# FIDELITY-WEIGHTED LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

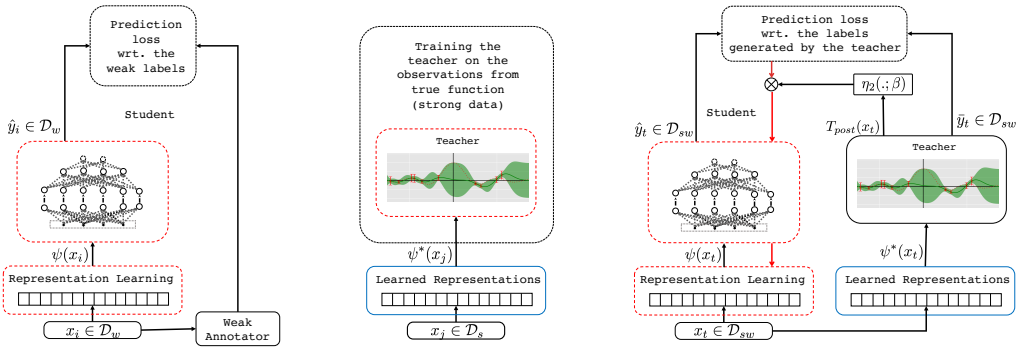
Training deep neural networks requires many training samples, but in practice training labels are expensive to obtain and may be of varying quality, as some may be from trusted expert labelers while others might be from heuristics or other sources of weak supervision such as crowd-sourcing. This creates a fundamental quality-versus-quantity trade-off in the learning process. Do we learn from the small amount of high-quality data or the potentially large amount of weakly-labeled data? We argue that if the learner could somehow know and take the label-quality into account when learning the data representation, we could get the best of both worlds. To this end, we propose "fidelity-weighted learning" (FWL), a semi-supervised student-teacher approach for training deep neural networks using weakly-labeled data. FWL modulates the parameter updates to a *student* network (trained on the task we care about) on a per-sample basis according to the posterior confidence of its label-quality estimated by a *teacher* (who has access to the high-quality labels). Both student and teacher are learned from the data. We evaluate FWL on two tasks in information retrieval and natural language processing where we outperform state-of-the-art alternative semi-supervised methods, indicating that our approach makes better use of strong and weak labels, and leads to better task-dependent data representations.

## 1 INTRODUCTION

The success of deep neural networks to date depends strongly on the availability of labeled data which is costly and not always easy to obtain. Usually it is much easier to obtain small quantities of high-quality labeled data and large quantities of unlabeled data. The problem of how to best integrate these two different sources of information during training is an active pursuit in the field of semi-supervised learning (Chapelle et al., 2006). However, for a large class of tasks it is also easy to define one or more so-called "weak annotators", additional (albeit noisy) sources of *weak supervision* based on heuristics or "weaker", biased classifiers trained on e.g. non-expert crowd-sourced data or data from different domains that are related. While easy and cheap to generate, it is not immediately clear if and how these additional weakly-labeled data can be used to train a stronger classifier for the task we care about. More generally, in almost all practical applications machine learning systems have to deal with data samples of variable quality. For example, in a large dataset of images only a small fraction of samples may be labeled by experts and the rest may be crowd-sourced using e.g. Amazon Mechanical Turk (Veit et al., 2017). In addition, in some applications, labels are intentionally perturbed due to privacy issues (Wainwright et al., 2012; Papernot et al., 2016).

Assuming we can obtain a large set of weakly-labeled data in addition to a much smaller training set of "strong" labels, the simplest approach is to expand the training set by including the weakly-supervised samples (all samples are equal). Alternatively, one may pretrain on the weak data and then fine-tune on observations from the true function or distribution (which we call strong data). Indeed, it has recently been shown that a small amount of expert-labeled data can be augmented in such a way by a large set of raw data, with labels coming from a heuristic function, to train a more accurate neural ranking model (Dehghani et al., 2017). The downside is that such approaches are oblivious to the amount or source of noise in the labels.

In this paper, we argue that treating weakly-labeled samples uniformly (i.e. each weak sample contributes equally to the final classifier) ignores potentially valuable information of the label quality. Instead, we propose Fidelity-Weighted Learning (FWL), a Bayesian semi-supervised approach that leverages a small amount of data with true labels to generate a larger training set with *confidence-weighted weakly-labeled samples*, which can then be used to modulate the fine-tuning process based on the fidelity (quality) of each weak sample. By directly modeling the inaccuracies




(a) Step 1 (b) Step 2 (c) Step 3

Figure 1: Illustration of Fidelity-Weighted Learning: Step 1: Pre-train student on weak data, Step 2: Fit teacher to observations from the true function, and Step 3: Fine-tune student on labels generated by teacher, taking the confidence into account. Red dotted borders and blue solid borders depict components with trainable and non-trainable parameters, respectively.

introduced by the weak annotator in this way, we can control the extent to which we make use of this additional source of weak supervision: more for confidently-labeled weak samples close to the true observed data, and less for uncertain samples further away from the observed data.

We propose a setting consisting of two main modules. One is called the student and is in charge of learning a suitable data representation and performing the main prediction task, the other is the teacher which modulates the learning process by modeling the inaccuracies in the labels. We explain our approach in much more detail in Section 2, but at a high level it works as follows (see Figure 1): We pretrain the student network on weak data to learn an initial task-dependent data representation which we pass to the teacher along with the strong data. The teacher then learns to predict the strong data, but crucially, *based on the student's learned representation*. This then allows the teacher to generate new labeled training data from unlabeled data, and in the process correct the student's mistakes, leading to a better final data representation and better final predictor.

We introduce the proposed FWL approach in more detail in Section 2. We then present our experimental setup in Section 3 where we evaluate FWL on a toy task and two real-world tasks, namely document ranking and sentence sentiment classification. In both latter cases, FWL outperforms competitive baselines and yields state-of-the-art results, indicating that FWL makes better use of the limited true labeled data and is thereby able to learn a better and more meaningful task-specific representation of the data. One may also view FWL from the perspective of Vapnik's learning with privileged information (LUPI) framework (Vapnik & Izmailov, 2015), for which we provide empirical support in Section 4.2. We discuss this connection in more detail in Appendix A.

## 2 Fidelity-Weighted Learning (FWL)

We now describe FWL, our proposed approach for semi-supervised learning when we have access to weak supervision (e.g. heuristics or weak annotators). We assume we are given a large set of unlabeled data samples, a heuristic labeling function called the *weak annotator*, and a small set of high-quality samples labeled by experts, called the *strong dataset*, consisting of tuples of training samples $x_i$ and their true labels $y_i$, i.e. $\mathcal{D}_s = \{(x_i, y_i)\}$. We consider the latter to be observations from the true target function that we are trying to learn. We use the weak annotator to generate labels for the unlabeled samples. Generated labels are noisy due to the limited accuracy of the weak annotator. This gives us the *weak dataset* consisting of tuples of training samples $x_i$ and their weak labels $\tilde{y}_i$, i.e. $\mathcal{D}_w = \{(x_i, \tilde{y}_i)\}$. Note that we can generate a large amount of weak training data $\mathcal{D}_w$ at almost no cost using the weak annotator. In contrast, we have only a limited amount of observations from the true function, i.e. $|\mathcal{D}_s| \ll |\mathcal{D}_w|$.

Our proposed setup comprises a neural network called the **student** and a Bayesian function approximator called the **teacher**. The training process consists of three phases which we summarize in Algorithm 1 and Figure 1.

**Step 1** *Pre-train the student on $\mathcal{D}_w$ using weak labels generated with the weak annotator.*

The goal of this stage is to learn a reasonably good representation of the data for the given task. The student function is a neural network consisting of two parts. The first part $\psi(.)$ learns the data representation and the second part $\phi(.)$ performs the prediction task (e.g. classification). Therefore

**Algorithm 1** Fidelity-Weighted Learning.

1: Train the student on samples from the weakly-annotated data $D_w$.
2: Freeze the representation-learning component $\psi(.)$ of the student and train teacher on the strong data $D_s = (\psi(x_j),y_j)$. Apply teacher to unlabeled samples $x_t$ to obtain soft dataset $D_{sw} = (x_t,\bar{y}_t,x_t^*)$ where $\bar{y}_t = T(x_t)$ is the soft label and $\Sigma(x_t)$ is the uncertainty provided by the teacher.
3: Train the student on samples from $D_{sw}$ with SGD and modulate the step-size $\eta_t$ according to the per-sample quality estimated using the teacher (Equation 1).

the overall function is $\hat{y} = \phi(\psi(x_i))$. The student is trained on all samples of the weak dataset $\mathcal{D}_w = \{(x_i,\tilde{y}_i)\}$. For brevity, in the following, we will refer to both data sample $x_i$ and its representation $\psi(x_i)$ by $x_i$ when it is obvious from the context.

**Step 2** *Train the teacher on the strong data $(\psi(x_j), y_j) \in \mathcal{D}_s$ represented in terms of the student representation $\psi(.)$ and then use the teacher to generate a soft dataset $\mathcal{D}_{sw}$ consisting of $\langle$sample,predicted label,confidence$\rangle$ pairs for **all** data samples.*

We use a Gaussian process as teacher to capture the label uncertainty in terms of the student representation, estimated w.r.t the strong data. We explain the finer details of the $\mathcal{GP}$ in Appendix D, and just present the overall description here. A prior mean and co-variance function is chosen for $\mathcal{GP}$. The learned embedding function $\psi(\cdot)$ in Step 1 is then used to map the data samples to dense vectors as input to the $\mathcal{GP}$. The $\mathcal{GP}$ is trained on this representation of the strong dataset to learn the posterior mean $\boldsymbol{m}_{\text{post}}$ (used to generate soft labels) and posterior co-variance $K_{\text{post}}(.,.)$ (which represents label uncertainty). We then create the *soft dataset* $\mathcal{D}_{sw} = \{(x_t,\bar{y}_t)\}$ using the posterior $\mathcal{GP}$, input samples $x_t$ from $\mathcal{D}_w \cup \mathcal{D}_s$, and predicted labels $\bar{y}_t$ with their associated uncertainties as computed by $T(x_t)$ and $\Sigma(x_t)$:

$$\begin{aligned} T(x_t) &= g(\boldsymbol{m}_{\text{post}}(x_t)) \\ \Sigma(x_t) &= h(K_{\text{post}}(x_t,x_t)) \end{aligned}$$

The generated labels are called *soft labels*. Therefore, we refer to $\mathcal{D}_{sw}$ as a soft dataset. $g(.)$ transforms the output of $\mathcal{GP}$ to the suitable output space. For example in classification tasks, $g(.)$ would be the softmax function to produce probabilities that sum up to one. For multidimensional-output tasks where a vector of variances is provided by the $\mathcal{GP}$, the vector $K_{\text{post}}(x_t,x_t)$ is passed through an aggregating function $h(.)$ to generate a scalar value for the uncertainty of each sample. Note that we train $\mathcal{GP}$ only on the strong dataset $\mathcal{D}_s$ but then use it to generate soft labels $\bar{y}_t = T(x_t)$ and uncertainty $\Sigma(x_t)$ for samples belonging to $\mathcal{D}_{sw} = \mathcal{D}_w \cup \mathcal{D}_s$.

In practice, we furthermore divide the space of data into several regions and assign each region a separate $\mathcal{GP}$ trained on samples from that region. By this division of space, we take advantage of the knowledge learned by several teachers, each an expert on its specific region of data space. As a nice side-effect, this also solves the scalability issues of $\mathcal{GP}$s in that we can increase the number of regions until the number of points in each region is tractable with a single $\mathcal{GP}$, and train these models in parallel. See Algorithm 2 in Appendix B for the detailed description.

**Step 3** *Fine-tune the weights of the student network on the soft dataset, while modulating the magnitude of each parameter update by the corresponding teacher-confidence in its label.*

The student network of Step 1 is fine-tuned using samples from the soft dataset $\mathcal{D}_{sw} = \{(x_t,\bar{y}_t)\}$ where $\bar{y}_t = T(x_t)$. The corresponding uncertainty $\Sigma(x_t)$ of each sample is mapped to a confidence value according to Equation 1 below, and this is then used to determine the step size for each iteration of SGD. So, intuitively, for data points where we have true labels, the uncertainty of the teacher is almost zero, which means we have high confidence and a large step-size for updating the parameters. However, for data points where the teacher is not confident, we down-weight the training steps of the student. This means that at these points, we keep the student function as it was trained on the weak data in Step 1.

More specifically, we update the parameters of the student by training on $\mathcal{D}_{sw}$ using stochastic gradient descent (SGD):

$$\begin{aligned} \boldsymbol{w}^* &= \underset{\boldsymbol{w}\in\mathcal{W}}{\operatorname{argmin}} \frac{1}{N} \sum_{(x_t,\bar{y}_t)\in\mathcal{D}_{sw}} l(\boldsymbol{w},x_t,\bar{y}_t) + \mathcal{R}(\boldsymbol{w}), \\ \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta_t(\nabla l(\boldsymbol{w},x_t,\bar{y}_t) + \nabla\mathcal{R}(\boldsymbol{w})) \end{aligned}$$

where $l(\cdot)$ is the per-example loss, $\eta_t$ is the total learning rate, $N$ is the size of the soft dataset $\mathcal{D}_{sw}$, $\boldsymbol{w}$ is the parameters of the student network, and $\mathcal{R}(.)$ is the regularization term.

We define the total learning rate as $\eta_t = \eta_1(t)\eta_2(x_t)$, where $\eta_1(t)$ is the usual learning rate of our chosen optimization algorithm that anneals over training iterations, and $\eta_2(x_t)$ is a function of the label uncertainty $\Sigma(x_t)$ that is computed by the teacher for each data point. Multiplying these two terms gives us the total learning rate. In other words, $\eta_2$ represents the *fidelity* (quality) of the current sample, and is used to multiplicatively modulate $\eta_1$. Note that the first term does not necessarily depend on each data point, whereas the second term does. We propose

$$\eta_2(x_t) = \exp[-\beta\Sigma(x_t)], \tag{1}$$

to exponentially decrease the learning rate for data point $x_t$ if its corresponding soft label $\bar{y}_t$ is unreliable (far from a true sample). In Equation 1, $\beta$ is a positive scalar hyper-parameter. Intuitively, small $\beta$ results in a student which listens more carefully to the teacher and copies its knowledge, while a large $\beta$ makes the student pay less attention to the teacher, staying with its initial weak knowledge. More concretely speaking, as $\beta \to 0$ student places more trust in the labels $\bar{y}_t$ estimated by the teacher and the student copies the knowledge of the teacher. On the other hand, as $\beta \to \infty$, student puts less weight on the extrapolation ability of $\mathcal{GP}$ and the parameters of the student are not affected by the correcting information from the teacher.

## 3 EXPERIMENTS

The neural networks in the proposed method are implemented in TensorFlow (Abadi et al., 2015; Tang, 2016). GPflow (Matthews et al., 2017) is employed for developing the $\mathcal{GP}$ modules. For both tasks, we evaluate the performance of our method compared to the following baselines:

1. **WA**. The weak annotator, i.e. the unsupervised method used for annotating the unlabeled data.
2. $\mathbf{NN_W}$. The student trained only on weak data.
3. $\mathbf{NN_S}$. The student trained only on strong data.
4. $\mathbf{NN_{S+/W}}$. The student trained on samples that are alternately drawn from $\mathcal{D}_w$ without replacement, and $\mathcal{D}_s$ with replacement. Since $|\mathcal{D}_s| \ll |\mathcal{D}_w|$, it oversamples the strong data.
5. $\mathbf{NN_{W\to S}}$. The student trained on weak dataset $\mathcal{D}_w$ and fine-tuned on strong dataset $\mathcal{D}_s$.
6. $\mathbf{NN_{W^\omega\to S}}$. The student trained on the weak data, but the step-size of each weak sample is weighted by a fixed value $0 \leq \omega \leq 1$, and fine-tuned on strong data. As an approximation for the optimal value for $\omega$, we have used the mean of $\eta_2$ of our model (below).
7. **FWL** $\backslash\Sigma$. The student trained on the weakly labeled data and fine-tuned on examples labeled by teacher without taking the confidence into account. This baseline is similar to (Veit et al., 2017).
8. **FWL**. Our FWL model, i.e. the student trained on the weakly labeled data and fine-tuned on examples labeled by teacher using the confidence scores.

In the following, we apply FWL first to a toy problem and then to two different real tasks: *document ranking* and *sentiment classification*. We introduce each task and the results produced for it. The student network and teacher $\mathcal{GP}$ for each task is given in more detail in the appendix.

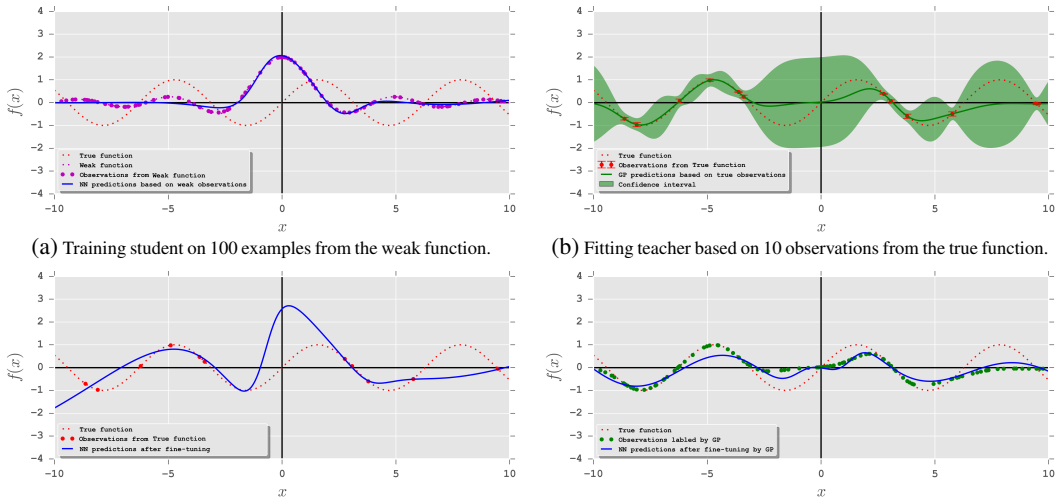
### 3.1 TOY PROBLEM

We first apply FWL to a one-dimensional toy problem to illustrate the various steps. Let $f_t(x) = \sin(x)$ be the true function (red dotted line in Figure 2a) from which a small set of observations $\mathcal{D}_s = \{x_j, y_j\}$ is provided (red points in Figure 2b). These observation might be noisy, in the same way that labels obtained from a human labeler could be noisy. A weak annotator function $f_w(x) = 2sinc(x)$ (magenta line in Figure 2a) is provided, as an approximation to $f_t(.)$.

The task is to obtain a good estimate of $f_t(.)$ given the set $\mathcal{D}_s$ of strong observations and the weak annotator function $f_w(.)$. We can easily obtain a large set of observations $\mathcal{D}_w = \{x_i, \tilde{y}_i\}$ from $f_w(.)$ with almost no cost (magenta points in Figure 2a).

We consider two experiments:

1. A neural network trained on weak data and then fine-tuned on strong data from the true function, which is the most common semi-supervised approach (Figure 2c).
2. A teacher-student framework working by the proposed FWL approach.




(a) Training student on 100 examples from the weak function.

(b) Fitting teacher based on 10 observations from the true function.

(c) Fine-tuning the student based on observations from the true function.

(d) Fine-tuning the student based on label/confidence from teacher.

Figure 2: Toy example: The true function we want to learn is $y = \sin(x)$ and the weak function is $y = 2sinc(x)$.

As can be seen in Figure 2d, FWL by taking into account label confidence, gives a better approximation of the true hidden function. We repeated the above experiment 10 times. The average RMSE with respect to the true function on a set of test points over those 10 experiments for the student, were as follows:

1. Student is trained on weak data (blue line in Figure 2a): $0.8406$,
2. Student is trained on weak data then fine tuned on true observations (blue line in Figure 2c): $0.5451$,
3. Student is trained on weak data, then fine tuned by soft labels and confidence information provided by the teacher (blue line in Figure 2d): $0.4143$ (best).

More details of the neural network and $\mathcal{GP}$ along with the specification of the data used in the above experiment are presented in Appendix D and F.1.

### 3.2 Document Ranking

This task is the core information retrieval problem and is challenging as the ranking model needs to learn a representation for long documents and capture the notion of relevance between queries and documents. Furthermore, the size of publicly available datasets with query-document relevance judgments is unfortunately quite small ($\sim 250$ queries). We employ a state-of-the-art pairwise neural ranker architecture as the student (Dehghani et al., 2017). In this model, ranking is cast as a regression task. Given each training sample $x$ as a triple of query $q$, and two documents $d^+$ and $d^-$, the goal is to learn a function $\mathcal{F}: \{<q, d^+, d^- >\} \rightarrow \mathbb{R}$, which maps each data sample $x$ to a scalar output value $y$ indicating the probability of $d^+$ being ranked higher than $d^-$ with respect to $q$.

**The student** follows the architecture proposed in (Dehghani et al., 2017). The first layer of the network, i.e. representation learning layer $\psi : \{< q, d^+, d^- >\} \rightarrow \mathbb{R}^m$ maps each input sample to an $m$- dimensional real-valued vector. In general, besides learning embeddings for words, function $\psi$ learns to compose word embedding based on their global importance in order to generate query/document embeddings. The representation layer is followed by a simple fully-connected feed-forward network with a sigmoidal output unit to predict the probability of ranking $d^+$ higher than $d^-$. The general schema of the student is illustrated in Figure 3. More details are provided in Appendix C.1.

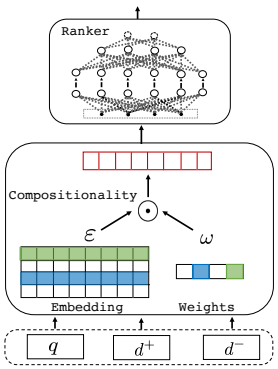



Figure 3: The student for the document ranking task.

**The teacher** is implemented by clustered $\mathcal{GP}$ algorithm. See Appendix D for more details.

**The weak annotator** is BM25 (Robertson & Zaragoza, 2009), a well-known unsupervised method for scoring query-document pairs based on statistics of the matched terms. More details are provided in Appendix E.1.

Table 1: Performance of the proposed FWL approach and baseline methods for ranking task. $^{\blacktriangle i}$ indicates that the improvements with respect to the baseline $i$ are statistically significant at the 0.05 level using the paired two-tailed t-test with Bonferroni correction.

| | Method | Robust04 | | ClueWeb | |
|---|---|---|---|---|---|
| | | MAP | nDCG@20 | MAP | nDCG@20 |
| 1 | $\text{WA}_{\text{BM25}}$ | $0.2503^{\blacktriangle 3}$ | $0.4102^{\blacktriangle 3}$ | $0.1021^{\blacktriangle 3}$ | $0.2070^{\blacktriangle 3}$ |
| 2 | $\text{NN}_{\text{W}}$ (Dehghani et al., 2017) | $0.2702^{\blacktriangle 13}$ | $0.4290^{\blacktriangle 13}$ | $0.1297^{\blacktriangle 13}$ | $0.2201^{\blacktriangle 13}$ |
| 3 | $\text{NN}_{\text{S}}$ | 0.1790 | 0.3519 | 0.0782 | 0.1730 |
| 4 | $\text{NN}_{\text{S+/W}}$ | $0.2763^{\blacktriangle 123}$ | $0.4330^{\blacktriangle 123}$ | $0.1354^{\blacktriangle 123}$ | $0.2319^{\blacktriangle 123}$ |
| 5 | $\text{NN}_{\text{W}\rightarrow\text{S}}$ | $0.2810^{\blacktriangle 123}$ | $0.4372^{\blacktriangle 123}$ | $0.1346^{\blacktriangle 123}$ | $0.2317^{\blacktriangle 123}$ |
| 6 | $\text{NN}_{\text{W}^{\omega}\rightarrow\text{S}}$ | $0.2899^{\blacktriangle 12345}$ | $0.4431^{\blacktriangle 12345}$ | $0.1320^{\blacktriangle 1234}$ | $0.2309^{\blacktriangle 1234}$ |
| 7 | FWL $\setminus\Sigma$ | $0.2980^{\blacktriangle 12345}$ | $0.4516^{\blacktriangle 12345}$ | $0.1386^{\blacktriangle 12345}$ | $0.2340^{\blacktriangle 12345}$ |
| 8 | FWL | $\mathbf{0.3124}^{\blacktriangle 1234567}$ | $\mathbf{0.4607}^{\blacktriangle 1234567}$ | $\mathbf{0.1472}^{\blacktriangle 1234567}$ | $\mathbf{0.2453}^{\blacktriangle 1234567}$ |

Description of the data with weak labels and data with true labels as well as setups of the document-ranking experiments is presented in Appendix F.2 in more details.

**Results and Discussions** We conducted k-fold cross validation on $\mathcal{D}_s$ (the strong data) and report two standard evaluation metrics for ranking: mean average precision (MAP) of the top-ranked $1000$ documents and normalized discounted cumulative gain calculated for the top $20$ retrieved documents (nDCG@20). Table 1 shows the performance on both datasets. As can be seen, FWL provides a significant boost on the performance over all datasets. In the ranking task, the student is designed in particular to be trained on weak annotations (Dehghani et al., 2017), hence training the network only on weak supervision, i.e. $\text{NN}_{\text{W}}$ performs better than $\text{NN}_{\text{S}}$. This can be due to the fact that ranking is a complex task requiring many training samples, while relatively few data with true labels are available.

Alternating between strong and weak data during training, i.e. $\text{NN}_{\text{S+/W}}$ seems to bring little (but statistically significant) improvement. However, we can gain better results by the typical fine-tuning strategy, $\text{NN}_{\text{W}\rightarrow\text{S}}$. We can gain improvement by fine-tuning the $\text{NN}_{\text{W}}$ using labels generated by teacher without considering their confidence score, i.e. FWL $\setminus\Sigma$. This means we just augmented the fine-tuning process by generating a fine-tuning set using teacher which is better than $\mathcal{D}_s$ in terms of quantity and $\mathcal{D}_w$ in terms of quality. This baseline is equivalent to setting $\beta = 0$ in Equation 1. However, we see a big jump in performance when we use FWL to include the estimated label quality from the teacher, leading to the best overall results.

### 3.3 Sentiment Classification

In sentiment classification, the goal is to predict the sentiment (e.g., positive, negative, or neutral) of a sentence. Each training sample $x$ consists of a sentence $s$ and its sentiment label $\tilde{y}$.

**The student** for the sentiment classification task is a convolutional model which has been shown to perform best on the dataset we used (Deriu et al., 2017; Severyn & Moschitti, 2015a;b; Deriu et al., 2016). The first layer of the network learns the function $\psi(.)$ which maps input sentence $s$ to a dense vector as its representation. The inputs are first passed through an embedding layer mapping the sentence to a matrix $S \in \mathbb{R}^{m \times |s|}$, followed by a series of 1d convolutional layers with max-pooling. The representation layer is followed by feed-forward layers and a softmax output layer which returns the probability distribution over all three classes. Figure 4 presents the general schema of the architecture of the student. See Appendix C.2 for more details.

**The teacher** for this task is modeled by a $\mathcal{GP}$. See Appendix D for more details.

**The weak annotator** is a simple unsupervised lexicon-based method (Hamdan et al., 2013; Kiritchenko et al., 2014), which estimate a distribution over sentiments for each sentence, based on sentiment labels of its terms. More details are provided in Appendix E.2.

Specification of the data with weak labels and data with true labels along with the detailed experimental setups are given in Appendix F.3.

**Results and Discussion** We report Macro-F1, the official SemEval metric, in Table 2. We see that the proposed FWL is the best performing among all the baselines.

For this task, since the amount of data with true labels are larger compared to the ranking task, the performance of $\text{NN}_{\text{S}}$ is acceptable. Alternately sampling from weak and strong data gives better

Table 2: Performance of the proposed FWL approach and baseline methods for sentiment classification task. $^{\blacktriangle i}$ indicates that the improvements with respect to the baseline#$i$ are statistically significant, at the 0.05 level using the paired two-tailed t-test, with Bonferroni correction.

| | Method | SemEval-14 | SemEval-15 |
|---|---|---|---|
| 1 | $\text{WA}_{\text{Lexicon}}$ | 0.5141 | 0.4471 |
| 2 | $\text{NN}_{\text{W}}$ | $0.6719^{\blacktriangle 13}$ | $0.5606^{\blacktriangle 1}$ |
| 3 | $\text{NN}_{\text{S}}$ | $0.6307^{\blacktriangle 1}$ | $0.5811^{\blacktriangle 12}$ |
| 4 | $\text{NN}_{\text{S+/W}}$ | $0.7032^{\blacktriangle 123}$ | $0.6319^{\blacktriangle 123}$ |
| 5 | $\text{NN}_{\text{W}\rightarrow\text{S}}$ | $0.7080^{\blacktriangle 123}$ | $0.6441^{\blacktriangle 123}$ |
| 6 | $\text{NN}_{\text{W}^{\omega}\rightarrow\text{S}}$ | $0.7166^{\blacktriangle 1234}$ | $0.6603^{\blacktriangle 12345}$ |
| 7 | $\text{FWL} \setminus \Sigma$ | $0.7202^{\blacktriangle 12345}$ | $0.6590^{\blacktriangle 12345}$ |
| 8 | FWL | $\mathbf{0.7470}^{\blacktriangle 1234567}$ | $\mathbf{0.6830}^{\blacktriangle 1234567}$ |
| 9 | $\text{SemEval}^{\text{Best}}$ | 0.7162 (Rouvier & Favre, 2016) | 0.6618 (Deriu et al., 2016) |

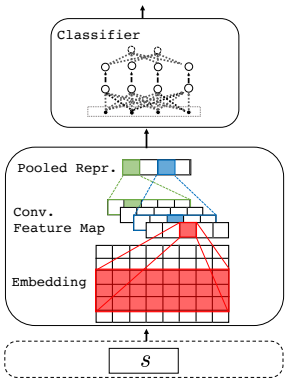



Figure 4: The student for the sentiment classification task.

results. Pretraining on weak labels then fine-tuning the network on true labels, further improves the performance. Weighting the gradient updates from weak labels during pretraining and fine-tuning the network with true labels, i.e. $\text{NN}_{\text{W}^{\omega}\rightarrow\text{S}}$ seems to work quite well in this task. Similar to the ranking task, fine-tuning $\text{NN}_{\text{S}}$ based on labels generated by $\mathcal{GP}$ instead of data with true labels, regardless of the confidence score, works better than standard fine-tuning.

Besides the baselines, we also report the best performing systems which are also convolution-based models (Rouvier & Favre 2016 on SemEval-14; Deriu et al. 2016 on SemEval-15). Using FWL and taking the confidence into consideration outperforms the best systems and leads to the highest reported results on both datasets.

# 4 Analysis

## 4.1 Handling the Bias-Variance Trade-off

As mentioned in Section 2, $\beta$ is a hyper-parameter that controls the contribution of weak and strong data to the training procedure. In order to investigate its influence, we fixed everything in the model and ran the fine-tuning stage with different values of $\beta \in \{0.0, 0.1, 1.0, 2.0, 5.0\}$ in all the experiments. Figure 5 illustrates the performance on the ranking (on Robust04 dataset) and sentiment classification tasks (on SemEval14 dataset). For both sentiment classification and ranking, $\beta = 1$ gives the best results. We also experimented on the toy problem with different values of $\beta$ in three cases: 1) having 10 observations from the true function (same setup as Section 3.1), marked as "Toy Data" in the plot, 2) having only 5 observations from the true function, marked as "Toy Data *" in the plot, and 3) having $f(x) = x + 1$ as the weak function, which is an extremely bad approximator of the true function, marked as "Toy Data **" in the plot. For the "Toy Data" experiment, $\beta = 1$ turned out to be optimal. However, for "Toy Data *", where we have an extremely small number of observations from the true function, setting $\beta$ to a higher value acts as a regularizer by relying more on weak signals, and eventually leads to better generalization. On the other hand, for "Toy Data **", where the quality of the weak annotator is extremely low, lower values of $\beta$ put more focus on the true observations. Therefore, $\beta$ lets us control the bias-variance trade-off in these extreme cases.

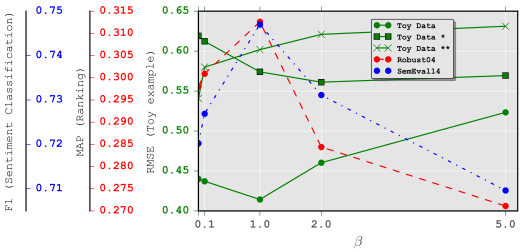
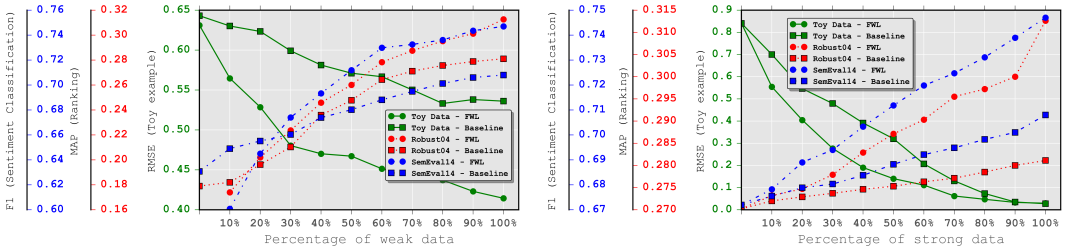



Figure 5: Effect of different values for $\beta$.

## 4.2 A Good Teacher is Better Than Many Observations

In this section, we study the rate of learning for the student as the amount of training data is varied. We performed two types of experiments for all tasks: In the first experiment, we use all the available strong data but consider different percentages of the entire weak dataset. In the second experiment, we fix




(a) Models trained on different amount weak data.

(b) Models trained on different amount of strong data.

Figure 6: Performance of FWL and the baseline model trained on different amount of data.

the amount of weak data and provide the model with varying amounts of strong data. We use standard fine-tuning with similar setups as for the baseline models[1].

Figure 6 presents the results of these experiments. In general, for all tasks and both setups, the student learns faster when there is a teacher. One caveat is in the case where we have a very small amount of weak data. In this case the student cannot learn a suitable representation in the first step, and hence the performance of FWL is pretty low, as expected. It is highly unlikely that this situation occurs in reality as obtaining weakly labeled data is much easier than strong data.

The empirical observation of Figure 6 that our model learns more with less data can also be seen as evidence in support of another perspective to FWL, called *learning using privileged information* (Vapnik & Izmailov, 2015). We elaborate more on this connection in Appendix A.

## 5 Related Work

Learning from imperfect labels has been thoroughly studied in the literature (Frénay & Verleysen, 2014). In the semi-supervised setup, some ideas were developed to utilize weakly or even unlabeled data. For instance, the idea of self-training (Rosenberg et al., 2005), pseudo-labeling (Lee, 2013; Hinton et al., 2015), and Co-training (Blum & Mitchell, 1998) are introduced for augmenting the training set by unlabeled data with predicted labels. As a common approach in semi-supervised learning, the unlabeled set can be used for learning the distribution of the data. In particular for neural networks, greedy layer-wise pre-training of weights using unlabeled data is followed by supervised fine-tuning (Hinton et al., 2006; Deriu et al., 2017; Severyn & Moschitti, 2015b;a; Go et al., 2009). Other methods learn unsupervised encoding at multiple levels of the architecture jointly with a supervised signal (Ororbia II et al., 2015; Weston et al., 2012).

On the other hand, some noise cleansing methods were proposed to remove or correct mislabeled samples (Brodley & Friedl, 1999). There are some studies showing that weak or noisy labels can be leveraged by employing a particular architecture or defining a proper loss function to avoid over-fitting to imperfections of the training data (Dehghani et al., 2017; Patrini et al., 2016; Beigman & Klebanov, 2009; Zeng et al., 2015; Bunescu & Mooney, 2007).

One direction of research focuses on modeling the pattern of the noise or weakness in the labels. For instance, methods that use a generative model to correct weak labels such that a discriminative model can be trained more effectively (Ratner et al., 2016; Rekatsinas et al., 2017; Varma et al., 2017). Furthermore, methods that aim to capture the pattern of the noise by inserting an extra layer or a separate module (Sukhbaatar et al., 2015; Veit et al., 2017) tries to infer better labels from noisy ones and use them to supervise the training of the network. Our proposed method can be categorized in this class.

## 6 Conclusion

Training neural networks using large amounts of weakly annotated data is an attractive approach in scenarios where an adequate amount of data with true labels is not available, a situation which often arises in practice. In this paper, we introduced fidelity-weighted learning (FWL), a new student-teacher framework for semi-supervised learning in the presence of weakly labeled data. We applied FWL to document ranking and sentiment classification, and empirically verified that FWL speeds up the training process and improves over state-of-the-art semi-supervised alternatives.

[1] Details on the experiments for the toy problem is provided in Appendix F.1

## REFERENCES


Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pp. 2200–2204, 2010.

Eyal Beigman and Beata Beigman Klebanov. Learning with annotation noise. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 280–287. Association for Computational Linguistics, 2009.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pp. 92–100, 1998.

Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

Razvan Bunescu and Raymond Mooney. Learning to extract relations from the web using minimal supervision. In *ACL*, 2007.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2006.

Gordon V. Cormack, Mark D. Smucker, and Charles L. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retr.*, 14(5):441–465, 2011.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *SIGIR'17*, 2017.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval*, pp. 1124–1128, 2016.

Jan Deriu, Aurelien Lucchi, Valeria De Luca, Aliaksei Severyn, Simon Müller, Mark Cieliebak, Thomas Hofmann, and Martin Jaggi. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th international International World Wide Web Conference (WWW'17)*, pp. 1045–1052, 2017.

Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15(1): 3873–3923, 2014.

Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

Hussam Hamdan, Frederic Béchet, and Patrice Bellot. Experiments with dbpedia, wordnet and sentiwordnet as resources for sentiment analysis in micro-blogging. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pp. 455–459, 2013.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Proceedings of AISTATS*, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, pp. 2, 2013.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, 2017.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS '13*, pp. 3111–3119, 2013.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Semeval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval*, pp. 1–18, 2016.

Alexander G Ororbia II, C Lee Giles, and David Reitter. Learning a deep hybrid model for semi-supervised text classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *InfoScale '06*, 2006.

Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. Making neural networks robust to label noise: a loss correction approach. *arXiv preprint arXiv:1609.03683*, 2016.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pp. 3567–3575, 2016.

Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. Holoclean: Holistic data repairs with probabilistic inference. *arXiv preprint arXiv:1702.00820*, 2017.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*, 2005.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pp. 451–463, 2015.

Mickael Rouvier and Benoit Favre. Sensei-lif at semeval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. *Proceedings of SemEval*, pp. 202–208, 2016.

Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 959–962. ACM, 2015a.

Aliaksei Severyn and Alessandro Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*, pp. 464–469, 2015b.

Yirong Shen, Matthias Seeger, and Andrew Y Ng. Fast gaussian process regression using kd-trees. In *Advances in neural information processing systems*, pp. 1225–1232, 2006.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. Training convolutional networks with noisy labels. In *Workshop contribution at ICLR 2015*, 2015.

Yuan Tang. Tf.learn: Tensorflow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251*, 2016.

Michalis K Titsias. Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Vladimir Vapnik and Rauf Izmailov. Learning using privileged information: similarity control and knowledge transfer. *Journal of machine learning research*, 16(20232049):55, 2015.

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5):544–557, 2009.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Paroma Varma, Bryan He, Dan Iter, Peng Xu, Rose Yu, Christopher De Sa, and Christopher Ré. Socratic learning: Correcting misspecified generative models using discriminative models. *arXiv preprint arXiv:1610.08123*, 2017.

Andreas Veit, Neil Alldrin, Gal Chechik, Ivan Krasin, Abhinav Gupta, and Serge Belongie. Learning from noisy large-scale datasets with minimal supervision. In *The Conference on Computer Vision and Pattern Recognition*, 2017.

Martin J Wainwright, Michael I Jordan, and John C Duchi. Privacy aware learning. In *Advances in Neural Information Processing Systems*, pp. 1430–1438, 2012.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pp. 639–655. Springer, 2012.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pp. 1753–1762, 2015.

# APPENDICES

## A CONNECTION WITH VAPNIK'S LEARNING USING PRIVILEGED INFORMATION

In this section we highlight the connections of our work with Vapnik's *learning using privileged information* (LUPI) (Vapnik & Vashist, 2009; Vapnik & Izmailov, 2015). FWL makes use of information from a small set of correctly labeled data to improve the performance of a semi-supervised learning algorithm. The main idea behind LUPI comes from the fact that humans learn much faster than machines. This can be due to the role that an *Intelligent Teacher* plays in human learning. In this framework, the training data is a collection of triplets

$$\{(x_1,y_1,x_1^*),...,(x_n,y_n,x_n^*)\}\sim P^n(x,y,x^*) \tag{2}$$

where each $(x_i,y_i)$ is a pair of feature-label and $x_i^*$ is the additional information provided by an intelligent teacher to ease the learning process for the student. Additional information for each $(x_i,y_i)$ is available only during training time and the learning machine must only rely on $x_i$ at test time. The theory of LUPI studies how to leverage such a teaching signal $x_i^*$ to outperform learning algorithms utilizing only the normal features $x_i$. For example, MRI brain images can be augmented with high-level medical or even psychological descriptions of Alzheimer's disease to build a classifier that predicts the probability of Alzheimer's disease from an MRI image at test time. It is known from statistical learning theory (Vapnik, 1998) that the following bound for test error is satisfied with probability $1-\delta$:

$$R(f)\leq R_n(f)+O\left(\left(\frac{|\mathcal{F}|_{VC}-\log\sigma}{n}\right)^{\alpha}\right), \tag{3}$$

where $R_n(f)$ denotes the training error over $n$ samples, $|\mathcal{F}|_{VC}$ is the VC dimension of the space of functions from which $f$ is chosen, and $\alpha\in[0.5,1]$. When the classes are not $separable$, $\alpha=0.5$ i.e. the machine learns at a slow rate of $O(n^{-1/2})$. For easier problems where classes are $separable$, $\alpha=1$ resulting in a learning rate of $O(n^{-1})$. The difference between these two cases is severe. The same error bound achieved for a separable problem with 10 thousand data points is only obtainable for a non-separable problem when 100 million data points are provided. This is prohibitive even when obtaining large datasets is not so costly. The theory of LUPI shows that an intelligent teacher can reduce $\alpha$ resulting in a faster learning process for the student. In this paper, we proposed a *teacher-student* framework for semi-supervised learning. Similar to LUPI, in FWL a student is supposed to solve the main prediction task while an intelligent teacher provides additional information to improve its learning. In addition, we first train the student network so that it obtains initial knowledge of weakly labeled data and learns a good data representation. Then the teacher is trained on truly labeled data enjoying the representation learnt by the student. This extends LUPI in a way that the teacher provides privileged information that is most useful for the current state of student's knowledge. FWL also extends LUPI by introducing several teachers each of which is specialized to correct student's knowledge related to a specific region of the data space.

Figure 6(a) provides evidence for the assumption that privileged information in our task can accelerate the learning process of the student. It shows how the privileged information from an intelligent teacher affects the exponent $\alpha$ of the error bound in Equation 3. Figure 6(b) shows the test error for various number of samples $|\mathcal{D}_s|$ with true label. As expected, In both extremes where $|\mathcal{D}_s|$ is too small or too large, the performance of our model becomes close to the models without a teacher. The reason is that student has enough strong samples to learn a good model of true function. In more realistic cases where $|\mathcal{D}_s|\ll|\mathcal{D}_w|$ but $|\mathcal{D}_s|$ is still large enough to be informative about $|\mathcal{D}_w|$, our model gives a lower test error than models without the intelligent teacher.

The theory of LUPI was first developed and proved for support vector machines by Vapnik as a method for knowledge transfer. Hinton introduced *Dark knowledge* as a spiritually close idea in the context of neural networks (Hinton et al., 2006). He proposed to use a large network or an ensemble of networks for training and a smaller network at test time. It turned out that compressing knowledge of a large system into a smaller system can improve the generalization ability. It was shown in (Lopez-Paz et al., 2015) that dark knowledge and LUPI can be unified under a single umbrella, called *generalized distillation*. The core idea of these models is *machines-teaching-machines*. As the name suggests, a machine is learning the knowledge embedded in another machine. In our case, student is correcting his knowledge by receiving privileged information about label uncertainty from teacher.

**Algorithm 2** Clustered Gaussian processes.

1: Let $N$ be the sample size, $n$ the sample size of each cluster, $K$ the number of clusters, and $c_i$ the center of cluster $i$.

2: Run K-means with $K$ clusters over all samples with true labels $\mathcal{D}_s = \{x_i, y_i\}$.

$$\text{K-means}(x_i) \rightarrow c_1, c_2, ..., c_K$$

where $c_i$ represents the center of cluster $C_i$ containing samples $D_s^{c_i} = \{x_{i,1}, x_{i,2}, ... x_{i,n}\}$.

3: Assign each of $K$ clusters a Gaussian process and train them in parallel to approximate the label of each sample.

$$\begin{aligned} \mathcal{GP}_{c_i}(\boldsymbol{m}_{\text{post}}^{c_i}, K_{\text{post}}^{c_i}) &= \mathcal{GP}(\boldsymbol{m}_{\text{prior}}, K_{\text{prior}}) | D_s^{c_i} = \{(\psi(x_{s,c_i}), y_{s,c_i})\} \\ T_{c_i}(x_t) &= g(\boldsymbol{m}_{\text{post}}^{c_i}(x_t)) \\ \Sigma_{c_i}(x_t) &= h(K_{\text{post}}^{c_i}(x_t, x_t)) \end{aligned}$$

where $\mathcal{GP}_{c_i}$ is trained on $\mathcal{D}_s^{c_i}$ containing samples belonging to the cluster $c_i$. Other elements are defined in Section 2

4: Use trained teacher $T_{c_i}(.)$ to evaluate the soft label and uncertainty for samples from $\mathcal{D}_{sw}$ to compute $\eta_2(x_t)$ required for step 3 of Algorithm 1. We use $T(.)$ as a wrapper for all teachers $\{T_{c_i}\}$.

Our framework extends the core idea of LUPI in the following directions:

- Trainable teacher: It is often assumed that the teacher in LUPI framework has some additional true information. We show that when this extra information is not available, one can still use the LUPI setup and define an implicit teacher whose knowledge is learned from the true data. In this approach, the performance of the final student-teacher system depends on a clever answer to the following question: which information should be considered as the privileged knowledge of teacher.
- Bayesian teacher: The proposed teacher is Bayesian. It provides posterior uncertainty of the label of each sample.
- Mutual representation: We introduced module $\psi(.)$ which learns a mutual embedding (representation) for both student and teacher. This is in particular interesting because it defines a two-way channel between teacher and student.
- Multiple teachers: We proposed a scalable method to introduce several teachers such that each teacher is specialized in a particular region of the data space.

## B DETAILED DESCRIPTION OF CLUSTERED $\mathcal{GP}$

We suggested using several $\mathcal{GP} = \{GP_{c_i}\}$ to cover the entire data space. This results in a better specialization of each teacher. In addition it solves the scalability issue when the size of the strong dataset $D_s$ on which we train $\mathcal{GP}$ is too large. Here we propose a method called *clustered Gaussian process* inspired by (Shen et al., 2006) to alleviate the issue of large sample size.

*Clustered* $\mathcal{GP}$: Let $N$ be the size of the dataset on which we train the teacher. Assume we allocate $K$ teachers to the entire data space. Therefore, each $\mathcal{GP}$ sees a dataset of size $n = N/K$. Then we use a simple clustering method (e.g. k-means) to find centroids of $K$ clusters $C_1, C_2, ..., C_K$ where $C_i$ consists of samples $\{x_{i,1}, x_{i,2}, ..., x_{i,n}\}$. We take the centroid $c_i$ of cluster $C_i$ as the representative sample for all its content. Note that $c_i$ does not necessarily belong to $\{x_{i,1}, x_{i,2}, ..., x_{i,n}\}$. We assign each cluster a $\mathcal{GP}$ trained by samples belonging to that cluster. More precisely, cluster $C_i$ is assigned a $\mathcal{GP}$ whose data points are $\{x_{i,1}, x_{i,2}, ..., x_{i,n}\}$. Because there is no dependency among different clusters, we train them in parallel to speed-up the procedure more. The pseudo-code of the clustered $\mathcal{GP}$ is presented in Algorithm 2. When the main issue is computational resources, we can first choose the number $n$ which is the maximum size of the dataset on which our resources allow to train a $\mathcal{GP}$, then find the number of clusters $K = N/n$ accordingly. The rest of the algorithm remains unchanged.

# C DETAILED ARCHITECTURE OF THE STUDENTS

## C.1 RANKING TASK

For the ranking task, the employed student is proposed in (Dehghani et al., 2017). The first layer of the network models function $\psi$ that learns the representation of the input data samples, i.e. $(q,d^+,d^-)$, and consists of three components: (1) an embedding function $\varepsilon:\mathcal{V}\rightarrow\mathbb{R}^m$ (where $\mathcal{V}$ denotes the vocabulary set and $m$ is the number of embedding dimensions), (2) a weighting function $\omega:\mathcal{V}\rightarrow\mathbb{R}$, and (3) a compositionality function $\odot:(\mathbb{R}^m,\mathbb{R})^n\rightarrow\mathbb{R}^m$. More formally, the function $\psi$ is defined as:

$$\begin{aligned}\psi(q,d^+,d^-)=[&\odot_{i=1}^{|q|}(\varepsilon(t_i^q),\omega(t_i^q))\,||\\ &\odot_{i=1}^{|d^+|}(\varepsilon(t_i^{d^+}),\omega(t_i^{d^+}))\,||\\ &\odot_{i=1}^{|d^-|}(\varepsilon(t_i^{d^-}),\omega(t_i^{d^-}))\,],\end{aligned} \tag{4}$$

where $t_i^q$ and $t_i^d$ denote the $i^{th}$ term in query $q$ respectively document $d$. The embedding function $\varepsilon$ maps each term to a dense $m$- dimensional real value vector, which is learned during the training phase. The weighting function $\omega$ assigns a weight to each term in the vocabulary. It has been shown that $\omega$ simulates the effect of inverse document frequency (IDF), which is an important feature in information retrieval (Dehghani et al., 2017).

The compositionality function $\odot$ projects a set of $n$ embedding-weighting pairs to an $m$- dimensional representation, independent from the value of $n$:

$$\bigodot_{i=1}^{n}(\varepsilon(t_i),\omega(t_i))=\frac{\sum_{i=1}^{n}\exp(\omega(t_i))\cdot\varepsilon(t_i)}{\sum_{j=1}^{n}\exp(\omega(t_j))}, \tag{5}$$

which is in fact the normalized weighted element-wise summation of the terms' embedding vectors. Again, it has been shown that having global term weighting function along with embedding function improves the performance of ranking as it simulates the effect of inverse document frequency (IDF). In our experiments, we initialize the embedding function $\varepsilon$ with word2vec embeddings (Mikolov et al., 2013) pre-trained on Google News and the weighting function $\omega$ with IDF.

The representation layer is followed by a simple fully connected feed-forward network with $l$ hidden layers followed by a softmax which receives the vector representation of the inputs processed by the representation learning layer and outputs a prediction $\tilde{y}$. Each hidden layer $z_k$ in this network computes $z_k=\alpha(W_k z_{k-1}+b_k)$, where $W_k$ and $b_k$ denote the weight matrix and the bias term corresponding to the $k^{th}$ hidden layer and $\alpha(.)$ is the non-linearity. These layers follow a sigmoid output. We employ the cross entropy loss:

$$\mathcal{L}_t=\sum_{i\in B}[-y_i\log(\hat{y}_i)-(1-y_i)\log(1-\hat{y}_i)], \tag{6}$$

where $B$ is a batch of data samples.

## C.2 SENTIMENT CLASSIFICATION TASK

The student for the sentiment classification task is a convolutional model which has been shown to perform best in the dataset we used (Deriu et al., 2017; Severyn & Moschitti, 2015a;b; Deriu et al., 2016). The first layer of the network learns the function $\psi$ which maps input sentence $s$ to a vector as its representation consists of an embedding function $\varepsilon:\mathcal{V}\rightarrow\mathbb{R}^m$, where $\mathcal{V}$ denotes the vocabulary set and $m$ is the number of embedding dimensions.

This function maps the sentence to a matrix $S\in\mathbb{R}^{m\times|s|}$, where each column represents the embedding of a word at the corresponding position in the sentence. Matrix $S$ is passed through a convolution layer. In this layer, a set of $f$ filters is applied to a sliding window of length $h$ over $S$ to generate a feature map matrix $C$. Each feature map $c_i$ for a given filter $F$ is generated by $c_i=\sum_{k,j}S[i:i+h]_{k,j}F_{k,j}$, where $S[i:i+h]$ denotes the concatenation of word vectors from position $i$ to $i+h$. The concatenation of all $c_i$ produces a feature vector $c\in\mathbb{R}^{|s|-h+1}$. The vectors $c$ are then aggregated over all $f$ filters into a feature map matrix $C\in\mathbb{R}^{f\times(|s|-h+1)}$.

We also add a bias vector $b \in R^f$ to the result of a convolution. Each convolutional layer is followed by a non-linear activation function (we use ReLU(Nair & Hinton, 2010)) which is applied element-wise. Afterward, the output is passed to the max pooling layer which operates on columns of the feature map matrix $C$ returning the largest value: $pool(c_i) : \mathbb{R}^{1\times(|s|-h+1)} \rightarrow \mathbb{R}$ (see Figure 4). This architecture is similar to the state-of-the-art model for Twitter sentiment classification from Semeval 2015 and 2016 (Severyn & Moschitti, 2015b; Deriu et al., 2016).

We initialize the embedding matrix with word2vec embeddings (Mikolov et al., 2013) pretrained on a collection of 50M tweets[2].

The representation layer then is followed by a feed-forward layer similar to the ranking task (with different width and depth) but with softmax instead of sigmoid as the output layer which returns $\hat{y}_i$, the probability distribution over all three classes. We employ the cross entropy loss:

$$\mathcal{L}_t = \sum_{i \in B} \sum_{k \in K} -y_i^k \log(\hat{y}_i^k), \tag{7}$$

where $B$ is a batch of data samples, and $K$ is a set of classes.

## D Detailed Architecture of the Teachers

We use Gaussian Process as the teacher in all the experiments. For each task, either regression or (multi-class) classification, in order to generate soft labels, we pass the mean of $\mathcal{GP}$ through the same function $g(.)$ that is applied on the output of the student network for that task, e.g. softmax, or sigmoid. For binary classification or one dimensional regression, $\Sigma(x_t)$ is scalar and $h(.)$ is identity. For multi-class classification or multi-dimensional regression tasks, $h(.)$ is an aggregation function that takes variance over several dimensions and outputs a single measure of variance. As a reasonable choice, the aggregating function $h(.)$ in our sentiment classification task (three classes) is *mean* of variances over dimensions.

In the teacher, linear combinations of different kernels are used for different tasks in our experiments.

**Toy Problem:** We use standard Gaussian process regression[3] with this kernel:

$$k(x_i, x_j) = k_{\text{RBF}}(x_i, x_j) + k_{\text{White}}(x_i, x_j) \tag{8}$$

**Document Ranking:** We use sparse variational GP regression[4] (Titsias, 2009) with this kernel:

$$k(x_i, x_j) = k_{\text{Matern3/2}}(x_i, x_j) + k_{\text{Linear}}(x_i, x_j) + k_{\text{White}}(x_i, x_j) \tag{9}$$

**Sentiment Classification:** We use sparse variational GP for multiclass classification [5] (Hensman et al., 2015) with the following kernel:

$$k(x_i, x_j) = k_{\text{RBF}}(x_i, x_j) + k_{\text{Linear}}(x_i, x_j) + k_{\text{White}}(x_i, x_j) \tag{10}$$

where,

$$k_{\text{RBF}}(x_i, x_j) = \exp\left(\frac{\|x_i - x_j\|^2}{2l^2}\right)$$
$$k_{\text{Matern3/2}}(x_i, x_j) = \left(1 + \frac{\sqrt{3}\|x_i - x_j\|}{l}\right)\exp\left(-\frac{\sqrt{3}\|x_i - x_j\|}{l}\right)$$
$$k_{\text{Linear}}(x_i, x_j) = \sigma_0^2 + x_i.x_j$$
$$k_{\text{White}}(x_i, x_j) = constant_value, \quad \forall x_1 = x_2 \text{ and 0 otherwise}$$

[2] collected using Twitter API. We are planning to release the list of tweet ids along with the code for replicating our experiments.
[3] http://gpflow.readthedocs.io/en/latest/notebooks/regression.html
[4] http://gpflow.readthedocs.io/en/latest/notebooks/SGPR_notes.html
[5] http://gpflow.readthedocs.io/en/latest/notebooks/multiclass.html

We empirically found $l = 1$ satisfying value for the length scale of RBF and Matern3/2 kernels. We also set $\sigma_0 = 0$ to obtain a homogeneous linear kernel. The constant value of $K_{White}(.,.)$ determines the level of noise in the labels. This is different from the noise in weak labels. This term explains the fact that even in true labels there might be a trace of noise due to the inaccuracy of human labelers.

We set the number of clusters in the clustered $\mathcal{GP}$ algorithm for the ranking task to $50$ and for the sentiment classification task to $30$.

# E WEAK ANNOTATORS

## E.1 DOCUMENT RANKING

The weak annotator in the document ranking task is BM25 (Robertson & Zaragoza, 2009), a well-known unsupervised retrieval method. This method heuristically scores a given pair of query-document based on the statistics of their matched terms. In the pairwise document ranking setup, $\tilde{y}_i$ for a given sample $x_j = (q, d^+, d^-)$ is the probability of document $d^+$ being ranked higher than $d^-$: $\tilde{y}_i = P_{q,d^+,d^-} = {s_{q,d^+}}/{s_{q,d^+} + s_{q,d^-}}$, where $s_{q,d}$ is the score obtained from the weak annotator.

## E.2 SENTIMENT CLASSIFICATION

The weak annotator for the sentiment classification task is a simple lexicon-based method (Hamdan et al., 2013; Kiritchenko et al., 2014). We use SentiWordNet03 (Baccianella et al., 2010) to assign probabilities (positive, negative and neutral) for each token in set $\mathcal{D}_w$. We use a bag-of-words model for the sentence-level probabilities (i.e. just averaging the distributions of the terms), yielding a noisy label $\tilde{y}_i \in \mathbb{R}^{|K|}$, where $|K| = 3$ is the number of classes. We found empirically that using soft labels from the weak annotator works better than assigning a single hard label.

# F DATA COLLECTION, PARAMETERS AND SETUP

## F.1 TOY PROBLEM

**Weak/True Data** In all the experiments with the toy problem, we have randomly sampled 100 data points from the weak function and 10 data points from the true function. We introduce a small amount of noise to the observation of the true function to model the noise in the human labeled data.

**Setup** The neural network employed in the toy problem experiments is a simple feed-forward network with the depth of 3 layers and width of 128 neurons per layer. We have used $tanh$ as the nonlinearity for the intermediate layers and a linear output layer. As the optimizer, we used Adam (Kingma & Ba, 2014) and the initial learning rate has been set to $0.001$. For the teacher in the toy problem, we fit only one $\mathcal{GP}$ on all the data points (i.e. no clustering). Also during fine-tuning, we set $\beta = 1$.

**Setup of experiments in Section 4.2** We fixed everything in the model and tried running the fine-tuning stage with different values for $\beta \in \{0.0, 0.1, 1.0, 2.0, 5.0\}$ in all the experiments. For the experiments on toy problem in Section 4.2, the reported numbers are averaged over 10 trials. In the first experiment (i.e. Figure 6a), the size of sampled data data is: $|\mathcal{D}_s| = 50$ and $|\mathcal{D}_w| = 100$ (Fixed) and for the second one (i.e. Figure 6a): $|\mathcal{D}_w| = 100$ and $|\mathcal{D}_s| = 10$ (fixed).

## F.2 RANKING TASK

**Collections** We use two standard TREC collections for the task of ad-hoc retrieval: The first collection (*Robust04*) consists of 500k news articles from different news agencies as a homogeneous collection. The second collection (*ClueWeb*) is ClueWeb09 Category B, a large-scale web collection with over 50 million English documents, which is considered as a heterogeneous collection. Spam documents were filtered out using the Waterloo spam scorer [6] (Cormack et al., 2011) with the default threshold $70\%$.

**Data with true labels** We take query sets that contain human-labeled judgments: a set of 250 queries (TREC topics 301–450 and 601–700) for the Robust04 collection and a set of 200 queries (topics 1-200) for the experiments on the ClueWeb collection. For each query, we take all documents judged as

[6] http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

relevant plus the same number of documents judged as non-relevant and form pairwise combinations among them.

**Data with weak labels** We create a query set $Q$ using the unique queries appearing in the AOL query logs (Pass et al., 2006). This query set contains web queries initiated by real users in the AOL search engine that were sampled from a three-month period from March 2006 to May 2006. We filtered out a large volume of navigational queries containing URL substrings ("http", "www.", ".com", ".net", ".org", ".edu"). We also removed all non-alphanumeric characters from the queries. For each dataset, we took queries that have at least ten hits in the target corpus using our weak annotator method. Applying all these steps, We collect 6.15 million queries to train on in Robust04 and 6.87 million queries for ClueWeb. To prepare the weakly labeled training set $\mathcal{D}_w$, we take the top $1{,}000$ retrieved documents using BM25 for each query from training query set $Q$, which in total leads to $\sim |Q| \times 10^6$ training samples.

**Setup** For the evaluation of the whole model, we conducted a 3-fold cross-validation. However, for each dataset, we first tuned all the hyper-parameters of the student in the first step on the set with true labels using batched GP bandits with an expected improvement acquisition function (Desautels et al., 2014) and kept the optimal parameters of the student fixed for all the other experiments. The size and number of hidden layers for the student is selected from $\{64{,}128{,}256{,}512\}$. The initial learning rate and the dropout parameter were selected from $\{10^{-3}{,}10^{-5}\}$ and $\{0.0{,}0.2{,}0.5\}$, respectively. We considered embedding sizes of $\{300{,}500\}$. The batch size in our experiments was set to $128$. We use ReLU (Nair & Hinton, 2010) as a non-linear activation function $\alpha$ in student. We use the Adam optimizer (Kingma & Ba, 2014) for training, and *dropout* (Srivastava et al., 2014) as a regularization technique.

At inference time, for each query, we take the top $2{,}000$ retrieved documents using BM25 as candidate documents and re-rank them using the trained models. We use the Indri[7] implementation of BM25 with default parameters (i.e., $k_1 = 1.2$, $b = 0.75$, and $k_3 = 1{,}000$).

### F.3 Sentiment Classification Task

**Collections** We test our model on the twitter message-level sentiment classification of SemEval-15 Task 10B (Rosenthal et al., 2015). Datasets of SemEval-15 subsume the test sets from previous editions of SemEval, i.e. SemEval-13 and SemEval-14. Each tweet was preprocessed so that URLs and usernames are masked.

**Data with true labels** We use train (9,728 tweets) and development (1,654 tweets) data from SemEval-13 for training and SemEval-13-test (3,813 tweets) for validation. To make your results comparable to the official runs on SemEval we us SemEval-14 (1,853 tweets) and SemEval-15 (2,390 tweets) as test sets (Rosenthal et al., 2015; Nakov et al., 2016).

**Data with weak labels** We use a large corpus containing 50M tweets collected during two months for both, training the word embeddings and creating the weakly annotated set $\mathcal{D}_w$ using the lexicon-based method explained in Section 3.3.

**Setup** Similar to the document ranking task, we tuned hyper-parameters for the student in the first step with respect to the true labels of the validation set using batched GP bandits with an expected improvement acquisition function (Desautels et al., 2014) and kept the optimal parameters fixed for all the other experiments. The size and number of hidden layers for the classifier and is selected from $\{32{,}64{,}128\}$. We tested the model with both, 1 and 2 convolutional layers. The number of convolutional feature maps and the filter width is selected from $\{200{,}300\}$ and $\{3{,}4{,}5\}$, respectively. The initial learning rate and the dropout parameter were selected from $\{1E-3, 1E-5\}$ and $\{0.0, 0.2, 0.5\}$, respectively. We considered embedding sizes of $\{100{,}200\}$ and the batch size in these experiments was set to $64$. ReLU (Nair & Hinton, 2010) is used as a non-linear activation function in student. Adam optimizer (Kingma & Ba, 2014) is used for training, and *dropout* (Srivastava et al., 2014) as a regularizer.

[7] https://www.lemurproject.org/indri.php