Structure Discovery in Prompted Weak Supervision

Jinyan Su^{*,1} **Peilin Yu**^{*,†,2} **Jieyu Zhang**^{3} **Stephen H. Bach**^{2}
^{1}Cornell University ^{2}Brown University ^{3}University of Washington
† Correspondence to `peilin_yu@brown.edu`

Abstract

Prompted weak supervision (PromptedWS) applies pre-trained large language models (LLMs) as supervision sources in a weak supervision setup to efficiently distill information from LLMs and obtain labeled datasets at scale. We further extend the use of LLMs to address one of the key challenges in weak supervision: learning the dependency structure among noisy supervision sources. In this work, we highlight the challenge of structure discovery in PromptedWS. We propose a Structure Refining Module, a simple yet effective first approach based on the similarities of the prompts by taking advantage of the intrinsic structure in the embedding space. At the core of our method are Labeling Function Removal (LaRe) and Correlation Structure Generation (CosGen). Compared to previous methods that learn the dependencies from weak labels, our method finds the dependencies that are intrinsic in the embedding space. We show that Structure Refining Module improves the PromptedWS by up to 12.7 points on benchmark tasks.

1 Introduction

Distillation from computationally intensive large language models (LLMs) is a valuable technique for mitigating the high costs of running LLMs directly. However, if the LLM outputs used for distillation are noisy or biased, the distilled model might underperform the original model. Recent research [1, 2] in prompted weak supervision (PromptedWS) aims to integrate efficient distillation from LLMs with robust denoising strategies from programmatic weak supervision [3, 4]. In programmatic weak supervision, multiple noisy sources of labels are combined to infer probabilistic labels by resolving the agreements and disagreements. This is usually accomplished with a probabilistic label model. The probabilistic labels are then used to train an end model to solve the task. However, the sources can exhibit spurious correlations, resulting in a label model that cannot accurately resolve their conflicts. To address this issue, *structure discovery* methods for weak supervision aim to learn the statistical relationships among sources in the absence of ground truth labels [5, 6].

In PromptedWS, prompted LLMs are the sources of supervision. Unlike traditional sources that are usually written in code, prompts can be correlated due to sharing an underlying LLM and computationally expensive at the scale of distillation. These issues make existing structure discovery methods challenging to apply to PromptedWS. Nevertheless, prompted LLMs offer a unique opportunity to solve the weak supervision structure discovery problem: LLMs provide rich internal structure within their embedding space. The key idea lies in leveraging the internal representations of prompts to discover the underlying structures among them. Our findings suggest that the similarity between these prompt representations can serve as a predictor of potentially harmful correlations. Moreover, structure discovery using the embedding space can also bring significant advantages in computational efficiency. In light of these observations, we highlight the problem of structure discovery in PromptedWS as an important challenge for efficient and effective distillation of LLMs.

In this paper, we present Structure Refining Module, a simple yet effective first solution to this novel problem (see Figure 1). Our approach allows the end users to quickly and accurately discover

[*] Equal contribution

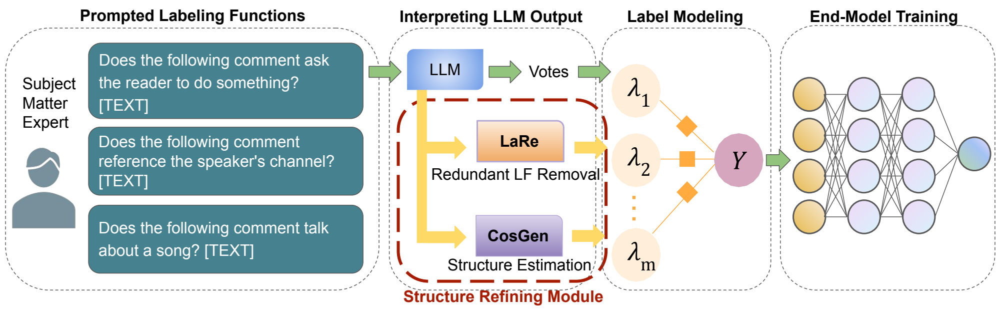
Prompted Labeling Functions
Subject Matter Expert
Does the following comment ask the reader to do something? [TEXT]
Does the following comment reference the speaker's channel? [TEXT]
Does the following comment talk about a song? [TEXT]
Interpreting LLM Output
LLM
Votes
LaRe
Redundant LF Removal
CosGen
Structure Estimation
Structure Refining Module
Label Modeling
λ1
λ2
λm
Y
End-Model Training

Figure 1: Prompted weak supervision workflow showing the Structure Refining Module as a plugin.

the correlations while significantly reducing the computational cost for both structure learning and label modeling. Structure Refining Module serves as a plug-in for existing PromptedWS workflows. To summarize, our main contributions are three-fold: (1) We propose a novel method for efficient structure learning in prompted weak supervision by leveraging similarity in the embedding space of prompted labeling functions. (2) We demonstrate the effectiveness of our proposed method by showing that it outperforms the prompted weak supervision baseline by an average of 5.9 points while saving significant computation. (3) We conduct comprehensive ablation and analysis experiments. We validate the core assumptions and analyze the contribution of each component of our method.

2 Background & Related Work

Programmatic weak supervision As an alternative to costly manual labeling, programmatic weak supervision [7, 3, 4] estimates the accuracy of weak sources without *any* labeled data. In programmatic weak supervision, users encode various weak supervision sources such as user-written heuristics [3, 8, 9], knowledge bases [10, 11], pre-trained models [12–14, 1, 2], and third-party tools [15] into labeling functions (LFs), which can either vote for a label or abstain.

Integrating LLMs into weak supervision Prompting is a powerful technique for many few-shot and zero-shot applications with large language models [16]. Prompting has been used to create and modify datasets in many ways [17–24]. Recent studies explored integrating prompting and programmatic weak supervision [1, 2]. Prompting also offers a unique opportunity to relax rigid code-based programmatic weak supervision sources.

Structure learning in weak supervision Multiple methods for learning statistical dependencies among LFs have been proposed [5, 6, 25]. Bach et al. [5] and Varma et al. [25] learn dependency structures from the votes of the LFs. Varma et al. [6] infer the structure through static analysis of the code specifying the LFs, with additional assumptions that the LFs operate over domain-specific primitives. With the dependency structure, many methods can then incorporate this information into the label modelling process for improved label quality, either through factor functions [7, 26] or graph structures [27–29].

Prompted weak supervision setup In a PromptedWS [1, 2] setup, given a dataset $D = \{x_1, x_2, \cdots, x_n\}$ of classification task with n unlabeled data points and each $x_i (i \in [n])$ with an unobserved true label $y_i \in \mathcal{Y}$, the goal is to infer the true label using weak supervision. We are also given a small set of heuristics created manually for labeling the unlabeled data. For example, in the case of labeling spam comments, we may find that many spam examples contains call to action, for example, asking readers to subscribe to a YouTube channel. In PromptedWS, these heuristics are expressed as natural language prompts, which can be fed into LLMs for answers. We then can map the answer to a label according to the heuristics. Formally, by observing the features in the development set, the SME could design a set of heuristics $\mathcal{H} = \{h_1, \cdots, h_m\}$ as well as a label map $\mathcal{M} : \mathcal{S} \rightarrow \mathcal{Y} \cup \{\varnothing\}$. The heuristics are encapsulated by a prompt template. For example, the prompt template could be

```
Does the following comment ask the reader to do something?  [TEXT]
```

where `[TEXT]` is a placeholder for the text of each comment. By feeding this template to the pre-trained LLM $\mathcal{A}$, it returns a text: $s = \mathcal{A}(h, x) \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible strings that could be returned from the LLM. Then the label map $\mathcal{M}$ maps the returned answer s to one of the class labels within the label space $\lambda_i \in \mathcal{Y}$ to vote for a label or maps s to a special symbol $\varnothing$, indicating abstaining and not voting for any label. For the above example, the label map would map positive responses such as `YES` and `True` to the SPAM label and all the other responses to abstentions.

Combining the above prompt template and label map together, we get a set of *prompted labeling functions*: $S_{\text{LF}} = \{\mathcal{M}(\mathcal{A}(h_1, \cdot)), \cdots, \mathcal{M}(\mathcal{A}(h_m, \cdot))\}$. Applying the above m LFs to the unlabeled dataset $D = \{x_1, \cdots, x_n\}$ would create a $n \times m$ label matrix L. The rest of the workflow remains the same as a standard weak supervision framework: a *label model* is used to aggregates the votes in L to produce probabilistic estimates of the true label. While many label modeling methods assume conditional independence among the LFs given the latent label [7, 3], recent label models may consider user-specified dependency structures [28]. Finally an appropriate *end model* is trained by minimizing the expected empirical risk with respect to the probabilistic estimates of the true labels.

3 Approach

Our method contains two sequential components: ***La****beling function* ***Re****moval* (LaRe) and ***Co****rrelation* ***S****tructure* ***Gen****eration* (CosGen). LaRe is designed to remove heavily correlated LFs while balancing performance and efficiency. We also introduce CosGen to estimate the dependency structures for the prompted LFs. Pseudocode of our method can be found in Appendix Section B

LaRe The goal of LaRe is to remove redundant prompted LFs in order to reduce computation need while maintaining labeling accuracy. To begin, the user may decide how many LFs should be removed by examining the similarity matrix among the prompted LFs set S_{LF}. Higher concentrations in the similarity matrix indicate higher correlations or possibility of redundant LFs to be removed. In addition, more LFs could lead to more redundant LFs to be removed when the labeling function set S_{LF} is large. After deciding how many labeling functions to be removed, potentially redundant LFs are then identified, removed based on their similarity matrix M. Next is to rank the pairwise similarities for each LF pair. Then at each step, we find the LF pair that has the largest similarity and remove the LF with a larger index to keep our algorithm deterministic. Removing the redundant LF results in a new set of LFs $S^{'}_{\text{LF}}$, where $|S^{'}_{\text{LF}}| = m - m_r$ and m_r is the total number of LFs removed.

CosGen CosGen aim to estimate the dependency among LFs based on the correlations indicated in the similarity matrix $M^{'}$, which contains only the correlation information of the refined LF set $S^{'}_{\text{LF}}$. One key intuition is to construct dependency for highly correlated LFs indicated by $M^{'}$. However, the structure directly estimated from this method may be non-identifiable and cannot be used for label modeling. In order to guarantee the structure to be identifiable, we first find two LFs that have the smallest correlation and assume these two to be mutually independent while at the same time, also to be independent with all the other LFs. Then we find the rest dependencies by assigning edges to LFs with high similarity. The hyperparameter for CosGen is the number of edges (m_e) in the graph and it dictates the sparsity of the graph. In practice, we can also replace this parameter to a percentage threshold on the total number of possible edges. Note that $m_e \leq \frac{(m-2)\cdot(m-3)}{2}$. The structure inferred from our similarity matrix can be then passed to the label model.

4 Experimental Results

Dataset We evaluate on three datasets: YouTube [30], SMS [31], and Spouse [32]. (Please refer to the summary of the datasets in Appendix Section D.) We follow the prompted labeling functions (LFs) from [1], which are translated from LFs from WRENCH [33].

Baselines We empirically evaluate the our method against three baselines: (1) the standard weak supervision with LFs in WRENCH [33]. (2) PromptedWS [1] (3) PromptedWS with weak supervision structure learning (WSSL) [25] by passing the structure learned from WSSL to the label model in the PromptedWS) We use the evaluation metrics specified by WRENCH [33] for the three datasets.

Setup We follow the conventional workflow of programmatic weak supervision. For each dataset in our analysis, we assume the training splits are unlabeled, the LFs (either prompted LFs in PromptedWS or code-based LFs from WRENCH) are applied to the unlabeled training splits to generate weak labels. Throughout our experiments, we use T0++ [34]. In all of our experiments, unless otherwise specified, we use FlyingSquid [28] as our label model to combine and denoise the labelers' votes into probabilistic labels, which are then used to train a RoBERTa[35]-based end-model. We pick the hyperparameters with the validation split provided by WRENCH. More details about the experiments can be found in Appendix Section E.

Results Table 1 outlines the performance of the 3 baselines. On average, our method improves 13.7 points compared to WRENCH, 5.9 points compared to PromptedWS and 1.1 points compared to PromptedWS+WSSL. Experimental results demonstrate that our Structure Refining Module can

Table 1: Performance mean and standard error of the 4 approaches with 5 random seeds.

	YouTube(Acc)	SMS(F1)	Spouse(F1)
WRENCH [33]	94.6(0.5)	93.5(0.7)	25.5(1.8)
PromptedWS [1]	94.8(1.2)	90.0(4.1)	52.1(1.3)
PromptedWS + WSSL [25]	93.0(1.3)	**94.5(1.7)**	63.9(0.9)
PromptedWS + Structure Refining Module	**95.6(0.4)**	94.2(1.4)	**64.8(0.2)**

significantly improve the performance over the vanilla PromptedWS. PromptedWS in complementary with our method can achieve superior performance, outperforming the WRENCH in all the three datasets used in our experiment. Additionally, our method outperforms or achieves comparable result to the state-of-the-art structure learning method WSSL [25].

5 Ablation and Analysis

Similarities reveal correlations Our goal is to learn more expressive correlations that are invariant to data. We formulate experiments to validate the idea that if two labeling functions (LFs) are correlated, removing one may not significantly harm the model's performance. We show that after removing one of the most correlated LFs, the result either improved or slightly dropped below the baseline. This further suggests that the similarity matrix can reveal information about the real correlations of LFs. For details about this ablation, please refer to Appendix Section F.

Removing LFs can improve performance In practice, the number or portion of removal should be more fine-grained and tailored to the total number of labeling functions for better performance. We conduct experiments to compare the performance of LaRe under different removal rates and exam the effect of LFs removal. Our experiments (See Appendix Section G) show that using LaRe alone could improve the performance.

Performance v.s. efficiency One benefit of removing redundant LFs is efficiency: when the total number of labeling functions is reduced, we are prompting less examples to the LLMs and therefore save more computations and time. Our experiments show that it is not always a trade-off but sometimes could be a win-win with Structure Refining Module.

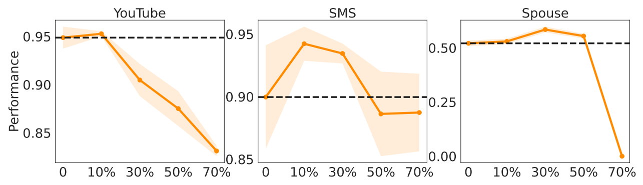

Figure 2: Performance on different removal rate. Dashed lines are PromptedWS without any removal. Showing results of removing 10%, 30%, 50%, 70% of the LFs.

Correlation structure generation We compare the CosGen module of our Structure Refining Module with weak supervision structure learning (WSSL) [25], a state-of-the-art structure learning for programmatic weak supervision that learns a sparse structure from data. We evaluate and analyze both methods in terms of performance and efficiency. The detailed performance comparisons are given in Table 8 and Appendix Section H. The time and computational cost for CosGen is negligible while the time of learning structures using WSSL depends on the data and the number of LFs.

As shown in Table 2, structure learning with WSSL takes around 0.1s for YouTube and Spouse dataset and around 24s to learn a structure of SMS for WSSL. In contrast, CosGen produces a structure that is as effective as WSSL, but with negligible computation time.

Table 2: WSSL runtime for inferring LFs structures.

	YouTube	SMS	Spouse
time(s)	0.11	24.76	0.15

6 Conclusion

In this paper, we propose Structure Refining Module, a novel approach for efficient structure discovery and management in prompted weak supervision setup. Our approach asks large language models for information beyond the inference output, leveraging similarities in the embedding space to detect redundant LFs and learn the dependency structures among prompted LFs. We demonstrate the effectiveness of our approach through extensive experiments and provide comprehensive ablation analysis for Structure Refining Module. We believe that the Structure Refining Module is a valuable tool for weak supervision practitioners who wish to optimize their workflow for both efficiency and performance. In future work, we plan to further investigate scalable and robust integration of Large Language Models into Programmatic Weak Supervision workflows.

Acknowledgements

We would like to thank Jiayou Zhang for his helpful suggestions and comments. This material is based on research sponsored by Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under agreement number FA8750-19-2-1006. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) or the U.S. Government. We gratefully acknowledge support from Google and Cisco. Disclosure: Stephen Bach is an advisor to Snorkel AI, a company that provides software and services for data-centric artificial intelligence.

Authors' Note. The first two authors contributed equally. Co-first authors can prioritize their names when adding this paper's reference to their resumes.

References

[1] R. Smith, J. A. Fries, B. Hancock, and S. H. Bach, "Language models in the loop: Incorporating prompting into weak supervision," *arXiv preprint arXiv:2205.02318*, 2022.

[2] P. Yu and S. Bach, "Alfred: A system for prompted weak supervision," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, (Toronto, Canada), pp. 479–488, Association for Computational Linguistics, July 2023.

[3] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, p. 269, NIH Public Access, 2017.

[4] J. Zhang, C.-Y. Hsieh, Y. Yu, C. Zhang, and A. Ratner, "A survey on programmatic weak supervision," *arXiv preprint arXiv:2202.05433*, 2022.

[5] S. H. Bach, B. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," in *International Conference on Machine Learning*, pp. 273–282, PMLR, 2017.

[6] P. Varma, B. D. He, P. Bajaj, N. Khandwala, I. Banerjee, D. Rubin, and C. Ré, "Inferring generative model structure with static analysis," *Advances in neural information processing systems*, vol. 30, 2017.

[7] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," *Advances in neural information processing systems*, vol. 29, 2016.

[8] Y. Meng, J. Shen, C. Zhang, and J. Han, "Weakly-supervised neural text classification," in *proceedings of the 27th ACM International Conference on information and knowledge management*, pp. 983–992, 2018.

[9] A. Awasthi, S. Ghosh, R. Goyal, and S. Sarawagi, "Learning from rules generalizing labeled exemplars," *arXiv preprint arXiv:2004.06025*, 2020.

[10] C. Liang, Y. Yu, H. Jiang, S. Er, R. Wang, T. Zhao, and C. Zhang, "Bond: Bert-assisted open-domain named entity recognition with distant supervision," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1054–1064, 2020.

[11] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 541–550, 2011.

[12] S. H. Bach, D. Rodriguez, Y. Liu, C. Luo, H. Shao, C. Xia, S. Sen, A. Ratner, B. Hancock, H. Alborzi, *et al.*, "Snorkel drybell: A case study in deploying weak supervision at industrial scale," in *Proceedings of the 2019 International Conference on Management of Data*, pp. 362–375, 2019.

[13] J. Zhang, B. Wang, X. Song, Y. Wang, Y. Yang, J. Bai, and A. Ratner, “Creating training sets via weak indirect supervision,” *arXiv preprint arXiv:2110.03484*, 2021.

[14] P. Yu, T. Ding, and S. H. Bach, “Learning from multiple noisy partial labelers,” in *Artificial Intelligence and Statistics (AISTATS)*, 2022.

[15] P. Lison, A. Hubin, J. Barnes, and S. Touileb, “Named entity recognition without labelled data: A weak supervision approach,” *arXiv preprint arXiv:2004.14723*, 2020.

[16] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing,” *arXiv preprint arXiv:2107.13586*, 2021.

[17] T. Schick and H. Schütze, “Generating datasets with pretrained language models,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[18] J. Ye, J. Gao, Q. Li, H. Xu, J. Feng, Z. Wu, T. Yu, and L. Kong, “ZeroGen: Efficient zero-shot learning via dataset generation,” *arXiv preprint arXiv:2202.07922*, 2022.

[19] Y. K. Chia, L. Bing, S. Poria, and L. Si, “RelationPrompt: Leveraging prompts to generate synthetic data for zero-shot relation triplet extraction,” in *Findings of the Association for Computational Linguistics*, 2022.

[20] Y. Wu, M. Gardner, P. Stenetorp, and P. Dasigi, “Generating data to mitigate spurious correlations in natural language inference datasets,” in *Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[21] L. Bonifacio, H. Abonizio, M. Fadaee, and R. Nogueira, “InPars: Data augmentation for information retrieval,” *arXiv preprint arXiv:2202.05144*, 2022.

[22] H. Lang, M. Agrawal, Y. Kim, and D. Sontag, “Co-training improves prompt-based learning for large language models,” *arXiv preprint arXiv:2202.00828*, 2022.

[23] Y. Elazar, N. Kassner, S. Ravfogel, A. Ravichander, E. Hovy, H. Schütze, and Y. Goldberg, “Measuring and improving consistency in pretrained language models,” *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1012–1031, 2021.

[24] R. Zhang, Y. Yu, P. Shetty, L. Song, and C. Zhang, “PRBoost: Prompt-based rule discovery and boosting for interactive weakly-supervised learning,” in *Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[25] P. Varma, F. Sala, A. He, A. Ratner, and C. Ré, “Learning dependency structures for weak supervision models,” in *International Conference on Machine Learning*, pp. 6418–6427, PMLR, 2019.

[26] C. Shin, W. Li, H. Vishwakarma, N. Roberts, and F. Sala, “Universalizing weak supervision,” *arXiv preprint arXiv:2112.03865*, 2021.

[27] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, “Training complex models with multi-task weak supervision,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4763–4771, 2019.

[28] D. Fu, M. Chen, F. Sala, S. Hooper, K. Fatahalian, and C. Ré, “Fast and three-rious: Speeding up weak supervision with triplet methods,” in *International Conference on Machine Learning*, pp. 3280–3291, PMLR, 2020.

[29] P. Varma, F. Sala, S. Sagawa, J. Fries, D. Fu, S. Khattar, A. Ramamoorthy, K. Xiao, K. Fatahalian, J. Priest, *et al.*, “Multi-resolution weak supervision for sequential data,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[30] T. C. Alberto, J. V. Lochter, and T. A. Almeida, “Tubespam: Comment spam filtering on youtube,” in *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pp. 138–143, IEEE, 2015.

[31] J. M. Gómez Hidalgo, G. C. Bringas, E. P. Sánz, and F. C. García, “Content based sms spam filtering,” in *Proceedings of the 2006 ACM symposium on Document engineering*, pp. 107–114, 2006.

[32] D. P. Corney, D. Albakour, M. Martinez-Alvarez, and S. Moussa, “What do a million news articles look like?,” in *NewsIR@ ECIR*, pp. 42–47, 2016.

[33] J. Zhang, Y. Yu, Y. Li, Y. Wang, Y. Yang, M. Yang, and A. Ratner, “WRENCH: A comprehensive benchmark for weak supervision,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

[34] V. Sanh, A. Webson, C. Raffel, S. H. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, T. L. Scao, A. Raja, *et al.*, “Multitask prompted training enables zero-shot task generalization,” *arXiv preprint arXiv:2110.08207*, 2021.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[36] J. Su, P. Yu, J. Zhang, and S. H. Bach, “Leveraging large language models for structure learning in prompted weak supervision,” in *IEEE International Conference on Big Data*, 2023.

[37] N. Dalvi, A. Dasgupta, R. Kumar, and V. Rastogi, “Aggregating crowdsourced binary ratings,” in *Proceedings of the 22nd international conference on World Wide Web*, pp. 285–294, 2013.

[38] M. Joglekar, H. Garcia-Molina, and A. Parameswaran, “Comprehensive and reliable crowd assessment algorithms,” in *2015 IEEE 31st International Conference on Data Engineering*, pp. 195–206, IEEE, 2015.

[39] R. E. Schapire and Y. Freund, “Boosting: Foundations and algorithms,” *Kybernetes*, vol. 42, no. 1, pp. 164–166, 2013.

[40] A. Balsubramani and Y. Freund, “Scalable semi-supervised aggregation of classifiers,” *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[41] Y. Zhang, X. Chen, D. Zhou, and M. I. Jordan, “Spectral methods meet em: A provably optimal algorithm for crowdsourcing,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3537–3580, 2016.

[42] A. Blum and T. Mitchell, “Combining labeled and unlabeled data with co-training,” in *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100, 1998.

[43] S. R. Cachay, B. Boecking, and A. Dubrawski, “Dependency structure misspecification in multi-source weak supervision models,” *arXiv preprint arXiv:2106.10302*, 2021.

[44] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

A Code and Extended Experiments

Please find the code of the preliminary experiments included in this paper from `https://github.com/BatsResearch/su-bigdata23-code`. For more extensive experiments and analysis, please refer to our 2023 IEEE BigData Paper [36].

B Pseudocode

Our method contains two sequential components: ***La**beling function **Re**moval* (LaRe) and ***Co**rrelation **S**tructure **Gen**eration* (CosGen). LaRe is designed to remove heavily correlated LFs while balancing performance and efficiency. It can be especially helpful when the total number of LFs is very large. In addition to LaRe, we introduce CosGen to estimate the dependency structures for the prompted LFs. We describe the pseudocode in 1.

Algorithm 1 Pseudocode for Structure Refining Module

```
1:  Input: Similarity matrix $M$, original Prompted LF set $S_{\text{LF}}$, number of LF to be removed: $m_r$,
    number of edges in the graph: $m_e$.
2:  $S_r = \varnothing, \mathcal{E} = \varnothing$
3:  // Labeling function removal (LaRe)
4:  for t $\in [m_r]$ do
5:      $(i, j) = \arg\max M_{i,j}$
6:      $k = \max\{i, j\}$
7:      $S_r \leftarrow S_r \cup \{k\}$
8:      $M_{k,:} \leftarrow 0, M_{:,k} \leftarrow 0$
9:  end for
10: $S'_{\text{LF}} \leftarrow S_{\text{LF}} / S_r$
11: // Correlation structure generation (CosGen)
12: Let $M'$ be the similarity matrix of LFs in $S'_{\text{LF}}$
13: $(i, j) = \arg\min M'_{i,j}$
14: $M'_{i,:} \leftarrow 0$, $M'_{:,i} \leftarrow 0$, $M'_{j,:} \leftarrow 0$, $M'_{:,j} \leftarrow 0$
15: for $t \in [m_e]$ do
16:     $(i, j) = \arg\max M'_{i,j}$
17:     $\mathcal{E} = \mathcal{E} \cup \{(i, j)\}$
18:     $M'_{i,j} \leftarrow 0, M'_{j,i} \leftarrow 0$
19: end for
20: return Refined labeling function set $S'_{\text{LF}}$, edges in the graph $\mathcal{E}$
```

C Extended Related Work

Programmatic Weak Supervision: Manually labeling training data is prohibitively expensive and time-consuming. A common alternative is to use weak supervision sources. Estimating the accuracy of multiple sources is well studied in many areas such as crowd sourcing[37, 38], boosting [39–41], and co-training [42]. In this paper, we focus on programmatic weak supervision [7, 3, 4], which estimates the accuracy of weak sources without *any* labeled data. In programmatic weak supervision, users encode various weak supervision sources such as user-written heuristics [3, 8, 9], knowledge bases [10, 11], pre-trained models [12–14, 1, 2], and third-party tools [15] into labeling functions (LF), which can then give votes on the true labels of the unlabeled data.

Integrating LLMs into Weak Supervision: Prompting is a powerful technique for many few-shot and zero-shot applications with large language models [16]. Prompting has been used to create and modify datasets in many ways [17–24]. Recent studies explored integrating prompting and weak supervision [1]. Prompting also offers a unique opportunity to relax rigid code-based programmatic weak supervision sources. In this work, we aim to make the integration of LLMs into weak supervision more effective by mitigating the downside of hazardous correlations among the prompted LFs.

Structure Learning in Weak Supervision: It can be highly beneficial to capture the statistical dependencies among LFs during label modeling since model misspecification can lead to incorrect estimates of the true labels [43]. Multiple methods for learning such dependencies have been proposed [5, 6, 25]. Bach et al. [5] and Varma et al. [25] learn dependency structures from the votes of the LFs. Varma et al. [6] infer the structure through static analysis of the code specifying the LFs, with additional assumptions that the LFs operate over domain-specific primitives. With the dependency structure in hand, many methods can then incorporate this information into the label modelling process for improved label quality, either through factor functions [7, 26] or graph structures [27–29]. The structure obtained by our Structure Refining Module can be incorporated into any such method.

D Dataset Details

Table 3: Summary statistics for text classification dataset used in our experiments. P(positive) is the class balance of the positive label calculated by the relative frequency of all gold labeled splits with standard error in the parentheses. Note that since we focus on PromptedWS, all the labeling functions we refer to here are prompted LFs as defined in [1].

Name	Task	Class	P(positive)	#LFs	#Augmented LFs	Train	Valid	Test
YouTube	Spam Detection	HAM,SPAM	0.488(0.02)	10	20	1586	120	250
SMS	Spam Detection	HAM,SPAM	0.132(< 0.01)	73	146	4571	500	500
Spouse	Relation extraction	NOT SPOUSE, SPOUSE	0.074(< 0.01)	11	22	22254	2811	2701

In our experiments, we evaluated three datasets: YouTube [30], SMS [31], and Spouse [32]. The datasets are obtained and loaded using the WRENCH benchmark for weak supervision [33]. Table 3 provides summary statistics.

E Experimental Details

The approaches we compared to are:

1. **WRENCH benchmark** [33]: The original WRENCH code-based LFs are released as part of the benchmark. Here, we use Majority vote as the label model since it performed best when using RoBERTa as the end model.
2. **PromptedWS** [1]: The prompted versions of WRENCH LFs, where the code-written LFs are replaced with prompts.
3. **PromptedWS with WSSL** [25]: The structure learning method we compared to is called Weak supervision structure learning (WSSL) [25], which is a robust PCA based algorithm for learning dependencies. This structure learning algorithm takes weak labels as inputs and return a graph representing the dependencies among LFs. Here, we input the weak labels from PromptedWS to get a graph and pass this dependency graph to the label model Flyingsquid.
4. **PromptedWS with Structure Refining Module:** The same framework as PromptedWS but with refining module (proposed in our paper) to refine the LFs and return the dependency structure. The dependency graph is then passed to the label model FlyingSquid.

For baselines involving PromptedWS, we use T0++ [34] as the large language model, which is a free, publicly available 11B parameter model based on T5 architecture [44]. T0++ is trained using a large dataset of supervised tasks transformed into instruction prompted training data and can achieve a better zero shot classification performance that often matches or exceeds GPT-3, which is much larger. Note that, 4 WRENCH datasets (IMDB, Yelp, AG News, TREC) were used for training T0++, we therefore exclude them from our analysis. The T0++ model requires 42 GB of GPU to efficiently run locally without parameter offloading. We use 2 NVIDIA A100 GPUs for inference.

Evaluation Metrics: We report default metrics specified in the WRENCH benchmark [33] for direct comparison: for YouTube dataset, we report accuracy and for SMS and spouse dataset, we report F1. Our performance metrics are reported as the mean and standard error of 5 independent runs using different random seeds.

E.1 Prompted Labeling Functions

The prompted labeling functions used in PromptedWS are illustrated in Table 4 (YouTube and Spouse) and Table 5 (SMS) respectively. They are taken from Smith et al. [1]. Note that when using Google translator to create the augmented labeling functions, there might be some labeling functions that are wrongly translated or some labeling functions that are the same after translation, we didn't do any human intervention about this because we expect our method to be robust enough to these cases. Denote the labeling function set provided in WRENCH as S_{LF}, after augmenting, total number of LFs is doubled as $2 \cdot |S_{\text{LF}}|$.

Table 4: Prompted labeling functions for YouTube and Spouse datasets. The YouTube dataset has class labels HAM and SPAM, and the dataset has class labels NOT SPOUSE and SPOUSE. The label map transforms the answer "yes" to the value denoted by the label column(either HAM or SPAM for YouTube dataset and either NOT SPOUSE or SPOUSE for spouse dataset) and we consider other answers as abstain. The original (not augmented) labeling functions are from Smith et al. [1].

Dataset	Template	Label
YouTube LFs	Does the following comment talk about a song?\n\n[TEXT]	HAM
	Is the following comment fewer than 5 words?\n\n [TEXT]	HAM
	Does the following comment mention a person's name?\n\n [TEXT]	HAM
	Does the following comment express a very strong sentiment?\n\n [TEXT]	HAM
	Does the following comment express a subjective opinion?\n\n [TEXT]	HAM
	Does the following comment reference the speaker's channel or video? \n\n [TEXT]	SPAM
	Does the following comment ask you to subscribe to a channel?\n\n [TEXT]	SPAM
	Does the following comment have a URL?\n \n[TEXT]	SPAM
	Does the following comment ask the reader to do something?\n\n [TEXT]	SPAM
	Does the following comment contain the words "check out"? \n\n [TEXT]	SPAM
Spouse LFs	Context: [TEXT]\n \n Are [PERSON1] and [PERSON2] family members?	NOT SPOUSE
	Context: [TEXT]\n \n Is [PERSON1] said to be a family member?	NOT SPOUSE
	Context: [TEXT]\n \n Is [PERSON2] said to be a family member?	NOT SPOUSE
	Context: [TEXT]\n \n Are [PERSON1] and [PERSON2] dating?	NOT SPOUSE
	Context: [TEXT]\n \n Are [PERSON1] and [PERSON2] co-workers?	NOT SPOUSE
	Context: [TEXT]\n \n Is there any mention of "spouse" between the entities [PERSON1] and [PERSON2]?	SPOUSE
	Context: [TEXT]\n \n Is there any mention of "spouse" before the entity [PERSON1]?	SPOUSE
	Context: [TEXT]\n \n Is there any mention of "spouse" before the entity [PERSON2]?	SPOUSE
	Context: [TEXT]\n \n Do [PERSON1] and [PERSON2] have the same last name?	SPOUSE
	Context: [TEXT]\n \n Did [PERSON1] and [PERSON2] get married?	SPOUSE
	Context: [TEXT]\n \n Are [PERSON1] and [PERSON2] married?	SPOUSE

Table 5: Prompted labeling functions for the SMS dataset. The template asks whether there are certain keywords in the text and the label map transforms the answer "yes" to the value denoted by the label column (either HAM or SPAM) and we consider other answers as abstain. The original (not augmented) labeling functions are from Smith et al. [1].

Template	Does the following text message contain the words "[KEYWORDS]"?\n \n [TEXT]	Label
[KEYWORDS]	??1.50, ??500, ??5000, call for offer, cash prize, chat date, chat to, childporn, credits, dating call, direct, expires now, fantasies call, free phones, free price, free ringtones, free sex, free tone, guaranteed free, guaranteed gift, hard live girl, important lucky, inviting friends, latest, latest offer, message call, new mobiles, no extra, password, please call, sms reply, unlimited calls, urgent award guaranteed, urgent prize, voucher claim, welcome reply, win shopping, winner reward, won call, won cash, won cash prize, won claim	SPAM
	I, I can did, I it, I miss, I used to, adventuring, amrita, can't talk, did u got, do you, fb, goodo, hee hee, i'll, jus, link, maggi, mine, my kids, noisy, praying, shit, should I, thanks, that's fine, thats nice, u how 2, we will, where are, wtf, your I	HAM

F Extended Analysis: From Labeling Function Similarities to Correlations

Table 6: Toy experiment of removing one of the most correlated LFs. The first row is the result of PromptedWS without moving any LFs, and the second and third rows are the result of PromptedWS after removing one LF from the most correlated LFs pairs.

	YouTube	SMS	Spouse
PromptedWS	94.8(1.2)	90.0(4.1)	52.1(1.3)
Remove most correlated LF	94.7(0.7)	93.0(0.8)↑	53.0(1.2)↑
	95.3(0.3)↑	91.4(2.2)↑	52.3(1.8)↑

Previous work [5, 25] characterizes LFs correlation structures based on their output, which can be sensitive to the data and can result in spurious correlations. We aim to learn more expressive correlations that are invariant to data. To achieve this, we focus on finding correlations intrinsic to the LFs. Here We may consider each LF as a task and capture their correlations through their respective task embeddings. We use T0++ [34] to embed each LFs with last layer embeddings and compute cosine similarities matrix. The heatmap visualization of the similarity matrix for the YouTube dataset is given in Figure 3.

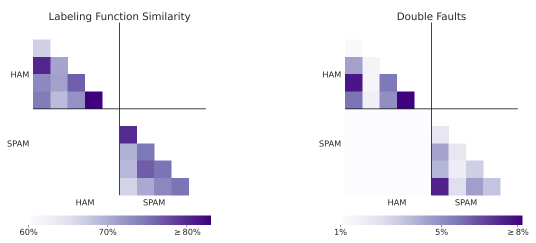

Figure 3: Visualization of the similarity matrix for labeling functions in the YouTube dataset compared with double faults, i.e., examples on which both labeling functions make a mistake.

The similarity information has significant overlaps with the correlated errors among LFs, which can be used to remove the most correlated LFs and still achieve good performance. We construct a toy experiment to further exam our assumption. Our toy experiment is based on the idea that if two labeling functions (LFs) are correlated, removing one may not significantly harm the model's performance. Removing the most correlated LF leads to more accurate weights assigned to each LF, resulting in a more accurate end model. Our toy experiment results align with the assumption. As shown in Table 6, after removing one of the most correlated LFs, the result either improved or slightly dropped below the baseline. This further suggests that the similarity matrix can reveal information about the real correlations of LFs.

G Extended Analysis: Effect of Labeling Function Removal

In this subsection, we exam the contribution of LaRe in our Structure Refining Module. The core idea for LaRe is that when LFs are highly correlated and redundant, removing some can improve performance and efficiency. To extensively understand effect of LFs removal, we only consider the LaRe by setting $m_e = 0$ for our Structure Refining Module. We examined the impact of removing LFs based on correlation by conducting experiments where we removed 10%, 30%, 50%, and 70% of labeling functions.

Table 7: We documented the number of labeling functions removed as well as accuracy or F1 metric for each experiment. We report the mean and standard error with 5 random runs and the best results are indicated in bold. We highlight results that outperform PromptedWS with arrows. We also estimate the total number of token saved when running with Train/Test/Valid splits.

					Estimated number of saved tokens		
		Performance(Acc/F1)	#(LF removed)	# (prompts saved)	Train	Test	Valid
Youtube	PromptedWS	94.8 ± 1.2	0	0	0	0	0
	removing 10%	**95.3 ± 0.3** ↑	1	1586	70900	11369	5186
	removing 30%	90.5±1.7	3	4758	212700	34107	15558
	removing 50%	87.5±1.9	5	7930	354500	56845	25930
	removing 70%	83.1±0.5	7	11102	496300	79583	36302
SMS	PromptedWS	90.0 ± 4.1	0	0	0	0	0
	removing 10%	**94.2± 1.4** ↑	7	31997	1343069	144984	147903
	removing 30%	93.5± 0.8 ↑	22	100562	4221074	455664	464838
	removing 50%	88.6±3.4	36	164556	6907212	745632	760644
	removing 70%	88.7±3.1	51	233121	9785217	1056312	1077579
Spouse	PromptedWS	52.1 ± 1.3	0	0	0	0	0
	removing 10%	53.0± 1.2 ↑	1	22254	1980962	234659	249159
	removing 30%	**58.5± 1.3** ↑	3	66762	5942886	703977	747477
	removing 50%	55.4±1.1 ↑	6	133524	11885772	1407954	1494954
	removing 70%	0±0	8	178032	-	-	-

As shown in 4, compared to PromptedWS, using LaRe alone could improve the performance. Even though labeling functions used in PromptedWS are carefully selected to avoid correlation and facilitate weak supervision, remove 10% of the labeling functions (on YouTube and SMS) and remove 30% of the labeling functions (on Spouse) still improve the results, suggesting that there still could be some redundancies with manually selected LFs. Note that our experiments are relatively coarse by setting the removal rate to be 10%, 30%, 50%, 70% for all the dataset because the total number of labeling functions varies from 10 to 146 for the six group experiments, so the effects of “removing 70%" could be different on different dataset based on the total number of LFs as well as the redundancies of the total LFs set.

Overall, the experiments show that the LaRe in our Structure Refining Module could help to improve the performance. Note that whether the performance improve or drop depends on the trade-off between the performance gain by handling the dependencies and the performance drop due to the reduced information from removing labeling functions, so the performance plot over removal isn't perfectly concave. Also, we should not only care about the point that achieves the best performance, but all the points where the performance are better or equivalent to the PromptedWS (namely, the all the points above or near the dashed line in Figure 4), since these are points indicating an improved efficiency without negatively affect performance.

Performance and efficiency: trade-off or win-win? One benefit of removing redundant LFs is efficiency: when the total number of labeling functions is reduced, we are prompting less examples to the LLMs and therefore save more computations and time. When weighing performance and efficiency, it's important to consider the feasibility of removing LFs in terms of the influence on performance. However, as our experiments show, it is not always a trade-off but sometimes could be a win-win in PromptedWS with our Structure Refining Module. In Figure 4, for YouTube dataset, removing 10% of the labeling functions improves the performance compared to not removal, while saving more computations. The only trade-off comes down to if the users want to further remove LFs to save more resources. For SMS, removing 10% or 30% both improve performance, and users can choose based on their priorities. Sacrificing some performance can help save more computational costs by removing 50% or even 70% of labeling functions. By removing 70% of the labeling functions, though the performance drops 1.3 points, it reduces 51 labeling functions, and saved in total of 233121 prompts. For the Spouse dataset, removing 30% of LF is a win-win scenario, both improving performance and saving computational resources. We demonstrate this trade-off (or potentially win-win) in Table 7.

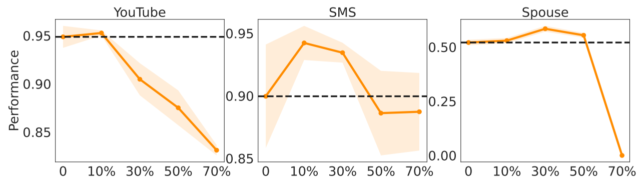

Figure 4: Performance on different removal rate. Dashed lines are PromptedWS without any removal. Showing results of removing 10%, 30%, 50%, 70% of the LFs.

H Extended Analysis: Analyzing Correlation Structure Generation

Table 8: Comparison of CosGen, weak supervision structure learning(WSSL) and without structures.

	YouTube	SMS	Spouse
PromptedWS	**94.8(1.2)**	90.0(4.1)	52.1(1.3)
PromptedWS + CosGen	94.2(0.5)	93.4(1.2)	**64.6(0.5)**
PromptedWS + WSSL	93.0(1.3)	**94.5(1.7)**	63.9(0.9)

Another core component of our Structure Refining Module is Correlation Structure Generation (CosGen). The main output of CosGen is the estimated structure $\mathcal{E}$, which contains all the labeling function pairs that are considered as correlated. In this subsection, we compare the CosGen module of our Structure Refining Module with weak supervision structure learning (WSSL) [25], a state-of-the-art structure learning for programmatic weak supervision that learns a sparse structure from data. We compare, evaluate and analyze both methods in terms of performance and efficiency. The performance comparisons are given in Table 8.

Table 8 shows that compared to PromptedWS without any consideration of correlations, passing the structure is extremely effective on Spouse dataset, improved 12.5 F1 score with CosGen and improved 11.8 F1 score using the WSSL. For SMS dataset, passing the structure also improves the results, boosting 3.4 Acc score and 4.5 Acc score using CosGen and the structure from WSSL respectively. Although both the structure from our refining module and the structure learned from data help to deal with dependencies and can effectively improve the performance, CosGen is much more efficient than WSSL. The time and computational cost for CosGen is negligible while the time of learning structures using WSSL depends on the data and the number of LFs.

As shown in Table 9, structure learning with WSSL takes around 0.1s for YouTube and Spouse dataset and around 24s to learn a structure of SMS for WSSL. Note that running a label model without any structures for SMS dataset only takes less than 1s. Compared to the label model run time, 24s for computing the structure alone should be considered as expensive and should not be ignored. The same holds for both YouTube and Spouse dataset: when using WSSL to learn a structure, the time spend on obtaining the structure is longer than label model runtime. In contrast, CosGen produces a structure that is as effective as WSSL, but with negligible computation time.

Table 9: WSSL runtime for inferring LFs structures.

	YouTube	SMS	Spouse
time(s)	0.11	24.76	0.15