Data Programming using Continuous and Quality-Guided Labeling Functions

Oishik Chatterjee
Department of CSE
IIT Bombay, India
oishik@cse.iitb.ac.in

Ganesh Ramakrishnan
Department of CSE
IIT Bombay, India
ganesh@cse.iitb.ac.in

Sunita Sarawagi
Department of CSE
IIT Bombay, India
sunita@iitb.ac.in

Abstract

Scarcity of labeled data is a bottleneck for supervised learning models. A paradigm that has evolved for dealing with this problem is data programming. An existing data programming paradigm allows human supervision to be provided as a set of discrete labeling functions (LF) that output possibly noisy labels to input instances and a generative model for consolidating the weak labels. We enhance and generalize this paradigm by supporting functions that output a continuous score (instead of a hard label) that noisily correlates with labels. We show across five applications that continuous LFs are more natural to program and lead to improved recall. We also show that accuracy of existing generative models is unstable with respect to initialization, training epochs, and learning rates. We give control to the data programmer to guide the training process by providing intuitive quality guides with each LF. We propose an elegant method of incorporating these guides into the generative model. Our overall method, called CAGE, makes the data programming paradigm more reliable than other tricks based on initialization, sign-penalties, or soft-accuracy constraints.

1 Introduction

Modern machine learning systems require large amounts of labelled data. For many applications, such labelled data is created by getting humans to explicitly label each training example. A problem of perpetual interest in machine learning is reducing the tedium of such human supervision via techniques like active learning, crowd-labeling, distant supervision, and semi-supervised learning. A limitation of all these methods is that supervision is restricted to the level of individual examples.

A recently proposed (Ratner et al. 2016) paradigm is that of Data Programming. In this paradigm, humans provide several labeling functions written in any high-level programming language. Each labeling function (LF) takes as input, an example and either attaches a label to it or backs off. We illustrate such LFs on one of the five tasks that we experimented with, *viz.*, that of labeling mention of a pair of people names in a sentence as defining the *spouse* relation or not. The users construct heuristic patterns as LFs for identifying spouse relation in a sentence containing an entity pair (E_1, E_2). A LF can assign +1 to indicate that the *spouse* relation is true for the candidate pair (E_1,E_2), -1 to mean that no *spouse* relation, and 0 to mean that the LF in unable to assert anything for this example. Specifically for the *spouse* relation extraction task, Table 1 lists six LFs.

In isolation, each LF may neither be always correct nor complete. LFs may also produce conflicting labels. For the purpose of illustration, consider a text snippet ‘Michelle Obama is the mother of Malia and Sasha and the wife of Barack Obama’. For the candidate pair (‘Michelle Obama’, ’Barack Obama’), LF1 and LF4 in Table 1 assign a label 1 whereas LF2 assigns the label -1.

Ratner et al. (2016) presented a generative model for consensus on the noisy and conflicting labels assigned by the discrete LFs to determine probability of the correct labels. Labels thus obtained could be used for training any supervised model/classifier and evaluated on a test set. In this paper, we present two significant extensions of the above data programming paradigm.

First, the user provided set of LFs might not be complete in their discrete forms. LF1 through LF3 in Table 1 that look for words in various hand-crafted dictionaries, may have incomplete dictionaries. A more comprehensive alternative could be to design continuous valued LFs that return scores derived from soft match between words in the sentence and the dictionary. As an example, for LF1 through LF3, the soft match could be obtained based on cosine similarity of pre-trained word embedding vectors (Mikolov et al. 2013) of a word in the dictionary with a word in the sentence. This could enable an LF to provide a continuous class-specific score to the model, instead of a hard class label (when triggered). In Table 2, we list a continuous LF corresponding to each LF from Table 1. Such continuous LFs can expand the scope of matching to semantically similar words beyond the pre-specified words in the dictionary. For example: in the sentence `1) <Allison>, 27, and <Ricky>, 34, wed on Saturday surrounded by friends.`, the word ’wed’ is semantically similar to ’married’ and would be detected by our continuous LF but missed by the discrete ones in Table 1.

More generally across applications, human experts are very often able to identify real-valued scores that correlate strongly with the label but find it difficult to discretize that score into a hard label. More examples of such scores in-

SpouseDict = {'spouse', 'married', wife', 'husband', 'ex-wife', 'ex-husband'}
FamilyDict = {'father', 'mother', 'sister', 'brother', 'son', 'daughter','grandfather', 'grandmother', 'uncle', 'aunt','cousin' }$\bigotimes${+",+'-in-law'}
OtherDict = {'boyfriend', 'girlfriend' 'boss', 'employee', 'secretary', 'co-worker'}
SeedSet = {('Barack Obama', 'Michelle Obama '), ('Jon Bon Jovi','Dorothea Hurley'),('Ron Howard','Cheryl Howard'),.....}

Id	Description
LF1	If some word in **SpouseDict** is present between E_1 and E_2 or within 2 words of either, return 1 else return 0
LF2	If some word in **FamilyDict** is present between E_1 and E_2, return -1 else return 0.
LF3	If some word in **OtherDict** is present between E_1 and E_2, return -1 else return 0.
LF4	If both E_1 and E_2 occur in **SeedSet**, return 1 else return 0.
LF5	If the number of word tokens lying between E_1 and E_2 are less than 4, return 1 else return 0.

Table 1: Discrete LFs based on dictionary lookups or thresholded distance for the *spouse* relationship extraction task

Id	Class	Description
LF1	+1	max [cosine(word-vector(u), word-vector(v))-0.8]$_+$: $u \in$ **SpouseDict** and $v \in$ {words between E_1, E_2}.
LF2	-1	max [cosine(word-vector(u), word-vector(v))-0.8]$_+$: $u \in$ **FamilyDict** and $v \in$ {words between E_1, E_2}.
LF3	-1	max [cosine(word-vector(u), word-vector(v))-0.8]$_+$: $u \in$ **OtherDict** and $v \in$ {words between E_1, E_2}.
LF4	-1	max [0.2 - Norm-Edit-Dist(E_1, E_2, u, v)]$_+$: $(u, v), (v, u) \in$ **SeedSet**.
LF5	+1	[1 - (number of word tokens between E_1 and E_2)/5.0]$_+$

Table 2: Continuous LFs corresponding to some of the discrete LFs in Table 1 for the *spouse* relationship extraction task

clude TF-IDF match with prototypical documents in a text-classification task, distance among entity pairs for a relation extraction task, and confidence scores from hand-crafted classifiers.

We extend existing generative models (Ratner et al. 2016; Bach et al. 2017) to support continuous LFs. In addition to modeling the consensus distribution over all LFs (continuous and discrete), we model the distribution of scores for each continuous LF. In Section 4 in the supplementary material, we illustrate through an example, why the model for continuous LFs is not a straightforward extension of the discrete counterpart.

Our second extension is designed to remove the instability of existing generative training based on unsupervised likelihood. Across several datasets we observed that the accuracy obtained by the existing models was highly sensitive to initialization, training epochs, and learning rates. In the absence of labeled validation set, we cannot depend on prevalent tricks like early stopping to stabilize training. We give control to the data programmer to guide the training process by providing intuitive accuracy guides with each LF. In the case that the labeler develops each LF after inspecting some examples, the labeler would naturally have a rough estimate of the fraction q of examples that would be correctly labeled by the LF, amongst all examples that trigger the LF. Even otherwise, it might not be too difficult for the labeler to intuitively specify some value of q. We show that such a q serves as a user-controlled quality guide that can effectively guide the training process. For the case of continuous LFs, we use q as a rough estimate of the mean score of the continuous LF whenever it triggers correctly. A quality guide of $q = 0.9$ for LF1 would imply that when LF1 triggers correctly the average embedding score is 0.9. This is easier than choosing a hard threshold on the embedding score to convert it to a discrete LF. We provide an elegant method of guiding our generative training with these user-provided accuracy estimates, and show how it surpasses simpler methods like sign penalty and data constraints. Empirically, we show our method stabilizes unsupervised likelihood training even with very crude estimates of q. We study stability issues of the existing model with respect to training epochs and demonstrate that the proposed model is naturally more stable. We refer to our overall approach as CAGE, which stands for _**C**ontinuous _**A**nd quality _**G**uided lab_**E**ling functions. In summary, CAGE makes the following contributions:
1) It enhances the expression power of data programming by supporting continuous labeling functions (generalizations of discrete LFs) as the unit of supervision from humans.
2) It proposes a carefully parameterized graphical model that outperforms existing models even for discrete LFs, and permits easy incorporation of user priors for continuous LFs.
3) It extends the generative model through quality guides, thereby increasing its stability and making it less sensitive to initialization. Its training is based on a principled method of regularizing the marginals of the joint model with user-provided accuracy guides.
We present extensive experiments on five datasets, comparing various models for performance and stability, and present the significantly positive impact of CAGE . We show that our method of incorporating user guides leads to more reliable training than obvious ideas like sign penalty and constraint based training.

2 Our Approach: CAGE

Let $\mathcal{X}$ denote the space of input instances and $\mathcal{Y} = \{1, \ldots, K\}$ denote the space of labels and $P(\mathcal{X}, \mathcal{Y})$ denote their joint distribution. Our goal is to learn a model to associate a label y with an example $\mathbf{x} \in \mathcal{X}$. Unlike standard supervised learning, we are not provided true labels of sampled instances during training. Let the sample of m unlabeled instances be $\mathbf{x}_1, \ldots, \mathbf{x}_m$. Instead of the true y's we are provided a set of n labeling functions (LFs) $\lambda_1, \lambda_2, \ldots \lambda_n$ such that each LF λ_j can be either discrete or continuous. Each LF λ_j is attached with a class k_j and on an instance $\mathbf{x}_i$ outputs a discrete label $\tau_{ij} = k_j$ when triggered and $\tau_{ij} = 0$ when not triggered. If λ_j is continuous, it also outputs a score $s_{ij} \in (0, 1)$. This is a form of weak supervision that implies that when a LF is triggered on an instance $\mathbf{x}_i$, it is proposing that the true label y should be k_j, and if continuous it is attaching a confidence proportional to s_{ij} with its labeling.

But to reliably and accurately infer the true label y from such weak supervision without any labeled data, we need to exploit the assumption that τ_{ij} is positively correlated with y. We allow the programmer of the LF to make this assumption explicit by attaching a guess on the fraction q_j^t of triggering of the LF where the true y agrees with τ_{ij}. We show in the experiments that crude guesses suffice. Intuitively, this says that the user expects q_j^t fraction of examples for which the LF value has been triggered to be correct. Additionally, for continuous LF the programmer can specify the quality index q_j^c denoting the average score of s_j when there is such agreement.

Our goal is to learn to infer the correct label by creating consensus among outputs of the LFs. Thus, the model of CAGE imposes a joint distribution between the true label y and the values τ_{ij}, s_{ij} returned by each LF λ_j on any data sample $\mathbf{x}_i$ drawn from the hidden distribution $P(\mathcal{X}, \mathcal{Y})$.

$$P_{\theta,\pi}(y, \tau_i, \mathbf{s}_i) = \frac{1}{Z_\theta} \prod_{j=1}^{n} \psi_\theta(\tau_{ij}, y) \left(\psi_\pi(\tau_{ij}, s_{ij}, y)\right)^{\text{cont}(\lambda_j)} \tag{1}$$

where $\text{cont}(\lambda_j)$ is 1 when λ_j is a continuous LF and 0 otherwise. And θ, π denote the parameters used in defining the potentials psi_θ, ψ_π coupling discrete and continuous variables respectively. In this factorization of the joint distribution we make the natural assumption that each LF independently provides its supervision on the true label. The main challenge now is designing the potentials coupling various random variables so that: (a) The parameters (θ, π) can be trained reliably using unlabeled data alone. This partially implies that the number of parameters should be limited. (b) The model should be expressive enough to fit the joint distribution on the τ_j and s_j variables across a variety of datasets without relying on labeled validation dataset for model selection and hyper-parameter tuning. (c) Finally, the potentials should reflect the bias of the programmer on the quality in providing the true y. We will show how without such control, it is easy to construct counter-examples where the standard likelihood-based training may fail miserably.

With these goals in mind, and after significant exploration we propose the following form of potentials. For the discrete binary τ_{ij} variables, we chose these simple potentials:

$$\psi_\theta(\tau_{ij}, y) = \begin{cases} \exp(\theta_{jy}) & \text{if } \tau_{ij} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \tag{2}$$

Thus, for each LF we have K parameters corresponding to each of the class labels. An even simpler alternative would be to share the θ_{jy} across different ys as used in (Bach et al. 2017) but that approach imposes undesirable restrictions on the distributions it can express. We elaborate on that in Section 2.3.

For the case of continuous LFs the task of designing the potential $\psi_\pi(s_{ij}, \tau_{ij}, y)$ that is trainable with unlabeled data and captures user bias well turned out to be significantly harder. Specifically, we wanted a form that is suited for scores that can be interpreted as confidence probabilities (that lie between 0 and 1), and capture the bias that s_{ij} is high when τ_{ij} and y agree, and low otherwise. For confidence variables, a natural parametric form of density is the beta density. The beta density is popularly expressed in terms of two independent parameters $\alpha > 0$ and $\beta > 0$ as $P(s|\alpha, \beta) \propto s^{\alpha-1}(1-s)^{\beta-1}$. Instead of independently learning these parameters, we chose an alternative representation that allows expression of user prior on the expected s. We write the beta in terms of two alternative parameters: the mean parameter q_j^c and the scale parameter π_1. These are related to α and β as $\alpha = q_j^c \pi_1$ and $\beta = (1 - q_j^c)\pi_1$.

We define our continuous potential as:

$$\psi_\pi(\tau_{ij}, s_{ij}, y) = \begin{cases} \mathit{Beta}(s_{ij}; \alpha_a, \beta_a) & \text{if } k_j = y \,\&\, \tau_{ij} \neq 0, \\ \mathit{Beta}(s_{ij}; \alpha_d, \beta_d) & \text{if } k_j \neq y \,\&\, \tau_{ij} \neq 0, \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

where $\alpha_a = q_j^c \pi_{jy}$ and $\beta_a = (1 - q_j^c)\pi_{jy}$ are parameters of the agreement distribution and $\alpha_d = (1 - q_j^c)\pi_{jy}$ and $\beta_d = q_j^c \pi_{jy}$ are parameters of the disagreement distribution, where π_{jy} is constrained to be strictly positive. To impose $\pi_{jy} > 0$ while also maintaining differentiability, we reparametrize π_{jy} as $\exp(\rho_{jy})$. Thus, we require K parameters for each continuous LF, which is the same as for a discrete LF. The Beta distribution would normally require $2K$ parameters but we used the user provided qualify guide in that special manner shown above to share the mean between the agreeing and disagreeing Beta.

We experimented with a large variety of other potential forms before converging on the above. We will elaborate on alternatives in the experimental section.

With these potentials, the normalizer Z_θ of our joint distribution (Eqn 1) can be calculated as

$$\begin{aligned} Z_\theta &= \sum_y \prod_j \sum_{\tau_j \in \{k_j, 0\}} \psi_\theta(\tau_j, y) \int_{s_j=0}^{1} \psi_\pi(\tau_j, s_j, y) \\ &= \sum_{y \in \mathcal{Y}} \prod_j (1 + \exp(\theta_{jy})) \end{aligned} \tag{4}$$

The normalizer reveals two further facets of our joint distribution. First our continuous potentials are defined such that when summed over s_j-s we get a value of 1, hence the normalizer is independent of the continuous parameters π. That is, the continuous potentials $\psi_\pi(\tau_{ij}, s_{ij}, y)$ are locally normalized Bayesian probabilities $P(s_{ij}|\tau_{ij}, y)$. Second, the discrete potentials are not locally normalized; the $\psi_\theta(\tau_j, y)$ cannot be interpreted as $\Pr(\tau_j|y)$ because by normalizing them globally we were able to learn the interaction among the LFs better. We will show empirically that either the full Bayesian model with potentials $P(y), P(\tau_{ij}|y)$, and $P(s_{ij}|\tau_{ij}, y)$ or the fully undirected model where the $\psi_\pi(\tau_{ij}, s_{ij}, y)$ potential is un-normalized are both harder to train.

2.1 Training CAGE

Our training objective can be expressed as:

$$\max_{\theta,\pi} LL(\theta,\pi|D) + R(\theta,\pi|\{q_j^t\}) \tag{5}$$

The first part maximizes the likelihood on the observed τ_i and $\mathbf{s}_i$ values of the training sample $D = \mathbf{x}_1, \ldots, \mathbf{x}_m$ after marginalizing out the true y. It can be expressed as:

$$LL(\theta,\pi|D) = \sum_{i=1}^{m} \log \sum_{y\in\mathcal{Y}} P_{\theta,\pi}(\tau_i, \mathbf{s}_i, y) \tag{6}$$

$$= \sum_{i=1}^{m} \log \sum_{y\in\mathcal{Y}} \prod_{j=1}^{n} \psi_j(\tau_{ij}, y) \left(\psi_j(s_{ij}, \tau_{ij}, y)\right)^{\text{cont}(\lambda_j)} - m \log Z_\theta$$

By CAGE$_{-\text{G}}$, we will hereafter refer to the model in Eqn 1 that has parameters learnt by maximizing only this (first) likelihood part of the objective in Eqn 6 and not the second part $R(\theta,\pi|\{q_j^t\})$. $R(\theta,\pi|\{q_j^t\})$ is a regularizer that guides the parameters with the programmer's expectation of the quality of each LF. We start by motivating the need for the regularizer by showing simple cases that can cause the likelihood-only training to yield poor accuracy.

Example 1: Sensitivity to Initialization Consider a binary classification task where the n LFs are perfect oracles that trigger only on instances whose true label matches k_j. Assume all λ_j are discrete. The likelihood of such data can be expressed as:

$$LL(\theta) = \sum_{i=1}^{m} \log(\exp(\sum_{j:k_j=y} \theta_{j1}) + \exp(\sum_{j:k_j=y} \theta_{j2}))$$

$$- m \log(\prod_j (1 + \exp(\theta_{j1})) + \prod_j (1 + \exp(\theta_{j2}))) \tag{7}$$

The value of the above likelihood is totally symmetric in θ_{j1} and θ_{j2} but the accuracy is not. We will get 100% accuracy only when the parameter for the agreeing case: θ_{jk_j} is larger than θ_{jy} for $y \neq k_j$, and 0% accuracy if θ_{jk_j} is smaller. A trick is to initialize the θ parameters carefully so that the agreeing parameters θ_{jk_j} do have larger values. However, even such careful initialization can be forgotten in less trivial cases as we show in the next example.

Example 2: Failure in spite of good initialization Consider a set S1 of r LFs that assign a label of 1 and remaining set S2 of $n - r$ LFs that assign label 2. Let each true class-2 instance trigger one or more LF from S1 and one or more LF from S2. Let each true class-1 instance trigger *only* LFs from S1. When we initialize LFs in set S1 such that $\theta_{j1} - \theta_{j2} > 0$ and LFs in set S2 have $\theta_{j2} - \theta_{j1} > 0$, we can get good accuracy. However, as training progresses the likelihood will be globally maximized when both sets of LFs favor the same class on all instances. If we further assume that the true class distribution is skewed, the $LL(\theta)$ objective quickly converges to this useless maxima. This scenario is not artificial. Many of the real datasets (e.g. the LFs of Spouse relation extraction data in Table 1) exhibit such trends.

A straight-forward fix of the above problem is to impose a penalty on the sign of $\theta_{jk_j} - \theta_{jy}$. However, since the θs of LFs interact via the global normalizer Z_θ this condition is neither necessary nor sufficient to ensure that in the joint model $P_\theta(y,\tau)$ the values of y and k_j agree more than disagree. For globally conditioned models the parameters cannot be easily interpreted, and we need to constrain at the level of the joint distribution.

One method to work with the whole distribution is to constrain the conditional $P_\theta(y|\tau_i)$ over the instances where the LF triggers and constrain that the accumulated probability of the agreeing y is at least q_j^t as follows:

$$R(\theta|\{q_j^t\}, D) = \sum_j \text{softplus}\left(\sum_{i:\tau_{ij}=k_j} (q_j^t - P_\theta(\tau_i, k_j))\right) \tag{8}$$

We call this the data-driven constrained training method and refer to it as CAGE$_{\text{dataG}}$. However, a limitation of this constraint is that in a mini-batch training environment it is difficult to get enough examples per batch for reliable estimation of the empirical accuracy, particularly for LFs that trigger infrequently. Next we present our method of incorporating the user guidance into the trained model to avoid such instability.

2.2 Data-independent quality guides in CAGE

Our final approach that worked reliably was to regularize the parameters so that the learned joint distribution of y and τ_j matches the user-provided quality guides q_j^t over all y, τ_j values from the joint distribution $P_{\theta,\pi}$. By default, this is the regularizer that we employ in CAGE.

The q_j^t guide is the user's belief on the fraction of cases where y and τ_j agree when $\tau_j \neq 0$ (LF λ_j triggers). Using the joint distribution we can calculate this agreement probability as $P_\theta(y = k_j|\tau_j = k_j)$. This probability can be computed in closed form by marginalizing over all remaining variables in the model in Equation 1 as follows:

$$\begin{aligned} P_\theta(y = k_j|\tau_j = k_j) &= \frac{P_\theta(y = k_j, \tau_j = k_j)}{P_\theta(\tau_j = k_j)} \\ &= \frac{\mathrm{M}_j(k_j)\prod_{r\neq j}(1 + \mathrm{M}_r(k_j))}{\sum_{y\in\mathcal{Y}} \mathrm{M}_j(y)\prod_{r\neq j}(1 + \mathrm{M}_r(y))} \end{aligned}$$

where $\mathrm{M}_j(y) = \exp(\theta_{jy})$. We then seek to minimize the KL distance between the user provided q_j^t and the model calculated precision $P_\theta(y = k_j|\tau_j = k_j)$ which turns out to be:

$$\begin{aligned} R(\theta|\{q_j^t\}) = \sum_j & q_j^t \log P_\theta(y = k_j|\tau_j = k_j) \\ & + (1 - q_j^t) \log(1 - P_\theta(y = k_j|\tau_j = k_j)) \end{aligned} \tag{9}$$

Specifically, when the CAGE model is restricted only to discrete LFs while also incorporating the quality guide in Eqn 9 into the objective in Eqn 6, we refer to the approach as CAGE$_{-\text{C}}$. Further, when the quality guide in Eqn 9 is dropped from CAGE$_{-\text{C}}$, we refer to the approach as CAGE$_{-\text{C}-\text{G}}$.

2.3 Relationship of CAGE with existing models

We would like to point out that the following two simplifications in CAGE lead to existing well known models (Ratner et al. 2016; Ratner et al. 2017), *viz.*, (i) Coupling the θ_{yj} parameters, (ii) Ignoring quality guides, and (iii) Not including continuous potentials. The design used in (Bach et al. 2017) is to assign a single parameter θ_j for each LF and share it across y as:

$$\psi_j^{\text{snorkel}}(\tau_{ij}, y) = \begin{cases} \exp(\theta_j) & \text{if } \tau_{ij} \neq 0, y = k_j, \\ \exp(-\theta_j) & \text{if } \tau_{ij} \neq 0, y \neq k_j, \\ 1 & \text{otherwise.} \end{cases} \quad (10)$$

After ignoring quality guides and continuous LF, we note that a choice of $\theta_{j,+1} = -\theta_{j,-1}$ makes CAGE exactly same as the model in Snorkel. However, we found the Snorkel's method of parameter sharing incorporates an unnecessary bias that $P_\theta(\tau_{ij} = 0|y = k_j) = 1 - P_\theta(\tau_{ij} = 0|y \neq k_j)$. Also, Snorkel's pure likelihood-based training is subject to all the sensitivity to parameter initialization and training epochs that we highlighted in Section 2.1. We show in the experiments how each of the three new extensions in CAGE is crucial to getting reliable training with the data programming paradigm.

3 Empirical Evaluation

In this section we (1) evaluate the utility of continuous LFs vis-a-vis discrete LFs, (2) demonstrate the role of the quality guides for the stability of the unsupervised likelihood training of CAGE as well as Snorkel, and (3) perform a detailed ablation study to justify various design elements of our generative model and its guided training procedure.

3.1 Datasets and Experiment Setup

We perform these comparisons on five different datasets.
Spouse: (spo) This is a relation extraction dataset that proposes to label candidate pairs of entities in a sentence as expressing a 'spouse' relation or not. Our train-dev-test splits and set of discrete LFs shown in Table 1 are the same as in (Ratner et al. 2016) where it was first used. For each discrete LF that checks for matches in a dictionary D of keywords we create a continuous LF that returns s_j as the maximum of cosine similarity of their word embeddings as shown in Table 2. We used pre-trained vectors provided by Glove (Pennington, Socher, and Manning 2014).
SMS spam (sms) is a binary spam/no-spam classification dataset with 5574 documents split into 3700 unlabeled-train and 1872 labeled-test instances. Nine LFs are created based on (i) presence of three categories of words which are highly likely to indicate spam (ii) presence of 2 categories of trigger words in certain contexts, (iii) reference to keywords indicative of first/second or third person, (iv) text characteristics such as number of capitalized characters, presence of special characters, *etc.* and finally a LF that is (v) associated with the negative class, always triggers and serves as the class prior. The LFs are explained in the supplementary in Section 6. The continuous LFs are created in the same way as in Spouse based on word-embedding similarity, number of capitalized characters, *etc.*

CDR: (cdr 2018) This is also a relation extraction dataset where the task is to detect whether or not a sentence expresses a 'chemical cures disease' relation. The train-dev-test splits and LFs are the same as in (Ratner et al. 2016). We did not develop any continuous LF for CDR.

Dedup: This datasetPublicly available at https://www.cse.iitb.ac.in/~sunita/alias/ comprises of 32 thousand pairs of noisy citation records with fields like Title, Author, Year *etc.* The task is to detect if the record pairs are duplicates. We have 18 continuous LFs corresponding to various text similarity functions (such as Jaccard, TF-IDF similarity, 1-EditDistance, *etc.*) computed over one or more of these fields. Each of these LFs is positively correlated with the duplicate label; we create another 18 with the score as 1-similarity for the negative class. The dataset is highly skewed with only 0.5% of the instances as duplicate. All LFs here are continuous.

Iris: Iris is a UCI dataset with 3 classes. We split it into 105 unlabeled train and 45 labeled test examples. We create LFs from the 4 features of the data as follows: For each feature f and class y, we calculate f's mean value $\overline{f}_y$ amongst the examples of y from labeled data and create a LF that returns the value $1 - \text{norm}(f - \overline{f}_y)$ - where $\text{norm}(f - \overline{f}_y)$ is the normalized distance from this mean. This gives us a total of $4 \times 3 = 12$ continuous LFs. Each such LF has a corresponding discrete LF that is triggered if the feature is closest to the mean of its corresponding class.

Ionosphere: This is another 2-class UCI dataset, that is split to 245 unlabeled train and 106 labeled test instances. 64 continuous LFs are created in a manner similar to Iris.

Training Setup For each dataset and discrete LF we arbitrarily assigned a default discrete quality guide $q_j^t = 0.9$ and for continuous LFs $q_j^c = 0.85$. We used learning rate of 0.01 and 100 training epochs. Parameters were initialized favorably — so for the agreeing parameter initial $\theta_{jk_j=1}$ and for disagreeing parameter initial $\theta_{ky} = -1, y \neq k_j$. For Snorkel this is equivalent to $\theta_j = 1$. Only for CAGE that is trained with guides we initialize all parameters to 1. We show in Section 5 that CAGE is insensitive to initialization whereas others are not.

Evaluation Metric We report F1 as our accuracy measure on all binary datasets and for the multi-class dataset Iris we measure micro F1 across the classes. From our generative model, as well as Snorkel, the predicted label on a test instance $\mathbf{x}_i$ is the y for which the joint generative probability is highest, that is: $\text{argmax}_y P(y, \tau_i, \mathbf{s}_i)$. Another measure of interest is the accuracy that would be obtained by training a standard discriminative classifier $P_\mathbf{w}(y|\mathbf{x})$ with labeled data as the probabilistically labeled $P(y|\mathbf{x}_i) \propto P(y, \tau_i, \mathbf{s}_i)$ examples $\mathbf{x}_i$ from the generative model. In the first part of the experiment we measure the accuracy of labeled data produced by the generative model. In Section 9 in the Supplementary, we present accuracy from a trained discriminative model from such dataset. We implemented our model in Pytorch.[1]

	Spouse	CDR	SMS	Ion	Iris	Dedup
Majority	0.17	0.53	0.23	0.79	0.84	-
Snorkel	0.41	0.66	0.34	0.70	0.87	-
CAGE$_{-\text{C}-\text{G}}$	0.48	0.69	0.34	0.81	0.87	-
CAGE$_{-\text{C}}$	0.50	0.69	0.45	0.82	0.87	-
CAGE	**0.58**	**0.69**	**0.54**	**0.97**	**0.87**	**0.79**

Table 3: Overall Results (F1) with predictions from various generative models contrasted with the Majority baseline.

3.2 Overall Results

In Table 3, we compare the performance of CAGE in terms of F1, against the following alternatives: (i) **Majority:** This is a simple baseline, wherein, the label on which a majority of the LFs show agreement is the inferred consensus label. (ii) **Snorkel:** See Section 2.3. (iii) **CAGE$_{-\text{C}-\text{G}}$:** Our model without continuous LFs and quality guides (See Section 2.2). (iv) **CAGE$_{-\text{C}}$:** Our model with quality guides but without continuous LFs (See Section 2.2).

From this table we make three important observations: (1) Comparing Snorkel with CAGE$_{-\text{C}-\text{G}}$ that differs only in decoupling Snorkel's shared θ_j parameters, we observe that the shared parameters of Snorkel were indeed introducing undesirable bias. (2) Comparing CAGE$_{-\text{C}-\text{G}}$ and CAGE$_{-\text{C}}$ we see the gains due to our quality guides. (3) Finally, comparing CAGE$_{-\text{C}}$ and CAGE we see the gains because of the greater expressibility of continuous LFs. These LFs required negligible additional human programming effort beyond the discrete LFs. Compared to Snorkel our model provides significant overall gains in F1. For datasets like Dedup which consist only of continuous scores, CAGE is the only option.

We next present a more detailed ablation study to tease out the importance of different design elements of CAGE.

3.3 Role of the Quality Guides

We motivate the role of the quality guides in Figure 1 where we show test F1 for increasing training epochs on three datasets. In these plots we considered only discrete LFs. We compare Snorkel and our model with (CAGE$_{-\text{C}}$) and without (CAGE$_{-\text{C}-\text{G}}$) these guides. Without the quality guides, all datasets exhibit unpredictable swings in test F1. These swings cannot be attributed to over-fitting since in Spouse and SMS F1 improves later in training with our quality guides. Since we do not have labeled validation data to choose the correct number of epochs, the quality guides are invaluable in getting reliable accuracy in unsupervised learning.

Next, we show that the stability provided by the quality guides (q_j^t) is robust to large deviations from the true accuracy of a LF. Our default q_j^t value was 0.9 for all LFs irrespective of their true accuracy. We repeated our experiments with a precision of 0.8 and got the same accuracy across training epochs (Figure 4 in Supplementary). We next ask if knowing the true accuracy of a LF would help even more and how robust our training is to distortion in the user's guess from the true accuracy. We calculated true accuracy of each LF on the devset and distorted this by a Gaussian noise with variance σ. In Figure 2 we present accuracy after 100 epochs on two datasets with increasing distortion (σ). On CDR CAGE's accuracy is very robust to distorted q_j^t but guides are important as we see from Figure 1(c). On Spouse accuracy is highest with perfect values of q_j^t (Sigma=0) but it stays close to this accuracy up to a distortion of 0.4.

Sensitivity to Initialization: We carefully initialized parameters of all models except CAGE . With random initialization all models without guides (Snorkel and CAGE$_{-\text{G}}$) provide very poor accuracy. Exact numbers are in the supplementary material (*c.f.* Figure 3).

[1]Code available at https://github.com/oishik75/CAGE.

	Spouse	**CDR**	**Sms**	**Ion**
CAGE$_{-\text{C}-\text{G}+-\text{P}}$	0.48	0.69	0.34	0.81
CAGE$_{-\text{C}-\text{G}}$	0.48	0.69	0.34	0.81
CAGE$_{\text{-C,dataG}}$	0.48	0.69	0.34	0.81
CAGE$_{-\text{C}}$	**0.50**	**0.69**	**0.45**	**0.82**

Table 4: Comparing different methods of incorporating user's quality guide on discrete LFs.

Method of Enforcing Quality Guides In Table 4, we compare F1 for the following choices: (i) **CAGE$_{-\text{C}-\text{G}+-\text{P}}$**: Our model with the objective in Eqn (6) augmented with the sign penalty $\max(0, \theta_{jy \neq k_j} - \theta_{jk_j})$ instead of the regularizer in Eqn (9). (ii) **CAGE$_{-\text{C}-\text{G}}$**: Our model without guides (iii) **CAGE$_{\text{-C,dataG}}$**: The data-driven method of incorporating quality guides (See Section 2.1), and (iv) **CAGE$_{-\text{C}}$**: our data independent regularizer of the model's marginals with q_j^t (Eqn 9). From Table 4 we observe that CAGE$_{-\text{C}}$ is the only one that provides reliable gains. Thus, it is not just enough to get quality guides from users, we need to also design sound methods of combining them in likelihood training.

3.4 Structure of the Potentials

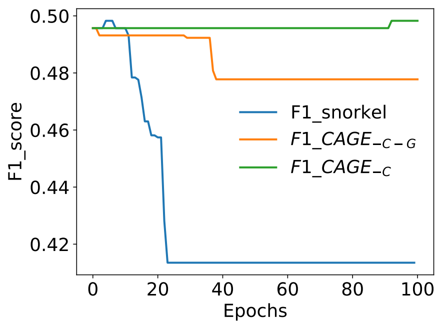
In defining the joint distribution $P_{\theta,\pi}(y, \tau_i, \mathbf{s}_i)$ (Eq 1) we used undirected globally normalized potentials for the discrete LFs(Eqn 2). We compare with an alternative where our joint is a pure directed Bayesian network with potentials $Pr(\tau_j|y) = \exp(\theta_{jy}/(1 + \exp(\theta_{jy})$ on each discrete LF and a class prior $\Pr(y)$. We observe that the undirected model is better able to capture interaction among the LFs with the global normalization Z_θ.

	Spouse	**CDR**	**Sms**	**Ion**	**Iris**
Directed	0.15	0.49	0.59	0.86	**0.89**
CAGE	**0.58**	**0.69**	**0.54**	**0.97**	0.87

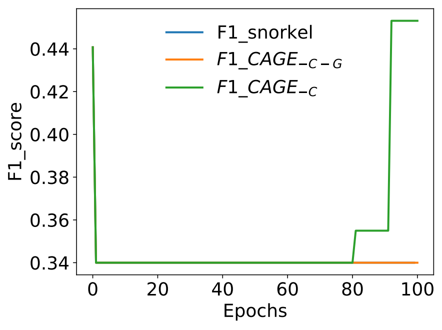
We repeated other ablation experiments where the continuous potentials are undirected and take various forms. The results appear in the Supplementary in Sect 7 and show that local normalization is crucially important for modeling s_j of continuous LFs.

Related Work

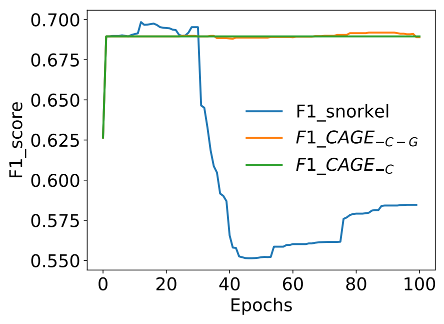
Several consensus-based prediction combination algorithms (Gao et al. 2009; Kulkarni et al. 2018) exist that



(a) Spouse



(b) SMS



(c) CDR

Figure 1: F1 with increasing number of training epochs compared across snorkel, CAGE$_{-\mathrm{C}-\mathrm{G}}$and CAGE$_{-\mathrm{C}}$, for three datasets. For each dataset, in the absence of guides, we observe unpredictable variation in test F1 as training progresses.

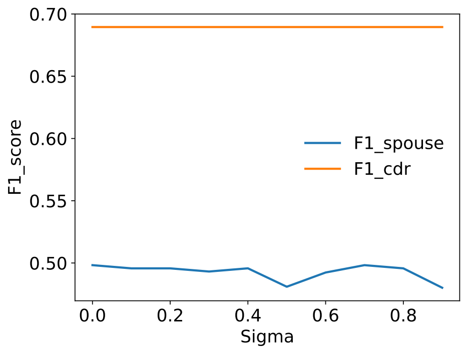


Figure 2: F1 with increasing distortion in the guess of the LF quality guide, q_j^t.

combine multiple model predictions to counteract the effects of data quality and model bias. There also exist label embedding approaches from the extreme classification literature (Yeh et al. 2017) that exploit inter-label correlation. While these approaches assume that the imperfect labeler's knowledge is fixed, (Fang et al. 2012) present a self-taught active learning paradigm, where a crowd of imperfect labelers learn complementary knowledge from each other. However, they use instance-wise reliability of labelers to query only the most reliable labeler without any notion of consensus. A recent work by (Chang, Amershi, and Kamar 2017) presents a collaborative crowd sourcing approach. However, they are motivated by the problem of eliminating the burden of defining labeling guidelines a priori and their approach harnesses the labeling disagreements to identify ambiguous concepts and create semantically rich structures for post-hoc label decisions.

There is work in the crowd-labeling literature that makes use of many imperfect labelers (Kulkarni et al. 2018; Raykar et al. 2010; Yan et al. 2011; Dekel and Shamir 2009) and accounts for both labeler and model uncertainty to propose probabilistic solutions to (a) adapt conventional supervised learning algorithms to learn from multiple subjective labels; (b) evaluate them in the absence of absolute gold standard; (c) estimate reliability of labelers. (Donmez and Carbonell 2008) propose a *proactive learning* method that jointly selects the optimal labeler and instance with a decision theoretic approach. Some recent literature has also studied the augmenting neural networks with rules in first order logic to either guide the individual layers (Li and Srikumar 2019) or train model weights within constraints of the rule based system using a student and teacher model (Hu et al. 2016)

Snorkel (Ratner et al. 2016; Bach et al. 2017; Ratner et al. 2017; Hancock et al. 2018; Varma et al. 2019) relies on domain experts manually developing heuristic and noisy LFs. Similar methods that rely on imperfect sources of labels are (Bunescu and Mooney 2007; Hearst 1992) relying on heuristics, (Mintz et al. 2009) on distant supervision and (Jawanpuria, Nath, and Ramakrishnan 2015) on learning conjunctions discrete of (discrete) rules. The aforementioned literature focuses exclusively on labeling suggestions that are discrete. We present a generalized generative model to aggregate heuristic labels from continuous (and discrete) LFs while also incorporating user accuracy priors.

Conclusion We presented a data programming paradigm that lets the user specify labeling functions which when triggered on instances can also produce continuous scores. The unsupervised task of consolidating weak labels is inherently unstable and sensitive to parameter initialization and training epochs. Instead of depending on un-interpretable hyperparameters which can only be tuned with labeled validation data which we assume is unavailable, we let the user guide the training with interpretable quality guesses. We carefully designed the potentials and the training process to give the user more interpretable control.

Acknowledgements This research was partly sponsored by a Google India AI/ML Research Award and partly sponsored by IBM Research, India (specifically the IBM AI Horizon Networks - IIT Bombay initiative).

References

[Bach et al. 2017] Bach, S. H.; He, B. D.; Ratner, A.; and Ré, C. 2017. Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 273–282.

[Bunescu and Mooney 2007] Bunescu, R. C., and Mooney, R. J. 2007. Learning to extract relations from the web using minimal supervision. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

[cdr 2018] 2018. Chemical - Disease Relation Extraction Task. https://github.com/HazyResearch/snorkel/tree/master/tutorials/cdr. [Online; accessed 31-March-2018].

[Chang, Amershi, and Kamar 2017] Chang, J. C.; Amershi, S.; and Kamar, E. 2017. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI), Denver, CO, USA*, 2334–2346.

[Dekel and Shamir 2009] Dekel, O., and Shamir, O. 2009. Good learners for evil teachers. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML), Montreal, Qubec, Canada*, 233–240.

[Donmez and Carbonell 2008] Donmez, P., and Carbonell, J. G. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM), Napa Valley, California, USA*, 619–628.

[Fang et al. 2012] Fang, M.; Zhu, X.; Li, B.; Ding, W.; and Wu, X. 2012. Self-taught active learning from crowds. In *12th IEEE International Conference on Data Mining (ICDM), Brussels, Belgium*, 858–863.

[Gao et al. 2009] Gao, J.; Liang, F.; Fan, W.; Sun, Y.; and Han, J. 2009. Graph-based consensus maximization among multiple supervised and unsupervised models. In *23rd Annual Conference on Neural Information Processing Systems (NIPS), Vancouver, Canada*, 585–593.

[Hancock et al. 2018] Hancock, B.; Varma, P.; Wang, S.; Bringmann, M.; Liang, P.; and Ré, C. 2018. Training classifiers with natural language explanations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), Melbourne, Australia*, 1884–1895.

[Hearst 1992] Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics, COLING, Nantes, France*, 539–545.

[Hu et al. 2016] Hu, Z.; Ma, X.; Liu, Z.; Hovy, E. H.; and Xing, E. P. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany*.

[Jawanpuria, Nath, and Ramakrishnan 2015] Jawanpuria, P.; Nath, J. S.; and Ramakrishnan, G. 2015. Generalized hierarchical kernel learning. *Journal of Machine Learning Research* 16:617–652.

[Kulkarni et al. 2018] Kulkarni, A.; Uppalapati, N. R.; Singh, P.; and Ramakrishnan, G. 2018. An interactive multi-label consensus labeling model for multiple labeler judgments. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI), New Orleans, Louisiana, USA*, 1479–1486.

[Li and Srikumar 2019] Li, T., and Srikumar, V. 2019. Augmenting neural networks with first-order logic. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL), Florence, Italy*, 292–302.

[Mikolov et al. 2013] Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. In *Workshop Track Proceedings of 1st International Conference on Learning Representations, (ICLR), Scottsdale, Arizona, USA*.

[Mintz et al. 2009] Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing (IJCNLP), Singapore*, 1003–1011.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar*, 1532–1543.

[Ratner et al. 2016] Ratner, A. J.; Sa, C. D.; Wu, S.; Selsam, D.; and Ré, C. 2016. Data programming: Creating large training sets, quickly. In *Proceedings of Annual Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain*, 3567–3575.

[Ratner et al. 2017] Ratner, A.; Bach, S. H.; Ehrenberg, H. R.; Fries, J. A.; Wu, S.; and Ré, C. 2017. Snorkel: Rapid training data creation with weak supervision. *PVLDB* 11(3):269–282.

[Raykar et al. 2010] Raykar, V. C.; Yu, S.; Zhao, L. H.; Valadez, G. H.; Florin, C.; Bogoni, L.; and Moy, L. 2010. Learning from crowds. *The Journal of Machine Learning Research* 11:1297–1322.

[sms] Sms spam collection data set. http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/.

[spo] Spouse Relation Extraction Task. https://github.com/HazyResearch/snorkel/tree/master/tutorials/intro.

[Varma et al. 2019] Varma, P.; Sala, F.; He, A.; Ratner, A.; and Ré, C. 2019. Learning dependency structures for weak supervision models. In *Proceedings of the 36th International Conference on Machine Learning (ICML), Long Beach, California, USA*, 6418–6427.

[Yan et al. 2011] Yan, Y.; Rosales, R.; Fung, G.; and Dy, J. G. 2011. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning (ICML), Bellevue, Washington, USA*, 1161–1168.

[Yeh et al. 2017] Yeh, C.; Wu, W.; Ko, W.; and Wang, Y. F. 2017. Learning deep latent space for multi-label classification. In *Proceedings of the Thirty-First AAAI Conference* *on Artificial Intelligence, San Francisco, California, USA*, 2838–2844.

Data Programming using Continuous and Quality-Guided Labeling Functions (Supplementary Material)

4 Challenges in modeling continuous LFs: Illustration through an Example

Sentence id	Discrete		Continuous	
	LF1 (+1)	LF2 (-1)	LF1 (+1)	LF2 (-1)
S_1 (+1)	1	0	$1,\ 1.0$	$1,\ 0.84$
S_2 (-1)	0	1	$1,\ 0.75$	$1,\ 1.0$

Table 5: Triggering of the continuous and discrete versions of functions LF1 and LF2 for the two example sentences S_1 and S_2.

In this section, we highlight some challenges that we attempt to address while modeling continuous LFs. We illustrate the challenges through two example sentences. Beside each sentence id, we state the value of the true label (± 1). While the candidate pair in S_1 is an instance of the spouse relationship, the pair in S_2 is not:
$\langle S_1, +1\rangle$: It's no secret that Brandi Glanville's relationship with ex-husband <Eddie Cibrian> and his wife, <LeAnn Rimes>, has been a strained one – but Glanville tells ETonline it's getting better.
$\langle S_2, -1\rangle$: Afterwards, <Christian Brady>, Dean of the Schreyer Honors College and father of <Mack Brady>, whom the game is named after, addressed the lacrosse team.

In Table 5, we present the outputs of both discrete and continuous LFs on the examples S_1 and S_2. For S_1, in the discrete case, correct consensus can easily be performed to output the true label +1 as LF1 (designed for class +1) gets triggered whereas LF2 (designed for class -1) is not triggered. correct consensus is challenging since both LF1 and LF2 produce non-trivial scores. Let us say that we threhold the score of each continuous LF to be able to mimic the discrete case; *e.g.*, score above the threshold will mean that the LF is triggered and otherwise, not. However, the threshold would depend on the particular LF (possibly 0.75 for LF1 and 0.84 for LF2), and is tricky to estimate in an unsupervised setup such as ours. To address this challenge, we significantly extend and modify the existing generative model (Ratner et al. 2016; Bach et al. 2017) to support continuous LFs. In addition to modeling the triggering distribution of each LF (continuous or discrete), we also model the distribution of scores for each continuous LF.

5 Robustness to Parameter Initialization

We trained our models with random Gaussian (0,0.1) initialization and compared with our agreeing initialization. In Figure 3 we show how CAGE$_{-\text{C}}$is able to recover from any initialization whereas methods without guides fare even worse with random initialization.

6 Labeling Functions employed in the SMS-Spam dataset

The SMS Spam dataset[2] is a collection of raw SMS text. The task is to classify each SMS as spam or not. We employed 9 discrete and 6 continuous labeling functions. We briefly discuss each labeling function that we employed in this task, beginning with the discrete labeling functions:

1. Three labeling functions based on the presence of trigger words which are highly likely to indicate spam. Each labeling function below, associated with the 'spam' class is triggered if any of the words in the SMS matches at least one of the words associated with that LF

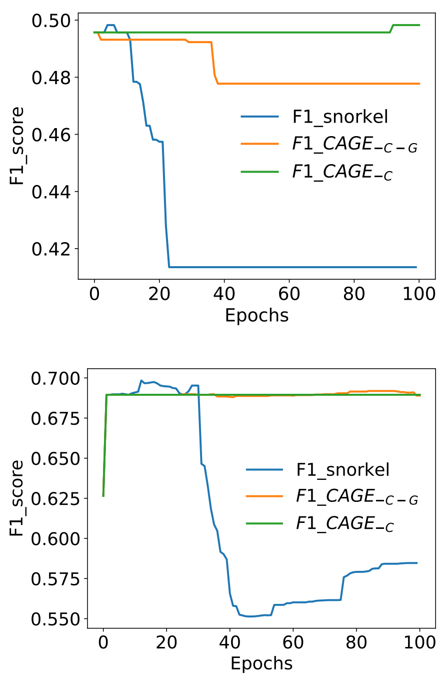
   (a) LF1: matches one of the trigger words in the set trigWord1 = {'free','credit','cheap','apply','buy','attention','shop','sex','soon', 'now','spam'}.

   ```
   def LF1(c):
    return (1,1) if len(trigWord1.intersection(c['text'].split())) > 0
    else (0,0)
   ```

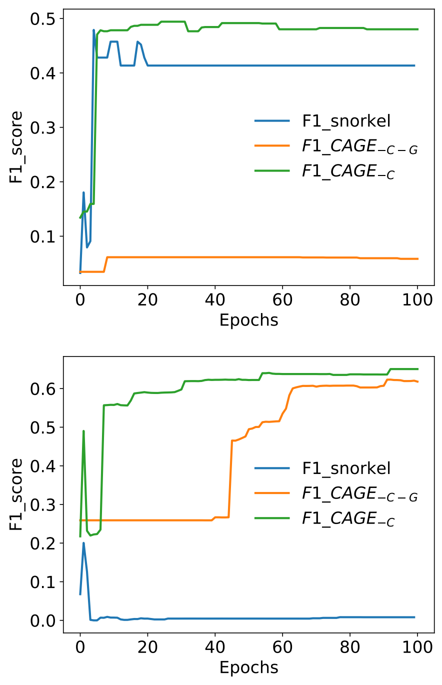
   (b) LF2: matches one of the trigger words in the set trigWord2 = {'gift','click','new','online','discount','earn','miss','hesitate', 'exclusive','urgent'}.

   ```
   def LF2(c):
    return (1,1) if len(trigWord2.intersection(c['text'].split())) > 0
    else (0,0)
   ```

[2] http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/



(a) Agreeing initialization



(b) Random Initialization.

Figure 3: F1 with increasing number of training epochs compared across snorkel, CAGE$_{-C-G}$and CAGE$_{-C}$, for two datasets: Spouse (top-row) and CDR(bottom-row)

(c) LF3: matches one of the trigger words in the set trigWord3 = {’cash’,’refund’,’insurance’,’money’,’guaranteed’,’save’,’win’,’teen’, ’weight’,’hair’}.

```
def LF3(c):
 return (1,1) if len(trigWord3.intersection(c[’text’].split())) > 0
 else (0,0)
```

2. Two labeling functions associated with the ‘spam’ class based on the presence of words/phrases in other parts of speech along with some **context**, which are highly likely to indicate spam:

(a) LF4: matches inquiries in the **contex** of a person’s being free or not free along with some overlap with the set of words notFreeWords = {’toll’,’Toll’,’freely’,’call’,’meet’,’talk’,’feedback’},

```
def LF4(c):
 return (-1,1) if ’free’ in c[’text’].split() and
  len(notFreeWords.intersection(c[’text’].split()))>0
 else (0,0)
```

(b) LF5: matches inquiries in the **contex** of a person’s being free or not free along with substring match with the set of words/phrases notFreeSubstring = {’not free’,’you are’,’when’,’wen’},

```
def LF4(c):
 return (-1,1) if ’free’ in c[’text’].split() and
  re.search(ltp(notFreeSubstring),c[’text’], flags= re.I)
 else (0,0)
```

3. Two labeling functions associated with the ‘spam’ class based on the presence of first, second or third person keyword matches:

(a) LF6: matches first and second person keywords in firstAndSecondPersonWords = {’I’,’i’,’u’,’you’,’ur’,’your’,’our’,’we’,’us’,’youŕe,’},

```
def LF6(c):
 return (-1,1) if ’free’ in c[’text’].split() and
  len(person1.intersection(c[’text’].split()))>0
 else (0,0)
```

(b) LF7: matches third person keywords in thirdPersonWords = {'He','he','She','she','they','They','Them','them','their','Their'},

```
def LF7(c):
 return (-1,1) if 'free' in c['text'].split() and
  len(thirdPersonWords.intersection(c['text'].split()))>0
 else (0,0)
```

4. Function based on text characteristics such as number of capitalized characters. The function below is triggered if the number of capital characters exceeds a threshold (6 in this case).

```
def LF8(c):
 return (1,1)
  if (sum(1 for ch in c['text'] if ch.isupper()) > 6)
 else (0,0)
```

5. Class Prior labeling function: The function below is always triggered for the negative class and serves as a class prior.

```
def LF9(c):
 return (-1,1)
```

All of our continuous labeling functions are continuous versions of the discrete labeling functions described above. For those discrete labeling functions that match dictionary entries, the continuous counterpart computes the maximum word vector based similarity of the text (or textual context) across all entires in the dictionary. They are listed below:

(i) Three labeling functions based on the presence of trigger words which are highly likely to indicate spam:

(a) CLF1: Continuous version of LF1:

```
def CLF1(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in trigWord1:
  sc=max(sc,get_similarity(word_vectors,w))
 return (1,sc)
```

(b) CLF2: Continuous version of LF2:

```
def CLF2(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in trigWord2:
  sc=max(sc,get_similarity(word_vectors,w))
 return (1,sc)
```

(c) CLF3: Continuous version of LF3:

```
def CLF3(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in trigWord3:
  sc=max(sc,get_similarity(word_vectors,w))
 return (1,sc)
```

(ii) Two labeling functions associated with the 'spam' class based on the presence of words/phrases in other parts of speech along with some **context**, which are highly likely to indicate spam:

(a) CLF4: Continuous version of LF4:

```
def CLF4(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in notFreeWords:
  sc=max(sc,get_similarity(word_vectors,w))
 return (1,sc)
```

(b) CLF5: Continuous version of LF5:

```
def CLF5(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in notFreeSubstring:
    sc=max(sc,get_similarity(word_vectors,w))
   return (1,sc)
```

(iii) Two labeling functions associated with the 'spam' class based on the presence of first, second or third person keyword matches:

  (a) CLF6: Continuous version of LF6:

```
def CLF6(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in firstAndSecondPersonWords:
  sc=max(sc,get_similarity(word_vectors,w))
 return (-1,sc)
```

  (b) CLF7: Continuous version of LF7

```
def CLF7(c):
 sc = 0
 word_vectors = get_word_vectors(c['text'].split())
 for w in thirdPersonWords:
  sc=max(sc,get_similarity(word_vectors,w))
 return (-1,sc)
```

(iv) Continuous version of LF8; the value it returns increases with the number of capital characters:

```
def CLF8(c):
 l = sum(1 for ch in c['text'] if ch.isupper())
 return (1, 1-np.exp(float(-l/2)))
```

(v) Class Prior labeling function: This remains the same as before.

```
def CLF8(c):
 return (-1,1)
```

7 Form of continuous potential

We explored many different generic forms of potentials for continuous LFs and we compare them with the specific beta-distribution form. These potentials $\psi(s_j, \tau_j, y)$ return a score increasing in s_j when there is agreement, and decreasing in s_j when there is disagreement (for positive θ_j):

1. Treat s as a weight: $\theta_{jy}\tau_j s_j$.
2. Thresholded: $\theta_{jy}\tau_j \max(s_j - \pi_j, 0)$
3. Thresholded Sigmoid: $\theta_{jy}\tau_j \text{sigmoid}(s_j - \pi_j)$
4. Logit (Global conditioning on Beta): $\theta_{jy}\tau_j \log \frac{s}{1-s}$
5. Half clipped Gaussian instead of Beta with the Gaussian mean at '1' when τ_j and y agree, and at 0 when they disagree. The only learned parameter is then the variance of the Gaussian which we represent as π_{jy}.

$$P(s_{ij}|y, l_{ij}, k_j) = \begin{cases} HG(1 - s_{ij}; \text{scale} = \theta_{jy0}) & \text{if } k_j = y \;\&\; l_{ij} \neq 0, \\ HG(s_{ij}; \text{scale} = \theta_{jy1}) & \text{if } k_j \neq y \;\&\; l_{ij} \neq 0, \\ 1 & \text{otherwise.} \end{cases} \quad (11)$$

In Table 6 we compare accuracy with these alternative forms of potentials. We appreciate the difficulty of choosing the right continuous potentials by observing that for most of these other choices the accuracy is really poor.

8 Example: Failure in spite of good initialization of the Snorkel Model

Consider another dataset consisting of n LFs that are just slightly better than random, that is their agreement with the true y when they trigger is just (0.5+ϵ)*100%. A simple majority voting on the noisy LF labels will give nearly 100% accuracy when n is large. On the other hand, with the Snorkel model even with favorable initializations ($\theta_j = 1, \forall j$), the final values of the parameters on covergence is often poor. In most training runs, the accuracy dropped to 0 after a few training iterations. Our decoupled model is able to perform well in this case without accuracy guides.

$\psi(s_j, \tau_j, y)$	**Spouse**	**CDR**	**Sms**	**Dedup**	**Ionosphere**	**Iris**
$\theta_{jy}\tau_j s_j$	0	0.49	0	0.26	0.96	0.87
$\theta_{jy}\tau_j \max(s_j - \pi_j, 0)$	0	0.49	0	0	0.83	0.87
$\theta_{jy}\tau_j \text{sigmoid}(s_j - \pi_j)$	0	0.49	0	0	0.80	0.87
$\theta_{jy}\tau_j \log \frac{s}{1-s}$	0.46	0	0	0	0	0.44
CAGE	0.58	0.61	0.54	0.79	0.97	0.87

Table 6: Comparison of Continuous Potentials

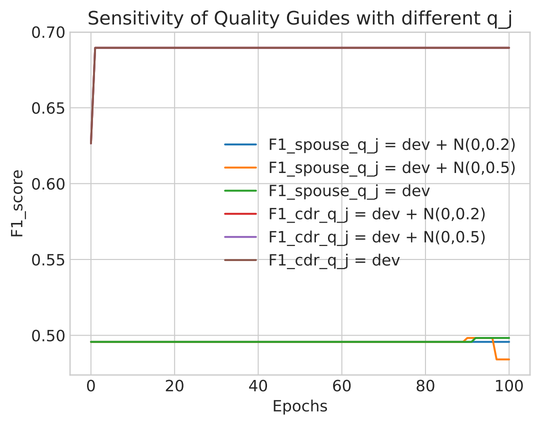


Figure 4: F1 with increasing training epochs for three settings of q_j^c: (1) where $q_j^c = 0.9$ for all LFs, (2) where $q_j^c = 0.8$ for all LFs, and (3) where q_j^c = true accuracy of LF λ_j + a Gaussian noise of variance 0.2. We observe that across two different datasets, the accuracy remains the same and stable for all three quality guides. Without the guides, we already saw in Figure 1 of the main paper how the F1 swings with training epochs.

9 Results by training Discriminative Classifier

Data Programming is hinged upon creation of probabilistic training labels for any discriminative model with a standard loss function. In Table 7, we report precision, recall and F1 scores on the test data by training a logistic regression classifier on labeled data created by Snorkel as well as our generative model CAGE . A takeaway of the following results is that, through the use of continuous LFs and quality guides, the discriminative model generalizes much better beyond the heuristics encoded in the LFs.

	Spouse			**CDR**			**Sms**		
	P	R	F1	P	R	F1	P	R	F1
Snorkel	0.30	0.66	0.41	0.41	0.98	0.58	0.20	0.96	0.34
CAGE	0.65	0.47	0.55	0.58	0.86	0.69	0.52	0.66	0.58

Table 7: Discriminative Classifier trained on data labeled using the generative models and evaluated on held-out test data.