LibMultiLabel: a Library for Multi-label Classification

LibMultiLabel Project Authors*

February 23, 2024

Abstract

LibMultiLabel is an open source software for binary, multi-class and multi-label classification, supporting various neural network architectures and linear classifiers. LibMultiLabel can be found at `https://www.csie.ntu.edu.tw/~cjlin/libmultilabel/` This paper provides the mathematical formulations and implementation details of LibMultiLabel.

Contents

*See contributors at https://github.com/ASUS-AICS/LibMultiLabel/graphs/contributors

1 Metrics

Metrics are functions that represent the performance of models during evaluation. When predicting, we use a model to calculate the scores for an instance associated with labels. For example, let $\boldsymbol{w}$ be the weight of a linear model for label l. Then for a given instance $\boldsymbol{x}$, the score of $\boldsymbol{x}$ for label l is calculated by $\boldsymbol{w}^T\boldsymbol{x}$. This score will be used to decide whether this instance is associated with label l. For this reason, we called this score decision value. If a given instance $\boldsymbol{x}$ has the label l, we say that the label l is relevant to $\boldsymbol{x}$.

For a given data instance, let L be the number of labels and

$$\begin{aligned} \boldsymbol{p} &= \begin{bmatrix} p_1 & p_2 & \cdots & p_L \end{bmatrix} \in \mathbb{R}^L, \\ \hat{\boldsymbol{y}} &= \begin{bmatrix} \hat{y_1} & \hat{y_2} & \cdots & \hat{y_L} \end{bmatrix} \in \{0,1\}^L, \\ \boldsymbol{y} &= \begin{bmatrix} y_1 & y_2 & \cdots & y_L \end{bmatrix} \in \{0,1\}^L \end{aligned} \tag{1}$$

be the decision values, the predictions, and the ground truths associated with the instance respectively. The value of 1 indicates a relevant label and 0 indicates an irrelevant label. Define $I_p = \{i_1, i_2, ..., i_L\}$ to be the sorted index of $\boldsymbol{p}$ by decision values.

1.1 Precision and Recall at K

Precision@K aims to check that among the top-K predictions for a given instance, how many labels are relevant to the instance. So, precision@K for the instance is defined as follows

$$\text{P@K} = \frac{\text{\#relevant labels in the top-}K\text{ predictions}}{K} = \frac{\sum_{s=1}^{K} y_{i_s}}{K}. \tag{2}$$

On the other hand, recall@K shows that among labels associated with the given instance, how many are in the top-K predictions. Recall@K for the instance can be defined as follows

$$\text{R@K} = \frac{\text{\#relevant labels in the top-}K\text{ predictions}}{\text{\#relevant labels}} = \frac{\sum_{s=1}^{K} y_{i_s}}{\sum_{s=1}^{L} y_s}. \tag{3}$$

Note that, we set $\text{R@K} = 0$ if the number of relevant labels associated with the instance is zero.

The values of P@K and R@K over the entire dataset is the average of D instances, calculated as follows:

$$\begin{aligned} \text{P@K} &= \frac{1}{D}\sum_{j=1}^{D} \text{P@K for the } j\text{th instance}, \\ \text{R@K} &= \frac{1}{D}\sum_{j=1}^{D} \text{R@K for the } j\text{th instance}. \end{aligned}$$

1.2 R-Precision at K

For instances where the number of relevant labels is less than K, even with a perfect prediction, P@K will be smaller than 1. This is because

$$\text{P@K} = \frac{\text{\#relevant labels in the top-}K\text{ predictions}}{K} < \frac{K}{K} = 1.$$

On the other hand, when K is smaller than the number of relevant labels, then even with a perfect prediction, R@K will be smaller than 1. The reason is

$$\text{R@K} = \frac{\text{\#relevant labels in the top-}K\text{ predictions}}{\text{\#relevant labels}} < \frac{\text{\#relevant labels}}{\text{\#relevant labels}} = 1.$$

For example, if the instance associates with two labels, then the value of P@5 for a perfect prediction is 0.4 and the value of R@1 for a perfect prediction is 0.5. For this reason, we cannot ensure that the values of P@K and R@K over different datasets for perfect predictions are always 1. To ensure the maximum value of the metric is 1, R-Precision at K (RP@K) may be used.

RP@K is very similar to P@K and R@K as the only difference is in the denominator. The denominators of P@K and R@K are K and the number relevant labels respectively. Instead, the denominator of RP@K is

$$\min(K, \text{\#relevant labels of the instance}).$$

With this change, the maximum value of RP@K is always 1. The definition of RP@K for the instance is as follows

$$\text{RP@K} = \frac{\text{\#relevant labels in the top-}K\text{ predictions}}{\min(K, \text{\#relevant labels of the instance})} = \frac{\sum_{s=1}^{K} y_{i_s}}{\min(K, \text{\#relevant labels of the instance})}.$$

Similarly, the value of RP@K over the entire dataset is the average of D instances, calculated as follows:

$$\text{RP@K} = \frac{1}{D}\sum_{j=1}^{D} \text{RP@K for the } j\text{th instance.}$$

1.3 Normalized Discounted Cumulative Gains at K

When the number of relevant labels in the top-K predictions are the same for two predictions, then by (2) and (3), these two predictions will have the same value of P@K and R@K. In this case, these metrics cannot discriminate between the two predictions. For example, consider the ground truth and two predictions for an instance as follows

$$\begin{aligned} \text{ground truth} &= [0, 1, 1, 0, 0], \\ \text{decision values of prediction 1} &= [0.1, 0.3, 1.0, -0.3, -0.7], \\ \text{decision values of prediction 2} &= [0.8, 0.2, 0.7, -0.1, -0.5]. \end{aligned}$$

In this case, P@5 for these two predictions are both 0.4, but have different orders of labels.

To understand why this matters, consider a search engine. If these are the search results, we hope that positive labels appear first. That is, positive labels have higher ranks. From this perspective, prediction 1 is better than prediction 2 in the example above.

To solve this problem, we use another metric called normalized discounted cumulative gains at K (NDCG@K). Before introducing how to compute NDCG@K, we need to understand what DCG@K and IDCG@K are.

Discounted cumulative gains at K (DCG@K) measures the top-K predictions by taking discounts for different ranks. With this metric, the above two predictions will have different values of DCG@K and we can use these values to compare which is better. DCG@K for the instance is defined as follows

$$\text{DCG@K} = \sum_{s=1}^{K} \frac{y_{i_s}}{\log_2(s+1)}.$$

A problem with DCG@K is that it is not comparable across instances with a different number of relevant labels. For example, consider these two instances

$$\begin{aligned} \text{grouth truth } 1 &= [0, 1, 0, 0, 0], \\ \text{decision values of prediction } 1 &= [0.1, 1.2, -0.9, -0.7, -0.5], \\ \text{grouth truth } 2 &= [1, 0, 1, 0, 1], \\ \text{decision values of prediction } 2 &= [0.3, 1.0, 0.4, -0.9, 0.1]. \end{aligned}$$

Then DCG@5 for these two instances will be 1 and 1.52 respectively. Despite the first instance having the best possible prediction, it has a lower DCG@5 than the second instance. To solve this problem, one way is to consider the ratio of DCG@K for a prediction and DCG@K for the best prediction. DCG@K for the best prediction is called ideal DCG@K (IDCG@K) and this ratio called normalized DCG@K (NDCG@K).

IDCG@K is the maximum value of DCG@K. The maximum value of DCG@K occurs when all of the relevant labels are ranked higher then irrelevant labels. In other words, let $I = \min(K, \|\boldsymbol{y}\|_0)$. Note that $\|\boldsymbol{y}\|_0$ is the 0-norm of $\boldsymbol{y}$, which is the number of non-zero elements of $\boldsymbol{y}$. Then the maximum value of DCG@K occurs when the top-I predictions for the given instance are all relevant. Thus, the expression of IDCG@K for the instance is defined as

$$\text{IDCG@K} = \sum_{i=1}^{\min(k, \|\boldsymbol{y}\|_0)} \frac{1}{\log_2(i+1)}.$$

NDCG@K shows how close the prediction is to the best possible prediction, calculated as follows:

$$\text{NDCG@K} = \frac{\text{DCG@K for the instance}}{\text{IDCG@K for the instance}}.$$

The value of NDCG@K over the entire dataset is the average of each instance, calculated as follows:

$$\text{NDCG@K} = \frac{1}{D} \sum_{j=1}^{D} \text{NDCG@K for the } j\text{th instance.}$$

1.4 F-measure

In the above, we introduced some ranking measures. They only check top-K predictions and K is usually a small number. When we need to consider the whole predictions, a ranking metric may not be a good choice. Instead, we may choose some classification measures like the F-measure. F-measure is one of the most used performance measures for information retrieval systems. It is the harmonic means of precision (P) and recall (R).

Precision shows that among predictions for all instances, how many positive predictions are correct. Recall shows that among positive instances, how many are predicted. Precision and recall for label l are expressed as follows:

$$P_l = \frac{\#\text{TP for label } l}{\#(\text{TP} + \text{FP}) \text{ for label } l} \quad \text{and} \quad R_l = \frac{\#\text{TP for label } l}{\#(\text{TP} + \text{FN}) \text{ for label } l},$$

where TP, FP, and FN are defined in Table 1.

Table 1: Definition of TP, FP, FN, and TN.

prediction \ ground truth	True	False
True	TP (true positive)	FP (false positive)
False	FN (false negative)	TN (true negative)

Then the F-measure for label l is

$$F_l = \frac{2 \cdot P_l \cdot R_l}{P_l + R_l} = \frac{2\#\text{TP for label } l}{(2\#\text{TP} + \#\text{FP} + \#\text{FN}) \text{ for label } l}$$

To extend the F-measure from single-label to multi-label, two approaches are developed in Tague (1981). The first is the macro-average F-measure, which is the unweighted mean of label F-measures,

$$\text{Macro-F1} = \frac{1}{L}\sum_{l=1}^{L} F_l = \frac{1}{L}\sum_{l=1}^{L} \frac{2\#\text{TP for label } l}{(2\#\text{TP} + \#\text{FP} + \#\text{FN}) \text{ for label } l}.$$

Some use a different way, denoted as Macro*-F1, by calculating the average percision and recall over all labels first.

$$\bar{P} = \frac{1}{L}\sum_{l=1}^{L} P_l = \frac{1}{L}\sum_{l=1}^{L} \frac{\#\text{TP for label } l}{\#(\text{TP} + \text{FP}) \text{ for label } l},$$

$$\bar{R} = \frac{1}{L}\sum_{l=1}^{L} R_j = \frac{1}{L}\sum_{l=1}^{L} \frac{\#\text{TP for label } l}{\#(\text{TP} + \text{FN}) \text{ for label } l},$$

$$\text{Macro*-F1} = \frac{2 \cdot \bar{P} \cdot \bar{R}}{\bar{P} + \bar{R}}.$$

Opitz and Burst (2021) suggest that Macro*-F1 is less suitable to use.

The other multi-label measure is the micro-average F-measure, which calculates total TP, FP, and FN first.

$$\text{Micro-F1} = \frac{\sum_{l=1}^{L} \#\text{TP for label } l}{\sum_{l=1}^{L} (2\#\text{TP} + \#\text{FP} + \#\text{FN}) \text{ for label } l}.$$

.

1.5 Choosing the Suitable Metrics

The choice of metrics should be motivated by the use case of the model. No metric fits every scenario equally well.

For example, if the model is used as a large-scale search engine, then the number of labels will be enormous. In this case, only the first few dozens of search results are important because no user will read every one of the results. For this reason, multi-label problems with a large amount of labels are often only concerned about the top few predictions. In this case, we might use P@K or NDCG@K with a choice of K that reflects the use case well.

In contrast, multi-label problems with a small amount of labels are often concerned with predicting all the labels correctly. For example, illness prediction in medical data is usually concerned about every label. In such a case, we may choose to use Macro-F1 and Micro-F1.

2 Handling zero-shot labels

In some cases, there exist labels that only appear in the test data. These labels are called zero-shot labels. We provide an option include_test_labels in LibMultiLabel to handle these labels in evaluation. This option can be true or false to decide whether to include zero-shot labels for evaluation. In this section, we illustrate some details on how to choose a correct value of include_test_labels.

2.1 The Default Behavior in LibMultiLabel

The default value of include_test_labels is false because of the following reasons. Consider the case that models do not handle the zero-shot labels. If we include these labels for evaluation, the ranking measures are not affected. However, the classification measures such as Macro-F1 or Micro-F1 become different. In particular, because the F-measure of zero-shot labels is zero, the resulting Macro-F1, which is the unweighted mean of label F-measures, can be significantly different. In this situation, the zero-shot labels should not be included in the evaluation. Popular software such as scikit-learn does not include test labels for evaluation, and we hope to be consistent with them.

2.2 When to Set the Option to be True?

Sometimes, we may need to include zero-shot labels for evaluation. For example, if a paper experiments with approaches to handle zero-shot labels and report some classification measures, then to compare their results, the option include_test_labels should be true. For example, Chalkidis et al. (2019) propose the dataset EURLEX57K, which contains zero-shot labels. In their experimental results, they include zero-shot labels for evaluation and report Micro-F1. To compare with their results, we must set include test labels to be true.

2.3 Remarks on Labels in Training/Validation Sets

No matter which value of include_test_labels, we always consider the combined label set of training and validation sets. The reason is that training and validation instances are considered as all available data, so those labels that only appear in validation sets should not be regarded as zero-shot labels. Further, it is possible that we conduct training/validation splits several times. For example, we adopt the cross-validation strategy in our linear solvers. Therefore, it is better to consider the same label set across splits.

3 Linear Methods

Linear methods are the methods based on linear classifiers trained with bag-of-words (BOW) features. Specifically, let $\mathcal{D}$ be the set of documents and $\mathcal{T}$ be the set of all terms appearing in $\mathcal{D}$. Given a term

$t \in \mathcal{T}$ and a document $d \in \mathcal{D}$, the associated BOW feature is the l2-normalized TF-IDF generated by

$$\begin{aligned}
\textsf{normalized-tf-idf}(d,t) &= \frac{\textsf{tf-idf}(d,t)}{\sqrt{\sum_{s\in\mathcal{T}} \textsf{tf-idf}^2(d,s)}},\\
\textsf{tf-idf}(d,t) &= \textsf{tf}(d,t)\cdot \textsf{idf}(t),\\
\textsf{tf}(d,t) &= \text{number of times } t \text{ occurs in } d,\\
\textsf{idf}(t) &= \log\left(\frac{1+|\mathcal{D}|}{1+\textsf{df}(t)}\right)+1,\\
\textsf{df}(t) &= \text{number of documents conatining } t.
\end{aligned}$$

Consider a set of training instances $\{(\boldsymbol{x}_j, \boldsymbol{y}_j)\}_{j=1}^{D}$ where D is the number of instances, L is the number of labels, n is the number of features, $\boldsymbol{x}_j \in \mathbb{R}^n$ is BOW features, and $\boldsymbol{y}_j \in \{-1,1\}^L$ is a label vector such that

$$y_{jl} = \begin{cases} 1, & \text{if } \boldsymbol{x}_j \text{ is associated with the label } l,\\ -1, & \text{otherwise.}\end{cases}$$

Note that here we use $+1/-1$ instead of $+1/0$ in (1) to indicate relevant/irrelevant labels.

In LibMultiLabel, currently all linear methods except a tree-based setting aim to learn a $f : \mathbb{R}^n \to \mathbb{R}^L$ which is composed of L decision functions.

$$f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_L(\boldsymbol{x})).$$

In this section, we begin with introducing linear methods in the software and then discuss implementation details.

3.1 One-vs-rest (Binary Relevance)

The one-versus-rest setting, also known as binary relevance, trains a binary classification problem for each label on data with/without that label. That is, for the lth label, we solve the corresponding lth binary classification problem

$$\boldsymbol{w}_l = \arg\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{j=1}^{D} \xi(y_{jl}\boldsymbol{w}^T\boldsymbol{x}_j), \tag{4}$$

where C is a penalty parameter. For the loss function ξ, we support logistic regression and linear SVM through LIBLINEAR (Fan et al., 2008). After the training process ends, for any test instance $\boldsymbol{x}$, the decision function of label l is $f_l(\boldsymbol{x}) = \boldsymbol{w}_l^T\boldsymbol{x}$. Various ways can be applied on decision values to make predictions. The most used method is to use $f_l(\boldsymbol{x})$ as a binary classifier so that

$$\begin{cases} \boldsymbol{x} \text{ is predicted to have the label } l & \text{if } f_l(\boldsymbol{x}) > 0,\\ \text{otherwise} & \text{if } f_l(\boldsymbol{x}) \le 0.\end{cases} \tag{5}$$

Alternatively, in some applications, labels corresponding to the largest K values of $f_l(\boldsymbol{x}), \forall l$ are predicted to be associated with $\boldsymbol{x}$, where K is a number specified by users.

3.2 Thresholding

It is known that under the one-vs-rest setting, for some infrequent labels, the two-class problem (4) is highly imbalanced. Sometimes an instance is predicted to have no labels at all. Thresholding is a technique to address this issue and it is effective to optimize the Macro-F1 score (Lewis et al., 1996; Yang, 1999; Fan and Lin, 2007). Lin and Lin (2023) proposed the method that automatically decides a threshold Δ_l for label l through a cross-validation procedure so that the decision function becomes

$$f_l(\boldsymbol{x}) = \boldsymbol{w}_l^T \boldsymbol{x} + \Delta_l.$$

Therefore, this method is more expensive than one-vs-rest. See Section 4.3 and supplementary D of Lin and Lin (2023) for details of the thresholding method.

3.3 Cost-Sensitive

Another scheme to solve the class imbalance problem is cost-sensitive learning, which uses a higher loss on positive training instances. Parambath et al. (2014) give some theoretical support showing that the F1 score can be optimized through cost-sensitive learning. For the label l, they extend problem (4) of one-vs-rest to

$$\boldsymbol{w}_l = \arg\min_{\boldsymbol{w}} \ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\left(\frac{2-t}{t}\right)\sum_{j:y_{jl}=1} \xi(y_{jl}\boldsymbol{w}^T\boldsymbol{x}_j) + C\sum_{j:y_{jl}=-1} \xi(y_{jl}\boldsymbol{w}^T\boldsymbol{x}_j), \tag{6}$$

where $(2-t)/t$ is the cost of false negatives, and $t \in (0, 1]$. In LibMultiLabel, for each label, a pre-defined grid of (C, t) pairs are checked to find the one leading to the best validation F1 score. The best pair is then applied to the whole training set to get the final decision function of the corresponding label. Therefore, this method is more expensive than one-vs-rest.

3.4 Choice of Solvers in LIBLINEAR

The classification problems (4) and (6) are solved with LIBLINEAR (Fan et al., 2008). All two-class classification methods in LIBLINEAR are supported. We choose the coordinate descent method to solve the dual problem of L2-loss SVM as the default. The reason is that for large sparse data (e.g. documents), the results of solving the dual problem and solving the primal problem are similar, but solving the dual problem is faster than solving the primal problem. Depending on the requirements, the solver can be changed by specifying the option `liblinear_options`. See Fan et al. (2008) for details of each solver.

3.5 Run Time Analysis

In general, the run time of the optimization problems (4) and (6) depends on the dimensions (number of instances and features), data values and parameters. While we may not be able to calculate the run time without knowing the data set, a rough comparison of the run time of different linear methods can be by counting the optimization problems solved. The reason is that all optimization problems solved (including those in the cross-validation procedures) have comparable dimensions.

- One-vs-rest: Because one problem (4) is solved for each label, the total number of optimization problmes solved is L.

- Thresholding: In the cross-validation, there has 3 folds for each label, so the number of optimization problems solved is $3L$.
- Cost-sensitive: This method finds the optimal t value by a grid search on checking the cross-validation performance at each t value. Let $\#t$ be the number of t values searched and f be the number of folds. The number of optimization problems solved is $\#t \cdot fL$. For one-vs-rest and thresholding, by default the regularization parameter $C = 1$ is used across all optimization problems, but in (6), $C(2-t)/t$ is the regularization parameter on the training loss of positive data. It is known that the training time increases with a larger regularization parameter. With $t \in (0, 1]$, we have $C(2-t)/t > C$, so the run time of cost-sensitive may be longer than the count of optimization problems suggests.

3.6 Space Anaylsis

All linear methods require the weights $(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L)$ to be stored. We store weights as dense vectors so the space consumption is $O(nL)$. More specifically, each entry of weights is a float number, which needs 8 bytes to store. So the space consumption of weights is $8nL$ bytes.

3.7 The -B Option in LIBLINEAR

LIBLINEAR (Fan et al., 2008) has the option -B for adding a regularized bias term. Given a parameter value B and a bias term b, the weights $\boldsymbol{w}$ and features $\boldsymbol{x}$ are augmented with an additional dimension:

$$\boldsymbol{w}' = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix} \qquad \boldsymbol{x}' = \begin{bmatrix} \boldsymbol{x} \\ B \end{bmatrix}$$

Problem (4) is then modified as

$$\begin{aligned} \boldsymbol{w}'_l &= \arg\min_{\boldsymbol{w}'} \frac{1}{2}\boldsymbol{w}'^T\boldsymbol{w}' + C\sum_{j=1}^{D} \xi(y_{jl}\boldsymbol{w}'^T\boldsymbol{x}'_j) \\ &= \arg\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \frac{1}{2}b^2 + C\sum_{j=1}^{D} \xi(y_{jl} \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}^T \begin{bmatrix} \boldsymbol{x}_j \\ B \end{bmatrix}), \end{aligned}$$

We note the following implementation details. Since the prediction of LibMultiLabel is not performed through LIBLINEAR, it is convenient to acquire b in both training and prediction processes. To that end, if users specify the -B option, we augment the value B as an additional feature of the data in LibMultiLabel and strip -B from the options before passing training data to LIBLINEAR.

By default, we use -B 1 because empirically this seems to be useful.

3.8 Cross-validation Data Splits

In cost-sensitive, a cross-validation procedure is performed for each value of $C(2-t)/t$. The data splits, i.e. the subsets of data chosen for training or validation, in each cross-validation procedure may be the same or different for each $C(2-t)/t$ value. The supplementary of Lin et al. (2022) showed that the difference of having the same or different data splits is insignificant. We chose to have the same data splits for each $C(2-t)/t$ value.

References

I. Chalkidis, E. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos. Large-scale multi-label text classification on EU legislation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 6314–6322, 2019. doi: 10.18653/v1/P19-1636.

R.-E. Fan and C.-J. Lin. A study on threshold selection for multi-label classification. Technical report, Department of Computer Science, National Taiwan University, 2007.

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*, 9:1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 298–306, 1996.

L.-C. Lin, C.-H. Liu, C.-M. Chen, K.-C. Hsu, I.-F. Wu, M.-F. Tsai, and C.-J. Lin. On the use of unrealistic predictions in hundreds of papers evaluating graph representations. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2022. URL `https://www.csie.ntu.edu.tw/~cjlin/papers/multilabel-embedding/multilabel_embedding.pdf`.

Y.-J. Lin and C.-J. Lin. On the thresholding strategy for infrequent labels in multi-label classification. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023. doi: 10.1145/3583780.3614996. URL `https://www.csie.ntu.edu.tw/~cjlin/papers/thresholding/smooth_acm.pdf`.

J. Opitz and S. Burst. Macro F1 and Macro F1, 2021. arXiv preprint arXiv:1911.03347.

S. A. P. Parambath, N. Usunier, and Y. Grandvalet. Optimizing F-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

J. M. Tague. Information retrieval experiment. In K. S. Jones, editor, *The pragmatics of information retrieval experimentation*, chapter 5, pages 59–102. Butterworths, London, 1981.

Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1/2): 69–90, 1999.