GPU. It took less than $\approx$ 1 hour to generate the hypotheses for each dataset using the same GPU.

Verifier For the verifier, we use `bert-base-uncased` for BERT as a task model where we use the final layer of BERT `[CLS]` token output representations with a maximum of 3 epochs. We optimize the models by using AdamW (Loshchilov and Hutter, 2018). We set a batch size of 32 for both labeled and unlabeled data, a learning rate of 2e-5, a gradient clip of 1.0, and no weight decay. For MixUp, we set the beta distribution hyperparameter $\alpha = 0.4$ for λ in Eq. (1), following previous studies that observed $\alpha = 0.4$ to yield the best performance in text classification tasks (Thulasidasan et al., 2019; Som et al., 2020). We utilize a sharpening function on probability distribution produced by the verifier, BERT, for numerical stability as follows:

$$\text{Sharpen}(P(y|\mathbf{x}), T) = \frac{P(y|\mathbf{x})^{\frac{1}{T}}}{||P(y|\mathbf{x})^{\frac{1}{T}}||_1}$$

where $||.||_1$ is l_1-norm of the vector, T is a temperature hyper-parameter and set as 0.5. We report the mean and standard deviation across three training runs with random restarts.

All experiments are conducted on two NVIDIA RTX A5000 GPUs with a total time for fine-tuning all models being under 24 hours. For semi-supervised learning baseline methods, we use batch size 16 across all datasets. We set $\tau = 0.95$ in FixMatch (Sohn et al., 2020), set $\tau = 0.95$ in FlexMatch (Zhang et al., 2021), and $\lambda = 0.3$ to obtain τ in FreeMatch (Wang et al., 2023b).

To evaluate VerifyMatch on other sentence-pair classification tasks such as fact verification and paraphrase detection, we do the following steps: For the fact verification task, we randomly sample 2,500 labeled data per class from FEVER original training data and then construct 2,500 pseudo-generated samples per class. For the paraphrase detection task, we also randomly sampled 2,500 labeled data per class from QQP (Iyer et al., 2017) original training data and then generate 10,000 pseudo-generated samples per class.

A.2 Various Lage Language Models (LLMs)

To understand the impact of leveraging various LLMs in the pseudo-generated data construction of VerifyMatch instead of using LoRA Llama 3, we explore zero-shot prompting Llama 2, LoRA Llama 2, and fully fine-tune GPT-2, and show results in Table 7. Interestingly, we observe better performance when using fully fine-tuned GPT-2 than zero-shot prompting Llama 2 and LoRA Llama 2. We posit that this is because the fully fine-tuning GPT-2 generates more meaningful hypotheses for each unlabeled premise with reasonable class labels than zero-shot prompting Llama 2 and LoRA Llama 2. Still, VerifyMatch which leverages LoRA Llama 3 in pseudo-generated data construction achieves the best performance in general, which proves the effectiveness of our design choice.

A.3 Examples of Pseudo-generated Data

We show examples from the pseudo-generated data on MNLI in Table 8 in comparison of full fine-tuning GPT-2, Zero-shot prompting Llama 2, LoRA Llama 2 and LoRA Llama 3. We make the following observations: (1) Llama 3 (LoRA) and

	RTE	SICK	SNLI-2.5k	MNLI-2.5k$_m$	MNLI-2.5k$_{mm}$
VerifyMatch (Ours, LoRA Llama 3)	**71.03**$_{2.1}$	**86.96**$_{0.8}$	**82.06**$_{0.3}$	**74.20**$_{0.5}$	74.10$_{0.3}$
VerifyMatch w/ Zero-shot Prompting Llama 2	69.82$_{2.5}$	85.93$_{0.5}$	81.33$_{0.3}$	73.08$_{0.5}$	73.55$_{0.8}$
VerifyMatch w/ LoRA Llama 2	69.77$_{1.8}$	<u>86.92</u>$_{0.6}$	81.16$_{0.8}$	73.49$_{0.2}$	73.90$_{0.2}$
VerifyMatch w/ Full Fine-tuning GPT-2	<u>70.84</u>$_{2.2}$	86.81$_{0.3}$	<u>81.85</u>$_{0.6}$	<u>73.92</u>$_{0.3}$	**<u>74.38</u>**$_{0.5}$

Table 7: The comparison of our method with different Large Language Models (LLMs) in pseudo-generated data construction. Bold text shows the best performance. Underlined text shows the best performance among other LLM settings than ours.

Premise	Generated Hypothesis				Target Label
	Full fine-tuning GPT-2	Zero-shot Prompting Llama 2	LoRA Llama 2	LoRA llama 3	
The Turks adopted the Indian cuisine and costume as well as a modified form of the Hindu caste system.	A similar style was developed by Turkey in India, with its custom for dancing on horseback	The Turks adopted the Indian cuisine and costume as well as a modified form of the Hindu caste system in order to create a distinct cultural identity that would set them apart from their neighbors.	The Turks took on Indian cuisine and fashion.	The Turks adopted Hinduism.	Entailment
During World War II, the US built an Air Force base here that later became the international airport.	The air force was established in Germany during WWI when it started building a facility there to serve as this country's national aviation terminal for future	The US built an Air Force base here in order to protect its interests during World War II.	The US built an air force base in the country during World War II.	The US built an Air Force base here during World War II.	Entailment
However, the evidence was strong for a harm reduction effect across various indicators of risk and re-injury.	There is no significant difference in risks from these factors to different outcomes at all levels except as shown above	Despite the evidence of a harm reduction effect, the program was not effective in reducing injuries	The evidence was not strong for a harm reduction effect across various indicators of risk and re-injury.	The evidence was weak for a harm reduction effect.	Contradiction
Even so, its ascendancy is troublesome.	The Ascendant's descent has been very successful and it doesn't require any special precautions or procedures to gain control of the land at all times for you as a member in this country	Despite its ascendancy, it will face a decline in popularity.	It is a blessing to have the ascendancy.	Its ascendancy is a good thing.	Contradiction
Sisters are, you know, Mr. Hastings.	I'll tell the truth about my father and mother's divorce because they were just divorcing in December of that year (as I did)	Sisters are, on average, more likely to have a positive relationship with their siblings than with their parents.	Sisters are like the best of friends.	Sisters are the best, Mr. Hastings.	Neutral
And responsibilities of any groups or committees of senior managers, and the relationships between them.	The relationship to these leaders is important because they have a lot more than just one person in their office at all times that you can get from an executive officer	The distribution of responsibilities among senior managers and the relationships between them are critical to the effective functioning of an organization, but they do not have a direct impact on the organization's overall performance.	The responsibilities of the groups and committees are clearly defined and understood.	The responsibilities of the senior managers are to make sure the company is successful.	Neutral

Table 8: The comparisons of generated hypothesis among full fine-tuning GPT-2, Zero-shot Prompting Llama 2, LoRA Llama 2, and LoRA Llama 3 in pseudo-generated data construction of VerifyMatch.

		Competence Test			Distraction Test						Noise Test	
		Antonymy		Numerical	Word Overlap		Negation		Length Mismatch		Spelling Error	
		m	mm	Reasoning	m	mm	m	mm	m	mm	m	mm
FT BERT		4.41	5.68	60.53	64.44	64.42	64.28	65.05	60.53	61.55	60.08	60.32
VerifyMatch w/ GPT-2	RTE	10.05	11.55	64.57	68.15	69.09	66.43	67.52	69.47	69.92	67.60	68.14
VerifyMatch w/ Llama 3		12.26	10.16	66.49	68.54	68.88	67.76	68.44	69.96	70.04	68.85	69.20
FT BERT		1.27	0.63	27.35	26.69	26.66	29.67	29.18	29.06	28.86	32.93	31.56
VerifyMatch w/ GPT-2	SICK	6.79	0.81	33.53	30.48	31.52	30.05	30.23	35.33	38.08	35.15	33.78
VerifyMatch w/ Llama 3		6.93	1.16	34.21	31.20	30.83	31.16	31.55	38.54	40.28	33.39	34.25
FT BERT		50.16	48.17	32.32	35.54	31.82	40.54	40.49	43.99	43.19	41.57	40.96
VerifyMatch w/ GPT-2	SNLI	73.99	89.85	35.42	40.77	41.01	44.76	44.20	53.14	54.79	50.81	50.35
VerifyMatch w/ Llama 3		75.16	86.65	36.11	40.86	45.53	45.88	45.56	55.30	55.47	51.12	51.58
FT BERT		18.64	21.91	24.76	47.46	49.28	36.38	32.27	59.42	60.68	58.41	52.12
VerifyMatch w/ GPT-2	MNLI	25.56	25.95	31.89	43.61	43.63	36.54	36.81	61.05	62.17	57.09	57.85
VerifyMatch w/ Llama 3		27.76	28.80	32.38	44.45	44.09	35.98	36.12	62.23	62.38	56.63	57.16

Table 9: The comparison of stress test accuracy (%) of the baseline fine-tuning BERT (FT BERT) and our method with different large language models (LLMs). GPT-2 refers to full fine-tuning GPT-2 and Llama 3 refers to LoRA tuning Llama 3.

GPT-2 generate higher quality hypotheses / pseudo-labels compared with Llama 2 (LoRA) and Llama 2 (zero-shot) although the GPT-2 generations are longer in length compared with Llama 3; in some cases, GPT-2 hallucinates; (2) Llama 2 (zero-shot) generates in most cases longer hypotheses compared with the other LLMs; (3) Llama 2 (zero-shot) yields slightly better generation results compared with Llama 2 (LoRA) especially on the contradiction class on which Llama 2 (LoRA) often simply adds negation words in front of the verb; (4) Llama 2 (zero-shot), although it was better than Llama 2 (LoRA), in many cases we observed that it was generating hypotheses by copying the premise and then generating additional tokens; and (5) None of the four LLMs introduced toxicity or vulgarity content in the generated hypotheses. Hence, we conclude that Llama 3 and GPT-2 could be the best competitor LLMs to use in VerifyMatch.

A.4 Baseline prompting

To obtain the baseline prompting results of in-context and zero-shot learning on Large Language Models (LLMs), we construct the prompts as follows by referring to Brown et al. (2020): `{premise} \nQuestion: {hypothesis} True, False, or Neither?\nAnswer: ` For in-context

	RTE	SICK	SNLI-2.5k	MNLI-2.5k$_m$	MNLI-2.5k$_{mm}$
VerifyMatch (Ours, w/ LoRA Llama 3)	**71.03**$_{2.1}$	**86.96**$_{0.8}$	**82.06**$_{0.3}$	**74.20**$_{0.5}$	74.10$_{0.3}$
VerifyMatch (w/ Full FT GPT-2)	70.84$_{2.2}$	86.81$_{0.3}$	81.85$_{0.6}$	73.92$_{0.3}$	**74.38**$_{0.5}$
w/o mismatched PG data (w/o $\mathcal{L}_{mm}$)	70.03$_{1.2}$	86.21$_{0.1}$	81.15$_{0.5}$	72.45$_{0.9}$	72.58$_{0.7}$
w/o unconfident PG data (w/o $\mathcal{L}_{unconf}$)	69.54$_{1.5}$	86.46$_{0.1}$	80.48$_{0.3}$	72.83$_{0.1}$	73.01$_{0.1}$
w/ Lower Weights	69.44$_{1.1}$	86.29$_{0.3}$	80.14$_{0.2}$	72.86$_{0.7}$	73.14$_{0.4}$
w/ Single GPT-2	69.31$_{1.8}$	86.05$_{0.6}$	81.05$_{0.3}$	71.02$_{0.6}$	71.49$_{0.3}$
w/ Fixed Threshold	66.11$_{1.3}$	85.97$_{0.7}$	80.24$_{0.5}$	72.91$_{0.2}$	73.22$_{0.3}$
w/ Median-Conf	69.94$_{1.8}$	86.36$_{0.3}$	80.91$_{0.2}$	73.05$_{0.1}$	73.47$_{0.2}$

Table 10: The results comparisons of ablation study using GPT-2 full fine-tuning in pseudo-generated data construction of VerifyMatch.

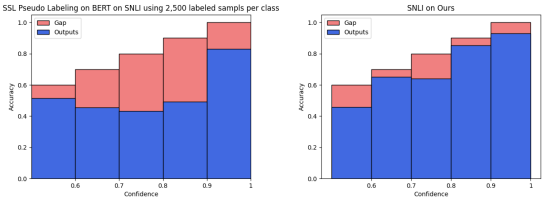

Figure 2: Reliability diagrams of SNLI on BERT using a fixed high-confidence threshold in self-training (left), and our proposed method (right) with pseudo-generated data by GPT-2.

	RTE	SICK	SNLI	MNLI$_m$	MNLI$_{mm}$
FT BERT, 500 labeled data	58.16	81.48	63.35	55.79	56.88
VerifyMatch, 500 labeled data (GPT-2)	**65.65**	**83.83**	**74.41**	**63.17**	**64.47**
FT BERT, 1,000 labeled data	60.90	84.63	71.89	64.85	65.37
VerifyMatch, 1,000 labeled data (GPT-2)	**70.84**	**86.81**	**77.52**	**67.91**	**68.43**

Table 11: The comparison on various low-resource settings using full fine-tuning GPT-2.

	RTE	SICK	SNLI-2.5k	MNLI-2.5k$_m$	MNLI-2.5k$_{mm}$
7,500 PG samples	70.03	86.04	81.05	72.27	72.65
15,000 PG samples	**70.84**	86.81	**81.85**	**73.92**	**74.38**
30,000 PG samples	69.93	**86.92**	81.26	73.63	73.53
45,000 PG samples	69.78	86.43	81.11	72.35	73.48
60,000 PG samples	69.82	86.47	80.82	72.71	72.96

Table 12: The comparison of our method varying the number of pseudo-generated (PG) samples using full fine-tuning GPT-2.

learning, we concatenate randomly selected 10 labeled samples (around 3 labeled samples per class) at the beginning of the prompts with their answers. We follow the same evaluation protocol provided by Gao et al. (2021) to report results.

A.5 Overconfidence

MixUp is widely known for preventing the model from being overly confident in its predictions and reducing miscalibration errors. To explore whether the VerifyMatch method also relieves the miscalibration problem, we plot the reliability diagram of our proposed method on the SNLI dataset and compare it to a method of using fixed high-threshold (i.e., 0.9) in Figure 2. We observe that VerifyMatch alleviates the overconfidence problem, as the gap between accuracy and confidence in each bin in the reliability diagrams is reduced compared to the baseline method.

A.6 Robustness Analysis

To explore the robustness of our method for NLI, we test on NLI stress test (Naik et al., 2018) and show results in Table 9. The stress test is developed based on the weakness of NLI models in various aspects such as, presence of a negation word such as 'no' causes the model to predict the sample as contradiction class (i.e., *negation*), and word overlap between premise and hypothesis results in model to predict the sample as entailment class (i.e., *word overlap*), etc. To this end, we evaluate 11 different tests that are divided into three parts: (1) competence test, (2) distraction test, and (3) noise test. We compare the fine-tuning (FT) BERT baseline method and our proposed approach, VerifyMatch, with leveraging full fine-tuning GPT-2, and LoRA Llama 3 in the pseudo-generated data construction, and show results in Table 9. We observe that our method shows better performance than the FT BERT baseline in general, all across the stress tests, which proves the robustness of our VerifyMatch.

A.7 VerifyMatch with full fine-tuning GPT-2

We observe the best performance when using full fine-tuning GPT-2, compared to LoRA Llama 2 and zero-shot prompting Llama 2 in pseudo-generated data construction. Consequently, we conduct the same ablation study as in Section 6, focusing on the full fine-tuning GPT-2, instead of leveraging LoRA Llama 3 in the pseudo-generated data construction of VerifyMatch. The results are presented in Table 10. Specifically, we compare the test accuracy (%) of VerfiyMatch (1) without leveraging mismatched pseudo-generated data (i.e., removing $\mathcal{L}_{mm}$ in the final training objective), (2) without using unconfident pseudo-generated data (i.e., removing $\mathcal{L}_{unconf}$), (3) with a denoising technique of lowering mismatched and unconfident pseudo-generated samples instead of using MixUp (i.e.,

w/ Lower weights), (4) using a single fully fine-tuned GPT-2 instead of using class-specifically fully fine-tuned GPT-2 (i.e., w/ Single GPT-2), (5) using a fixed threshold in identifying unconfident pseudo-generated samples (i.e., w/ Fixed Threshold), and (6) using the median confidence in identifying unconfident pseudo-generated samples (i.e., w/ Median-Conf). We observe there is a performance drop in all cases compared to VerifyMatch either leveraging full fine-tuned GPT-2 or LoRA Llama 3, which proves the effectiveness of each component in our proposed method.

We also explore VerifyMatch with full fine-tuning GPT-2 by lowering the number of labeled samples per class to 500 and 1,000 and show results in Table 11. The size of the pseudo-generated data constructed by fully fine-tuned GPT-2 remains the same as 15,000 samples per class. Notably, our proposed method consistently outperformed the baseline on all datasets, demonstrating its effectiveness. Furthermore, we vary the number of pseudo-generated data generated from class-specifically fine-tuned GPT-2 (from 7,500 to 60,000 samples per class), while using the size of labeled samples as 2,500 samples per class, and show results in Table 12. We observe the performance achieves the best performance when leveraging 15,000 pseudo-generated samples per class in general. We observe that using 60,000 PG samples per class results in performance degradation on all datasets. We posit that this is because the large number of pseudo-generated samples (e.g., more than 30,000 per class) compared to the limited labeled data (e.g., 2,500 per class) weakens the influence of the high-quality labeled samples in general.