Snorkel MeTaL: Weak Supervision for Multi-Task Learning

Alex Ratner, Braden Hancock, Jared Dunnmon, Roger Goldman, Christopher Ré
Stanford University
Stanford, CA
{ajratner,bradenjh,jdunnmon,chrismre}@cs.stanford.edu,goldmanr@stanford.edu

ABSTRACT

Many real-world machine learning problems are challenging to tackle for two reasons: (i) they involve multiple sub-tasks at different levels of granularity; and (ii) they require large volumes of labeled training data. We propose Snorkel MeTaL, an end-to-end system for multi-task learning that leverages *weak* supervision provided at *multiple levels of granularity* by domain expert users. In MeTaL, a user specifies a problem consisting of multiple, hierarchically-related sub-tasks—for example, classifying a document at multiple levels of granularity—and then provides *labeling functions* for each sub-task as weak supervision. MeTaL learns a re-weighted model of these labeling functions, and uses the combined signal to train a hierarchical multi-task network which is automatically compiled from the structure of the sub-tasks. Using MeTaL on a radiology report triage task and a fine-grained news classification task, we achieve average gains of 11.2 accuracy points over a baseline supervised approach and 9.5 accuracy points over the predictions of the user-provided labeling functions.

ACM Reference Format:
Alex Ratner, Braden Hancock, Jared Dunnmon, Roger Goldman, Christopher Ré. 2018. Snorkel MeTaL: Weak Supervision for Multi-Task Learning. In *DEEM'18: International Workshop on Data Management for End-to-End Machine Learning, June 15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209889.3209898

1 INTRODUCTION

An increasing number of real-world systems today are not just utilizing machine learning as a sub-component, but are in fact wholesale transitioning to "Software 2.0"-style architectures [4] where machine learning models are the principle deployed artifact. Especially for complex tasks e.g. involving vision, speech, control, and more, this approach has the advantages of better generalization as well as a more homogeneous and modular form factor. However, most real-world tasks in fact consist of *multiple* sub-tasks; and while machine learning models may perform well on the individual tasks, connecting them together leads to complex pipelines which lead to cascading errors and break the clean, "Software 2.0"-style paradigm of single model deployment.

In our experience building and interacting with users of Snorkel [13], we found exactly this obstacle across a range of application domains and settings. For example, when working with scientific and government collaborators on information extraction tasks, the need for multiple connected models—e.g. for candidate extraction, classification, and entity linking—quickly became apparent. More broadly, the sub-tasks were generally related both by constraints on their output values and shared features of their input values. Current multi-task and/or joint learning approaches do support shared structure between tasks, but still require large volumes of hand-labeled training data for *each* task—a prohibitive requirement for most settings.

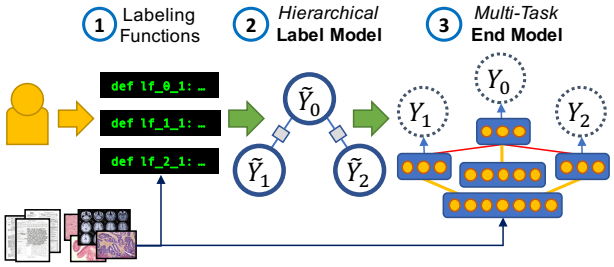

Figure 1: In MeTaL domain expert users write *labeling functions* (1) to provide weak supervision for a hierarchy of tasks. This weak supervision is de-noised by a generative *label model* (2), and then used to train a multi-task *end model* (3) that is auto-compiled based on the structure of the tasks.

To handle this multi-task supervision challenge, we propose Snorkel MeTaL, a prototype system at the confluence of two recent technical trends: multi-task learning through shared structure in deep neural networks, and weak supervision denoised using statistical methods. In multi-task learning [2], similar parameters are shared between separate tasks, the idea being that the inductive bias of multiple tasks will lead to better learned representations. However, hand-labeling large volumes of training data for *multiple* separate tasks is still a prohibitive burden. Thus, we propose to use standard multi-task modeling techniques, but to supervise them in a novel *weaker* way, extending the recently proposed paradigm of *data programming* [12], wherein domain expert users write *labeling functions* rather than hand-label training data. In MeTaL, we first denoise the outputs of these labeling functions—now written for a set of related tasks—using a generative model, and then use this as weak supervision for an auto-compiled multi-task network.

In this initial work, we tackle the restricted but important case of *hierarchically related* tasks, which characterizes many fine-grained classification tasks that can be broken into a set of hierarchical decisions, allowing users to provide weak supervision at multiple levels of granularity. For example, suppose our goal is to train a machine learning model to perform fine-grained classification of financial news articles. By breaking this down into a hierarchy of tasks—e.g., company vs. money market articles, then sub-classes for

DEEM'18, June 15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-5828-6/18/06…$15.00
https://doi.org/10.1145/3209889.3209898

each—we potentially make it easier to use available weak supervision at various different levels of granularity. We may, for instance, be able to use existing classifiers or other noisy labels for the 'company vs. money market' task, and then rely on pattern-matching heuristics to provide labels for the finer-grained sub-tasks. This collection of multi-task weak supervision signals can then be used to train a multi-task end model that will generalize beyond the weak supervision sources.

In the MeTaL pipeline (Figure 1), users first specify a hierarchical *task schema* and a set of *labeling functions* (LFs) for each task, which collectively encode weak supervision signals at various levels of granularity. We then use a hierarchical generative *label model* to learn the accuracies of these LFs, and use their re-weighted and combined outputs as a set of probabilistic training labels for a multi-task discriminative *end model*, which is dynamically compiled based on the type of input and structure of sub-tasks.

We start by briefly reviewing related work in Section 2. Then, we describe the two main components of the MeTaL pipeline: first, in Section 3, the hierarchical *label model* used to learn the accuracies of the user-defined labeling functions; and then, in Section 4, the multi-task *end model* trained with the weak supervision. Finally, Section 5 details the results of applying MeTaL to two fine-grained classification problems—a financial news article classification task and a radiology report triage task—where we report average gains of 11.2 points of accuracy over a baseline approach and of 9.5 points of accuracy over the weak supervision heuristics used.

2 RELATED WORK

In *multi-task learning (MTL)* [2], separate machine learning models for multiple tasks share subsets of their parameters, or are regularized to have similar parameters, with the aim of sharing inductive bias between tasks. A body of recent work has explored multi-task learning for modern deep learning models [15]. In particular, when hierarchy may be present amongst tasks, recent work has explored sharing intermediate layers according to these hierarchies [5, 16].

Modern supervised learning models often have large numbers of trainable parameters and thus require large labeled training sets; the idea of *weak supervision* approaches is to use higher-level and/or noisier signals instead of hand-labeled data [7, 8, 17]. Recent approaches model the accuracies of weak sources without labeled data [1, 12, 14], and propose end-to-end systems for weak supervision based on these techniques [13]. Other recent work considers learning classifier accuracies over logically-related tasks without labeled data [3, 11]. Here, we propose a system at the confluence of both trends.

3 HIERARCHICAL WEAK SUPERVISION

In the MeTaL pipeline, users start by specifying a *task schema*, which describes the hierarchical relationships between the set of tasks (and which in future work could potentially be learned). Our ultimate goal is to train a standard multi-task network–however we propose to supervise it in a novel way. Rather than hand-labeling individual data points, users provide two inputs for each task: unlabeled data, and a set of LFs, which are simply functions that, given a data point, either output a label or abstain [12]. For example, LFs could encode pattern-based heuristics, distant supervision from existing knowledge bases, crowdsourced labels, existing weak classifiers, or

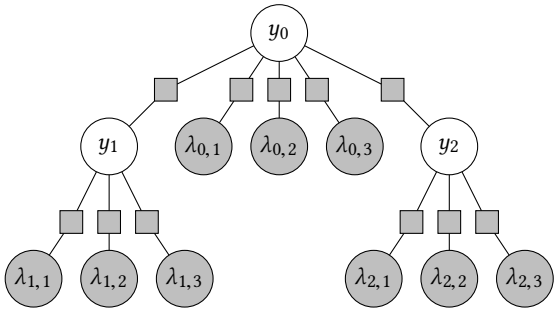

Figure 2: An example hierarchical generative label model, where a four-class classification problem is broken into three hierarchical sub-tasks y_0, y_1, y_2, each supervised by three LFs. An assignment of *task labels* $(y_0 = 2, y_1 = \emptyset, y_2 = 1)$ would correspond to an *end label* $y = 3$.

arbitrary domain heuristics [13]. LFs can have arbitrary, unknown accuracies—but by looking at the matrix of their agreements, we can estimate their accuracies without labeled data [12]. For MTL, we want to leverage LFs at multiple levels of granularity, outputting labels for different but hierarchically related tasks.

To handle this novel setting, we represent the outputs of the labeling functions via a hierarchical generative model. To simplify our exposition, we focus on the setting where the task schema represents a fine-grained classification problem with *end labels* $y \in \{1, \ldots, K\}$, which has been decomposed into a decision tree over sub-tasks $y_0, \ldots, y_{T-1}$, each with *task labels* $y_t \in \{1, \ldots, K_t\}$. For non-root tasks, we also include a "null" task label $\emptyset$, indicating that the task is not relevant for the data point.

Example 3.1. Suppose our task is to classify financial news articles (see Section 5) with end labels $y \in$ {Corporate Performance, Corporate Funding, Money Markets, Commodity Markets}, which we break down into three sub-tasks,

- $y_0 \in$ {*Corporate*, *Markets*}
- $y_1 \in$ {*Performance*, *Funding*, $\emptyset$}
- $y_2 \in$ {*Money*, *Commodity*, $\emptyset$},

related by a task schema $\{(0,1),(0,2)\}$ indicating that tasks 1 and 2 are both children of task 0. Here, end label $y =$ "Money Markets" would correspond to $y_0 =$ "Markets", $y_1 = \emptyset$, $y_2 =$ "Money".

For each task y_t, we consider having m_t labeling functions $\lambda_{t,1}, \ldots, \lambda_{t,m_t}, \lambda_{t,i} : \mathcal{X} \mapsto \{0\} \cup \{1, \ldots, k_t\}$, where 0 represents an abstention. Additionally, let $M = \sum_t m_t$. We assume these labeling functions are conditionally independent, i.e. $\lambda_{t,i} \perp \lambda_{t,j \neq i} \mid y_t$. We then define the generative label model distribution as:

$$p_\theta(\lambda, y) = \frac{1}{Z_\theta} \exp\left(\theta^T h(\lambda, y)\right)$$

where $Z_\theta = \sum_{y', \lambda'} \exp\left(\theta^T h(\lambda', y')\right)$ is the partition function and h is a vector of sufficient statistics. For each task t and every labeling function $\lambda_{t,j}$, we define two types of sufficient statistics,

$$h^{lab}_{t,j}(\lambda, y) = \mathbb{1}\{y_t \neq \emptyset\}\mathbb{1}\{\lambda_{t,j} \neq 0\}$$
$$h^{acc}_{t,j}(\lambda, y) = h^{lab}_{t,j}(\lambda, y)\left(\mathbb{1}\{\lambda_{t,j} = y_t\} - \mathbb{1}\{\lambda_{t,j} \neq y_t\}\right)$$

with corresponding parameters $\theta^{lab}_{t,j}$ and $\theta^{acc}_{t,j}$, representing the *labeling propensity* and *accuracy* (resp.) of $\lambda_{t,j}$ *when* $y_t \neq \emptyset$. Finally,

we assume that the end labels $1, \ldots, K$ are mutually exclusive via hard constraints. Our goal now is to learn a model $\hat{\theta}$ using *only unlabeled data*, $U = \{x_1, \ldots, x_N\}$, which best explains the observed labels. Intuitively, we can learn the parameters of the labeling functions from observing their overlapping agreements and disagreements. Concretely, we minimize the empirical negative log marginal likelihood, given the observed $M \times N$ label matrix $\bar{\lambda}$ (see [12]):

$$L(\theta, \bar{\lambda}) = -\sum_{x_i \in U} \log \sum_{y} p_\theta(\bar{\lambda}_i, y)$$

The gradient of our objective is:

$$\nabla_\theta L(\theta, \bar{\lambda}) = \sum_{x_i \in U} \left(\mathbb{E}_{\lambda, y \sim p_\theta} \left[h(\lambda, y) \right] - \mathbb{E}_{y \sim p_\theta(\cdot \mid \bar{\lambda}_i)} \left[h(\bar{\lambda}_i, y) \right] \right)$$

We compute the gradient update in closed form, using belief propagation for the second term on the RHS, which given our tree structure gives the exact computation in one pass up and down the tree. The predictions of the learned label model are then used as probabilistic training labels in the next step.

4 WEAKLY-SUPERVISED MTL

Current state-of-the-art MTL models are painstakingly configured by hand for each new configuration of tasks [5, 15, 16]. In MeTaL, a multi-task deep neural network (the *end model*) is automatically configured in PyTorch [10] based on the provided task schema, using the following three configurable building blocks:

- *Input Module:* To support multiple types of input data, MeTaL's end model accepts a plug-in input module of arbitrary complexity with parameters jointly learned at training time, which maps from a raw data point x to a vector of dimension D_0.
- *Intermediate Module:* Given a hierarchy of tasks with depth d, MeTaL constructs a corresponding hierarchy of d intermediate modules. By default, we use linear layers (i.e. $f^{int}_{W,b}(x) = Wx + b$) but these are easily replaced by other more complex modules.
- *Task Heads:* As is standard in modern MTL network designs, each task y_t has a separate layer attached to the shared layers. By default, we use a linear layer for each task.

Once the network is constructed, it is then trained using the *probabilistic* training labels output by the label model in Section 3.

In MeTaL, we consider a novel, dynamically-configured architecture (HierMTL) whereby task heads are attached to intermediate layers corresponding to their depth in the supervision hierarchy, and then outputs of child task heads are passed up as additional inputs to their parents (Figure 3). We also consider two simpler architectures, both sharing the same input and intermediate layer architecture: (i) a non-MTL network (SingleTask) where a final layer simply predicts the end label y; and (ii) a simpler MTL network (FlatMTL) where each of the same task heads is attached at the same intermediate layer. We empirically evaluate the different architectures in Section 5.

5 EXPERIMENTS

We now seek to empirically validate the design of Snorkel MeTaL in the context of fine-grained classification problems. We assess how end task predictive performance is affected by:

(1) Accepting weak supervision at multiple levels of granularity;

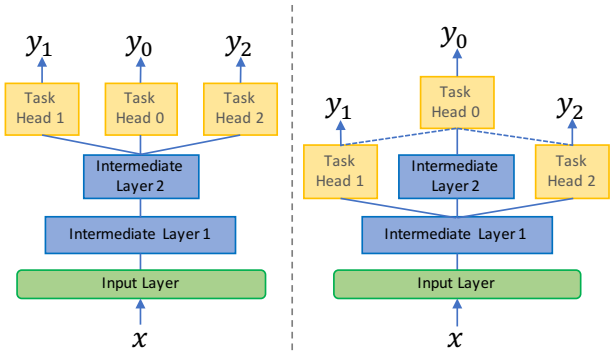

Figure 3: The two multi-task network architectures, for an example three-task problem with root task y_0 and child tasks y_1, y_2. Boxes are linear layers, and lines are nonlinearity connections. In the left panel (FlatMTL), the *task heads* are all connected to the topmost intermediate layer. In the right panel (HierMTL), the task heads are connected to intermediate layers corresponding to their hierarchical positions, with child outputs passed as input to parent task heads.

(2) Using our weakly-supervised end model (i.e. the full MeTaL pipeline), versus just using the predictions of the label model;
(3) Learning and modeling the accuracies of the labeling functions;
(4) Using the HierMTL end model versus FlatMTL or SingleTask.

Labeled Development Set. In our experiments, we assume access to a small (50 data points) hand-labeled *development set*, which is used as a guide during LF development, and is then used for label model and end model cross-validation. We also use a randomly sampled hand-labeled *test set* for evaluation, as is standard.

Datasets. To make these assessments, we perform a series of experiments on two different classification tasks that have been formulated in this hierarchical multi-task setting. The first is a news classification task based on the Reuters Corpus Volume I (`RCV1`) dataset [6], described in Example 3.1. For our experiments we used an unlabeled training set of 1000 articles.

The second task is triaging radiology reports from the OpenI biomedical image repository (`OpenI`) [9] as one of four end labels, $y \in \{\text{Normal, Non-urgent, Urgent, Emergent}\}$, as determined by an expert radiologist, broken into a hierarchy of three sub-tasks: $y_0 \in \{\text{Acute, Non-acute}\}$ (root task), $y_1 \in \{\text{Urgent, Emergent}\}$ (acute leaf task), $y_2 \in \{\text{Normal, Non-urgent}\}$ (non-acute leaf task). For our experiments we used an unlabeled training set of 2630 reports.

Protocol. All experiments reported here use a simple word count featurizer, an identity input layer, and intermediate fully-connected linear layers with respective sizes of 250 and 50 and ReLU nonlinearities (see Figure 3). All reported numbers are averaged over 20 trials with identical settings other than random seeds.

Initial Results: Multi-Level Weak Supervision. We first confirm a basic premise motivating our system design: that using weak supervision (i.e. labeling functions) at multiple levels of granularity indeed provides a boost in accuracy. Since both of our experiments have two levels, this amounts to testing whether adding the root node weak supervision helps. Indeed, we observe an average increase in accuracy of 16.3 points by utilizing this signal. For the rest of the experiments, we assume all levels of weak supervision are used.

	Dev. Labels	Label Model	End Model
RCV1	75.4 ± 0.59	60.5 ± 0.02	**78.4** ± 1.39
OpenI	54.6 ± 6.28	72.9 ± 0.01	**74.0** ± 0.46
Average	65.0	66.7	**76.2**

Table 1: Accuracy of the end label predictions for the end model (i.e. full MeTaL pipeline), label model, and a simple baseline using the the development set labels as supervision.

Main Results: Full Pipeline Performance. Next, we consider three basic evaluations (Table 1):

(1) *Dev. Labels:* As a simple baseline, we consider training the end model (HierMTL) using the 50 hand-labeled data points in the development set
(2) *Label Model:* We use the probabilistic labels (i.e. the predictions) of the trained label model as our final predictions.
(3) *End Model:* The full MeTaL pipeline- we use the probabilistic labels from the label model to train the end MTL model (HierMTL), and evaluate its predictions

We see first of all that the end model—i.e., the full MeTaL pipeline—improves over the label model by an average 9.5 points of accuracy, showing its ability to effectively generalize beyond the labeling functions. Second, we see that the end model also outperforms the simple baseline (Dev. Labels) by an average 11.2 points in accuracy. We note, however, that RCV1 is fairly simple task, and therefore the performance gain over the tiny labeled development set is smaller than we might expect; in future work we plan to evaluate MeTaL on more challenging tasks, with more complex model architectures and parameter spaces that require far larger training sets.

	Hard MV	Soft MV	Trained Label Model
RCV1	56.4 ± 0.02	59.7 ± 0.02	**60.5** ± 0.02
OpenI	70.8 ± 0.06	72.3 ± 0.05	**72.9** ± 0.01
Average	63.6	66.0	**66.7**

Table 2: Label model ablation results, using hard majority vote, soft majority vote, and the full trained label model.

Label Model Ablation. In Table 2, we investigate the effect of learning and modeling the labeling function accuracies (Section 3). We consider three approaches, in order of increasing sophistication:

(1) *Hard MV:* Starting at the root task, we take the majority vote of the labeling functions at that task, and then proceed down the tree, breaking ties randomly;
(2) *Soft MV:* We use the predictions of the label model in Section 3, with all labeling function weights set at a fixed initial value (i.e. without training the model);
(3) *Trained Label Model:* We use the trained label model of Section 3.

End Model Ablation. In Table 3, we compare the results of the configurations of the end model described in Section 4. Note that we observe the best performance on each task in the HierMTL configuration. These results suggest that performance gains from multi-task learning can be driven not only by the usual separation of task representations, but also by ensuring that these separate task representations are connected in an alignment that directly mirrors the structure of the supervision input.

	SingleTask	FlatMTL	HierMTL
RCV1	76.3 ± 0.87	76.1 ± 1.75	**78.4** ± 1.39
OpenI	72.6 ± 0.57	73.1 ± 0.76	**74.0** ± 0.46
Average	74.5	74.6	**76.2**

Table 3: End model ablation results.

6 CONCLUSION

In this paper, we envision an extension of current "Software 2.0" trends in which *multi-task* pipelines are deployed as a single machine learning model. To support this, we propose Snorkel MeTaL, a prototype end-to-end system for weakly supervising multi-task networks compiled user task schemas and labeling functions. Empirical results show preliminary evidence that our approach provides performance gains. In future work, we plan to apply Snorkel MeTaL to more complex tasks and network architectures, to explore task schema structures beyond hierarchies, to better provide theoretical support, and to explore learned task schemas.

Acknowledgements. We gratefully acknowledge the support of DARPA under No. FA87501720095, NIH under No. U54EB020405, ONR under No. N000141712266, the Moore Foundation, Okawa Research Grant, American Family Insurance, Accenture, Toshiba, the Secure Internet of Things Project, Google, VMware, Qualcomm, Ericsson, Analog Devices, the NSF Graduate Research Fellowship under Grant No. DGE-114747, the Stanford Interdisciplinary Graduate and Bio-X fellowships, the Intelligence Community Postdoctoral Fellowship, and members of the Stanford DAWN project: Intel, Microsoft, Teradata, and VMware. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, DOE, NIH, ONR, or the U.S. Government.

REFERENCES

[1] S. H. Bach, B. He, A. J. Ratner, and C. Ré. Learning the structure of generative models without labeled data. In *ICML*, 2017.
[2] R. Caruana. Multitask learning: A knowledge-based source of inductive bias. In *ICML*, pages 41–48, 1993.
[3] T. Mitchell et. al. Never-ending learning. In *AAAI*, 2015.
[4] A. Karpathy. Software 2.0. medium.com/@karpathy/software-2-0-a64152b37c35.
[5] H. Kazuma, X. Caiming, T. Yoshimasa, and R. Socher. A joint many-task model: Growing a neural network for multiple NLP tasks. *CoRR*, abs/1611.01587, 2016.
[6] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5(Apr):361–397, 2004.
[7] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc ACL*, pages 1003–1011, 2009.
[8] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *ICML*, pages 567–574, 2012.
[9] National Institutes of Health. Open-i. 2017. URL https://openi.nlm.nih.gov/.
[10] A. et. al. Paszke. Automatic differentiation in pytorch. In *NIPS-W*, 2017.
[11] E. Platanios, H. Poon, T. M. Mitchell, and E. J. Horvitz. Estimating accuracy from unlabeled data: A probabilistic logic approach. In *NIPS*, pages 4364–4373, 2017.
[12] A.J. Ratner, C.M. De Sa, S. Wu, D. Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Adv Neural Inf Process Syst*, pages 3567–3575, 2016.
[13] A.J. Ratner, S.H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. In *VLDB*, 2018.
[14] T. Rekatsinas, M. Joglekar, H. Garcia-Molina, A. Parameswaran, and C. Ré. Slimfast: Guaranteed results for data fusion and source reliability. In *SIGMOD*, 2017.
[15] S. Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.
[16] A. Søgaard and Y. Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proc ACL*, volume 2, pages 231–235, 2016.
[17] T. Xiao, T. Xia, Y. Yang, C. Huang, and X. Wang. Learning from massive noisy labeled data for image classification. In *CVPR*, pages 2691–2699, 2015.