Semi-Supervised Data Programming with Subset Selection

Ayush Maheshwari [1], Oishik Chatterjee [1], Krishnateja Killamsetty [2], Ganesh Ramakrishnan [1] , Rishabh Iyer [2]
[1]Department of CSE, IIT Bombay, India
[2] The University of Texas at Dallas
{ayusham, oishik, ganesh}@cse.iitb.ac.in
{krishnateja.killamsetty, rishabh.iyer}@utdallas.edu

Abstract

The paradigm of data programming, which uses weak supervision in the form of rules/labelling functions, and semi-supervised learning, which augments small amounts of labelled data with a large unlabelled dataset, have shown great promise in several text classification scenarios. In this work, we argue that by not using any labelled data, data programming based approaches can yield sub-optimal performances, particularly when the labelling functions are noisy. The first contribution of this work is an introduction of a framework, SPEAR which is a semi-supervised data programming paradigm that learns a *joint model* that effectively uses the rules/labelling functions along with semi-supervised loss functions on the feature space. Next, we also study SPEAR-SS which additionally does subset selection on top of the joint semi-supervised data programming objective and *selects* a set of examples that can be used as the labelled set by SPEAR. The goal of SPEAR-SS is to ensure that the labelled data can *complement* the labelling functions, thereby benefiting from both data-programming as well as appropriately selected data for human labelling. We demonstrate that by effectively combining semi-supervision, data-programming, and subset selection paradigms, we significantly outperform the current state-of-the-art on seven publicly available datasets. [1]

1 Introduction

Modern machine learning techniques rely on large amounts of labelled training data for text classification tasks such as spam detection, (movie) genre classification, sequence labelling, *etc*. Supervised learning approaches have utilised such large amounts of labelled data and, this has resulted in huge successes in the last decade. However, the acquisition of labelled data, in most cases, entails a painstaking process requiring months of human effort. Several techniques such as active learning, distant supervision, crowd-consensus learning, and semi-supervised learning have been proposed to reduce the *annotation cost* (Settles et al., 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 in which 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.

1.1 Motivating Example

We illustrate the LFs on one of the seven tasks on which we experiment with, *viz.*, identifying spam/no-spam comments in the YouTube reviews. For some applications, writing LFs is often as simple as using keyword lookups or a regex expression. In this specific case, the users construct heuristic patterns as LFs for classifying spam/not-spam comments. Each LF takes a comment as an input and provides a binary label as the output; +1 indicates that the comment is spam, -1 indicates that the comment is not spam, and 0 indicates that the LF is unable to assert anything for the comment (referred to as an *abstain*). Table 1 presents a few example LFs for spam and non-spam classification.

In isolation, a particular LF may neither be always correct nor complete. Furthermore, the LFs

[1]The source code is available at https://github.com/ayushbits/Semi-Supervised-LFs-Subset-Selection

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

may also produce conflicting labels. In the past, generative models such as Snorkel (Ratner et al., 2016) and CAGE (Chatterjee et al., 2020) have been proposed for consensus on the noisy and conflicting labels assigned by the discrete LFs to determine the probability of the correct labels. Labels thus obtained could be used for training any supervised model/classifier and evaluated on a test set. We will next highlight a challenge in doing data programming using only LFs that we attempt to address. 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 a spam comment, the ones in S_2 and S_3 are not. In fact, these examples constitute one of the canonical cases that we discovered during the analysis of our approach in Section 4.4.

1. $\langle \boldsymbol{S_1}, +1 \rangle$: Please help me go to college guys! Thanks from the bottom of my heart. https://www.indiegogo.com/projects/
2. $\langle \boldsymbol{S_2}, -1 \rangle$: I love this song
3. $\langle \boldsymbol{S_3}, -1 \rangle$: This song is very good... but the video makes no sense...
4. $\langle \boldsymbol{S_4}, +1 \rangle$: https://www.facebook.com/teeLaLaLa

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. $\langle \boldsymbol{S_5}, -1 \rangle$: This song is prehistoric
6. $\langle \boldsymbol{S_6}, +1 \rangle$: Watch Maroon 5's latest ... www.youtube.com/watch?v=TQ046FuAu00

In Table 2, we present the outputs of the LFs as well as some n-gram features F1 ('.com') and F2 ('This song') on the observed training examples S_1, S_2, S_3 and S_4 as well as on the unseen test examples S_5 and S_6. For S_1, the correct consensus can easily be performed to output the true label +1, since 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 it possible to easily perform the correct consensus. Hence, we have treated S_1 and S_2 as unlabelled, indicating that we

Training data		LF outputs		Features	
id	Label	LF1(+1)	LF2(-1)	F1	F2
S_1	+1	1	0	1	0
S_2	-1	0	1	0	1
S_3	-1	0	0	0	1
S_4	+1	1	1	1	0
Test data					
S_5	-1	0	1	0	1
S_6	+1	0	0	1	0

Table 2: Example illustrating the insufficiency of using data programming using only LFs.

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, the correct consensus on S_3 and S_4 is challenging since both LF1 and LF2 either fire or do not. While the (n-gram based) features F1 and F2 appear to be informative and could potentially complement LF1 and LF2, we can easily see that correlating feature values with LF outputs is tricky in a completely unsupervised setup. To address this issue, we ask the following questions:

(A) What if we are 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) Can we precisely determine the subset of the unlabelled data that, when labelling would help us train a model (in conjunction with the labelling functions) that is most effective on the test set? In other words, instead of randomly choosing the labelled dataset for doing semi-supervised learning (part A), can we intelligently select the labelled subset? In the above example, choosing the labelled set as S_3, S_4 would be much more useful than choosing the labelled set as S_1, S_2.

As a solution to (A), in Section 3.3, we present a new formulation, SPEAR, in which the parameters over features and LFs are jointly trained in a semi-supervised manner. SPEAR expands as **S**emi-su**P**ervis**E**d d**A**ta p**R**ogramming. As for (B), we present a subset selection recipe, SPEAR-SS (in

Section 3.4), that recommends the sub-set of the data (*e.g.* S_3 and S_4), which, after labelling, would most benefit the joint learning framework.

1.2 Our Contributions

We summarise our main contributions as follows: To address (A), we present SPEAR (*c.f.*, Section 3.3), which is a novel paradigm for jointly learning the parameters over features and labelling functions in a semi-supervised manner. We jointly learn a parameterized graphical model and a classifier model to learn our overall objective. To address (B), we present SPEAR-SS (*c.f.*, Section 3.4), which is a subset selection approach to *select* the set of examples which can be used as the labelled set by SPEAR. 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. Moreover, we also show that the automatically selected subset performs comparably or even better than the hand-picked subset by humans in the work reported by Awasthi et al. (2020), further emphasising the benefit of subset selection for semi-supervised data programming. Our framework is agnostic to the underlying network architecture and can be applied using different underlying techniques without a change in the meta-approach. Finally, we evaluate our model on seven publicly available datasets from domains such as spam detection, record classification, and genre prediction and demonstrate significant improvement over state-of-the-art techniques. We also draw insights from experiments in synthetic settings (presented in the appendix).

2 Related Work

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. Chatterjee et al. (2020) proposed a graphical model, CAGE, that uses 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 employs user-controlled quality guides that incorporate labeller intuition into the model. However, these models completely disregard feature information that could provide additional information to learn the (graphical) model. 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); 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 learn the parameters jointly. Hu et al. (2016) proposed a student-teacher model that transfers rule information by assigning linear weight to each rule based on an agreement objective. The model we propose in this paper jointly learns parameters over features and rules in a semi-supervised manner rather than just weighing their outputs and can therefore be more expressive.

Other works such as Jawanpuria et al. (2011); Dembczyński et al. (2008) discovers simple and conjuctive rules from input features and assign weight to each rules for better generalization. Nagesh et al. (2012) induces rules from query language to build NER extractor while Kulkarni et al. (2018) uses active learning to derive consensus among labelers to label data.

Semi-Supervised Data Programming: The only work which, to our knowledge, combines rules with supervised learning in a joint framework is the work by Awasthi et al. (2020). 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 denoise over-generalized rules and train a classification model. Our approach differs from their work in two ways: a) we do not have information of rule exemplars - thus our labelled examples need not have any correspondence to any of the LFs (and may instead complement the LFs as illustrated in Table 2) and b) we employ a semi-supervised framework combined with graphical model for consensus amongst the LFs to train our model. We also study how to automatically select the seed set of labelled data, rather than having a human provide this seed set, as was done in (Awasthi et al., 2020).

Data Subset Selection: Finally, another approach that 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. Data subset selection techniques have been used to reduce end to end training time (Mirzasoleiman et al., 2019; Kaushal et al., 2019; Killamsetty et al., 2021) and to select unlabelled points in an active learning manner to label (Wei et al., 2015; Sener and Savarese, 2017) or for topic summarization (Bairi et al., 2015). In this paper, we present a framework (SPEAR-SS) of data subset selection for *selecting* a subset of unlabelled examples for obtaining labels complementary to the labelling functions.

3 Methodology

3.1 Problem Description

Let $\mathcal{X}$ and $\mathcal{Y} \in \{1...K\}$ be the feature space and label space, respectively. We also have access to m labelling functions (LF) λ_1 to λ_m. As mentioned in Section 1.1, each LF λ_j is designed to record some class; let us denote[2] by $k_j \in \{1...K\}$, the class associated with λ_j. The dataset consists of 2 components, *viz.*,
1. $\mathcal{L} = \{(\mathbf{x}_1, y_1, \mathbf{l}_1), (\mathbf{x}_2, y_2, \mathbf{l}_2), \ldots, (\mathbf{x}_N, y_N, \mathbf{l}_N)\}$, which denotes the labelled dataset and
2. $\mathcal{U} = \{(\mathbf{x}_{N+1}, \mathbf{l}_{N+1}), (\mathbf{x}_{N+2}, \mathbf{l}_{N+2}), \ldots, (\mathbf{x}_M, \mathbf{l}_M)\}$, which denotes the unlabelled dataset wherein $\mathbf{x}_i \in \mathcal{X}$, $y_i \in \mathcal{Y}$.
Here, the vector $\mathbf{l}_i = (l_{i1}, l_{i2}, \ldots, l_{im})$ denotes the firings of all the LFs on instance $\mathbf{x}_i$. Each l_{ij} can be either 1 or 0. $l_{ij} = 1$ indicates that the LF λ_j has fired on the instance i and 0 indicates it has not. All the labelling functions are discrete; hence, no continuous scores are associated with them.

3.2 Classification and Labelling Function Models

SPEAR has a feature-based classification model $f_\phi(\mathbf{x})$ which takes the features as input and predicts the class label. Examples of $f_\phi(\mathbf{x})$ we consider in this paper are logistic regression and neural network models. The output of this model is $P_\phi^f(y|\mathbf{x})$, *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 $P_\theta(\mathbf{l}_i, y)$ which, as specified in equation (1) for an example $\mathbf{x}_i$, is a generative model on the LF outputs and class label y.

[2]We use the association of LF λ_j with some class k_j only in the quality guide component (QG) of the loss in eqn. 3

Notation	Description
f_ϕ	The feature-based Model
P_ϕ^f	The label probabilities as per the feature-based model f_ϕ
P_θ	The label probabilities as per the LF-based Graphical Model
L_{CE}	Cross Entropy Loss: $L_{CE}\left(P_\phi^f(y\|\mathbf{x}), \bar{y}\right) = -\log\left(P_\phi^f(y=\bar{y}\|\mathbf{x})\right)$
H	Entropy function : $H(P_\phi^f(y\|\mathbf{x})) = -\sum_{\hat{y}} P_\phi^f(y=\hat{y}\|\mathbf{x}) \log P_\phi^f(y=\hat{y}\|\mathbf{x})$
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.

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

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

There are K parameters $\theta_{j1}, \theta_{j2}...\theta_{jK}$ for each LF λ_j, where K is the number of classes. The model makes the simple assumption that each LF λ_j independently acts on an instance $\mathbf{x}_i$ to produce outputs $l_{i1}, l_{1i}...l_{im}$. The potentials ψ_θ invoked in equation (1) are defined in equation (2). Z_θ is the normalization factor. We propose a joint learning algorithm with semi-supervision to employ both features and LF predictions in an end-to-end manner.

3.3 Joint Learning in SPEAR

We first specify the objective of SPEAR and thereafter explain each of its components in greater detail:

$$\begin{aligned} \min_{\theta,\phi} & \sum_{i\in\mathcal{L}} L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), y_i\right) + \sum_{i\in\mathcal{U}} H\left(P_\phi^f(y|\mathbf{x}_i)\right) \\ & + \sum_{i\in\mathcal{U}} L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), g(\mathbf{l}_i)\right) + LL_s(\theta|\mathcal{L}) \\ & + LL_u(\theta|\mathcal{U}) + \sum_{i\in\mathcal{U}\cup\mathcal{L}} KL\left(P_\phi^f(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i)\right) \\ & + R(\theta|\{q_j\}) \end{aligned} \tag{3}$$

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 be introducing.

First Component (L1): The first component (L1) of the loss $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), y_i\right) = -\log\left(P_\phi^f(y=y_i|\mathbf{x}_i)\right)$ is the standard cross-entropy loss on the labelled dataset $\mathcal{L}$ for the model P_ϕ^f.

Second Component (L2): The second component L2 is the semi-supervised loss on the unlabelled data $\mathcal{U}$. 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 $H\left(P_\phi^f(y|\mathbf{x}_i)\right)$ 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 $L_{CE}\left(P_\phi^f(y|\mathbf{x}_i), g(\mathbf{l}_i)\right)$ is the cross-entropy of the classification model using the hypothesised labels from CAGE (Chatterjee et al., 2020) on $\mathcal{U}$. Given that $\mathbf{l}_i$ is the output vector of all labelling functions for any $\mathbf{x}_i \in \mathcal{U}$, we specify the predicted label for $\mathbf{x}_i$ using the LF-based graphical model $P_\theta(\mathbf{l}_i, y)$ from eqn. (1) as: $g(\mathbf{l}_i) = \text{argmax}_y P_\theta(\mathbf{l}_i, y)$.

Fourth Component (L4): The fourth component $LL_s(\theta|\mathcal{L})$ is the (supervised) negative log likelihood loss on the labelled dataset $\mathcal{L}$ as per eqn. (3):

$$LL_s(\theta|\mathcal{L}) = -\sum_{i=1}^{N} \log P_\theta(\mathbf{l}_i, y_i)$$

Fifth Component (L5): The fifth component $LL_u(\theta|\mathcal{U})$ is the negative log likelihood loss for the unlabelled dataset $\mathcal{U}$ as per eqn. (3). Since the true label information is not available, the probabilities need to be summed over y: $LL_u(\theta|\mathcal{U}) = -\sum_{i=N+1}^{M} \log \sum_{y \in \mathcal{Y}} P_\theta(\mathbf{l}_i, y)$

Sixth Component (L6): The sixth component $KL(P_\phi^f(y|\mathbf{x}_i), P_\theta(y|\mathbf{l}_i))$ 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 summed over every example $\mathbf{x}_i \in \mathcal{U} \cup \mathcal{L}$. Through this term, we try and make the models agree in their predictions over the union of the labelled and unlabelled datasets.

Quality Guides (QG): As the last component in our objective, we use quality guides $R(\theta|\{q_j\})$ on LFs, which have been shown in (Chatterjee et al., 2020) to stabilise the unsupervised likelihood training while using labelling functions. Let q_j be the fraction of cases where λ_j correctly triggered, and let q_j^t be the user's belief on the fraction of examples $\mathbf{x}_i$ where y_i and l_{ij} agree. If the user's beliefs were not available, we consider the precision of the LFs on the validation set as the user's beliefs. Except for the SMS dataset, we take the precision of the LFs on the validation set as the quality guides. If $P_\theta(y_i = k_j|l_{ij} = 1)$ is the model-based precision over the LFs, the quality guide based loss can be expressed as $R(\theta|\{q_j^t\}) = \sum_j q_j^t \log P_\theta(y_i = k_j|l_{ij} = 1) + (1 - q_j^t) \log(1 - P_\theta(y_i = k_j|l_{ij} = 1))$. Throughout the paper, we consider QG always in conjunction with Loss **L5**.

In summary, the first three components (L1, L2 and L3) invoke losses on the supervised model f_ϕ. While L1 compares the output f_ϕ against the ground truth in the labelled set $\mathcal{L}$, L2 and L3 operate on the unlabelled data $\mathcal{U}$ by minimizing the entropy of f_ϕ (L2) and by calibrating the f_ϕ output against the noisy predictions $g(\mathbf{l}_i)$ of the graphical model $P_\theta(\mathbf{l}_i, y)$ for each $\mathbf{x}_i \in \mathcal{U}$ (L3). The next two components L4 and L5 focus on maximizing the likelihood of the parameters θ of $P_\theta(\mathbf{l}_i, y)$ over labelled $\mathbf{x}_i \in \mathcal{L}$ and unlabelled $\mathbf{x}_i \in \mathcal{U}$ datasets respectively. Finally, in L6, we compare the probabilistic outputs from the supervised model f_ϕ against those from the graphical model $P_\theta(\mathbf{l}, y)$ through a KL divergence based loss. We use the ADAM (stochastic gradient descent) optimizer to train the non-convex loss objective.

Previous data programming approaches (Bach et al., 2019; Chatterjee et al., 2020) adopt a cascaded approach in which they first optimise a variant of L5 to learn the θ parameters associated with the LFs and thereafter use the noisily generated labels using $g(\mathbf{l})$ to learn the supervised model f_ϕ using a variant of L3. In contrast, our approach learns the LF's θ parameters and the model's ϕ parameters jointly in the context of the unlabelled data $\mathcal{U}$.

We present synthetic experiments to illustrate the effect of SPEAR for data programming and semi-supervision in a controlled setting in which (i) the overlap between classes in the data is controlled and (ii) the labelling functions are accurate. The details of the synthetic experiments are provided in the appendix.

3.4 SPEAR-SS: Subset Selection with SPEAR

Suppose we are given an unlabelled data set $\mathcal{U}$ and a limited budget for data labelling because of the costs involved in it. It is essential for us to choose the data points that need to be labelled properly. We explore two strategies for selecting a subset of data points from the unlabelled set. We then obtain the labels for this subset, and run SPEAR on the combination of this labelled and unlabelled set. The two approaches given are intended to maximise diversity of the selected subset in the feature space. We complement both the approaches with Entropy Filtering (also described below).

Unsupervised Facility Location: In this approach, given an unlabelled data-set $\mathcal{U}$, we want to select a subset $\mathcal{S}$ such that the selected subset has maximum diversity with respect to the 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 $f_{\text{unsup}}(\mathcal{S}) = \sum_{i \in \mathcal{U}} \max_{j \in \mathcal{S}} \sigma_{ij}$ where σ_{ij} denotes the similarity score (in the feature space $\mathcal{X}$) between data instance $\mathbf{x}_i$ in unlabelled set $\mathcal{U}$ and data instance $\mathbf{x}_j$ in selected subset data $\mathcal{S}$. We employ a lazy greedy strategy to select the subset. In conjunction with Entropy Filtering described below,we call this technique *Unsupervised Subset Selection*.

Supervised Facility Location: The objective function for Supervised Facility Location (Wei et al., 2015) is $f_{\text{sup}}(\mathcal{S}) = \sum_{y \in \mathcal{Y}} \sum_{i \in \mathcal{U}_y} \max_{j \in \mathcal{S} \cap \mathcal{U}_y} \sigma_{ij}$. Here we assume that $\mathcal{U}_y \subseteq \mathcal{U}$ is the subset of data points with hypothesised label y. Simply put, $\mathcal{U}_y$ forms a partition of $\mathcal{U}$ based on the hypothesized labels obtained by performing unsupervised learning with labelling functions. In conjunction with Entropy Filtering, we call this technique *Supervised Subset Selection*.

Entropy Filtering: We also do a filtering based on entropy. In particular, we sort the examples based on maximum entropy and select fB number of data points[3], where B is the data selection budget (which was set to the size of the labelled set $|\mathcal{L}|$ in all our experiments). On the filtered dataset, we perform the subset selection, using either the supervised or unsupervised facility location as described above. Below, we describe the optimisation algorithm for subset selection.

Optimisation Algorithms and Submodularity: Both $f_{\text{unsup}}(\mathcal{S})$ and $f_{\text{sup}}(\mathcal{S})$ are submodular functions. We select a subset $\mathcal{S}$ of the filtered unlabelled data, by maximising these functions under a cardinality budget k (*i.e.*, a labelling budget). For cardinality constrained maximisation, a simple greedy algorithm provides a near-optimal solution (Nemhauser et al., 1978). Starting with $\mathcal{S}^0 = \emptyset$, we sequentially update

$$\mathcal{S}^{t+1} = \mathcal{S}^t \cup \underset{j \in \mathcal{U} \setminus \mathcal{S}^t}{\operatorname{argmax}} f(j|\mathcal{S}^t) \tag{4}$$

where $f(j|\mathcal{S}) = f(\mathcal{S} \cup j) - f(\mathcal{S})$ is the gain of adding element j to set $\mathcal{S}$. We iteratively execute the greedy step (4) until $t = k$ and $|\mathcal{S}^t| = k$. 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 $f(j|\mathcal{S})$ for the supervised and unsupervised facility location functions. We then significantly optimize this simple greedy algorithm via a lazy greedy algorithm (Minoux, 1978) via memoization and precompute statistics (Iyer and Bilmes, 2019).

[3] In our experiments, we set $f = 5$

Dataset	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{U}\rvert$	#Rules/LFs	Precision	%Cover	|Test|
MIT-R	1842	64888	15	80.7	14	14256
YouTube	100	1586	10	78.6	87	250
SMS	69	4502	73	97.3	40	500
Census	83	10000	83	84.1	100	16281
Ionosphere	73	98	64	65	100	106
Audit	162	218	52	87.5	100	233
IMDB	284	852	25	80	58.1	500

Table 4: Statistics of datasets and their rules/LFs. Precision refers to micro precision of rules. %Cover is the fraction of instances in $\mathcal{U}$ covered by at least one LF. Size of Validation set is equal to $|\mathcal{L}|$.

4 Experiments

In this section, we (1) evaluate our joint learning against state-of-the-art approaches and (2) demonstrate the importance of subset selection over random subset selection. We present evaluations on seven datasets on tasks such as text classification, record classification and sequence labelling.

4.1 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 the validation data size to be equal to the 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.

We use the following datasets: **(1) YouTube**: A spam classification on YouTube comments; **(2) SMS Spam Classification (Almeida et al., 2011)**, which is a binary spam classification dataset containing 5574 documents; **(3) MIT-R (Liu et al., 2013)**, is a sequence labelling task on each token with following labels: *Amenity, Prices, Cuisine, Dish, Location, Hours, Others*; **(4) IMDB**, which is a plot summary based movie genre binary classification dataset, and the LFs (and the labelled set) are obtained from the approach followed by Varma and Ré (2018); **(5) Census** (Dua and Graff, 2017), **(6) Ionosphere**, and **(7) Audit**, which are all UCI datasets. The task in the Census is to predict whether a person earns more than $50K or not.

Ionosphere is radar binary classification task given a list of 32 features. The task in the Audit is to classify suspicious firms based on the present and historical risk factors.

Statistics pertaining to these datasets are presented in Table 4. Since we compare performances against models that adopt different terminology, we refer to rules and labelling functions interchangeably. For fairness, we restrict the size of the validation set and keep it equal to the size $|\mathcal{L}|$ 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. Note that we mostly outperform them even with their larger-sized validation set as can be seen in Table 5. More details on training and validation set size are given in the appendix.

4.2 Baselines

In Table 5, we compare SPEAR and SPEAR-SS against other following standard methods on seven datasets.

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

$\mathcal{L} + \mathcal{U}_{maj}$: We train the baseline classifier $P_\theta(y|x)$ on the labelled data $\mathcal{L}$ along with $\mathcal{U}_{maj}$ where labels on the $\mathcal{U}$ instances are obtained by majority voting on the rules/LFs. The training loss is obtained by weighing instances labelled by rules as $\min_\theta \sum_{(\mathbf{x}_i,\mathbf{l}_i)\in\mathcal{L}} -\log P_\theta(\mathbf{l}_i|\mathbf{x}_i) + \gamma \sum_{(\mathbf{x}_i,y_i)\in\mathcal{L}} -\log P_\theta(y_i|\mathbf{x}_i)$.

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.

$\mathcal{L}$+$\mathcal{U}_{Snorkel}$ (Ratner et al., 2016): 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): This is a method for joint learning of a 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 in addition to what we assume, Imply Loss can be considered as a skyline for our proposed approaches.

4.3 Results with SPEAR

SPEAR uses the 'best' combination of the loss components L1, L2, L3, L4, L5, L6. To determine the 'best' combination, we perform a grid search over various combinations of losses using validation accuracy/f1-score as the criteria for selecting the most appropriate loss combination. Imply Loss uses a larger-sized validation set to tune their models. In our experiments, we maintained a validation set size equal to the size of the labelled data. In Table 5, we observe that SPEAR performs significantly better than all other approaches on all but the MIT-R data-set. Please note that all results are based on the same hand-picked labelled data subset as was chosen in prior work (Awasthi et al., 2020; Varma and Ré, 2018), except for Audit and Ionosphere. Even though we do not have rule-exemplar information in our model, SPEAR achieves better gains than even ImplyLoss. Recall that the use of ImplyLoss can be viewed as a skyline approach owing to the additional exemplar information that associates labelling functions with specific labelled examples. The slightly lower performance of the 'best' SPEAR on the MIT-R data-set can be partially explained by the fact that there are no LFs corresponding to the '0' class label, owing to which our graphical model is not trained for all classes. However, as we will show in the next section, by suitably determining a subset of the data-set that can be labelled (using the facility location representation function), we achieve improved performance even on the MIT-R data-set (see Table 5). Also, note that in Table 5, we present results on two versions of Audit, one in which both the train and test set are balanced, and the other where the labelled training set is imbalanced. In the imbalanced case (where the number of positives are only 10%), we were unable to successfully run the ImplyLoss and Posterior-Reg models (and hence the '-'), despite communication with the authors. We see that SPEAR and similarly, SPEAR-SS (discussed below) significantly outperform the baselines by almost 40% in the imbalanced case. In the balanced case, the gains are similar to what we observe on the other datasets.

Methods	Datasets							
	YouTube [Accuracy]	SMS [F1]	MIT-R [F1]	IMDB [F1]	Census [Accuracy]	Ionosphere [F1]	Audit (Imb) [F1]	Audit (Bal) [F1]
Only-$\mathcal{L}$ (Handpicked)	90.7 (1.2)	90.0 (3.7)	74.1 (0.4)	72.2 (3.1)	78.3 (0.3)	92.7 (0.5)	24.7(2.6)	87.3 (0.9)
$\mathcal{L}$+$\mathcal{U}_{maj}$ (Handpicked)	+1.9 (1.1)	-0.3 (1.4)	+0.1(0.2)	+1.2 (0.3)	-0.9 (0.4)	+0.4 (0.7)	-4.8 (6)	-1.4(4.2)
L2R (Ren et al., 2018) (Handpicked)	-3.7 (5.1)	+0.7 (2.9)	-20.2(0.9)	+4.5(0.2)	+3.6(0.3)	-18.8 (0.3)	-1.2(3.2)	-3.0(4.9)
$\mathcal{L}$+$\mathcal{U}_{Snorkel}$ (Ratner et al., 2016) (Handpicked)	+0.9 (2.6)	+0.3 (4.5)	-0.3(0.2)	+0.6(1.8)	+1.7 (0.2)	-0.6 (0.5)	-7.4(3.4)	-0.6(4.2)
Posterior Reg (Hu et al., 2016) (Handpicked)	-1.9(1.6)	-3.3 (1.9)	-0.2 (0.2)	+1.1 (0.7)	-1.9 (0)	-0.1(0.7)	-	+0.1 (1.4)
ImplyLoss (Awasthi et al., 2020) (Handpicked)	+0.4 (0.5)	+0.9(0.9)	0.9(0.4)	+4.3 (1.5)	+3.4 (0.1)	-3.9(2.4)	-	+0.5(1)
SPEAR (Handpicked)	+3.7(0.5)	**+3.4**(0.9)	-0.8 (0.5)	+4.9(0.3)	**+3.7** (0.3)	**+5.4** (0.3)	+44 (0.9)	**+4.3** (0.9)
SPEAR-SS (Random Subset Selection)	+3.5	+1.8	-2.9	+4.0	-5.2	+4.7	+41.7	+2.0
SPEAR-SS (Unsupervised Subset Selection)	+3.9	+1.9	+2.6	-0.6	+2.5	+4.8	+43.5	+3.3
SPEAR-SS (Supervised Subset Selection)	**+4.2**	+3.2	**+2.9**	**+6.3**	+2.5	+5.1	**+44.5**	+3.5

Table 5: Performance of SPEAR and SPEAR-SS for three subset selection schemes on seven data-sets. All numbers reported are gains over the baseline method (Only-$\mathcal{L}$). All results are averaged over 5 runs. Numbers in brackets '()' represent standard deviation of the original score. Handpicked instances refers to instances selected from the dataset for designing LFs. These instances are taken directly from (Awasthi et al., 2020) to ensure fair comparison.

4.4 Results with SPEAR-SS

Recall that all results discussed so far (including those for SPEAR) on the Youtube, SMS, MIT-R, IMDB and Census datasets were based on the same 'hand-picked' labelled data subset as in prior work (Awasthi et al., 2020; Varma and Ré, 2018). In the case of Audit and Ionosphere, the labelled subset was randomly picked. In Table 5, we summarise the results obtained by employing supervised and unsupervised subset selection schemes for picking the labelled data-set and present comparisons against results obtained using (i) 'hand-picked' labelled data-sets, and (ii) random selection of the labelled set. In each case, the size of the subset is the same, which we set to be the size of the hand-picked labelled set. Our data selection schemes are applied to the 'best' SPEAR model obtained across various loss components. 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 unsupervised data selection approaches, the supervised one tends to perform the best, which means that using the hypothesised labels does help. Thirdly, we observe that YouTube, MIT-R, IMDB and Audit using the selected subset outperform prior work that employ hand picked data-set, whereas, in the case of SMS, Census and Ionosphere, we come close. Finally, our approach is more stable than other approaches as the standard deviation of SPEAR is low for 5 different runs across all the datasets.

As an illustration, the examples such as S_3 and S_4 referred to in Section 1.1 were precisely obtained through supervised subset selection in SPEAR-SS, to form part of the labelled dataset. As previously observed in Table 2, S_3 and S_4 *complement* (via n-grams features such as F1 and F2) the effect of the labelling functions LF1 and LF2 on the unlabelled examples such as S_1 and S_2, when included in the labelled set. Further detailed results with subset selection, *etc.* can be found in the appendix. In general, we observe that when the subset of instances selected for labelling is *complementary* to the labelling functions (as in our case), the performance is higher than when the labelled examples (exemplars) are inspired by labelling functions themselves as done in the work by Awasthi et al. (2020).

4.5 Significance Test

We employ the Wilcoxon signed-rank test (Wilcoxon, 1992) to determine whether there is a significant difference between SPEAR and Imply Loss (current state-of-the-art). Our null hypothesis is that there is not significant difference between SPEAR and Imply loss. For $n = 7$ instances, we observe that the one-tailed hypothesis is significant at $p < .05$, so we reject the null hypothesis. Clearly, SPEAR significantly outperforms Imply loss and, therefore, all previous baselines.

Similarly, we perform the significance test to assess the difference between SPEAR-SS and Imply Loss. As expected, the one-tailed hypothesis is significant at $p < 0.05$, which implies that our SPEAR-SS approach significantly outperforms Imply Loss, and thus all other approaches.

5 Conclusion

We study how data programming can benefit from labelled data by learning a model (SPEAR) that

Methods	Datasets			
	YouTube [Accuracy]	SMS [F1]	MIT-R [F1]	Census [Accuracy]
ImplyLoss (Awasthi et al., 2020)	94.1	93.2	74.3	81.1
SPEAR (Handpicked)	+0.3	**+0.2**	-0.9	**+0.9**
SPEAR-SS (Supervised Subset Selection)	**+0.8**	0.0	**+1.7**	-0.3

Table 6: Comparison of SPEAR and SPEAR-SS against ImplyLoss on subset of datasets from Table 5 for which ImplyLoss used a much larger validation set than $|\mathcal{L}|$. JL uses a validation set sizes equal to $|\mathcal{L}|$.

jointly optimises 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 our joint loss function. As another contribution, we also study some subset selection approaches to guide the selection of the labelled subset of examples. We present the performance of our models and present insights on both synthetic and real datasets. While outperforming previous approaches, our approach is often better than an exemplar-based (skyline) approach that uses the additional information of the association of rules with specific labelled examples.

Acknowledgements

We thank anonymous reviewers for providing constructive feedback. Ayush Maheshwari is supported by a Fellowship from Ekal Foundation (www.ekal.org). We are also grateful to IBM Research, India (specifically the IBM AI Horizon Networks - IIT Bombay initiative) for their support and sponsorship.

References

Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, 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*, pages 362–375.

Ramakrishna Bairi, Rishabh Iyer, Ganesh Ramakrishnan, and Jeff Bilmes. 2015. Summarization of multi-document topic hierarchies using submodular mixtures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 553–563.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. In *AAAI*.

Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. 2008. Maximum likelihood rule ensembles. In *Proceedings of the 25th international conference on Machine learning*, pages 224–231.

Dheeru Dua and Casey Graff. 2017. UCI machine learning repository.

Garrett B Goh, Charles Siegel, Abhinav Vishnu, and Nathan Hodas. 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*, pages 302–310.

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

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. 2016. Harnessing deep neural networks with logic rules. *CoRR*, abs/1603.06318.

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

Pratik Jawanpuria, Saketha N Jagarlapudi, and Ganesh Ramakrishnan. 2011. Efficient rule ensemble learning using hierarchical kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 161–168.

Vishal Kaushal, Rishabh Iyer, Suraj Kothawade, Rohan Mahadev, Khoshrav Doctor, and Ganesh Ramakrishnan. 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)*, pages 1289–1299. IEEE.

Krishnateja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. Glister: Generalization based data subset selection for efficient and robust learning. *In AAAI*.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.

Ashish Kulkarni, Narasimha Raju Uppalapati, Pankaj Singh, and Ganesh Ramakrishnan. 2018. An interactive multi-label consensus labeling model for multiple labeler judgments. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI), 2018*, pages 1479–1486. AAAI Press.

Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. 2013. Query understanding enhanced by hierarchical parsing structures. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77. IEEE.

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

Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. 2019. Data sketching for faster training of machine learning models. *arXiv preprint arXiv:1906.01827*.

Ajay Nagesh, Ganesh Ramakrishnan, Laura Chiticariu, Rajasekar Krishnamurthy, Ankush Dharkar, and Pushpak Bhattacharyya. 2012. Towards efficient named-entity rule induction for customizability. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 128–138, Jeju Island, Korea. Association for Computational Linguistics.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. 1978. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pages 3567–3575.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343.

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

Burr Settles, Mark Craven, and Lewis Friedland. 2008. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, pages 1–10. Vancouver, CA:.

Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. *Proc. VLDB Endow.*, 12(3):223–236.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. 2015. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pages 1954–1963.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

A Illustration of SPEAR on a synthetic setting

Through a synthetic example, 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. Consider a 3-class classification problem with overlap in the feature space as depicted in Figure 1. 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). We randomly pick 5 points from each class $i \in \{a, b, c\}$, and corresponding to each such point (x_i, y_i) we create a labelling function based on its coordinates:

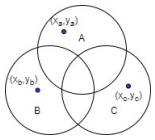

Figure 1: Synthetic data

- 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 7 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 first experiment (result in the first row of Table 7), 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.

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	**0.440**
L4 (1% labelled)+L5	0.28

Table 7: F1 scores on test data in the synthetic setting

We make the following important observations with respect to Table 7: (1) **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. (2) With just 1% labelled data on L1, we achieve 0.349 F1 score (using only the labelled data). (3) 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). (4) 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. (5) 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 7.

B 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).

Datasets	lr (f network)	lr (g network)	Batch Size
SMS	0.0001	0.01	256
MIT-R	0.0003	0.001	512
Census	0.0003	0.001	256
Youtube	0.0003	0.001	32
Ionosphere	0.003	0.01	32
Audit	0.0003	0.01	32
IMDB	0.0003	0.01	32

Table 8: Hyper parameter details for the different datasets

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 Ionosphere dataset, learning rate for f is set to 0.003. 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.

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.

C Optimisation Algorithms and Submodularity: Lazy Greedy and Memoization

Both $f_{\text{unsup}}(X)$ and $f_{\text{sup}}(X)$ are submodular functions, and for data selection, we select a subset

Loss Combination	Datasets				
	YouTube (Accuracy)	SMS (F1)	MIT-R (F1)	IMDB (F1)	Census (Accuracy)
L1+L2+L3+L4	94.6	93.1	72.5	73.6	82.0
L1+L2+L4+L6	92.0	91.9	69.7	73.3	81.3
L1+L3+L4+L6	94.4	93.2	29.8	74.4	81.0
L1+L2+L3+L4+L6	94.4	92.3	29.5	64.4	80.9
L1+L3+L4+L5+L6	**94.6**	**93.4**	**73.2**	**77.1**	**82.0**
L1+L2+L3+L4+L5+L6	94.5	93.0	72.8	76.9	81.9

Table 9: Performance on the test data, of various loss combinations from our objective function in equation (3). For each dataset, 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) contribute to the best model. Note that all combinations includes QG (Component 7).

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 et al., 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.

D Role of different components in the loss function

Given that our loss function 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 7 of the main paper), we attempt to choose the best performing JL combination of the 7 loss components, *viz.* L1, L2, L3, L4, L5, L6. To choose the 'best' JL combination, we evaluate the performance on the validation set of the different JL combinations. 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 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 dataset. Secondly, we observe that QGs do not cause significant improvement in the performance during training.