Data Programming using Semi-Supervision and Subset Selection

Ayush Maheshwari[†], Oishik Chatterjee[†], KrishnaTeja Killamsetty[*], Rishabh Iyer[*], Ganesh Ramakrishnan[†]
[†] Department of CSE, IIT Bombay, India
[*] The University of Texas at Dallas

Abstract

The paradigm of data programming (Bach et al. 2019) has shown a lot of promise in using weak supervision in the form of rules and labelling functions to learn in scenarios where labelled data is not available. Another approach which has shown a lot of promise is that of semi-supervised learning where we augment small amounts of labelled data with a large unlabelled dataset. In this work, we argue that by not using any labelled data, data programming based approaches can yield sub-optimal performance, particularly, in cases when the labelling functions are noisy. The first contribution of this work is to study a framework of joint learning which combines un-supervised consensus from labelling functions with semi-supervised learning and *jointly learns a model* to efficiently use the rules/labelling functions along with semi-supervised loss functions on the feature space. Next, we also study a subset selection approach to *select* the set of examples which can be used as the labelled set. We evaluate our techniques on synthetic data as well as four publicly available datasets and show improvement over state-of-the-art techniques[1].

Introduction

Modern machine learning techniques rely excessively on large amounts of labelled training data. Supervised learning approaches have utilised such large amounts of labelled data and this has resulted in huge successes in the last decade. However, acquisition of labelled data, in most cases, entails a pain-staking process requiring human supervision. Several techniques such as active learning, distant supervision, crowd-consensus learning, and semi-supervised learning have been proposed to reduce the *annotation cost* (Settles, Craven, and Friedland 2008). However, clean annotated labels continue to be critical for reliable results (Bach et al. 2019; Goh et al. 2018).

Recently, (Ratner et al. 2016) proposed a paradigm on data programming where several labelling functions (LF) written by humans, are used to weakly associate labels with the instances. In data programming, users encode the weak supervision in the form of labelling functions. On the other hand, traditional semi-supervised learning methods combine a small amount of labelled data with large unlabelled data (Kingma et al. 2014). In this paper, we leverage semi-supervision in the feature space for more effective data programming using labelling functions.

Id	Description
LF1	If `http` or `https` in comment text, then return +1 otherwise ABSTAIN (return 0)
LF2	If length of comment is less than 5 words, then return -1 otherwise ABSTAIN (return 0).(Non spam comments are often short)
LF3	If comment contains `my channel` or `my video`, then return +1 otherwise ABSTAIN (return 0).

Table 1: Three LFs based on keyword lookups or regex expression for the *YouTube* spam classification task

We illustrate such LFs on one of the four tasks that we experimented with, *viz.*, that of identifying spam/no-spam comments in the YouTube reviews. For some applications, writing LFs are often as simple as keyword lookups or a regex expression. In this specific case, the users construct heuristics patterns as LFs for identifying a comment as a spam or not a spam (ham). Each LF takes a comment as an input and provides a binary label as the output; +1 indicates that comment is a spam, -1 means no spam comment and 0 means that LF is unable to assert anything for this example. Based on the lists of common words and regex that indicate the comment as spam or non-spam, we present a subset of such LFs in Table 1.

In isolation, each LF may neither be always correct nor complete. LFs may also produce conflicting labels. For the purpose of illustration, consider a non-spam comment `For the previous version of my video please see https://somesite.com`. LF1 and LF3 in Table 1 assign a label +1 whereas LF2 assigns label 0 (Abstain). In the past, generative models such as Snorkel (Ratner et al. 2016) and CAGE (Chatterjee, Ramakrishnan, and Sarawagi 2019) have been discussed 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

[1]Source code of the paper at `https://github.com/ayushbits/Semi-Supervised-LFs-Subset-Selection`

Training data		LF outputs		Features	
id	label	LF1 (+1)	LF2 (-1)	F1	F2
S_1	-	1	0	0.96	0.84
S_2	-	0	1	0.75	0.99
S_3	-1	0	0	0.65	0.96
S_4	+1	1	1	0.95	0.77
Test data					
S_5	-1	0	1	0.70	0.96
S_6	+1	0	0	0.99	0.80

Table 2: Triggering of the labelling functions LF1 and LF2 as well as value of features F1 and F2 for the four example sentences in the training set, *viz.*, S_1, S_2, S_3 and S_4 as well as the two sentences S_5 and S_6 in the test set. While S_1 and S_2 are unlabelled, we illustrate how, when the labels of S_3 and S_4 are known, we can better predict labels on the seen training instances as well as unseen test instances S_5 and S_6.

supervised model/classifier and evaluated on a test set.

We will next highlight a challenge that we attempt to address in doing data programming using only LFs. For each of the following sentences $S_1 \ldots S_6$, that can constitute *an observed set of training instances*, we state the value of the true label (± 1). While the candidates in S_1 and S_4 are instances of the non-spam comment, the ones in S_2 and S_3 are not:

1. $\langle S_1, +1 \rangle$: Please help me go to college guys! Thanks from the bottom of my heart. https://www.indiegogo.com/projects/
2. $\langle S_2, -1 \rangle$: Are those real animals
3. $\langle S_3, -1 \rangle$:this video is very inaccurate, a tiger would rip her face of
4. $\langle S_4, +1 \rangle$: Get free gift http://swagbucks.com/p/register

Further, let us say we have a completely *unseen set of test instances*, S_5 and S_6, whose labels we would also like to predict effectively:

5. $\langle S_5, -1 \rangle$: prehistoric song..has been
6. $\langle S_6, +1 \rangle$: Come check out our parody of this!

In Table 2, we present the outputs of the LFs as well as some embedding-based features F1 and F2 on the observed training examples S_1, S_2, S_3 and S_4 as well as the unseen test examples S_5 and S_6. For S_1, 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. Similarly, for S_2, LF2 gets triggered whereas LF1 is not, making the correct consensus easy to perform. Hence, we have treated S_1 and S_2 as unlabelled, indicating that we could learn a model based on LFs alone without supervision if all we observed were these two examples and the outputs of LF1 and LF2. However, correct consensus on S_3 and S_4 is challenging since LF1 and LF2 either both fire or both do not. One possible way out could be that an oracle could help us threshold the score of each of the two features F1 and F2 and convert them into LFs that are able to mimic LF1 and LF2 respectively on S_1 and S_2 and therefore help improve consensus on S_3 and S_4; However, how could such a threshold be determined for each LF (possibly 0.65 for F1 and 0.77 for F2)? We can easily see that correlating feature values with LF outputs is tricky to estimate in a completely unsupervised setup. To address this issue, we ask the following questions:

- **(A)** What if were provided access to the true labels of a small subset of instances - in this case only S_3 and S_4? Could the (i) correlation of features values (*eg.* F1 and F2) with labels (*eg.* +1 and -1 respectively), modelled via a small set of labelled instances (*eg.* S_3 and S_4), in conjunction with (ii) the correlation of feature values (*eg.* F1 and F2) with LFs (*eg.* LF1 and LF2) modelled via a potentially larger set of unlabelled instances (*eg.* S_1, S_2), help improved prediction of labels for hitherto unseen test instances S_5 and S_6?
- **(B)** Could we determine precisely that size constrained subset of the training set, knowledge of whose true labels would help us train a model that is most effective on the test set?

As a solution to (A), we present a new formulation in which the parameters over features and LFs are jointly trained in a semi-supervised manner. As for (B), we present subset selection algorithms, with a recipe that recommends the sub-set of the data, which if labelled, would most benefit the joint learning framework.

Related Work and Our Contributions

Related Work

In this section, we review related work on data programming, learning with rules, unsupervised learning with labelling functions, semi-supervised learning and data subset selection.

Data Programming and Unsupervised Learning Snorkel (Ratner et al. 2016) has been proposed as a generative model to determine correct label probability using consensus on the noisy and conflicting labels assigned by the discrete LFs. CAGE (Chatterjee, Ramakrishnan, and Sarawagi 2019) has been proposed as a graphical model that used continuous valued LFs with scores obtained using soft match techniques such as cosine similarity of word vectors, TF-IDF score, distance among entity pairs, *etc*. Owing to its generative model, Snorkel is highly sensitive to initialisation and hyper-parameters. On the other hand, the CAGE model introduced user-controlled quality guides that incorporates labeller intuition into the model. However, these models treat LFs as complete black boxes and completely disregard feature information that could provide additional information to learn the (graphical) model. Both these models try to learn a combined model for the labelling functions in an unsupervised manner. However, in practical scenarios, some labelled data is always available (or could be made available by labelling a few instances), and hence, a completely unsupervised approach might not be the best solution. In this work, we augment these data programming approaches by designing a semi-supervised model that

incorporates feature information and LFs to jointly learn the parameters. Hu (Hu et al. 2016) proposed a student and teacher model that transfers rule information into the weight parameters of the neural network . However, they assign linear weight to each rule based on an agreement objective. The model we propose in this paper jointly learns paraemters over features and rules in a semi-supervised manner rather than just weigh their outputs and can therefore be more powerful. Several approaches exist that programatically create training sets - these include distant supervision, crowdsourcing, co-training, boosting, *etc*. Distant supervision (Craven, Kumlien, and others 1999; Mintz et al. 2009; Bunescu and Mooney 2007; Mallory et al. 2016) uses existing information from knowledge bases heuristically mapped to an unlabelled data set, often employing additional heuristic patterns to generate label training data from the unlabelled data. However, these approaches could be sensitive to noise in the data.

Semi-Supervised Learning Semi-supervised methods are of immense interest in applications such as image search (Fergus, Weiss, and Torralba 2009), natural language processing (Liang 2005), *etc.*, where unlabelled data is abundant, whereas obtaining class labels is expensive. These include approaches such as self-training (Rosenberg, Hebert, and Schneiderman 2005) in which the model is bootstrapped with samples of highly confidence predictions. These methods can be prone to error since poor confident predictions could get reinforced. Transductive SVMs (Joachims 1999) extend SVM with the aim of keeping unlabelled instances distant from the margin, however, efficient optimisation is a concern for large amounts of unlabelled data. Other approaches based on neural networks combine unsupervised and supervised learning by training with an additional penalty from an auto-encoder (Ranzato and Szummer 2008). Semi-supervised learning has also found applications in deep learning, such as in the image classification setting (Oliver et al. 2018).

Semi-Supervised Data Programming The only work, which to our knowledge, combines rules with supervised learning in a joint framework is work by (Awasthi et al. 2020). In their approach, they leverage both rules and labelled data by associating each rule with exemplars of correct firings (*i.e.*, instantiations) of that rule. Their joint training algorithms denoises over-generalized rules and trains a classification model. Our approach differs from their paper in two ways: a) we do not have information of rule exemplars and b) we employ a semi-supervised framework combined with graphical model for consensus amongst the LFs to train our model. Finally, as discussed in the next section, we also study how to automatically select the seed set of labeled data, rather than having a human provide this seed set, as was done in (Awasthi et al. 2020).

Data Subset Selection Finally, another approach which has been gaining a lot of attention recently is data subset selection. The specific application of data subset selection depends on the goal at hand. The main applications of data subset selection have been: i) reduction of end to end training time, (ii) reduction of labeling cost and time, and (iii) faster adaptation and better generalization. Most of the related work on data subset selection and core-sets has focused on the goal of reducing the training time. Existing techniques perform supervised data subset (*i.e.*, with knowledge of the labels) selection by approximating gradients in gradient-based algorithms (Mirzasoleiman, Bilmes, and Leskovec 2019) or approximating the loss functions in specialised models such as SVMs (Tsang, Kwok, and Cheung 2005), k-means and k-mediods (Har-Peled and Kushal 2007) and other unsupervised clustering algorithms (Bachem, Lucic, and Krause 2017). Other techniques include use of proxy functions or models for data selection (Wei et al. 2013; Coleman et al. 2018; Kaushal et al. 2019; Wei, Iyer, and Bilmes 2015). Data subset selection has also been applied to active learning, for selecting a subset of unlabelled data for labelling (Wei, Iyer, and Bilmes 2015; Sener and Savarese 2017) (*i.e.*, goal (ii) mentioned above). In this paper, we use the framework of data subset selection for *selecting* a subset of unlabelled data for obtaining labels. The goal of data selection considered here though is different from the data selection in active learning. In particular, we want to select a set of diverse examples such that the labelling functions perform very poorly on these examples. These examples would be the *most informative* from the perspective of the loss functions considered here.

Our Contributions

We now summarise our main contributions:
1) We present a novel formulation for jointly learning the parameters over features and Labelling Functions in a semi-supervised manner.
2) We jointly learn a parameterized graphical model and a classifier model to learn our overall objective.
3) We also study a subset selection approach to *select* the set of examples which can be used as the labelled set. We show, in particular, that through a principled data selection approach, we can achieve significantly higher accuracies than just randomly selecting the seed labelled set for semi-supervised learning with labelling functions.
4) Unlike the work of (Awasthi et al. 2020), we do not assume we have rule exemplars given, or that we are given a seed labelled set. Our framework can also leverage any state-of-the-art semi-supervised learning algorithm, though we focus mostly on Entropy Minimisation (Grandvalet and Bengio 2005) for the experiments in this paper.
5) Finally, we evaluate our model on four publicly available datasets from domains such as spam detection and record classification and show improvement over state-of-the-art techniques. We also draw insights from experiments in synthetic settings.

Methodology

Problem Description

Let X and Y be the feature space and label space respectively. We also have access to m labelling functions (LF) L_1

to L_m. The dataset consists of 2 components, *viz.*,

1. $L = \{(\mathbf{x}_1, y_1, \mathbf{l}_1), (\mathbf{x}_2, y_2, \mathbf{l}_2), \ldots, (\mathbf{x}_N, y_N, \mathbf{l}_N)\}$ which denotes the labelled dataset and
2. $U = \{(\mathbf{x}_{N+1}, \mathbf{l}_{N+1}), (\mathbf{x}_{N+2}, \mathbf{l}_{N+2}), \ldots, (\mathbf{x}_M, \mathbf{l}_M)\}$ which denotes the unlabelled dataset wherein $\mathbf{x}_i \in X$, $y_i \in Y$. Here, $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})$ denotes the firings of all the LFs on instance i. Each l_{ij} can be 1 or 0 where 1 indicates that the LF j has fired on the instance i and 0 indicates it has not. All the labelling functions are discrete and hence no continuous scores are associated with them.

Classification and Labelling Function Models

We have a feature-based classification model $f_\phi(\mathbf{x})$ which takes the features as input and predicts the class label. The output of this model is $p^f_\phi(y|\mathbf{x})$. *i.e.*, the probability of the classes given the input features. This model can be a simple classification model such as a logistic regression model or a simple neural network model.

We also use an LF-based graphical model $P_\theta(\mathbf{l}, y)$ which, as specified in equation (1), is a generative model on the LF outputs and the true class labels.

$$P_\theta(\mathbf{l}, y) = \frac{1}{Z_\theta} \prod_{j=1}^{m} \psi_\theta(l_j, y) \tag{1}$$

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

$$\begin{aligned} Z_\theta &= \sum_y \prod_j \sum_{l_j \in \{1,0\}} \psi_\theta(l_j, y) \\ &= \sum_{y \in Y} \prod_j (1 + \exp(\theta_{jy})) \end{aligned} \tag{3}$$

The graphical model is illustrated in Figure 1. The model makes the simple assumption that each LF independently acts on an instance to produce an output. The potentials ψ_θ invoked in equation (1) are defined in equation (2). There are K parameters for each LF, where K is the number of classes. This model can be thought of as the graphical model in CAGE (Chatterjee, Ramakrishnan, and Sarawagi 2019) restricted to the discrete labelling functions.

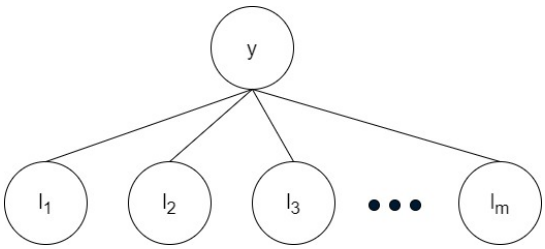

Figure 1: The LF-based Graphical model

We propose a joint learning algorithm with semi supervision to employ both features and LF predictions in an end-to-end manner.

Notation	Description
f_ϕ	The feature-based Model
p^f_ϕ	The label probabilities as per the feature-based model
P_θ	The label probabilities as per the LF-based Graphical Model
L	Cross Entropy Loss
H	Entropy function
g	Label Prediction from the LF-based graphical model
LL_s	Supervised negative log likelihood
LL_u	Unsupervised negative log likelihood summed over labels
KL	KL Divergence between two probability models
R	Quality Guide based loss

Table 3: Summary of notation used.

Training

We have the following objective which we want to minimise

$$\begin{aligned} \min_{\theta,\phi} & \sum_{i \in L} L(f_\phi(x_i), y_i) + \sum_{i \in U} H(f_\phi(x_i)) \\ & + \sum_{i \in U} L(f_\phi(x_i), g(l_i)) + LL_s(\theta|L) \\ & + LL_u(\theta|U) + \sum_{i \in U \cup L} KL(P_\theta(l_i), f_\phi(x_i)) + R(\theta|\{q_j\}) \end{aligned} \tag{4}$$

Before we proceed further, we refer the reader to Table 3 in which we summarise the notation built so far as well as the notation that we will soon introducing.

First Component (L1): The first component (L1) of the loss $L(f_\phi(x_i), y_i)$ is the standard cross-entropy loss on the labelled dataset for the model p^f_ϕ. It is defined to be: $L(f_\phi(x_i), y_i) = -\log\left(p^f_\phi(y = y_i|x_i)\right)$

Second Component (L2): The second component L2 is the semi-supervised loss on the unlabelled data. In our framework, we can use any unsupervised loss function. However, for this paper, we use the Entropy minimisation (Grandvalet and Bengio 2005) approach. Thus, our second component $H(f_\phi(x_i))$ is the entropy of the predictions on the unlabelled dataset. It acts as a form of semi-supervision by trying to increase the confidence of the predictions made by the model on the unlabelled dataset.

Third Component (L3): The third component $L(f_\phi(x_i), g(l_i))$ is the cross entropy of the classification model using the hypothesised labels from CAGE (Chatterjee, Ramakrishnan, and Sarawagi 2019). Denoting l_i as the output vector of all labelling functions on the i^{th} instance, we can specify the predicted label for the i^{th} instance using the LF-based graphical model as: $g(l_i) = \arg\max_y P_\theta(l_i, y)$

Fourth Component (L4): The fourth component $LL_s(\theta|L)$ is the (supervised) negative log likelihood loss on the labelled dataset as per equation (1): $LL_s(\theta|L) = -\sum_{i=1}^{N} \log P_\theta(\mathbf{l}_i, y_i)$

Fifth Component (L5): The fifth component $LL_u(\theta|U)$ is the negative log likelihood loss for the unlabelled dataset as per equation (1). Since the true label information is not available, the probabilities need to be summed over y, in a manner similar to the CAGE model.

$$LL_u(\theta|U) = -\sum_{i=N+1}^{M} \log \sum_{y \in Y} P_\theta(l_i, y)$$

Sixth Component (L6): The sixth component $KL(P_\theta(l_i), f_\phi(x_i))$ is the Kullback-Leibler (KL) divergence between the predictions of both the models, *viz.*, feature-based model f_ϕ and the LF-based graphical model P_θ over the entire dataset $U \cup L$. Through this term, we try and make the models agree in their predictions.

Quality Guides (QG): As a last component in our objective, we use quality guides $R(\theta|\{q_j\})$ on LFs which have been shown (Chatterjee, Ramakrishnan, and Sarawagi 2019) to stabilise the unsupervised likelihood training while using labelling functions. Let q_j be the fraction of cases where LF_i is correctly triggered. And let q_j^t be the user's belief on the fraction where y and l_j agree. If $P_\theta(y = k_j|l_j = k_j)$ is the model calculated precision over the LFs, the quality guide based loss can be written as follows:

$$R(\theta|\{q_j^t\}) = \sum_j q_j^t \log P_\theta(y = k_j|l_j = k_j) + (1 - q_j^t) \log(1 - P_\theta(y = k_j|l_j = k_j)) \quad (5)$$

Illustration of joint learning on a synthetic setting

Through a synthetic example depicted in Figure 2, we illustrate the effectiveness of our formulation of combining semi-supervised learning with labelling functions (i.e. combined Losses 1-6) to achieve superior performance. The example should also help illustrate how use of labelling function complements semi-supervised learning. Consider a 3-class classification problem with overlap in the feature space as depicted in Figure 2. The classes are A, B and C. Though we illustrate the synthetic setting in 2 dimensions, in reality, we performed similar experiments in three dimensions (and results were similar).

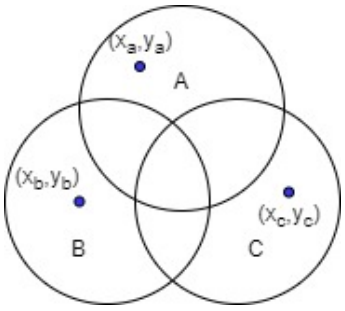

Figure 2: Synthetic data

We create labelling functions for each class as follows. We randomly pick 5 points from each class, and corresponding to each such point we create a labelling function based on its coordinates:

- LF_a: Consider the point (x_a, y_a). The corresponding LF will be: if $y \geq y_a$ return 1 (i.e. classify as class A) else return 0 (abstain).
- LF_b: Similarly for (x_b, y_b) the LF will return 1 if $x \leq x_b$ and else will return 0.
- LF_c: The LF corresponding to (x_c, y_c) will return 1 if $x \geq x_c$ and else will return 0.

These seemingly 15 weak labelling functions (5 for each class) actually aid in classification when the labelled example set is extremely small and the classifier is unable to get a good estimate of the class boundaries. This can be observed in Table 4 wherein we report the F1 score on a held out test dataset for models obtained by training on the different loss components. The results are reported in the case of three dimensions, wherein each circle was obtained as a 3-dimensional gaussian. The means for the three classes A, B and C were respectively, $(0, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$ and the variance for each class was set to $(1, 1, 1)$. The training and test sets had 1000 examples each, with roughly equal number of samples randomly generated from each class (gaussian). In the the first experiment (reported on the first row of Table 4), the training was performed on the L1 loss by treating the entire data-set of 1000 examples as labelled. In all the other experiments only 1% (10 examples with almost uniform distribution across the 3 classes) of the training set was considered to be labelled and the remaining (990 examples) were treated as unlabelled. We make the

Loss component used for training	F1 Score
L1 (on entire dataset as labelled)	0.584
L1 (1% labelled)	0.349
L1 (1% labelled) +L2	0.352
L1 (1% labelled)+L2+L3+L4 (1% labelled)+L5+L6+QG	**0.440**
L4 (1% labelled)+L5	0.28

Table 4: F1 score on a held out test subset in the synthetic setting for models obtained by training on the different loss components.

following important observations with respect to Table 4:

- **Skyline:** When the entire training data is treated as labelled and loss function L1 is minimized, we obtain a skyline model with F1 score of 0.584.
- With just 1% labelled data on L1, we achieve 0.349 F1 score (using only the labelled data).
- We obtain an F1 score of 0.28 using the labelling functions on the unlabelled data (for L5) in conjunction with the 1% labelled data (for L4).
- When the 1% labelled data (for L1) and the remaining observed unlabelled data (for L2) are used to train the semi-supervised model using L1+L2, an F1 score of 0.352 is obtained.
- However, by jointly learning on all the loss components, we observe an F1 score of 0.44. This is far better than the numbers obtained using only (semi)-supervised learning and those obtained using only the labelling functions. Understandably, this number is lower than the skyline of 0.584 mentioned on the first row of Table 4.

Data Subset Selection

Suppose we are given an unlabelled data set and a limited budget for data labelling because of the expensive costs involved in it. It is essential for us to be smart about choosing the data points that need to be labelled. We explore two subset selection strategies in this section, where we select a subset of data points from unlabelled data set and use it as labelled data such that we maximise the diversity of the feature set.
The two approaches are Unsupervised Facility Location and Supervised Facility Location. We complement both the approaches with Entropy Filtering (also described below).

Unsupervised Facility Location: In this approach, given a Unlabelled Data set V we want to select a subset X such that the selected subset has maximum diversity with respect to features. Inherently, we are trying to maximise the information gained by a machine learning model when trained on the subset selected. The objective function for Unsupervised facility location is:

$$f_{\text{unsup}}(X) = \sum_{i \in V} \max_{j \in X} s_{ij} \tag{6}$$

where s_{ij} denotes the similarity score between data instance i in unlabelled set V and data instance j in subset selected data X. We employ a lazy greedy strategy to select the best possible subset.

Supervised Facility Location : The objective function for Supervised Facility Location is:

$$f_{\text{sup}}(X) = \sum_{y \in Y} \sum_{i \in V_y} \max_{j \in X \cap V_y} s_{ij} \tag{7}$$

Assume $V_y \subseteq V$ is the subset of data points of hypothesised label y. In other words, V_y forms a partition based on the hypothesized labels.

Optimisation Algorithms and Submodularity: Both $f_{\text{unsup}}(X)$ and $f_{\text{sup}}(X)$ are submodular functions, and for data selection, we select a subset X of the unlabelled data, which maximises these functions under a cardinality budget (i.e. a labelling budget). For cardinality constrained maximisation, a simple greedy algorithm provides a near optimal solution (Nemhauser, Wolsey, and Fisher 1978). Starting with $X^0 = \emptyset$, we sequentially update $X^{t+1} = X^t \cup \text{argmax}_{j \in V \setminus X^t} f(j|X^t)$, where $f(j|X) = f(X \cup j) - f(X)$ is the gain of adding element j to set X. We run this till $t = k$ and $|X^t| = k$, where k is the budget constraint. It is easy to see that the complexity of the greedy algorithm is $O(nkT_f)$ where T_f is the complexity of evaluating the gain $f(j|X)$ for the supervised and unsupervised facility location functions. This simple greedy algorithm can be significantly optimized via a lazy greedy algorithm (Minoux 1978). The idea is that instead of recomputing $f(j|X^t), \forall j \notin^t$, we maintain a priority queue of sorted gains $\rho(j), \forall j \in V$. Initially $\rho(j)$ is set to $f(j), \forall j \in V$. The algorithm selects an element $j \notin X^t$, if $\rho(j) \geq f(j|X^t)$, we add j to X^t (thanks to submodularity). If $\rho(j) \leq f(j|X^t)$, we update $\rho(j)$ to $f(j|X^t)$ and re-sort the priority queue. The complexity of this algorithm is roughly $O(kn_R T_f)$, where n_R is the average number of re-sorts in each iteration. Note that $n_R \leq n$, while in practice, it is a constant thus offering almost a factor n speedup compared to the simple greedy algorithm. One of the parameters in the lazy greedy algorithms is T_f, which involves evaluating $f(X \cup j) - f(X)$. One option is to do a naïve implementation of computing $f(X \cup j)$ and then $f(X)$ and take the difference. However, due to the greedy nature of algorithms, we can use memoization and maintain a precompute statistics $p_f(X)$ at a set X, using which the gain can be evaluated much more efficiently (Iyer and Bilmes 2019). At every iteration, we evaluate $f(j|X)$ using $p_f(X)$, which we call $f(j|X, p_f)$. We then update $p_f(X \cup j)$ after adding element j to X. Both the supervised and unsupervised facility location functions admit precompute statistics thereby enabling further speedups.

Experiments

In this section, we (1) evaluate our joint learning against state-of-the-art approaches, (2) study the effect of the different components of losses in our joint optimisation framework and (3) demonstrate the importance of subset selection over random subset selection. We present evaluations on four datasets on tasks such as text classification, record classification and sequence labelling.

Datasets

We adopt the same experimental setting as in (Awasthi et al. 2020) for the dataset split and the labelling functions. However (for the sake of fairness) we set validation data size to be equal to size of the labelled data-set unlike (Awasthi et al. 2020) in which the size of the validation set was assumed to be much larger. In fact we find that even with a smaller size of the validation set, we are mostly able to outperform numbers reported in (Awasthi et al. 2020) using much larger validation sets.

1. **SMS Spam Classification (Almeida, Hidalgo, and Yamakami 2011)** is a binary spam/no-spam classification dataset containing 5574 documents split into 69 labelled set, 500 test set and 4502 as unlabelled set. There are 73 LFs for the dataset with a precision of 97.3%.
2. **Youtube Spam Classification (Alberto, Lochter, and Almeida 2015)** is a spam/no-spam comment classification on YouTube videos. The LFs are obtained from Snorkel's github page[2] that has 10 LFs. The dataset split contains 100 labelled instances, 250 test instances and 1586 unlabelled instances with a precision of 78.6%.
3. **Census Binary Classification(Dua and Karra Taniskidou 2019)** is a UCI dataset from the 1994 U.S. census containing a list of 13 features of each individual such as country of origin, age, education level, *etc*. The task is to predict whether a person earns more than $50K or not. The dataset has 83 LFs and it contains 83 labelled,

[2]https://github.com/snorkel-team/snorkel-tutorials/tree/master/spam

Dataset	|L|	|U|	#Rules/LFs	Precision	%Cover	|Valid|	|Test|
MIT-R	1842	64888	15	80.7	14	1842	14256
YouTube	100	1586	10	78.6	87	100	250
SMS	69	4502	73	97.3	40	69	500
Census	83	10000	83	84.1	100	83	16281

Table 5: Statistics of datasets and their rules/LFs. We will refer to rules and LFs interchangeably. Precision refers to micro precision of rules.

Methods	Datasets			
	Census (Accuracy)	YouTube (Accuracy)	SMS (F1)	MIT-R (F1)
Only-L	78.3	90.7	90.0	74.1
L+UMaj	-0.9	+1.9	-0.3	+0.1
L2R (Ren et al. 2018)	+3.6	-3.7	+0.7	-20.2
L+USnorkel (Ratner et al. 2016)	+1.7	+0.9	+0.3	-0.3
Posterior Reg (Hu et al. 2016)	-1.9	-1.9	-3.3	-0.2
ImplyLoss (Awasthi et al. 2020)	+3.4	+0.4	+0.9	**+0.9**
'Best' JL Approach	**+3.7**	**+3.7**	**+3.4**	-0.8

Table 6: Comparison of our 'Best' JL approach against various state of the art methods on four different datasets. Please note that all results are based on the same hand-picked labelled data subset as was chosen in (Awasthi et al. 2020). The numbers reported (below double lines) are gains over the baseline method (Only-L). All results are averaged over 5 runs.

Loss Combination	Datasets			
	Census (Accuracy)	YouTube (Accuracy)	SMS (F1)	MIT-R (F1)
L1+L2+L3+L4	82.3	94.6	93.1	72.5
L1+L2+L4+L6	81.3	92.8	91.9	69.7
L1+L3+L4+L6	81.0	94.7	93.2	29.8
L1+L2+L3+L4+L6	80.9	94.5	92.3	29.5
L1+L3+L4+L5+L6	74.0	92.8	91.5	72.0
L1+L2+L3+L4+L5+L6	74.0	94.6	92.9	70.9
L1+L3+L4+L5+L6+QG	**82.0**	94.6	**93.4**	**73.2**
L1+L2+L3+L4+L5+L6+QG	81.9	**94.4**	93.0	72.8

Table 7: Performance on the test data, of various loss combinations from our objective function in equation (4), reported for the four data-sets. For each data-set, the numbers in **bold** refer to the 'best' performing combination, determined based on performance on the validation data-set. In general, we observe that all the loss components (barring L2 for three datasets) contribute to the best model. QG refer to Quality Guides.

16281 test and 10K unlabelled instances with a precision of 84.1%.

4. **MIT-R (Liu et al. 2013)** contains sentences about restaurant search and is a sequence labelling tasks on each token with following labels : `Amenity`, `Prices`, `Cuisine`, `Dish`, `Location`, `Hours`, `Others`. The dataset has 15 LFs and contain 1842 labelled instances, 14256 test instances and 64888 unlabelled instances with a precision of 80.7%.

Some statistics pertaining to these datasets are presented in Table 5. Since we compare performances against models that adopt different jargon, we refer to rules (Awasthi et al. 2020) and labelling functions (LFs) (Bach et al. 2019; Chatterjee, Ramakrishnan, and Sarawagi 2019) interchangeably.

Network Architecture

To train our model on the supervised data L, we use a neural network architecture having two hidden layers with ReLU activation. We chose our classification network to be the same as (Awasthi et al. 2020). In the case of MIT-R and SMS, the classification network contain 512 units in each hidden layer whereas the classification network for Census has 256 units in its hidden layers. For the YouTube dataset, we used a simple logistic regression as a classifier network, again as followed in (Awasthi et al. 2020). The features as well as the labelling functions for each dataset are also directly obtained from Snorkel (Ratner et al. 2016) and (Awasthi et al. 2020). Please note that all experiments (barring those on subset selection) are based on the same hand-picked labelled data subset as was chosen in (Awasthi et al. 2020).

In each experiment, we train our model for 100 epochs and early stopping was performed based on the validation set. We use Adam optimizer with the dropout probability set to 0.8. The learning rate for f and g network are set to 0.0003 and 0.001 respectively for YouTube, Census and MIT-R datasets. For SMS dataset, learning rate is set to 0.0001 and 0.01 for f and g network. For each experiment, the numbers are obtained by averaging over five runs, each with a different random initialisation. The model with the best performance on the validation set was chosen for evaluation on the test set. As mentioned previously, the experimental setup in (Awasthi et al. 2020) surprisingly employed a large validation set For fairness, we restrict the size of the

validation set and keep it equal to the size of the labelled set. For all experiments involving comparison with previous approaches, we used code and hyperparameters from (Awasthi et al. 2020) but with our smaller sized validation set (though we mostly outperform them even with their larger sized validation set as seen in Table 9).

Following (Awasthi et al. 2020), we used binary-F1 as an evaluation measure for the SMS, macro-F1 for MIT-R datasets, and accuracy for the YouTube and Census datasets.

Role of Loss components in the Joint Learning (JL)

Given that our loss function in equation (4) has seven components (including the quality guides), a natural question is 'how do we choose among the different components for joint learning (JL)?' Another question we attempt to answer is 'whether all the components are necessary for JL?' For our final model (*i.e.*, the results presented in Tables 6 and 9), we attempt to choose the best performing JL combination of the 7 loss components, *viz.* L1, L2, L3, L4, L5, L6 and QG. To choose the 'best' JL combination, we evaluate the performance on the validation set of the different JL combinations listed in Table 7. Since we generally observe considerably weaker performance by selecting lesser than 3 loss terms, we restrict ourselves to 3 or more loss terms in our search. We report performance on the test data, of various JL combinations from our objective function in equation (4) for each of the four data-sets. For each data-set, the numbers in **bold** refer to the 'best' performing JL combination, determined based on performance on the validation data-set. The observations on the results are as follows. Firstly, we observe that all the loss components (barring L2 for three datasets) contribute to the best model. Furthermore, we observe that the best JL combination (picked on the basis of the validation set) either achieves the best performance or close to best among the different JL combinations as measured on the test data-set.

Joint Learning (JL) Results

In Table 6, we compare our joint learning (JL) approach against other standard methods (that are compared against by (Awasthi et al. 2020)) on four datasets. Below, we briefly describe these other methods:

Only-L: We train the classifier $P_\theta(y|x)$ only on the labelled data L using loss component L1. As explained earlier, following (Awasthi et al. 2020), we observe that a 2-layered neural network trained with small amount of labelled data is capable of achieving competitive accuracy. We choose this method as a baseline and report gains over these numbers.

L+Umaj: We train the baseline classifier $P_\theta(y|x)$ on the labelled data L along with U_{maj} where labels on the U instances are obtained by majority voting on the rules/LFs. The training loss is obtained by weighing instances labelled by rules/LFs as follows:

$$\min_\theta \sum_{(x_j,l_j)\in L} -\log P_\theta(l_j|x_j) + \gamma \sum_{(x_j,y_j)\in U_m aj} -\log P_\theta(y_j|x_j) \tag{8}$$

Learning to Reweight (L2R) (Ren et al. 2018): This method trains the classifier by an online training algorithm that assigns importance to examples based on the gradient direction.

L+USnorkel (Ratner et al. 2016) is Snorkel's generative model that models class probabilities based on discrete LFs for consensus on the noisy and conflicting labels.

Posterior Regularization (PR) (Hu et al. 2016) is method for joint learning of rule and feature network in a teacher-student setup.

Imply Loss (Awasthi et al. 2020): This approach uses additional information in the form of labelled rule exemplars and trains with denoised rule-label loss. Since it uses information over above what we assume, Imply Loss can be considered as a skyline for our proposed approaches.

'Best' JL Approach: As discussed in the previous Section , our final joint learning (JL) approach uses the 'best' combination of the loss components L1, L2, L3, L4, L5, L6 and QG. To determine the 'best' combination, we perform a grid search over various combination of losses using validation accuracy/f1-score as the criteria for selecting the most appropriate loss combination. Imply Loss (Awasthi et al. 2020) uses a large validation size to tune their models. In our experiments, we kept validation set equal to the size of the labelled data. Our approach performs better on Census, YouTube and SMS datasets in comparison to all other methods.

In Table 6, we observe that our 'best' JL approach performs significantly better than all the other approaches on 3 out of the 4 data-sets. Please note that all results are based on the same hand-picked labelled data subset as was chosen in (Awasthi et al. 2020). Even though, we do not have rule-exemplar information in our model, our joint learning method achieve better gains than even ImplyLoss. Recall that the use of ImplyLoss in (Awasthi et al. 2020) can be viewed as a skyline approach owing to the additional exemplar information that associates labelling functions/rules with specific labelled example.

The poor performance of the 'best' JL approach on the MIT-R data-set can be explained as follows. There are no LFs corresponding to the '0' class label, owing to which our graphical model is not trained for all classes. Therefore, we see a drop in F1-score on the MIT-R data-set. As we will show in the next section, by suitably determining a subset of the data-set that can be labelled, we achieve improved performance even on the MIT-R data-set (see Table 8).

Subset Selection Results

We note that all experiments reported so far until Table 6 were based on some 'hand picked' labelled data-set. In Table 8, we summarise the results for different subset selection schemes for picking the labelled data-set and compare them against the results from Table 6 which were based on 'hand-picked' labelled data-sets. Our data selection schemes are applied to the 'best' JL model obtained from Table 7.We observe that the best performing model for the supervised and unsupervised data selection tends to outperform the best model based on random selection. Secondly, we observe that between the supervised and un-supervised data selection approaches, the supervised one tends to perform the best, which means that using the hypothesised labels does

Methods	Datasets			
	Census (Accuracy)	YouTube (Accuracy)	SMS (F1)	MIT-R (F1)
Only-L (Handpicked)	78.3	90.7	90.0	74.1
L+UMaj (Handpicked)	-0.9	+1.9	-0.3	+0.1
L2R (Ren et al. 2018) (Handpicked)	+3.6	-3.7	+0.7	-20.2
L+USnorkel (Ratner et al. 2016) (Handpicked)	+1.7	+0.9	+0.3	-0.3
Posterior Reg (Hu et al. 2016) (Handpicked)	-1.9	-1.9	-3.3	-0.2
ImplyLoss (Awasthi et al. 2020) (Handpicked)	+3.4	+0.4	+0.9	0.9
'Best' JL (Handpicked)	**+3.7**	+3.7	**+3.4**	-0.8
'Best' JL + Unsupervised Subset Selection	-1.9	+3.9	+1.9	-0.2
'Best' JL + Random Subset Selection	-5.2	+3.5	+1.8	-2.9
'Best' JL + Supervised Subset Selection	-0.6	**+4.2**	+3.2	**+1.9**

Table 8: Performance of the three subset selection schemes applied on the 'Best' JL models from Table 7 for the four different data-sets. All numbers reported (below top double line) are gains over the baseline method (Only-L). All results are averaged over 5 runs.

Methods	Datasets			
	Census (Accuracy)	YouTube (Accuracy)	SMS (F1)	MIT-R (F1)
ImplyLoss (Awasthi et al. 2020)	81.1	94.1	93.2	**74.3**
Ours	**+0.9**	**+0.3**	**+0.2**	-0.9

Table 9: Comparison of our approach employing a much smaller validation set size against a more informed ImplyLoss (Awasthi et al. 2020) that uses exemplar information while also employing a much larger sized validation set.

help. Thirdly, we observe that on YouTube and MIT-R, the selected subset even outperforms the hand picked data-set from (Awasthi et al. 2020). Finally, we also note (in Tables 10, 11, 12 and 13) that the results with the random labelled set tend to have a higher variance compared to the other selection techniques (for example, on the SMS data-set). These results demonstrate the promise of using data subset selection for selecting labelled data-sets in semi-supervised learning with labelling functions.

Tables 10, 11, 12 and 13 demonstrate the results comparing the different selection schemes on the four data-sets across the different JL loss combinations. For each subset selection, we present in bold, the best performing combination on the validation set.

Conclusion

We present, how data programming can benefit from use of labelled data by learning a model that jointly optimizes the consensus obtained from labelling functions in an unsupervised manner along with semi-supervised loss functions designed in the feature space. We empirically assess the performance of the different components of the loss in our joint learning framework. As another contribution, we also study a subset selection approach to guide the selection of the subset of examples that can be used as the labelled data-set. We present performance of our models and present insights on both synthetic and real datasets. While outperforming previous approaches, we are often able to better even an examplar based (skyline) approach that uses the additional information of the association of rules with specific labelled examples.

References

Alberto, T. C.; Lochter, J. V.; and Almeida, T. A. 2015. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 138–143. IEEE.

Almeida, T. A.; Hidalgo, J. M. G.; and Yamakami, A. 2011. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, 259–262.

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

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

Bachem, O.; Lucic, M.; and Krause, A. 2017. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*.

Bunescu, R., and Mooney, R. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 576–583.

Chatterjee, O.; Ramakrishnan, G.; and Sarawagi, S. 2019. Data programming using continuous and quality-guided labeling functions. *arXiv preprint arXiv:1911.09860*.

Coleman, C.; Mussmann, S.; Mirzasoleiman, B.; Bailis, P.; Liang, P.; Leskovec, J.; and Zaharia, M. 2018. Select via proxy: Efficient data selection for training deep networks.

Craven, M.; Kumlien, J.; et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, volume 1999, 77–86.

Dua, D., and Karra Taniskidou, E. 2019. Uci machine learning repository. school of information and computer science, university of california, irvine, ca (2017).

Fergus, R.; Weiss, Y.; and Torralba, A. 2009. Semi-supervised learning in gigantic image collections. In *Advances in neural information processing systems*, 522–530.

Loss Combination	SMS Dataset			
	Unsupervised Subset Selection (F1)	Random Subset Selection (F1)	Supervised Subset Selection (F1)	HandPicked Dataset (F1)
L1+2+3+4	90.59	89.51	92.47	93.1
L1+2+4+6	90.57	50.50	90.11	91.9
L1+3+4+6	85.79	81.90	91.52	93.2
L1+2+3+4+6	87.39	35.47	90.37	92.3
L1+3+4+5+6	90.00	**91.84**	92.14	91.5
L1+2+3+4+5+6	89.61	91.51	91.91	92.9
L1+3+4+5+6+QG	91.85	90.24	**93.24**	**93.4**
L1+2+3+4+5+6+QG	**91.91**	91.79	92.10	93.0

Table 10: Performance of subset selection for SMS Dataset: QG refer to Quality Guides.

Loss Combination	Youtube Dataset			
	Unsupervised Subset Selection (Accuracy)	Random Subset Selection (Accuracy)	Supervised Subset Selection (Accuracy)	HandPicked Dataset (Accuracy)
L1+2+3+4	93.0	**94.233**	91.12	94.6
L1+2+4+6	94.12	92.88	93.12	92.8
L1+3+4+6	94.05	93.76	94.11	95.7
L1+2+3+4+6	93.52	91.86	94.32	94.5
L1+3+4+5+6	94.12	92.01	94.25	92.8
L1+2+3+4+5+6	90.15	91.15	**94.85**	94.6
L1+3+4+5+6+QG	**94.56**	89.14	93.47	94.6
L1+2+3+4+5+6+QG	93.52	89.11	91.25	**94.4**

Table 11: Performance of subset selection for YouTube Dataset: QG refer to Quality Guides.

Loss Combination	MITR Dataset			
	Unsupervised Subset Selection (F1)	Random Subset Selection (F1)	Supervised Subset Selection (F1)	HandPicked Dataset (F1)
L1+2+3+4	73.12	69.26	73.24	72.5
L1+2+4+6	76.52	**71.24**	**76.01**	69.7
L1+3+4+6	73.66	69.21	74.37	29.8
L1+2+3+4+6	73.17	64.88	72.23	29.5
L1+3+4+5+6	72.88	68.94	73.09	72.0
L1+2+3+4+5+6	72.12	68.51	71.07	70.9
L1+3+4+5+6+QG	**73.85**	70.17	72.69	**73.2**
L1+2+3+4+5+6+QG	73.73	67.55	73.56	72.8

Table 12: Performance of subset selection for MITR Dataset: QG refer to Quality Guides.

Loss Combination	CENSUS Dataset			
	Unsupervised Subset Selection (Accuracy)	Random Subset Selection (Accuracy)	Supervised Subset Selection (Accuracy)	HandPicked Dataset (Accuracy)
L1+2+3+4	**76.36**	70.17	77.12	**82.3**
L1+2+4+6	76.13	69.81	**77.65**	81.3
L1+3+4+6	75.60	71.14	73.56	81.0
L1+2+3+4+6	75.92	69.17	75.20	80.9
L1+3+4+5+6	72.14	68.15	73.12	74.0
L1+2+3+4+5+6	74.07	**72.58**	74.14	74.0
L1+3+4+5+6+QG	72.98	68.12	75.38	82.0
L1+2+3+4+5+6+QG	70.54	71.2	71.24	81.9

Table 13: Performance of subset selection for CENSUS Dataset: QG refer to Quality Guides.

Goh, G. B.; Siegel, C.; Vishnu, A.; and Hodas, N. 2018. Using rule-based labels for weak supervised learning: a chemnet for transferable chemical property prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 302–310.

Grandvalet, Y., and Bengio, Y. 2005. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, 529–536.

Har-Peled, S., and Kushal, A. 2007. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry* 37(1):3–19.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E. H.; and Xing, E. P. 2016. Harnessing deep neural networks with logic rules. *CoRR* abs/1603.06318.

Iyer, R., and Bilmes, J. 2019. A memoization framework for scaling submodular optimization to large scale problems. *arXiv preprint arXiv:1902.10176*.

Joachims, T. 1999. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, 200–209.

Kaushal, V.; Iyer, R.; Kothawade, S.; Mahadev, R.; Doctor, K.; and Ramakrishnan, G. 2019. Learning from less data: A unified data subset selection and active learning framework for computer vision. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1289–1299. IEEE.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, 3581–3589.

Liang, P. 2005. *Semi-supervised learning for natural language*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Liu, J.; Pasupat, P.; Wang, Y.; Cyphers, S.; and Glass, J. 2013. Query understanding enhanced by hierarchical parsing structures. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 72–77. IEEE.

Mallory, E. K.; Zhang, C.; Ré, C.; and Altman, R. B. 2016. Large-scale extraction of gene interactions from full-text literature using deepdive. *Bioinformatics* 32(1):106–113.

Minoux, M. 1978. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*. Springer. 234–243.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. 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 2-Volume 2*, 1003–1011. Association for Computational Linguistics.

Mirzasoleiman, B.; Bilmes, J.; and Leskovec, J. 2019. Data sketching for faster training of machine learning models. *arXiv preprint arXiv:1906.01827*.

Nemhauser, G. L.; Wolsey, L. A.; and Fisher, M. L. 1978. An analysis of approximations for maximizing submodular set functionsi. *Mathematical programming* 14(1):265–294.

Oliver, A.; Odena, A.; Raffel, C.; Cubuk, E. D.; and Goodfellow, I. J. 2018. Realistic evaluation of semi-supervised learning algorithms.

Ranzato, M., and Szummer, M. 2008. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*, 792–799.

Ratner, A. J.; De Sa, C. M.; Wu, S.; Selsam, D.; and Ré, C. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, 3567–3575.

Ren, M.; Zeng, W.; Yang, B.; and Urtasun, R. 2018. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, 4334–4343.

Rosenberg, C.; Hebert, M.; and Schneiderman, H. 2005. Semi-supervised self-training of object detection models. *WACV/MOTION* 2.

Sener, O., and Savarese, S. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Settles, B.; Craven, M.; and Friedland, L. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, 1–10. Vancouver, CA:.

Tsang, I. W.; Kwok, J. T.; and Cheung, P.-M. 2005. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research* 6(Apr):363–392.

Wei, K.; Liu, Y.; Kirchhoff, K.; and Bilmes, J. 2013. Using document summarization techniques for speech data subset selection. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 721–726.

Wei, K.; Iyer, R.; and Bilmes, J. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, 1954–1963.