Araida: Analogical Reasoning-Augmented Interactive Data Annotation

Chen Huang[♠♣], **Yiping Jin**[♡], **Ilija Ilievski**[◇], **Wenqiang Lei**[♠♣*], **Jiancheng Lv**[♠♣]
[♠]College of Computer Science, Sichuan University, China
[♣]Engineering Research Center of Machine Learning and Industry Intelligence, Ministry of Education, China
[♡]NLP Group, Pompeu Fabra University, Spain
[◇]ISEM, National University of Singapore, Singapore
wenqianglei@scu.edu.cn

Abstract

Human annotation is a time-consuming task that requires a significant amount of effort. To address this issue, interactive data annotation utilizes an annotation model to provide suggestions for humans to approve or correct. However, annotation models trained with limited labeled data are prone to generating incorrect suggestions, leading to extra human correction effort. To tackle this challenge, we propose Araida, an analogical reasoning-based approach that enhances automatic annotation accuracy in the interactive data annotation setting and reduces the need for human corrections. Araida involves an error-aware integration strategy that dynamically coordinates an annotation model and a k-nearest neighbors (KNN) model, giving more importance to KNN's predictions when predictions from the annotation model are deemed inaccurate. Empirical studies demonstrate that Araida is adaptable to different annotation tasks and models. On average, it reduces human correction labor by 11.02% compared to vanilla interactive data annotation methods.

1 Introduction

Data annotation is a challenging task that involves a tradeoff between annotation quality and budget. While some platforms offer a cost-effective solution by relying on ML models to annotate data automatically [1], the quality of such annotations is often compromised (Wang et al., 2022a). It is particularly true in the **limited data annotation** scenario where the annotation budget is limited or when unlabeled data are scarce (Ringger et al., 2007; Chaudhary et al., 2021; Huang et al., 2024).

Human-machine **interactive annotation** methods were introduced to reduce annotation effort while maintaining annotation quality (Klie et al., 2018, 2020; Le et al., 2021). As illustrated in Fig. 1, these methods introduce an *annotation model* to suggest labels (*model annotations*) to human annotators. The annotators accept a suggested label if it is correct. Otherwise, they have to correct the label manually. Compared to manual annotation, interactive annotation requires less human effort because human annotators only have to verify the model annotations instead of coming up with an answer from scratch, leading to potential speedup of the annotation process (Klie et al., 2020).

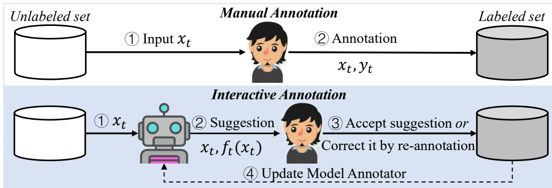

Figure 1: Comparison between manual annotation and interactive annotation.

Evidently, the annotation model's accuracy is crucial because incorrect suggestions require additional human effort to rectify. Existing methods update the annotation model based on previously accepted or corrected data (*ground-truth annotation*), aiming to reduce human corrections by improving prediction accuracy at each iteration (Klie et al., 2020; Wu et al., 2022). However, in the context of limited data annotation, the annotation model lacks sufficient labeled training data to reach a reasonable accuracy and is prone to providing incorrect suggestions (Rietz and Maedche, 2021). For example, in the span relation annotation example shown in Fig. 2 (blue), the annotation model continues to make mistakes on similar examples ([*car*, *tyre*]) even after the human annotator corrects the label '[*tree*, *leaf*]=>*component*'. As a result, this leads to more human corrections. Such a problem is crucial for interactive annotation and has been identified by recent work (Rietz and Maedche, 2021),

[*]Correspondence to Wenqiang Lei.

[1]For example, https://aws.amazon.com/sagemaker/groundtruth/.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10660–10675
August 11-16, 2024 ©2024 Association for Computational Linguistics

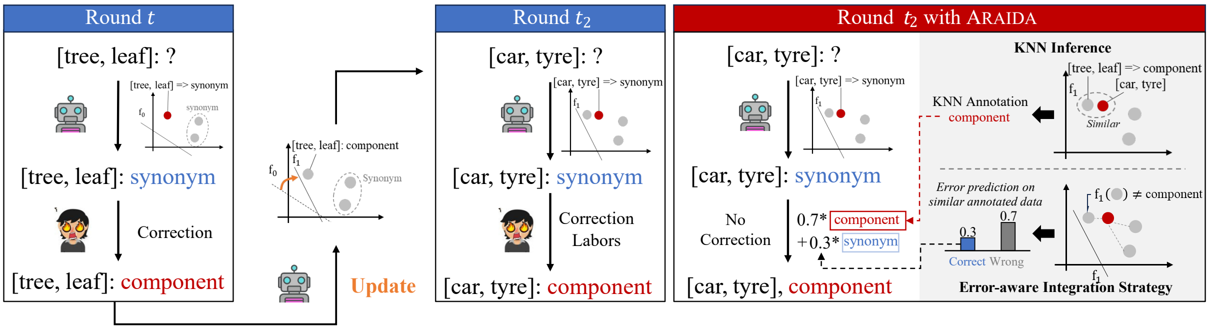

Figure 2: Example on span relation annotation. An under-trained annotation model results in more suggestion errors and increases human correction effort. ARAIDA improves the model annotation accuracy via the KNN model and the error-aware integration strategy for dynamical coordination of annotations.

but it has yet to be addressed.

Inspired by cognitive studies on efficient learning (Lake et al., 2017, 2015; Mitchell, 2021), finding that the human brain can learn from a few examples because our brain is continuously building analogies during the learning process of concepts to facilitate comprehension, we propose **A**nalogical **R**easoning-**A**ugmented **I**nteractive **D**ata **A**nnotation (ARAIDA), which is designed to improve interactive annotation under the limited data annotation setting. ARAIDA provides an annotation reference to the annotation model by retrieving previously human-labeled examples in the proximity of the example in consideration using the k-nearest neighbors (KNN) method. As illustrated in Fig. 2(red), the final suggestion combines the model annotation and the annotation reference provided by KNN via an error-aware integration strategy. This strategy dynamically coordinates the annotation model and KNN, giving more importance to KNN's prediction if the predicted label from the annotation model is estimated to be inaccurate.

We conduct simulated experiments for the limited data annotation task and estimate the human annotation effort based on the number of human corrections (or the number of suggestion errors) following Hwa (2000) and Kristjansson et al. (2004). We test ARAIDA on different word-level and sentence-level annotation tasks, combining with different annotation models (i.e., classic and LLM-based models). The result shows that ARAIDA consistently improves different annotation models' accuracy across various tasks. On average, it reduces human corrections by 11.02%. Further analysis attributes this improvement to the few-shot capability of the KNN module and the error-aware integration strategy that effectively synergizes complementary annotations. In summary, our contributions are as follows:

- Calling attention to the limited data annotation scenario. We highlight the under-trained problem of the annotation model, which is crucial in practice but overlooked in interactive annotation.
- Introducing ARAIDA that involves a KNN module and an error-aware integration strategy to alleviate the under-trained problem by facilitating coordination between the two model annotators (i.e., the vanilla annotation model and KNN).
- Demonstrating the efficacy of ARAIDA in enhancing suggestion accuracy, reducing human corrections, and showcasing its flexibility to combine with various annotation models through extensive experiments.

2 Related Work

Our research is tied to interactive data annotation, human analogical reasoning (KNN), and retrieval-based language models. We provide a literature review and highlight our differences.

Interactive Data Annotation. Interactive data annotation aims to reduce human annotation effort by incorporating an annotation model that suggests labels to human annotators during an interactive process (Klie et al., 2018, 2020; Le et al., 2021). The annotation model must be sample-efficient because, when we start a new annotation task, there are few labeled examples to learn from. Current studies focus on employing active learning (Klie et al., 2018; Laws et al., 2011; Casanova et al., 2020; Li et al., 2021; Huang et al., 2023) to prioritize annotating examples more likely to improve model accuracy. While active learning can reduce the required training data to some extent, it may not be effective in limited data annotation scenarios or when complex hypotheses or semantics are to be learned (Dasgupta, 2005; Rietz and Maedche, 2021). Another approach is to employ LLMs for automatic data annotation, which have demonstrated strong performance under zero-shot and few-shot settings (He et al., 2023; Gilardi et al., 2023). However, such performance might not be consistent for difficult tasks, as they may even perform worse than fine-tuned small language models (Xiao et al., 2023). Regardless of whether we use active learning and whether we use a classic or LLM-based annotation model, our empirical evidence demonstrates that ARAIDA can effectively decrease the amount of human corrections required.

KNN and Analogical Reasoning. While KNN has been extensively utilized in NLP community (Wang et al., 2019; Liu et al., 2023; Wang et al., 2022b), its underlying mechanism is often overlooked. To shed light on this, cognitive studies (Lake et al., 2017, 2015; Mitchell, 2021) revealed that the KNN inference process aligns with human analogical reasoning, enabling efficient learning (Lake et al., 2017, 2015). In particular, analogical reasoning establishes connections between relevant aspects of the current task and past experiences, forming abstractions that enhance human reasoning capabilities (Mitchell, 2021). In this context, KNN facilitates sample-efficient learning by leveraging similarities between the example to be labeled and previously annotated examples, resulting in exemplary solutions (Bautista et al., 2016), which reduce the training data requirement.

Retrieval-Based Language Models. There is growing interest in enhancing the output of language models by incorporating a retrieval module (usually KNN or alike) that interpolates with a datastore built from the training data (Khandelwal et al., 2019; Kassner and Schütze, 2020). Compared to vanilla language models, retrieval-based models ground the predictions in labeled training examples, potentially yielding better explainability and sample efficiency (Asai et al., 2023). This approach has shown promising results in tasks such as machine translation (Khandelwal et al., 2021; Liu et al., 2023), named entity recognition (Wang et al., 2022b), and question answering (Kassner and Schütze, 2020). While some studies have explored the use of dynamically adjusted combination weights between the language model and the retrieval module (Wang et al., 2021; Zheng et al., 2021; Jiang et al., 2021), our method differs significantly for two main reasons: 1) Different tasks. We are the pioneers in introducing KNN to the interactive data annotation task, whereas these methods are primarily designed for machine translation. 2) Different techniques. We adjust the weight by estimating the error of model predictions for each data point (e.g., sentence), whereas these methods learn the weight for each token without error estimation.

3 ARAIDA: The Proposed Method

We present ARAIDA, an analogical reasoning-based method for interactive data annotation that provides an annotation reference to the annotation model by retrieving previously human-labeled examples in the proximity of the example in consideration. We detail the KNN inference module in Section 3.1 and the error-aware integration strategy in Section 3.2. Finally, the optimization details are provided in Section 3.3.

Task Formalization and Overview. Let X denote the dataset that needs to be annotated, with C being the number of classes. Given a data batch x_t at time t, the annotation model f_t predicts label vectors $f_t(x_t) \in R^{|x_t| \times C}$, and the KNN module g_t infers label vectors $g_t(x_t) \in R^{|x_t| \times C}$ using previously annotated data stored in a datastore A_t. Then, we estimate the probability $\lambda_t \in R^{|x_t| \times 1}$ of the annotation model's predictions $f_t(x_t)$ being reliable, i.e., $argmax(f_t(x_t)) = y_t$, where $argmax(\cdot)$ returns indices of the classes with the highest predicted probability and $y_t \in R^{|x_t| \times 1}$ are the ground truth labels. Finally, we use λ_t to weigh the two predictions $f_t(x_t)$ and $g_t(x_t)$ through a linear weighted combination:

$$F_t(x_t) = \lambda_t \cdot f_t(x_t) + (1 - \lambda_t) \cdot g_t(x_t). \quad (1)$$

Notably, closed-source language models such as ChatGPT produce discrete labels rather than predicted distributions. Therefore, we cannot combine its predictions with KNN's using linear combination. In such case, we use binary values (0 or 1) for λ_t, which acts as a function allocation to determine whether f_t or g_t should apply to each example. Once the human approves or corrects the final suggestions, the datastore A_t is updated with the newly arrived data batch and its corresponding labels. In addition, the annotation model f_t (if applicable), KNN module g_t, and weighting strategy λ_t are updated via back-propagation.

3.1 KNN Inference

We utilize a weighted KNN to perform inference, defined as $g_t(x_t^i) = \frac{\sum_{a \in \rho_i} w_a y_a}{\sum_{a \in \rho_i} w_a}$, where $\rho_i \in A_t$ is the k nearest neighbors of each example x_t^i, y_a corresponds to the human annotation of each neighbor $a \in \rho_i$. The similarity between x_t^i and a is measured by $w_a = \frac{1}{d(x_t^i, a)}$, where $d(x_t^i, a) = \|w_{knn}(x_t^i - a)\|_2$ is a distance metric parameterized by w_{knn}. We use the similarity measure to retrieve ρ_i. To avoid overconfidence in KNN inference, we apply label smoothing to the labels of the retrieved neighbors. Specifically, we set $y_a = y_a(1-\alpha) + \alpha/C$, where $\alpha = 1 - \frac{1}{C}$.

Datastore Maintenance Strategy. The datastore A_t consisting of historically annotated data grows in size as the interactive annotation continues, causing the KNN's retrieval to be less time-efficient. To address this issue, we impose a constraint on the maximum datastore size using a pre-defined hyperparameter. We propose a class-aware maintenance strategy. Precisely, if A_t exceeds its budget, data that is from the majority class [2] and most similar to its class prototype[3] is discarded first. This strategy ensures that the datastore contains as many labeled data from different classes as possible while minimizing the impact on the class prototype. Appendix A.1 and A.3 present experiments using different datastore sizes and maintenance strategies.

3.2 Error-aware Integration Strategy

Motivation. A popular method to combine annotations from two models (i.e., annotation model and KNN) is to use a weighted linear combination with a constant weight [4] (Liu et al., 2023; Wang et al., 2022b). However, assuming that one model consistently outperforms the other on all unlabeled data is unrealistic. Furthermore, both models are updated with each new batch of data in interactive data annotation, and their relative performance will alter, making it infeasible to find the optimal weight through a one-off hyperparameter tuning. To address this issue, we propose an error-aware integration strategy that automatically assigns weights to different models, relying more on KNN inference when the annotation model's prediction is estimated to be inaccurate.

[2]The majority class refers to the class with the highest frequency in A_t.

[3]The class prototypes are the average of the feature vectors in A_t that belong to each class.

[4]Equivalent to when $\lambda_t(x_t)$ in Eq.1 is a constant.

Error Estimation of Model Annotation. We base on the intuition that if the model f_t consistently makes mistakes on previous examples similar to the current data point x_t^i, then its prediction $f_t(x_t^i)$ will likely be incorrect. To achieve this, we parameterize the integration strategy λ_t as a neural network to learn from the customized input.

- Customized Inputs. For each data point x_t^i, we derive the input $\varkappa_t^i$ to the integration strategy λ_t, which considers the local error estimation E_t^i and local density D_t^i. Specifically, E_t^i is a vector with elements $e_{t,j}^i = \mathbb{1}[argmax(f_t(a_j)) = y_{a_j}]$ indicates if the annotation model f_t predicted correctly on each annotated example $a_j \in \rho_i$ in the k nearest neighbors of x_t^i. The local density D_t^i is a distance vector, with each element being $d(x_t^i, a_j)$. These two vectors are combined using the element-wise multiplication operator $\odot$ to create the input: $\varkappa_t^i = D_t^i \odot E_t^i - D_t^i \odot (1 - E_t^i)$. Notably, $\varkappa_t^i$ measures the error regularity of x_t^i among its neighborhood, as the more positive values in the vector $\varkappa_t^i$, the less likely f_t would make an error on x_t^i.
- Learning Objectives. We collect the ground truth labels y_t through human feedback. To optimize the error-aware integration strategy λ_t, we use a mean squared error (MSE) loss, denoted as $\ell_d^t(y_t, f_t(x_t), \lambda_t) = \text{MSE}(\mathbb{1}[argmax(f_t(x_t)) = y_t], \lambda_t(\varkappa_t))$, where $\mathbb{1}[\cdot]$ indicates whether the ground truth labels y_t are the same as the predictions $f_t(x_t)$. The purpose of this loss function is to guide λ_t by encouraging it to predict errors made by f_t.

3.3 Optimization of ARAIDA

To simplify the optimization process, we independently optimize the annotation model f_t, KNN model g_t, and error-aware integration strategy λ_t. We treat human feedback y_t as the ground truth following previous studies on interactive annotation (Klie et al., 2018, 2020; Le et al., 2021). It is worth noting that ARAIDA supports any task-specific annotation model and uses its corresponding loss function ℓ_f to update the parameters. Combining the ℓ_d loss and the negative log-likelihood loss ℓ_g to optimize KNN, we formulate the final loss function as follows:

$$\mathcal{L}(f, g, \lambda) = \sum_{i=1}^{B_t} \ell_f(y_i, f_t(x_i)) + \ell_g(y_i, g_t(x_i)) + \ell_d(y_i, f_t(x_i), \lambda_t), \tag{2}$$

where B_t represents the total data accumulated until round t. There are two challenges to optimizing this objective function. Firstly, the operator used in KNN to retrieve the k nearest neighbors is not differentiable. To address this problem, we utilize the Gumbel-softmax-based reparameterization trick (Jang et al., 2016) to facilitate the optimization process. Secondly, the loss function $\mathcal{L}$ presents a bi-level optimization problem, where the optimization of λ_t is nested within the optimization problems of f_t and g_t. As a result, we update f_t, g_t, and λ_t iteratively using a coordinate-descent approach. Formally, at each optimization iteration k, we have network parameters θ_f^k, θ_g^k, and θ_λ^k corresponding to f^k, g^k, and λ^k. The update procedures are as follows:

$$\begin{aligned}\theta_f^{k+1} &= \theta_f^k - \bigtriangledown_f \mathcal{L}(f, g^k, \lambda^k),\\ \theta_g^{k+1} &= \theta_g^k - \bigtriangledown_g \mathcal{L}(f^k, g, \lambda^k),\\ \theta_\lambda^{k+1} &= \theta_\lambda^k - \bigtriangledown_\lambda \mathcal{L}(f^{k+1}, g^{k+1}, \lambda).\end{aligned} \tag{3}$$

4 Experiments

We conduct extensive experiments to assess ARAIDA's effectiveness in the limited data annotation scenario. Our primary focus is to assess whether ARAIDA can decrease the human effort required for corrections by providing more precise annotations at various stages of the annotation process (see Section 4.2). Furthermore, we perform a comprehensive examination to investigate the behavior and impact of KNN and the error-aware integration strategy (see Section 4.3). Additional analysis of our error-aware integration strategy is presented in Section 4.4. Lastly, we analyze the sensitivity of the parameters in Appendices A.

4.1 Experimental Setup

Tasks & Datasets. We experiment with word- and sentence-level annotation tasks. These tasks have been highlighted as crucial in various web applications (Yao et al., 2021; Marcos-Pablos and García-Peñalvo, 2020; Lee et al., 2022). To simulate the scenario of limited data annotation, we follow Dou et al. (2019) by imposing dataset size restrictions, ranging from $1K$ to $5K$. Table 1 overviews the dataset statistics.

- **Word-level annotation**. We focus on the knowledge graph completion task, which annotates the semantic classes of input word pairs (e.g., '[*tree*, *leaf*]=>*component*'). We use two benchmark knowledge graph datasets in our experiments, namely the WN18RR (Dettmers et al., 2018)[5] and Freebase (Bollacker et al., 2008)[6] dataset. We experiment with the eight most frequent classes for each dataset.

- **Sentence-level annotation**. We consider the sentiment classification task and experiment on two benchmark datasets, including SST-5 (Socher et al., 2013)[7], and IMDB (Maas et al., 2011)[8]. SST-5 dataset contains categories on a scale of 1-5 while IMDB contains two categories (positive/negative).

Dataset	# Val.	Classes
WN18RR	3,034	Hypernym; Derivation; Member; Component; Synset; Synonym; Verb group; Instance of hypernym;
FreeBase	5,116	Contains; Country; Track_role; Profession; Group_role; Adjoins; Film_release; Nutrient
IMDB	5,000	Positive; Negative
SST-5	1,101	Strong positive; Positive; Neutral; Negative; Strong negative

Table 1: Statistics of datasets. For each dataset, we randomly sample 5K examples from the original training dataset to form the unlabeled data, and the validation dataset is taken from the original dataset. Table 4 presents the mapping from the original categories to the categories we use.

Evaluation Metric. We aim to minimize the total human corrections (i.e., the total model suggestion errors) annotating a given amount of data using the interactive annotation process. Therefore, we report the *Machine Cumulative Accuracy* (MCA), defined as the total correct suggestions divided by the total suggestions for different dataset sizes. To assess the performance of each method in the limited data annotation scenario, we present the mean and the corresponding standard deviation by varying the dataset size ({1K, 2K, 3K, 4K, 5K}).

Annotation Models. To verify the generalizability of ARAIDA, we apply it in conjunction with different annotation models:

- Classic annotation models. We utilize lightweight annotation models following

[5]https://paperswithcode.com/dataset/wn18rr
[6]https://www.microsoft.com/en-us/download/details.aspx?id=52312
[7]https://nlp.stanford.edu/sentiment/code.html
[8]https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Annotation Model	Without Active Learning (AL)				With Active Learning (AL)			
	Word-level Annotation		Sentence-level Annotation		Word-level Annotation		Sentence-level Annotation	
	WN18RR	FreeBase	IMDB	SST-5	WN18RR	FreeBase	IMDB	SST-5
Dist./FT	50.44±1.02	32.47±12.37	70.18±8.43	36.02±3.14	47.77±0.91	26.92±10.85	66.98±6.31	35.20±3.76
Dist./FT + ARAIDA	**52.16±1.37**	**43.02±6.43**	**79.33±2.81**	**37.21±3.03**	**49.54±0.84**	**39.06±4.57**	**75.84±1.14**	**37.02±2.50**
LLaMa2	31.35±1.89	24.41±1.77	80.21±2.64	37.83±1.64	31.35±1.89	24.41±1.77	80.21±2.64	37.83±1.64
LLaMa2 + ARAIDA	**45.15±1.96**	**38.20±1.91**	**88.47±2.03**	**42.03±1.88**	**46.33±2.01**	**38.79±1.96**	**89.68±2.25**	**42.61±1.91**
LLaMa2$_{sft}$	58.24±2.79	53.11±1.38	94.06±9.02	46.84±6.30	59.28±2.57	55.39±2.01	95.13±10.15	47.45±6.17
LLaMa2$_{sft}$ + ARAIDA	**60.74±2.33**	**55.23±1.46**	**95.15±9.32**	**49.62±5.98**	**61.02±2.18**	**56.71±2.09**	**95.88±12.27**	**49.51±6.03**

Table 2: Machine cumulative accuracy (MCA) scores using various methods. We report the averaged results across varying amounts of data. LLaMa2 with AL has identical performance as LLaMa2 because the vanilla LLaMa2 model is not updated during the interactive annotation process. Thus, the data order does not impact the performance. When combined with ARAIDA, the performances w/ and w/o AL are different because the KNN and integration strategy models are updated.

previous works (Desmond et al., 2021; Chen et al., 2020; Hedderich et al., 2021). Specifically, for word-level tasks, we use a distributional model (Roller et al., 2014; Kober et al., 2021) with pretrained GloVe word embeddings embedding (Pennington et al., 2014)[9]. For sentence-level tasks, we use FastText (Joulin et al., 2017) to derive the sentence embeddings. We denote this baseline as *Dist./FT*.

- LLM-based annotators. We also use large language models (LLMs) as annotation models, whose few-shot and in-context learning capabilities might help with the limited data annotation process. We experiment with LLaMa2-7B (Touvron et al., 2023) and ChatGPT (Ouyang et al., 2022) [10]. We use zero-shot and few-shot prompts for ChatGPT (denoted as $ChatGPT_{zero}$ and $ChatGPT_{few}$). Detailed prompts can be found in Table 7. For LLaMa2, we consider both the vanilla *LLaMa2* that uses the same zero-shot prompts as ChatGPT$_{zero}$ and $LLaMa2_{sft}$, which is fine-tuned using an open-source toolkit[11] during the interactive annotation process. Fine-tuning data examples can be found in Table 8.

Impact of Active Learning. Regardless of the annotation model, the sequence of the data to annotate also affects the amount of human corrections. Intuitively, if we show unambiguous examples first, few corrections are needed. However, the annotation model and KNN may not learn to handle more challenging examples and may subsequently make more mistakes. Previous studies focused on applying active learning in interactive annotation to enhance the annotation models' sample efficiency (Laws et al., 2011; Klie et al., 2018; Li et al., 2021). To study the impact of active learning in the limited data annotation scenario, we compare an uncertainty-based active learning method with random data ordering for different annotation models with and without ARAIDA. Note that we omit the ChatGPT with AL results because we are unable to estimate its prediction uncertainty accurately.

Implementation Details. All experiments are carried out on a machine with Intel(R) Xeon(R) Gold 5317 CPU @ 3.00GHz and a GeForce RTX 3090 GPU. For simplicity, we implement our integration strategy using a three-layer, fully-connected network with ReLu activation and dropout. For KNN, we set $k = 20$ for ρ_t. KNN runs in the embedding space of *text-embedding-ada-002* (Neelakantan et al., 2022) when combining with ChatGPT. For Dist./FT and LLaMa2 models, KNN runs in the corresponding model's embedding space. Moreover, we leave the details of LLM prompts and mode fine-tuning examples in Appendix D.

4.2 Main Result

We evaluate the effectiveness of ARAIDA in enhancing the model annotation quality, hence reducing human correction effort. Table 2 and Figure 3 show the machine cumulative accuracy scores averaged across varying amounts of data ($\{1K, 2K, 3K, 4K, 5K\}$) for different experimental settings. We make the following observations:

ARAIDA reduces human corrections consistently. As illustrated in Table 2 and Figure 3, ARAIDA consistently improves the model suggestion accuracy across different annotation models and annotation tasks. Specifically, it achieves a

[9]We did not use contextualized embeddings because the word-level task in our experiment has no context.

[10]The `gpt-3.5-turbo` checkpoint in OpenAI's API (`https://platform.openai.com/docs/models/gpt-3-5`).

[11]`https://github.com/Alpha-VLLM/LLaMA2-Accessory`

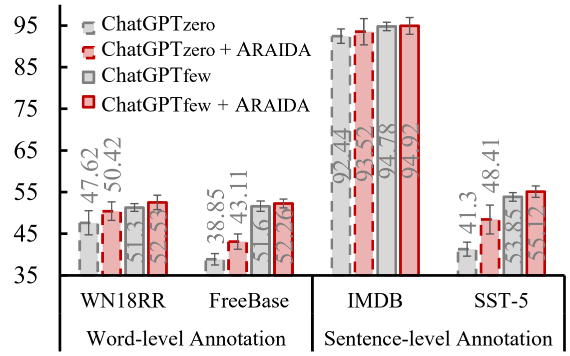

Figure 3: MCA scores using ChatGPT-based methods. We omit the ChatGPT with AL results because we are unable to estimate its prediction uncertainty. ARAIDA can further improve ChatGPT's performance

13.06% performance gain in MCA on Dist./FT, 3.85% on LLaMa2$_{sft}$, 16.80% on Dist./FT with AL, 2.61% on LLaMa2$_{sft}$ with AL, 30.48% on LLaMa2, 8.81% on ChatGPT$_{zero}$, and 1.55% on ChatGPT$_{few}$. On average, it achieves an 11.02% performance gain in model annotation accuracy, translating into a reduction in human correction effort in practice.

Active learning does not always help. When comparing the annotation models with and without active learning, we observe that active learning does not always improve model performance and can even harm the performance in some instances (e.g., Dist./FT). We hypothesize that the more challenging cases selected by active learning might require more training data for the models to correct their predictions (Dasgupta, 2005; Rietz and Maedche, 2021), which are unavailable in limited data annotation scenarios. Instead of relying on an annotation model alone, ARAIDA acts as a posthoc "plug-in" that fixes the annotation model's mistakes using retrieved annotation references and yields a robust improvement under various settings with and without active learning.

LLMs are strong annotation models. ARAIDA can improve them further. LLMs perform better than classic distributional models, especially for sentence-level tasks. Furthermore, LLaMa2 with fine-tuning consistently outperforms the vanilla LLaMa2 model, and few-shot ChatGPT consistently beats its zero-shot counterpart. Interestingly, comparing these two pairs of annotation models, we observe that ARAIDA brings a more substantial improvement to weaker models. When the annotation models are already strong (in the case of LLaMa2$_{sft}$ and ChatGPT$_{few}$), ARAIDA is more conservative in making corrections, yielding a smaller but consistent improvement. It demonstrates ARAIDA's robustness to combine with annotations models with different performances.

Annotation Model	Word-level Annotation		Sentence-level Annotation	
	WN18RR	FreeBase	IMDB	SST-5
	Dist./FT			
ARAIDA	**52.16±1.37**	**43.02±6.43**	**79.33±2.81**	**37.21±3.03**
- *w/o* KNN	50.44±1.02	32.47±12.37	70.18±8.43	36.02±3.14
- *w/o* $f(\cdot)$	50.28±3.01	41.52±5.87	78.48±0.68	35.02±1.37
- *w/* const.	51.85±4.77	41.78±6.78	78.58±3.94	37.07±2.68
	LLaMa2$_{sft}$			
ARAIDA	**60.74±2.33**	**55.23±1.46**	**95.15±9.32**	**49.62±5.98**
- *w/o* KNN	58.24±2.79	53.11±1.38	94.06±9.02	46.84±6.30
- *w/o* $f(\cdot)$	45.23±2.58	39.82±3.41	89.13±0.55	42.93±1.71
- *w/* const.	58.24±2.79	53.11±1.38	94.06±9.02	46.84±6.30
	ChatGPT$_{zero}$			
ARAIDA	**50.42±1.37**	**43.11±1.83**	**93.52±1.24**	**48.41±1.06**
- *w/o* KNN	47.62±2.89	38.85±2.23	92.44±0.95	41.30±1.72
- *w/o* $f(\cdot)$	48.73±2.64	41.30±5.04	90.61±0.26	45.84±1.12
- *w/* const.	48.73±2.64	41.30±5.04	92.44±0.95	45.84±1.12
	ChatGPT$_{few}$			
ARAIDA	**52.53±1.67**	**52.26±3.44**	**94.92±1.02**	**55.12±1.36**
- *w/o* KNN	51.30±1.72	51.60±3.15	94.78±0.99	53.85±2.03
- *w/o* $f(\cdot)$	48.73±2.64	41.30±5.04	90.61±0.26	45.84±1.12
- *w/* const.	51.30±1.72	51.60±3.15	94.78±0.99	53.85±2.03

Table 3: Ablation study on the KNN and the error-aware integration strategy modules. We report the MCA scores using various methods, averaging results with different amounts of data. Error-aware integration strategy effectively coordinates the two annotators.

4.3 Ablation Study

This section aims to perform a comprehensive examination to investigate the behavior and impact of the KNN and out integration strategy. We conduct an ablation study to analyze the effectiveness of each component. In particular, we consider the following baselines:

- **ARAIDA w/o KNN:** Using the annotation model alone to suggest labels. Equivalent to $\lambda_t(\cdot) = 1$.
- **ARAIDA w/o f($\cdot$):** Using KNN alone to suggest labels. Equivalent to $\lambda_t(\cdot) = 0$.
- **ARAIDA w/ const.:** Using a constant λ_t^* value for all examples, in contrast to ARAIDA, which varies λ_t for different examples. In our experiments, we report the result with the best λ_t^* tuned on the validation set.

The ablation test results are presented in Table 3. Due to limited space, we omit the result for vanilla LLaMa2, which is much weaker than other LLM-based baselines. The detailed observations are provided below.

KNN is a strong stand-alone annotator. Table 3 reveals that although KNN (ARAIDA *w/o f($\cdot$)*)

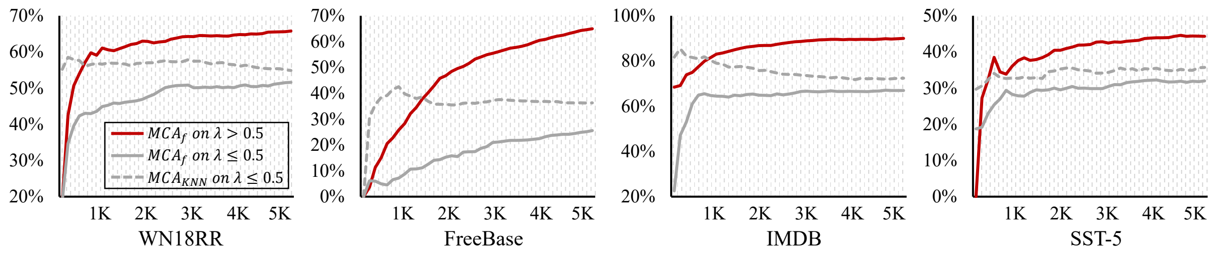

Figure 4: Analyzing our integration strategy with the Dist./FT model. The solid lines show the MAC scores of the annotation model $f(\cdot)$, separated by examples with $\lambda > 0.5$ (higher weights assigned to the annotation model $f(\cdot)$) and $\lambda \leq 0.5$ (higher weights assigned to KNN). The dotted line shows KNN's performance on the latter set.

does not match the performance of ARAIDA, it obtains comparable accuracy as using the annotation model alone (ARAIDA *w/o KNN*). This result is surprising, given KNN's simplicity compared to the annotation models (including LLMs). Such the results also highlight that significance and effectiveness of analogical reasoning, allowing humans to reason more effectively (Mitchell, 2021).

Error-aware integration strategy effectively coordinates the two annotators. Table 3 shows that ARAIDA's error-aware integration strategy achieves consistent performance gain compared to using a constant weight λ_t^*. This in turn also confirms the original intention behind our construction of the error-aware integration strategy: it is infeasible to find the optimal weight through a one-off hyperparameter tuning. Here, it is worth noting that ChatGPT outputs discrete labels rather than probabilistic vectors. In this case, the constant strategy reduces into an indicator function: if $\lambda_t^* > 0.5$, $F_t(x) = f_t(x), \forall x_t$; otherwise, $F_t(x) = g_t(x), \forall x_t$. In this case, ARAIDA *w/ const.* prefers ChatGPT or KNN based on their performance. However, it cannot improve the annotation quality beyond the individual components (i.e., the KNN and the annotation model) due to the discrete output of ChatGPT.

4.4 Qualitative Analysis

To shed light on how the error-aware integration strategy works, we measure the cumulative accuracy of the annotator model and KNN throughout the annotation process for each dataset and present the result with the Dist./FT annotation model in Figure 4. We also separate the cases where $\lambda > 0.5$ (higher weights assigned to the annotation model $f(\cdot)$) and $\lambda \leq 0.5$ (higher weights assigned to KNN). Our observations are as follows.

Firstly, we observe that there is a substantial gap between the two solid lines (MCA_f on $\lambda > 0.5$ and MCA_f on $\lambda \leq 0.5$), showing that our error estimation model effectively identifies cases where $f(\cdot)$ is likely to error and assigns it a lower weight. Secondly, KNN reaches a reasonable accuracy much faster than the annotation model at the initial stage of the interactive annotation process. Even as the number of annotated examples increases, its accuracy is still higher than the annotation model $f(\cdot)$ when $\lambda \leq 0.5$. This result reveals that KNN compensates for the performance deficiencies of $f(\cdot)$ on data where it is more likely to make a mistake. Therefore, combining KNN using the error-aware integration strategy in ARAIDA leads to an overall improvement in annotation quality, hence reducing human correction effort.

5 Conclusion

In interactive data annotation, an annotation model suggests labels to human annotators to verify. However, the annotation model is prone to errors when trained on limited labeled data. To tackle this challenge, we proposed ARAIDA, an approach inspired by analogical reasoning, to compensate for the performance deficiencies of the annotation model and correct its mistakes using an error-aware integration strategy. Extensive experiments demonstrated that ARAIDA is flexible to combine with different annotation models across various tasks and yields consistent improvement in label suggestion accuracy, which leads to a reduction of human correction effort. In this study, our method explores a new solution to bring more flexibility by allowing the human to design any preferred annotation model according to different annotation tasks. We are devoted to optimizing human-machine utilities by emphasizing the learning of task-specified concepts efficiently from a few human demonstrations. In future work, we plan to extend ARAIDA to other annotation tasks and develop it as a general toolkit that can benefit the NLP community.

6 Limitations

Human Studies. This work aims to reduce human correction effort in interactive data annotation. We follow previous work (Hwa, 2000; Kristjansson et al., 2004) to use the number of model suggestion errors to approximate the human correction effort needed. However, the actual effort needed depends on the particular example and the type of errors (e.g., whether it is obvious). Ideally, we would involve human annotators and measure the saving of annotation time. However, due to the large number of experimental settings, conducting human studies with each annotation model and ablation baseline was infeasible.

Error-Prone Human Annotation. This paper treats human annotations as ground truth following previous studies in interactive data annotation (Klie et al., 2018; Le et al., 2021). However, uncertainty and inconsistency of human annotations do occur. We refer readers to the literature on handling error-prone human annotation, such as crowd-sourced data annotation (Larson et al., 2020).

Although human annotation errors are not the focus of this work, we explore ARAIDA's performance under synthesized label noise conditions. We consider the crowd-sourced data annotation scenario and assume that each human annotator h_i makes mistakes with the latent probability $p_e^i \sim (0, 0.3)$. We set the total number of annotators $O = 10$ and sample their corresponding error probabilities $P_e = \{p_e^1, p_e^2, ..., p_e^O\}$. Then, we sample u_i from a uniform distribution $U(0, 1)$ for each annotation. If $u_i \leq p_e^i$, h_i assigns a randomly sampled incorrect label; otherwise, it assigns the correct one. We use majority voting of the 10 annotators to obtain the final annotations following Shirani et al. (2019).

We slightly modified ARAIDA's datastore maintenance strategy. When the datastore exceeds its budge, we discard the data from the majority class and most **dis-similar** to its class prototype instead of removing the most similar one (as in the original ARAIDA strategy). This strategy may help remove incorrectly labeled data. The experimental result in Figure 5 shows that ARAIDA still outperforms the baseline. The modified datastore maintenance strategy (ARAIDA-dis) further improves the performance by a slight margin. Further research and more rigorous experiments are required to address the human annotation noise problem in interactive annotation.

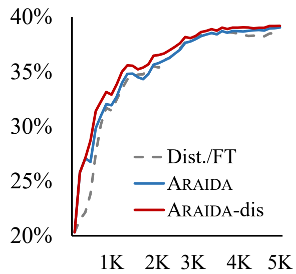

Figure 5: MAC scores of various methods with synthesized label noise on the SST-5 dataset. Dist./FT is used as the annotation model. ARAIDA-dis refers to ARAIDA with a modified datastore maintenance strategy.

Latency. The KNN component requires retrieving similar examples as the input data, which may limit our method's time efficiency when the datastore size is large. To address this problem, besides the proposed datastore management strategy, we can also employ an efficient similarity search library such as FAISS [12] to speed up the retrieval.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 62272330); in part by the Fundamental Research Funds for the Central Universities (No. YJ202219).

References

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46, Toronto, Canada. Association for Computational Linguistics.

Miguel A Bautista, Artsiom Sanakoyeu, Ekaterina Tikhoncheva, and Bjorn Ommer. 2016. Cliquecnn: Deep unsupervised exemplar learning. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Arantxa Casanova, Pedro O Pinheiro, Negar Rostamzadeh, and Christopher J Pal. 2020. Reinforced active learning for image segmentation. *arxiv*.

Aditi Chaudhary, Antonios Anastasopoulos, Zaid Sheikh, and Graham Neubig. 2021. Reducing confusion in active learning for part-of-speech tagging. *TACL*, 9:1–16.

[12]https://github.com/facebookresearch/faiss

Xu Chen, Changying Du, Xiuqiang He, and Jun Wang. 2020. Jit2r: A joint framework for item tagging and tag-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1681–1684.

Sanjoy Dasgupta. 2005. Coarse sample complexity bounds for active learning. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pages 235–242.

Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer. 2005. The forgetron: A kernel-based perceptron on a fixed budget. *NeurIPS*, 18.

Michael Desmond, Evelyn Duesterwald, Kristina Brimijoin, Michelle Brachman, and Qian Pan. 2021. Semi-automated data labeling. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 156–169. PMLR.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *EMNLP*, pages 1192–1197.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd-workers for text-annotation tasks. *arXiv preprint arXiv:2303.15056*.

Xingwei He, Zhenghao Lin, Yeyun Gong, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, Weizhu Chen, et al. 2023. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*.

Michael A Hedderich, Lukas Lange, and Dietrich Klakow. 2021. Anea: distant supervision for low-resource named entity recognition. *the Practical Machine Learning For Developing Countries Workshop at ICLR*.

Chen Huang, Yang Deng, Wenqiang Lei, Jiancheng Lv, and Ido Dagan. 2024. Selective annotation via data allocation: These data should be triaged to experts for annotation rather than the model.

Chen Huang, Peixin Qin, Wenqiang Lei, and Jiancheng Lv. 2023. Reduce human labor on evaluating conversational information retrieval system: A human-machine collaboration approach. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10876–10891, Singapore. Association for Computational Linguistics.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arxiv*.

Qingnan Jiang, Mingxuan Wang, Jun Cao, Shanbo Cheng, Shujian Huang, and Lei Li. 2021. Learning kernel-smoothed machine translation with retrieved examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7280–7290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *EACL*, pages 427–431. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. BERT-kNN: Adding a kNN search component to pretrained language models for better QA. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3424–3430, Online. Association for Computational Linguistics.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. *International Conference on Learning Representations*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. In *ICLR*.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The INCEpTION platform: Machine-assisted and knowledge-oriented interactive annotation. In *COLING: System Demonstrations*, pages 5–9, Santa Fe, New Mexico. ACL.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains. In *ACL*, pages 6982–6993, Online. ACL.

Thomas Kober, Julie Weeds, Lorenzo Bertolini, and David J. Weir. 2021. Data augmentation for hypernymy detection. In *EACL*, pages 1034–1048.

Trausti Kristjansson, Aron Culotta, Paul Viola, and Andrew McCallum. 2004. Interactive information extraction with constrained conditional random fields. In *AAAI*, volume 4, pages 412–418.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Stefan Larson, Adrian Cheung, Anish Mahendran, Kevin Leach, and Jonathan K Kummerfeld. 2020. Inconsistencies in crowdsourced slot-filling annotations: A typology and identification methods. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5035–5046.

Florian Laws, Christian Scheible, and Hinrich Schütze. 2011. Active learning with Amazon Mechanical Turk. In *EMNLP*, pages 1546–1556, Edinburgh, Scotland, UK. ACL.

Trung-Nghia Le, Tam V Nguyen, Quoc-Cuong Tran, Lam Nguyen, Trung-Hieu Hoang, Minh-Quan Le, and Minh-Triet Tran. 2021. Interactive video object mask annotation. In *AAAI*, volume 35, pages 16067–16070.

Yoonjoo Lee, John Joon Young Chung, Tae Soo Kim, Jean Y Song, and Juho Kim. 2022. Promptiverse: Scalable generation of scaffolding prompts through human-ai hybrid knowledge graph annotation. In *CHI Conference on Human Factors in Computing Systems*, pages 1–18.

Yanzeng Li, Bowen Yu, Li Quangang, and Tingwen Liu. 2021. Fitannotator: A flexible and intelligent text annotation system. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*, pages 35–41.

Shudong Liu, Xuebo Liu, Derek F Wong, Zhaocong Li, Wenxiang Jiao, Lidia S Chao, and Min Zhang. 2023. knn-tl: k-nearest-neighbor transfer learning for low-resource neural machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1878–1891.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Samuel Marcos-Pablos and Francisco J García-Peñalvo. 2020. Information retrieval methodology for aiding scientific database search. *Soft Computing*, 24(8):5551–5560.

Melanie Mitchell. 2021. Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 1505(1):79–101.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Tim Rietz and Alexander Maedche. 2021. Cody: An ai-based system to semi-automate coding for qualitative research. In *CHI*, pages 1–14.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop*, pages 101–108.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING: Technical Papers*, pages 1025–1036.

Amirreza Shirani, Franck Dernoncourt, Paul Asente, Nedim Lipka, Seokhwan Kim, Jose Echevarria, and Thamar Solorio. 2019. Learning emphasis selection for written text in visual media from crowd-sourced label distributions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1167–1172.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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*.

Ding Wang, Shantanu Prabhat, and Nithya Sambasivan. 2022a. Whose ai dream? in search of the aspiration in data annotation. In *CHI*, pages 1–16.

Dongqi Wang, Haoran Wei, Zhirui Zhang, Shujian Huang, Jun Xie, Weihua Luo, and Jiajun Chen. 2021. Non-parametric online learning from human feedback for neural machine translation. *arXiv*.

Shuhe Wang, Xiaoya Li, Yuxian Meng, Tianwei Zhang, Rongbin Ouyang, Jiwei Li, and Guoyin Wang. 2022b. k nn-ner: Named entity recognition with nearest neighbor search. *arXiv preprint arXiv:2203.17103*.

Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. 2019. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv*.

Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381.

Ruixuan Xiao, Yiwen Dong, Junbo Zhao, Runze Wu, Minmin Lin, Gang Chen, and Haobo Wang. 2023. FreeAL: Towards human-free active learning in the era of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14520–14535, Singapore. Association for Computational Linguistics.

Kaichun Yao, Chuan Qin, Hengshu Zhu, Chao Ma, Jingshuai Zhang, Yi Du, and Hui Xiong. 2021. An interactive neural network approach to keyphrase extraction in talent recruitment. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2383–2393.

Xin Zheng, Zhirui Zhang, Junliang Guo, Shujian Huang, Boxing Chen, Weihua Luo, and Jiajun Chen. 2021. Adaptive nearest neighbor machine translation. pages 368–374.

A Hyperparameter Analysis

In this section, we provide a detailed hyperparameter analysis of ARAIDA, including the datastore size A_t, the number of neighbors k for ρ_t, and the datastore maintenance strategy. We show results only for the Dist./FT annotation model for brevity.

A.1 Datastore Size

We tune the size of the datastore A_t in $\{100, 500, 1000, 2000\}$ and evaluate the machine cumulative accuracy of ARAIDA with or without active learning. We illustrate the results in Fig.6. A larger datastore size generally brings higher annotation performance since it allows us to maintain more data from past human-machine interactions. However, it also requires larger memory usage and causes longer latency. We found that a datastore size of 1000 is a reasonable tradeoff, which we utilize in our main experiments.

A.2 Top k for ρ_t

We tune the number of neighbors A_t in $\{5, 10, 20, 50\}$ and evaluate the machine cumulative accuracy of ARAIDA with or without active learning. As shown in Fig.7, $k = 20$ seems to perform well for all tasks.

A.3 Datastore Maintenance Strategy

We propose a class-aware datastore maintenance strategy for ARAIDA, which removes labeled examples from the majority class most similar to their class prototype. Compared to the conventional First-In-First-Out (FIFO) strategy (Dekel et al., 2005), our method ensures that 1) the datastore contains as much data from different classes as possible, and 2) the class prototype is least affected after the removal. We compare with two variants of ARAIDA, including ARAIDA *w/ FIFO* and ARAIDA *w/ class FIFO*. The former discards the oldest example regardless of the class; the latter discards the oldest example belonging to the majority class in the datastore.

As shown in Fig.8, ARAIDA *w/ FIFO* can suffer from a sudden decrease in performance, as it may remove important examples arbitrarily. After integrating the class information, ARAIDA *w/ class FIFO* removes the oldest analogy from the majority class, achieving a comparative performance to ARAIDA. However, ARAIDA still performs better when the number of classes increases (e.g., FreeBase).

B Results on Comparing to Fully Fine-tuned Model

We utilized up to 5K of data to simulate the limited data annotation task, and we updated the parameters of the annotation model as the interactive annotation process progressed. To validate the effectiveness of the fully fine-tuned model, we use the remaining data from the original dataset, excluding the 5K data and the validation data, as the training data to fully fine-tune the model. Next, we employ this model for interactive annotation. Since it is already fully fine-tuned, we do not update the model parameters during the annotation process. Due to time constraints, we are currently only considering scenarios where active learning has not been adopted. Additionally, we only fine-tune small annotation models (i.e., BERT and Dist./FT).

Based on the results in the Table 5, it is evident that the annotation performance of the fully fine-tuned model surpasses that of our method. This suggests that while our method currently offers a lightweight approach to aid data annotation, which can to some extent enhance the sample efficiency of the annotation model, there is still potential for improvement in future research.Nevertheless, it's important to emphasize that in real-world data an-

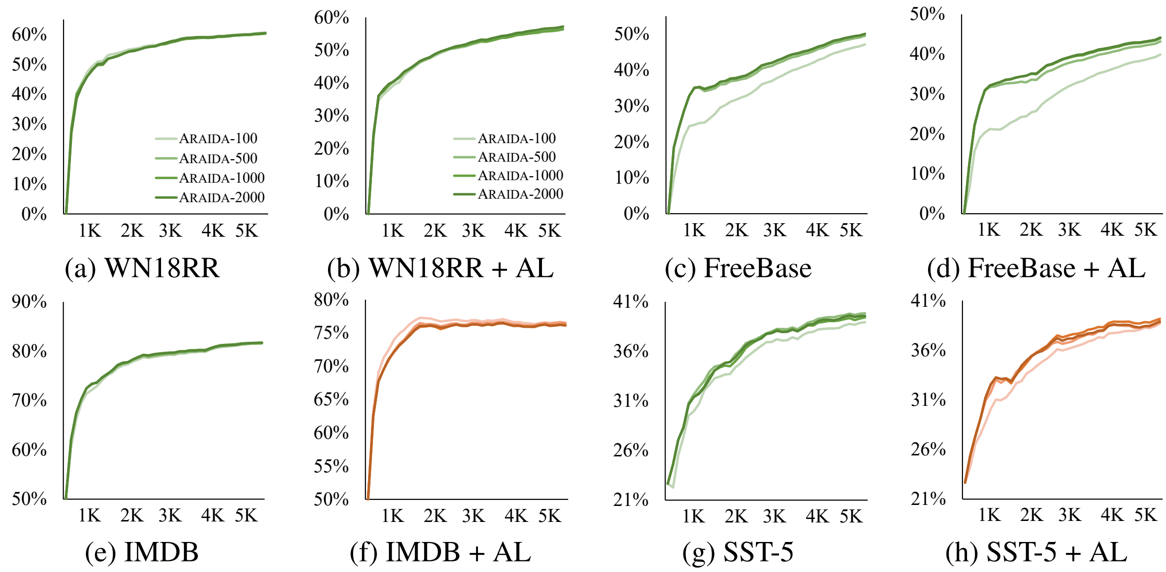

Figure 6: Machine Cumulative Accuracy of ARAIDA with different datastore sizes.

WN18RR		FreeBase	
Raw Class	**Mapped Class**	**Raw Class**	**Mapped Class**
_hypernym	hypernym	/location/location/contains	contains
_derivationally_related_form	derivation	/olympics/olympic_sport/athletes./olympics/olympic_athlete_affiliation/country	country
_member_meronym	member	/music/performance_role/track_performances./music/track_contribution/role	track_role
_has_part	component	/people/person/profession	profession
_synset_domain_topic_of	synset	/music/performance_role/regular_performances./music/group_membership/role	group_role
_instance_hypernym	instance of hypernym	/location/location/adjoin_s./location/adjoining_relationship/adjoins	adjoins
_also_see	synonym	/film/film/release_date_s./film/film_regional_release_date/film_release_region	film_release
_verb_group	verb group	/food/food/nutrients./food/nutrition_fact/nutrient	nutrient

Table 4: Class mapping details for WN18RR and FreeBase

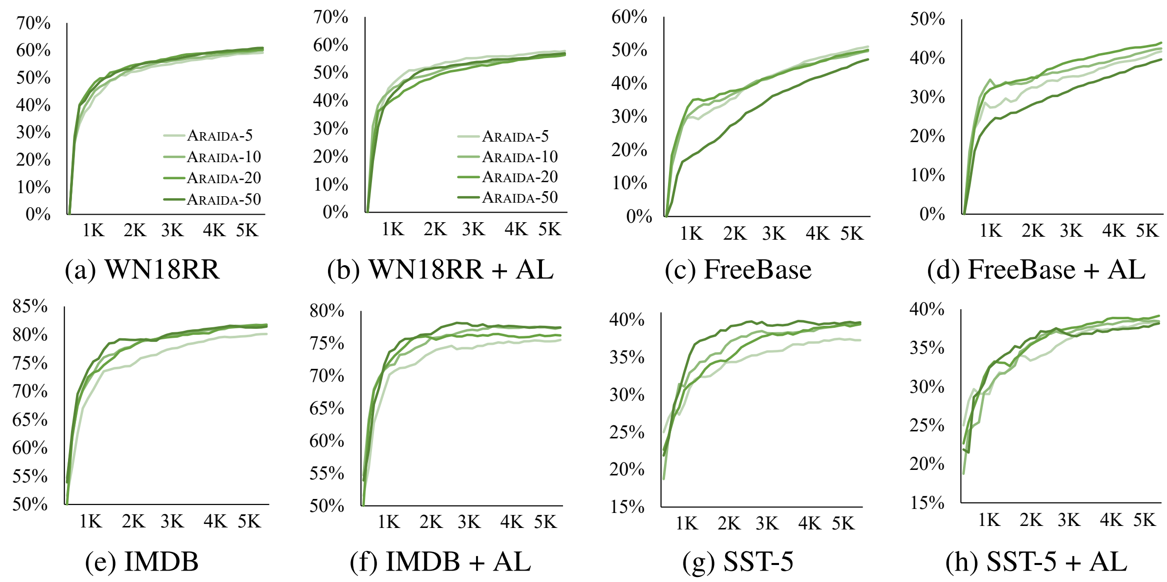

Figure 7: Machine Cumulative Accuracy of ARAIDA with different k for ρ_t.

notation scenarios, we typically don't have access to fully fine-tuned models initially, as we lack labeled datasets. While one might turn to LLMs like ChatGPT, our findings in Figure 3 indicate that our method could potentially improve upon ChatGPT even further.

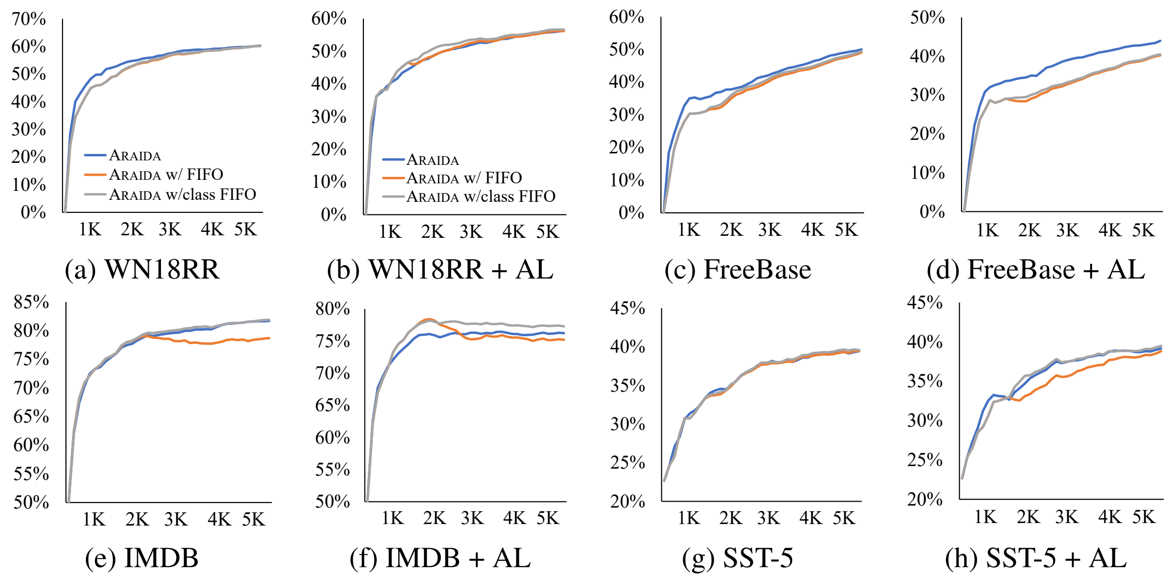

Figure 8: Machine Cumulative Accuracy of ARAIDA with different datastore maintenance strategies.

Table 5: Results on comparing to fully fine-tuned models

Model	WN18RR	FreeBase	IMDB	SST-5
BERT(fully finetuned)	75.13±1.12	60.47±1.24	90.04±0.42	46.59±1.20
BERT+ARAIDA	54.41±1.51	50.30±2.25	88.79±3.54	43.81±3.12
Dist./FT(fully finetuned)	72.64±1.05	58.50±1.36	84.87±0.81	42.12±1.44
Dist./FT+ARAIDA	52.16±1.37	43.02±6.43	79.33±2.81	37.21±3.03

C Results on Smaller-sized Transformer-based Model

With a wide array of language models to choose from, we faced the challenge of not being able to test all available models. To address this, we selected three prominent models (GloVe, LLaMa2, and ChatGPT) based on our hardware resource capabilities in our main experiments. While open to conducting further experiments, our focus was limited by time constraints, leading us to concentrate solely on fine-tuning BERT (w/o AL), with the outcomes detailed in Table 6. After implementing ARAIDA, we observed a significant enhancement in the quality of annotations by using our ARAIDA.

Table 6: Results on smaller-sized Transformer-based model

Model	WN18RR	FreeBase	IMDB	SST-5
BERT	53.32±1.02	46.12±3.23	85.38±7.16	41.08±3.41
BERT+ARAIDA	**54.41±1.51**	**50.30±2.25**	**88.79±3.54**	**43.81±3.12**
BERT+ARAIDA w/o f	51.03±2.11	44.63±4.04	80.29±1.09	38.51±1.45

D Prompts and Fine-Tuning Examples for LLM-Based Annotation Models

Table 7 presents prompts used for zero-shot and few-shot annotation. Table 8 shows fine-tuning examples for LLaMa2$_{sft}$.

Dataset	Prompts
IMDB	================== input: If you like original gut wrenching laughter...it. Great Camp!!!. output: positive input: I saw this movie when I was about 12 when it ... There are no rules. output: negative ================== You need to identify the sentiments of the following sentences, output positive or negative. {INPUTS}
SST-5	===================== input: The gorgeously elaborate continuation ...Tolkien 's Middle-earth. output: Strong Positive input: Singer/composer Bryan Adams contributes a slew of..., spirit of the piece. output: Positive input: You'd think by now ...with hearts of gold. output: Neutral input: This isn't a new idea . output: Negative input: A sour little movie at ... What was it all for ? output: Strong Negative ==================== Identify the sentiment of each paragraph, you have five options: 'Strong Positive', 'Positive', 'Neutral', 'Negative' or 'Strong Negative' {INPUTS}
WN18RR	============= input: ability unfitness output: antonym input: dissent debating output: entailment ... input: abandonment apostasy output: hypernym input: abandonment discard output: hyponym input: Afghanistan Afghan output: member input: abandonment abandonment output: synonym ============= Identify the semantic relation of the each word pair, you have eight options: 'component', 'synset',..., 'hypernym', 'derivation', 'member', 'synonym' You MUST only output the semantic relation word for each input! {INPUTS}
FreeBase	========== input: Libya Egypt output: adjoins input: Honolulu Punahou output: contains input: Bobsleigh Netherlands output: country input: Blackbriar Lithuania output: film_release input: Autoharp Guitar output: group_role input: IceCream Water output: nutrient input: Shriya Actor output: profession input: Cello Pennywhistle output: track_role ========= Identify the semantic relation of the each word pair, you have eight options: 'contains', 'country', 'track_role', 'profession', 'group_role', 'adjoins', 'film_release', 'nutrient'. only output the semantic relation. {INPUTS}

Table 7: Prompts for different datasets to obtain the annotation. We remove the few-shot demonstrations in the prompts in the zero-shot scenarios.

Dataset	**Finetuning data example for LLaMa2$_{sft}$**
IMDB	{"instruction": "Identify the sentiment of the following paragraph, output 'positive' or 'negative'.", "input": "The cast played Shakespeare.Shakespeare lost.I appreciate that this is trying to bring Shakespeare to the masses, but why ruin something so good. Is it because 'The Scottish Play' is my favorite Shakespeare? I do not know. What I do know is that a certain Rev Bowdler (hence bowdlerization) tried to do something similar in the Victorian era.In other words, you cannot improve perfection.I have no more to write but as I have to write at least ten lines of text (and English composition was never my forte I will just have to keep going and say that this movie, as the saying goes, just does not cut it.", "output": "negative"},
SST-5	{"instruction": "Identify the sentiment of the following paragraph, you have five options: 'Strong Positive', 'Positive', 'Neutral', 'Negative' or 'Strong Negative'.", "input": "The gorgeously elaborate continuation of The Lord of the Rings trilogy is so huge that a column of words can not adequately describe co-writer/director Peter Jackson 's expanded vision of J.R.R. Tolkien 's Middle-earth .", "output": "Strong Positive"},
WN18RR	{"instruction": "Identify the semantic relation of the following word pair, you have eight options: 'antonym',..., 'synonym'.", "input": "a.m. A.M.", "output": "synonym"},
FreeBase	{"instruction": "Identify the semantic relation of the following word pair, you have eight options: 'contains', ..., 'nutrient'.", "input": "Autoharp Guitar", "output": "group_role"},

Table 8: Finetuning data examples for LLaMa2$_{sft}$