A Pipeline for Optimizing F1-Measure in Multi-Label Text Classification

Bingyu Wang*, Cheng Li*, Virgil Pavlu*, Jay Aslam*
*College of Computer and Information Science
Northeastern University, Boston, MA, United States
{rainicy, chengli, vip, jaa}@ccs.neu.edu

***Abstract*—Multi-label text classification is the machine learning task wherein each document is tagged with multiple labels, and this task is uniquely challenging due to *high dimensional features* and *correlated labels*. Such text classifiers need to be regularized to prevent severe over-fitting in the high dimensional space, and they also need to take into account label dependencies in order to make accurate predictions under uncertainty. Many classic multi-label learning algorithms focus on incorporating label dependencies in the model training phase and optimize for the strict set-accuracy measure. We propose a new pipeline which takes such algorithms and improves their F1-performance with careful training regularization and a new prediction strategy based on support inference, calibration and GFM, to the point that classic multi-label models are able to outperform recent sophisticated methods (PDsparse, SPEN) and models (LSF, CFT, CLEMS) designed specifically to be multi-label F-optimal. Beyond performance and practical contributions, we further demonstrate that support inference acts as a strong regularizer on the label prediction structure.**

***Keywords*-multi-label; f-measure; text classification**

I. Introduction

Multi-label text classification is challenging, first due to label sets that exhibit complex dependency structures: Text labels such as `election` and `politics` are *dependent* variables in general, so independent predictions per label (Binary Relevance method) is unlikely to work well [1]. Labels can be *many*, so learning approaches dealing explicitly with the exponential number of label subsets (e.g. the PowerSet method [1]) are infeasible; even when feasible, they suffer from scarce data and limited label subsets observed during training. In recent years, there has been an growing interest in developing multi-label methods that are capable of modeling label dependencies in the meantime avoiding exponential computational complexity. Examples include probabilistic classifier chains [2], conditional random fields [3], and conditional Bernoulli mixtures [4]. And the second challenge is that the commonly used bag-of-words feature representation is high dimensional and sparse. For example, the WISE dataset has 301,561 unigram features. If ngram features are further included, the feature size would grow dramatically. Model training without careful regularization can lead to severe over-fitting. This is especially a problem for high-complexity classifiers.

Formally, in a multi-label classification problem, we are given a set of label candidates $\mathcal{L} = \{1, 2, ..., L\}$. Every datapoint $\mathbf{x} \in \mathbb{R}^D$ matches a subset of labels $\mathbf{y} \subseteq \mathcal{L}$, which is often also written in the form of a binary label vector $\mathbf{y} \in \{0, 1\}^L$, with each bit y_l indicating the presence or absence of the corresponding label. The goal is to build a classier $h : \mathbb{R}^D \mapsto \{0, 1\}^L$ which maps an instance to a subset of labels. The label subset $\mathbf{y}$ can be of arbitrary size (written as $|\mathbf{y}| = ||\mathbf{y}||_1$). The *subset accuracy* measure, which is usually reported, gives for each test datapoint a score of 1 if the exact label set is predicted and 0 otherwise and it is usually optimized by maximum likelihood over sets.

In practice however, and particularly in industry, the F-measure, which gives partial rewards for subset predictions based on overlap with the correct subset, is much better suited for many multi-label tasks than strict subset-accuracy. For example, in a medical note, a patient may present with multiple illnesses or undergo a procedure with multiple billing codes; predicting five out of six codes correctly is a considerable help to medical billing systems. Multi-label competitions organized by industrial companies, such as the Yelp business categorization[5] and the Greek Media[6], employ F-measure for evaluation. In this paper, we focus on medium-to-large scale multi-label prediction problems with up to hundreds of labels, such as medical billing codes, movie genres, review objects, patent classification and news categorization. These problems are of particular importance in practice and research: 1) large and complex enough that scalability and regularization need to be considered in the algorithm design; 2) not so large as to prohibit interesting algorithms due to computational requirements; 3) where partially correct predictions are valuable.

We demonstrate how to regularize model complexity during training, and how to regularize the label search space during prediction. This separation allows any multi-label algorithm to give optimal F-measure predictions, so long as it outputs label-set joint probabilities. Specifically, we regularize the classifier during training using the *Elastic-net* (L1+L2) penalty in order to reduce model complexity.At prediction time, we apply *support inference* to restrict the label space to sets encountered in the training set, use *Isotonic Regression* to produce calibrated marginal probabilities and use the General F-measure Maximizer (*GFM*) to make F1-optimal predictions, see Figure 1.

F-measure and Optimal F1 Predictions. The F-measure is by far the most widely used metric for label/tag prediction

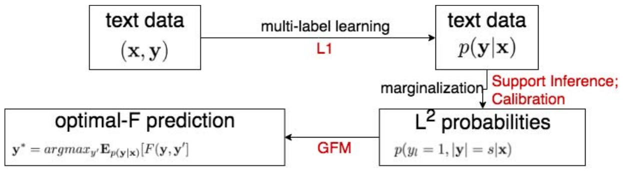
text data
$(\mathbf{x}, \mathbf{y})$
multi-label learning
L1
text data
$p(\mathbf{y}|\mathbf{x})$
marginalization
Support Inference; Calibration
L^2 probabilities
$p(y_l = 1, |\mathbf{y}| = s|\mathbf{x})$
GFM
optimal-F prediction
$\mathbf{y}^* = argmax_{y'} \mathbf{E}_{p(\mathbf{y}|\mathbf{x})}[F(\mathbf{y}, \mathbf{y}']$

Figure 1: Proposed Pipeline

because it assigns partial credit to "almost correct" answers and handles label imbalance well. Let $\mathbf{y}$ be the ground truth label vector, and let $\mathbf{y}'$ be the predicted label vector. The **F1-measure** for an instance is defined as

$$F(\mathbf{y}, \mathbf{y}') = \frac{2\sum_{l=1}^{L} y_l y'_l}{\sum_{l=1}^{L} y_l + \sum_{l=1}^{L} y'_l} \tag{1}$$

which is the harmonic mean between precision and recall. The reported "instance-F1" is the average of F1-measure values over test instances.

Making an optimal prediction based on a trained model to maximize the F1-measure cannot be done, in general, by ranking all labels by their relevance and selecting the top labels. It may seem surprising, in light of the existence of many methods which make predictions by thresholding marginal label probabilities. However, one can see why making predictions based on **label probabilities** alone is suboptimal: many metrics, including F1-measure, are not decomposable over individual labels, and thus in general require the classifier to take into account label dependencies through **joint label probability estimation**. There exist a few methods which explicitly take into account the F1-measure during training [7], [8], [9], but the popular methods that provide a joint estimation in the form of $p(\mathbf{y}|\mathbf{x})$ are trained by standard maximum likelihood estimation without considering the F1-measure as an objective. For such methods, it is still possible to use an F1-optimal prediction strategy post-training, that is, output $\mathbf{y}^*$ which maximizes the expected F1-measure:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}'} \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) \cdot F(\mathbf{y}, \mathbf{y}') \tag{2}$$

The General F-measure Maximizer (**GFM**) [10] is an efficient algorithm that finds the F1-optimal prediction for a given instance based on some probability estimations. The GFM algorithm does not work directly with a joint estimation $p(\mathbf{y}|\mathbf{x})$, but rather, some L^2 marginal distributions (defined more precisely in Section III-B). The paper [10] proposed two ways of obtaining these L^2 marginals (or probabilities per instance): 1) a model which directly estimates L^2 marginals from data, and 2) the use of a probabilistic joint estimator $p(\mathbf{y}|\mathbf{x})$ and sampling to generate the required L^2 probabilities. We find that option 1) is very difficult, perhaps unsolvable, although it is indeed appealing as a theoretical exercise. We instead develop an efficient solution based on 2) with a critical change: after training the joint estimator $p(\mathbf{y}|\mathbf{x})$, we derive the required L^2 marginals using **support inference** and **Calibration**. These marginals are then fed into GFM to produce the F1-optimal prediction (Figure 1).

II. Multi-label Classifiers

For each multi-label classifier considered, we describe its probabilistic formulation $p(\mathbf{y}|\mathbf{x})$ and its standard argmax prediction method, which gives $\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$. It is worth noting that the argmax prediction, although very common, only provides the optimal prediction for the exact-set accuracy measure, but not for the F1-measure [10], [11].

Binary Relevance (BR) [1] assumes that all labels are independent and thus the joint over all labels is a product of marginals, see (3). This independence assumption simplifies both training and prediction: L binary classifiers (logistic regressions) are trained, one for each label; the argmax prediction simply predicts each label independently.

$$p(\mathbf{y}|\mathbf{x}) = \prod_{l=1}^{L} p(y_l|\mathbf{x}) \tag{3}$$

Probabilistic Classifier Chain (PCC) [2] decomposes the joint density estimator into a product of conditionals using the chain rule, in (4). During training, one logistic regression is trained for each label based on both features and all previous labels; it models label dependency, but the pre-set order of labels in the decomposition is critical. The exact argmax prediction is generally intractable. Beam search is often used as an approximate argmax prediction procedure [12], which is applied in our experiments.

$$p(\mathbf{y}|\mathbf{x}) = p(y_1|\mathbf{x})p(y_2|\mathbf{x}, y_1) \cdots p(y_L|\mathbf{x}, y_1, .., y_{L-1}) \tag{4}$$

Pair-wise Conditional Random Field (CRF) [3] defines a log-linear model with potential functions for feature-label pairs and label-label pairs.

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\{ & \sum_{l=1}^{L}\sum_{d=1}^{D} w_{ld} x_d \mathbb{1}[y_l = 1] \\ + \sum_{l=1}^{L}\sum_{m=1}^{L} & (w_{lm1}\mathbb{1}[y_l = 0, y_m = 0] + w_{lm2}\mathbb{1}[y_l = 0, y_m = 1] \\ & + w_{lm3}\mathbb{1}[y_l = 1, y_m = 0] + w_{lm4}\mathbb{1}[y_l = 1, y_m = 1])\} \end{aligned} \tag{5}$$

where $Z(\mathbf{x})$ is the normalization constant, and all w are estimated weights. All weights are trained jointly using maximum likelihood estimation. Both computing the partition function $Z(\mathbf{x})$ and the exact argmax predictions are intractable, as an exponential number of label combinations are involved. [3] suggests using support inference to solve the intractability issues: the idea is to restrict $\mathbf{y}$ values only to label combinations observed in the training set when computing scores and probabilities. While the authors propose support inference only as an approximation, we think it is actually the main reason CRF works well: support inference adds strong regularization and dependency effects.

Conditional Bernoulli Mixtures (CBM) [4] represents the joint as a mixture of K components, each with independent label classifiers, shown in (6). The multi-class classifier (a multi-nomial logistic regression) π decides the mixing coefficient for each mixture component. Inside each component, the joint is factorized into marginals, estimated by L binary classifiers b (binary logistic regressions). Both the multi-class classifier and the binary classifiers are trained jointly by EM algorithm. The exact argmax can be computed efficiently using dynamic programming.

$$p(\mathbf{y}|\mathbf{x}) = \sum_{k=1}^{K} \pi(z = k|\mathbf{x}) \prod_{l=1}^{L} b(y_l|\mathbf{x}, z = k) \qquad (6)$$

Table I: Datasets Characteristics

	BIBTEX	IMDB	OHSUMED	RCV1	WISE	WIPO
domain	bkmark	genre	medical	news	articles	patent
source	Mulan	crawled*	MEKA*	Mulan	WISE2014	HRSVM
labels	159	27	23	101	203	188
label sets	2,058	2,122	1,042	494	3,536	155
features	1,836	27,228	16,344	47,236	301,561	74,435
instances	7,395	34,157	13,929	6,000	64,857	1,710
cardinality	2.40	2.52	1.66	3.23	1.45	4.00
inst/label	112	2537	1007	188	463	36

Note: cardinality = average number of labels per instance; inst/label = the average number of training instances per label. Except for IMDB, all datasets are publicly available. "*"= we processed the source documents and recomputed all-unigram feature values, because the published feature matrix did not include all unigrams and the pre-processing was unclear.

III. Training and Prediction for Optimal F

We propose a classifier training and prediction pipeline which works with all the existing multi-label classifiers described above. It adds careful regularization to the classifier training and employs a new prediction strategy. With these enhancements, most multi-label classifiers studied here are able to outperform recent other methods (see Table V).

A. Training Regularization

L1 Regularization. L2 regularization is the most commonly used regularization techniques for text classifiers. However, because the number of model parameters (for the classifiers described above) grows at least linearly with the number of labels and the number of bag-of-words/ngrams features, and both of which are large on multi-label text data, using only L2 regularization leads to a large number of model parameters and hence over-fitting.

One remedy for the high dimensionality is to apply the L1 regularization to perform feature selection by shrinking some irrelevant feature weights to zero. However, L1 alone can often pick only one out of many highly correlated features, possibly hurting the generalization — so L2 is still necessary for spreading weights across correlated features. In our pipeline we regularize all model training (except CRF, which is difficult) with the elastic-net regularization [13], which combines both L1 and L2 regularization, in the form $\lambda\{\alpha||w||_1 + (1-\alpha)||w||_2^2\}$, to get the best of both worlds .

B. Prediction Strategy

At prediction time, the task is to find the Bayes optimal prediction $\mathbf{y}^*$ that gives the highest expected F1-measure by formula (2) under the predictive distribution $p(\mathbf{y}|\mathbf{x})$.

GFM: optimal prediction for F1-measure. The General F-Measure Maximizer (**GFM**) algorithm [10] is an exact and efficient algorithm for computing $\mathbf{y}^*$ in $\Theta(LT^2)$ time, where T is the average number of labels per instance. However, the GFM algorithm does not work directly with a joint estimation $p(\mathbf{y}|\mathbf{x})$ provided by standard classifiers, but rather, some marginal distributions of the form

$$p(y_l = 1, |\mathbf{y}| = s \mid \mathbf{x}), \ \forall l, s \in \{1, ..., L\} \qquad (7)$$

where $|\mathbf{y}|$ stands for the number of relevant labels in $\mathbf{y}$. This formula can be read as, for example, "the probability of the given document having $s = 5$ relevant labels and y_l =`election` is one of them". Obviously, there are (no more than) L^2 probabilities in this form, per instance.

Support Inference. [10] proposes to sample from the joint $p(\mathbf{y}|\mathbf{x})$ and then compute the GFM input probabilities based on the samples. However sampling is ineffective for large L and low-confidence ("flat") joint density that spreads probability mass over many label sets. Our proposed way of producing the GFM L^2 input probabilities from the joint $p(\mathbf{y}|\mathbf{x})$ is through support inference, which only considers those label combinations $\mathbf{y}$ in the training set and marginalizes over their probabilities. This is more efficient than sampling, and also provides some additional regularization effect on the label structures, as demonstrated later.

At first glance, support inference seems to have the limitation of not considering unseen label combinations. In reality this limitation only appears *during marginalization*, and is largely mitigated by GFM in the prediction step. It is not hard to show that although support inference only considers existing combinations, support inference + GFM can output unseen combinations. Thus support inference provides a regularized probability estimation by only assigning probability mass to observed combinations, and GFM takes this regularized probability estimation as the input and outputs F optimal prediction, which could potentially be an unobserved label combination. We observe this strategy to work remarkably well for many classifiers and datasets.

Calibration. It is often the case that the probability estimations given by the classifiers are *uncalibrated*, meaning that the probabilities do not align well with the actual prediction accuracy. One can further calibrate these probabilities on a validation set using some calibration method such as Isotonic Regression [14]. We find that calibrating the L^2 marginal probabilities produced by support inference helps GFM make better predictions. Putting together, our proposed overall prediction strategy is to first run support inference to compute $p(\mathbf{y}|\mathbf{x})$ for each $\mathbf{y}$ in the training set, and then marginalize over them to get the required L^2 marginal probabilities, and then run Isotonic Regression to calibrate these probabilities, and finally run GFM on the calibrated L^2 probabilities to make a prediction.

Table II: F-measure on test w/ and w/o L1(L), Support Inference(S), GFM(G) and Calibration(C)

Data	Model	Standard	SG	L	LS	LG	LSG	LSCG
BIBT	BR	37.8	44.5	39.8	44.4	40.2	45.4	**48.1**
	CRF\ L1	-	-	-	46.5	-	49.4	**49.5**
	PCC	37.4	45.3	39.5	45.0	40.1	47.3	**48.2**
	CBM	44.0	45.9	45.3	46.9	40.4	49.5	**50.4***
IMDB	BR	59.4	61.8	59.6	59.7	61.0	61.4	**63.8**
	CRF\ L1	-	-	-	63.0	-	66.6	**67.1***
	PCC	59.6	63.9	60.1	60.2	61.5	62.8	**64.4**
	CBM	61.6	65.1	62.2	62.2	64.8	65.2	**66.2**
OHSU	BR	60.2	67.9	63.6	68.0	64.3	69.1	**71.0**
	CRF\ L1	-	-	-	66.4	-	69.6	**70.5**
	PCC	62.5	70.1	64.7	68.4	65.8	70.4	**72.1**
	CBM	68.7	70.3	69.5	70.3	65.4	71.7	**72.6***
RCV1	BR	72.1	73.7	73.8	74.6	74.9	75.1	**76.1**
	CRF\ L1	-	-	-	74.4	-	75.8	**76.1**
	PCC	71.0	73.6	72.7	72.8	74.3	74.1	**74.4**
	CBM	76.6	77.3	77.3	78.5	77.9	**79.2***	78.7
WISE	BR	68.0	77.3	72.8	79.0	73.0	79.3	**80.1**
	CRF\ L1	-	-	-	77.7	-	79.0	**79.4**
	PCC	70.7	76.0	74.6	76.7	77.1	**78.0**	-
	CBM	77.9	78.6	79.8	79.8	73.6	80.3	**81.5***
WIPO	BR	63.4	71.2	69.5	73.2	70.0	**74.0**	68.0
	CRF\ L1	-	-	-	70.3	-	72.2	**72.5**
	PCC	68.8	71.5	70.2	70.4	70.6	**72.3**	54.6
	CBM	63.0	70.8	69.6	72.5	70.3	**74.3***	71.3

Note:**bold**: best in row; *: best in dataset; "-": N/A (CRF requires S); "CRF\L1": CRF w/o L1.

IV. Experimental Results & Analysis

Datasets and Experiment Setup. The multi-label text datasets used in experiments are shown in Table 1. We adopt the given train/test split whenever it is provided; otherwise we use a random 20% of the data as the test set. Hyper parameter tuning for all algorithms is done by cross validation on the training set and F1-measure on the test set is reported. For methods involving random initializations or sampling, reported results are averaged over 3 runs. When applying L1 regularization, we use L1 penalty together with the basic L2 penalty in the elastic-net form $\lambda\{\alpha||w||_1+(1-\alpha)||w||_2^2\}$, and we tune the overall strength λ and the L1 ratio α. When L1 penalty is not included, we only keep L2 penalty by setting $\alpha = 0$ and we only tune λ.

A. Analysis: L1 Regularization

First we analyze the regularization effects of L1 penalty during training. The experiment results are summarized in Table II. The letters "L, S, G" stand for "L1", "support inference" and "GFM", respectively. Each column uses a different subset of these techniques. The "Standard" column does not use any of these. It follows the convention that uses only L2 penalty to regularize logistic regression learners, trains each model until full convergence, and performs argmax prediction during prediction. This column serves as a baseline. Comparing the column "LSG" with the column "SG", we can see that overall introducing some L1 penalty improves performance on 5 out of 6 datasets—BIBTEX, OHSUMED, RCV1, WISE, WIPO, but not IMDB. Also the difference L1 makes is a function mainly of the dataset, and less of the classifier (CRF is excluded due to the absence of L1).

Table III: Model size and feature used

Data	BR			CBM		
	L2 model size(MB)	L1L2 feature used	L1L2 model size	L2 model size(MB)	L1L2 feature used	L1L2 model size
BIBT	7	100%	26%	135	100%	4%
IMDB	20	66%	21%	355	99%	10%
OHSU	10	53%	34%	177	68%	6%
RCV1	48	70%	12%	910	77%	2%
WISE	1.4(G)	14%	1%	13(G)	24%	<1%
WIPO	294	42%	2%	6G	77%	2%

Note: Percentages of the L2 model/feature size after adding L1 in BR and CBM. Base L2 models use all features.

Each dataset has some intrinsic properties such as the number of relevant documents per label, the diversity of the topics, the total number of documents and the total number of features, that dictate how many features have to be used in order to explain the given labels/topics well and how many model parameters can be reliably estimated based on the given dataset size, and these factors in turn influence how much improvement L1 feature selection can bring in.

Apart from improving test F1 performance, L1 also shrinks the model sizes massively. The model size is measured by the disk space the model occupies. Table III compares the sizes of models trained with only L2 versus models trained with both L1 and L2 penalty. Generally, adding L1 shrinks models to no more than 10% of its original sizes for CBM. On some datasets, such as RCV1, WISE and WIPO, the shrunk CBM models are only about 1%. Interesting, if we look at the total number of features selected by the classifiers, the reduction in feature size is not as dramatic as the reduction in model size. This is in direct contrast with binary classification, where these two reductions mostly agree. By looking into the trained multi-label classifiers, we notice that although many features are relevant for some labels and are thus included in the classifiers, for each individual label, only a few features actually have non-zero weights. Thus, although the union of relevant features for all labels can be large, each label predictor can be a small model that includes a few features, and the entire multi-label classifier is therefore quite compact.

B. Analysis: GFM, Support Inference, Calibration

We now analyze the effect of each step in the proposed prediction strategy. Experiment results are summarized in Table II. The new letter "C" stands for "Calibration", and all others stay the same. The "L" column uses argmax prediction as described in Section II. Comparing the "L" column with the "LSCG" column, it is clear that the proposed prediction strategy performs better than argmax prediction in terms of F1 for almost all methods on all datasets. Furthermore, the effect of calibration can be observed by comparing the "LSCG" column with the "LSG" column. Adding calibration consistently boosts the performance by 1 percent on all datasets except WIPO.

The "LG" column only uses GFM predictor but not

Table IV: F-measure on CRF w/ and w/o label-label pair.

	BIBT		IMDB		OHSU		RCV1		WISE		WIPO	
pairwise	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
CRF w/o GFM	46.9	46.5	61.3	63.0	66.1	66.4	73.8	74.4	78.2	77.7	70.7	70.3
CRF w/ GFM	49.4	49.4	66.1	66.6	69.8	69.6	75.8	75.8	79.4	79.0	71.8	72.2

support inference. For each method, we sample 1000 times based on the estimated joint and use samples to compute the marginals probabilities required by the GFM predictor, as described in Section III-B (The CRF numbers are missing as there is no straight forward way of sampling from CRF). Comparing "LG" with column "L", we see that GFM alone gives some improvement for BR and PCC, but is less effective for CBM. However, CBM clearly benefits from GFM in conjunction with support inference.

The "LS" column only uses support inference but not GFM prediction. We use support inference to restrict the label combinations to those observed in the training set, and among them, we pick the one with the highest probability. Comparing "L" with the "LS" column, we see that adding the support inference alone consistently improves the test performance (except for CRF, for which support inference is always used, as described in Section II). The improvement is most substantial on BR, which did not estimate any label dependencies during training, and is relatively small on CRF, PCC and CBM, which already estimated some label dependencies during training. Thus support inference acts as a regularizer on the label structure.

The Role of Support Inference It is known that if CRF as described in Section II only contains label-feature interaction but not label-pair interaction, and if the partition function is computed exactly by summing overall all label combinations (conceptually), then the resulting model is mathematically equivalent to BR. In this case, theoretically there is no need to compute the partition function approximately using support combinations. So the authors in [3] only use support inference when CRF contains label pair interactions and thus the model does not factorize and one has to somehow compute the partition function approximately. Support inference there was deemed purely as an approximate inference procedure. But our results show, perhaps surprisingly, that support inference in fact helps on BR – in other words, even if we could compute the CRF partition function exactly, it is still beneficial to compute it approximately, using support inference. Table IV shows that CRF without label pair terms does almost equally well as the one with label pair terms. This demonstrates support inference as a simple yet powerful regularizer, besides its original role as an approximation.

V. Comparisons with Related Methods

Theoretically GFM can produce the Bayesian optimal prediction given the L^2 marginal probability inputs. It may appear that if the end goal is to produce these L^2 probabilities as input to the GFM algorithm, it should be more straightforward to estimate these L^2 probabilities directly rather than going through a joint estimation first, which by itself is quite challenging (the joint evolves 2^L probabilities in general). In fact, it is conceptually not hard to derive an algorithm (called Label Square Functions (**LSF**)) to estimate the L^2 marginals directly, as suggested in [10]. We can estimate each $p(y_l = 1, |\mathbf{y}| = s \mid \mathbf{x})$ using a binary logistic regression. Another option is to estimate $p(y_l = 1|\mathbf{x})$ with a binary logistic regression, and then $p(|\mathbf{y}| = s|\mathbf{x}, y_l = 1)$ with another multinomial logistic regression and then multiply their probabilities. However, in practice, Table V shows that this direct estimation approach does not perform well: directly predicting the number of relevant labels $|\mathbf{y}|$ by a classifier is a very hard and unnatural task.

Besides the BR, PCC, CRF, CBM and LSF models with logistic regression learners described so far, there also exist other multi-label methods, some of which are quite effective for certain metrics. However, once F1-measure is concerned, most of these other methods lack an explicit probability estimation, which is indispensable for GFM prediction. One example is the BR model with linear SVM learners. We take the widely used Liblinear package [15] for state-of-the-art linear SVM and Logistic Regression implementations, with carefully tuned hyper parameters, and notice that the predictions from the package have much lower F1-measure compared with our proposed BR + LSCG (see Table V).

The **PD-Sparse** method [16] is recently proposed for extremely large scale multi-label classification. It employs a Dual Fully-Corrective Block-Coordinate Frank-Wolfe algorithm that exploits both primal and dual sparsity to achieve high efficiency. However, PD-Sparse only computes a non-probabilistic score for each label and ranks labels by scores. It does not provide a straightforward way of predicting a set of labels for each instance. The original implementation provided by the authors ask the users to provide the desired number of labels per instance and returns the top labels with highest scores as predictions. Because the correct number of labels varies greatly from instance to instance, predicting a fixed number of labels for all instances results in sometimes low precision (when the specified number of labels is more than necessary), sometimes low recall (when the specified number of labels is less than necessary), and overall low F1-measure. Since PD-Sparse does not provide probability estimations, GFM cannot be plugged in to predict optimal F1. We tried to make the PD-Sparse predictions more adaptive by tuning the threshold of the label scores to maximize the F1-measure, but PD-Sparse still performs much worse than our proposed methods (see Table V).

There are also several neural network based multi-label classification methods [17], [18], [19], [20]. We run the code associated with the recently proposed Structured Prediction Energy Networks (**SPEN**) [17] with carefully tuned hyper parameters as suggested by the authors and observe that SPEN's performance to be less competitive (see Table V), possibly due to over-fitting in high dimensional data with neural network's high model capacity.

Table V: F-measure comparisons with other methods.

Method	BIBT	IMDB	OHSU	RCV1	WISE	WIPO
BR SVM + L2	37.8	59.9	60.9	73.4	70.0	64.7
BR SVM + L1	39.3	59.0	63.5	73.0	70.0	68.1
BR LR + L2	38.1	60.0	61.1	72.3	68.6	64.3
BR LR + L1	39.0	60.5	61.4	73.4	70.4	68.7
LIFT	31.5	-	54.4	70.2	-	61.6
SPEN + L2	39.0	61.1	61.7	65.3	-	65.9
PDsparse+L1L2	40.7	62.3	67.3	75.0	74.5	67.5
CFT	23.5	-	-	53.5	-	62.7
CLEMS	42.5	-	52.6	72.4	-	67.1
LSF	43.9	59.8	65.0	73.6	76.7	71.1
BR+LSCG†	48.1	63.8	71.0	76.1	80.1	68.0
CRF+LSCG†	49.5	**67.1**	70.5	76.1	79.4	**72.5**
CBM+LSCG†	**50.4**	66.2	**72.6**	**78.7**	**81.5**	71.3

Note: †: our method; '-': indicates failed runs with 56 core and 256GB RAM.

The **LIFT** algorithm [21] constructs features specific to each label by conducting clustering analysis on its positive and negative instances, and then performs training and testing by querying the clustering results. We run the code provided by the authors and follow the suggested hyper parameters and report the results in Table V. LIFT does not perform well and could not finish on two datasets.

There are several approaches that seek to optimize the F-measure directly during training. [9] provides an up-to-date overview on different F-measure maximization methods. [22] uses a graph-cut algorithm and has poor scalability on high dimensional text datasets. There are three methods that use a cost-sensitive approach to optimize F-measure score during training [23], [24], [7]. We tested the Condensed Filter Tree method (**CFT**) [23] and the cost-sensitive label embedding with multidimensional scaling method (**CLEMS**) [24] and found both to perform poorly and their training to be also slow (see Table V). [8] studies F-measure maximization with conditionally independent label subsets. This method has a strong assumption which makes it hard to apply to real data.

VI. Conclusion

In this paper our main goal is to develop a pipeline in order to reuse classic multi-label models to achieve high F1 scores on multi-label text data. We show that most multi-label classification algorithms can be used in the pipeline as long as they produce a joint estimator $p(\mathbf{y}|\mathbf{x})$. We show that with careful training regularization and special prediction strategy based on Support Inference, Calibration and GFM, these classic methods can outperform recent sophisticated methods (PDsparse, SPEN) and models (LSF, CFT, CLEMS) designed specifically to be multi-label F-optimal.

References

[1] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *Int J Data Warehousing and Mining*, vol. 2007, pp. 1–13, 2007.

[2] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, pp. 333–359, 2011.

[3] N. Ghamrawi and A. McCallum, "Collective multi-label classification," in *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM, 2005, pp. 195–200.

[4] C. Li, B. Wang, V. Pavlu, and J. A. Aslam, "Conditional bernoulli mixtures for multi-label classification," in *ICML*, 2016, pp. 2482–2491.

[5] Yelp, "Automatically categorizing yelp businesses," Sep 2015, https://engineeringblog.yelp.com/amp/2015/09/automatically-categorizing-yelp-businesses.html.

[6] WISE, "Greek media monitoring multilabel classification (wise 2014)," July 2014, www.kaggle.com/c/wise-2014.

[7] S. P. Parambath, N. Usunier, and Y. Grandvalet, "Optimizing f-measures by cost-sensitive classification," in *Advances in Neural Information Processing Systems*, 2014, pp. 2123–2131.

[8] M. Gasse and A. Aussem, "F-measure maximization in multi-label classification with conditionally independent label subsets," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 619–631.

[9] I. Pillai, G. Fumera, and F. Roli, "Designing multi-label classifiers that maximize f measures: State of the art," *Pattern Recognition*, vol. 61, pp. 394–404, 2017.

[10] W. Waegeman, K. Dembczyński, A. Jachnik, W. Cheng, and E. Hüllermeier, "On the bayes-optimality of f-measure maximizers," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3333–3388, 2014.

[11] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence and loss minimization in multi-label classification," *Machine Learning*, vol. 88, no. 1-2, pp. 5–45, 2012.

[12] A. Kumar, S. Vembu, A. K. Menon, and C. Elkan, "Beam search algorithms for multilabel learning," *Machine learning*, vol. 92, no. 1, pp. 65–89, 2013.

[13] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, vol. 33, no. 1, p. 1, 2010.

[14] R. E. Barlow, "Statistical inference under order restrictions; the theory and application of isotonic regression," Tech. Rep., 1972.

[15] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[16] I. E. Yen, X. Huang, K. Zhong, P. Ravikumar, and I. S. Dhillon, "Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification," in *Proceedings of the 33nd International Conference on Machine Learning*, 2016, code: http://www.cs.utexas.edu/~xrhuang/PDSparse/.

[17] D. Belanger and A. McCallum, "Structured prediction energy networks," in *ICML*, 2016.

[18] J. Nam, J. Kim, E. L. Mencía, I. Gurevych, and J. Fürnkranz, "Large-scale multi-label text classification revisiting neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 437–452.

[19] M. Cissé, C. M. Al-Shedivat, and S. Bengio, "Adios: Architectures deep in output space," in *ICML*, 2016.

[20] V. Mnih, H. Larochelle, and G. E. Hinton, "Conditional restricted boltzmann machines for structured output prediction," *arXiv preprint arXiv:1202.3748*, 2012.

[21] M.-L. Zhang and L. Wu, "Lift: Multi-label learning with label-specific features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 1, pp. 107–120, 2015.

[22] J. Petterson and T. S. Caetano, "Submodular multi-label learning," in *Advances in Neural Information Processing Systems*, 2011, pp. 1512–1520.

[23] C.-L. Li and H.-T. Lin, "Condensed filter tree for cost-sensitive multi-label classification." in *ICML*, 2014, pp. 423–431.

[24] K.-H. Huang and H.-T. Lin, "Cost-sensitive label embedding for multi-label classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1725–1746, 2017.