On the Use of Unrealistic Predictions in Hundreds of Papers Evaluating Graph Representations

Li-Chung Lin,[1] Cheng-Hung Liu,[1] Chih-Ming Chen,[2] Kai-Chin Hsu,[3]
I-Feng Wu,[4] Ming-Feng Tsai,[2] Chih-Jen Lin[1]
[1]National Taiwan University
[2]National Chengchi University
[3]University of Southern California
[4]ASUS Intelligent Cloud Services
r08922141@ntu.edu.tw, ericliu8168@gmail.com, 104761501@nccu.edu.tw, kaichinh@usc.edu,
ifengwu1518@gmail.com, mftsai@nccu.edu.tw, cjlin@csie.ntu.edu.tw

Abstract

Prediction using the ground truth sounds like an oxymoron in machine learning. However, such an unrealistic setting was used in hundreds, if not thousands of papers in the area of finding graph representations. To evaluate the multi-label problem of node classification by using the obtained representations, many works assume that the number of labels of each test instance is known in the prediction stage. In practice such ground truth information is rarely available, but we point out that such an inappropriate setting is now ubiquitous in this research area. We detailedly investigate why the situation occurs. Our analysis indicates that with unrealistic information, the performance is likely over-estimated. To see why suitable predictions were not used, we identify difficulties in applying some multi-label techniques. For the use in future studies, we propose simple and effective settings without using practically unknown information. Finally, we take this chance to compare major graph-representation learning methods on multi-label node classification.

1 Introduction

Recently unsupervised representation learning over graphs has been an important research area. One of the primary goals is to find embedding vectors as feature representations of graph nodes. Many effective techniques (e.g., Perozzi, Al-Rfou, and Skiena 2014; Tang et al. 2015; Grover and Leskovec 2016) have been developed and widely applied. This research area is very active as can be seen from the tens of thousands of related papers.

The obtained embedding vectors can be used in many downstream tasks, an important one being node classification. Because each node may be associated with multiple labels, this application falls into the category of multi-label problems in machine learning. In this study, we point out that in many (if not most) papers using node classification to evaluate the quality of embedding vectors, an unrealistic setting was adopted for prediction and evaluation. Specifically, in the prediction stage, the number of labels of each test instance is assumed to be known. Then according to decision values, this number of top-ranked labels is considered to be associated with the instance. Because information on the number of labels is usually not available in practice, this setting violates the machine learning principle that ground-truth information should not be used in the prediction stage. Unfortunately, after surveying numerous papers, we find that this inappropriate setting is so ubiquitous that many started thinking it is a standard and valid one.

While the research community should move to use appropriate settings, some detailed investigation is needed first. In this work, we aim to do so by answering the following research questions.

- Knowing this unrealistic setting has been commonly used, how serious is the situation and why does it occur?

  To confirm the seriousness of the situation, we identify a long list of papers that have used the unrealistic predictions. Our analysis then indicates that with unrealistic information, the performance is likely over-estimated. Further, while the setting clearly cheats, it roughly works for some node classification problems that are close to a multi-class one with many single-labeled instances.
- What are suitable settings without using unknown information? Are there practical difficulties for researchers to apply them?

  After explaining that multi-label algorithms and/or tools may not be readily available, we suggest pragmatic solutions for future studies. Experimental comparisons with the unrealistic setting show that we can effectively optimize some commonly used metrics such as Macro-F1.
- Because of the use of unrealistic predictions, past comparisons on methods to generate embedding vectors may need to be re-examined. Can we give comparisons under appropriate multi-label predictions?

  By using suitable prediction settings, our results give new insights into comparing influential methods on representation learning.

This paper is organized as follows. Sections 2-3 address the first research question, while Sections 4 and 5 address the second and the third research questions, respectively. Finally, Section 6 concludes this work. Programs and supplementary materials are available at https://www.csie.ntu.edu.tw/~cjlin/papers/multilabel-embedding/

2 Unrealistic Predictions in Past Works

After finding the embedding vectors, past studies on representation learning experiment with various applications. An important downstream task is node classification, which is often a multi-label classification problem.

In machine learning, multi-label classification is a well-developed area with many available training methods. The most used one may be the simple one-versus-rest setting, also known as binary relevance. This method has been adopted by most works on representation learning. The main idea is to train a binary classification problem for each label on data with/without that label. The binary optimization problem on label-feature pairs $(y_i, \boldsymbol{x}_i)$, where $y_i = \pm 1$ and $i = 1, \ ..., \#$ training instances, takes the following form.

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum\nolimits_i \xi(y_i \boldsymbol{w}^T \boldsymbol{x}_i), \tag{1}$$

where $\xi(\cdot)$ is the loss function, $\boldsymbol{w}^T\boldsymbol{w}/2$ is the regularization, and C is the regularization parameter.[1] Now embedding vectors $\boldsymbol{x}_i, \ \forall i$ are available and fixed throughout all binary problems. Then for each label, the construction of problem (1) is simply by assigning

$$y_i = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \text{ is associated with the label,} \\ -1, & \text{otherwise.} \end{cases}$$

Because representation learning aims to get a low-dimensional but informative vector, a linear classifier is often sufficient in the downstream task. For the loss function, logistic regression is usually considered, and many use the software LIBLINEAR (Fan et al. 2008) to solve (1).

To check the performance after the training process, we find that hundreds, if not thousands of papers[2] in this area used the following procedure.

- Prediction stage: for each test instance, assume

  the number of labels of this instance is known.

  Predict this number of labels by selecting those with the largest decision values from all binary models.
- Evaluation stage: many works report Micro-F1 and Macro-F1.

Clearly, this setting violates the principle that in the prediction stage, ground-truth information should not be used. The reason is obvious that in the practical model deployment, such information is rarely available.

In particular, some influential works with thousands of citations (e.g., Perozzi, Al-Rfou, and Skiena 2014; Tang et al. 2015) employed such unrealistic predictions, and many subsequent works followed. The practice is now ubiquitous and here we quote the descriptions in some papers.

- Chanpuriya and Musco (2020): "As in Perozzi, Al-Rfou, and Skiena (2014) and Qiu et al. (2018), we assume that the number of labels for each test example is given."
- Schlötterer et al. (2019): "we first obtain the number of actual labels to predict for each sample from the test set. ... This is a common choice in the evaluation setup of the reproduced methods."

Interestingly, we find that such unrealistic predictions were used long before the many recent studies on representation learning. An example is as follows.

- Tang and Liu (2009): "we assume the number of labels of unobserved nodes is already known and check the match of the top-ranking labels with the truth."[3]

Our discussion shows how an inappropriate setting can eventually propagate to an entire research area. Some works did express concerns about the setting. For example,

- Faerman et al. (2018): "Precisely, this method uses the actual number of labels k each test instance has. ... In real world applications, it is fairly uncommon that users have such knowledge in advance."
- Liu and Kim (2018): "we note that at the prediction stage previous approaches often employs information that is typically unknown. Precisely, they use the actual number of labels m each testing node has (Perozzi, Al-Rfou, and Skiena 2014; Qiu et al. 2018). ... However, in real-world situations it is fairly uncommon to have such prior knowledge of m."

To be realistic, Faerman et al. (2018); Liu and Kim (2018) predict labels by checking the sign of decision values.[4] We name this method and give its details as follows.

- one-vs-rest-basic: for a test instance $\boldsymbol{x}$,

$$\boldsymbol{w}^T\boldsymbol{x} \begin{cases} \geq 0 \\ < 0 \end{cases} \Rightarrow \begin{cases} \boldsymbol{x} \text{ predicted to have the label,} \\ \text{otherwise.} \end{cases} \tag{2}$$

Their resulting Macro-F1 and Micro-F1 are much lower than works that have used unknown information.

If so many works consider an unrealistic setting for predictions, they probably have reasons for doing so. Some papers explain the difficulties that lead to their assumption of knowing the number of labels.

- Li, Zhu, and Zhang (2016): "As the datasets are not only multi-class but also multi-label, we usually need a thresholding method to test the results. But literature gives a negative opinion of arbitrarily choosing thresholding methods because of the considerably different performances. To avoid this, we assume that the number of the labels is already known in all the test processes."
- Qiu et al. (2018): "To avoid the thresholding effect (Tang, Rajan, and Narayanan 2009), we assume that the number of labels for test data is given (Perozzi, Al-Rfou, and Skiena 2014; Tang, Rajan, and Narayanan 2009)."

[1] In some situations a bias term is considered, so $\boldsymbol{w}^T\boldsymbol{x}_i$ is replaced by $\boldsymbol{w}^T\boldsymbol{x}_i + b$.

[2] See a long list compiled in supplementary materials.

[3] Tang and Liu (2009) stated that "Such a scheme has been adopted for other multi-label evaluation works (Liu, Jin, and Yang 2006)". However, we found no evidence that Liu, Jin, and Yang (2006) assumed that the number of labels is known.

[4] More precisely, if logistic regression is used, they check if the probability is greater than 0.5 or not. This is the same as checking the decision value in (2).

To see what is meant by the thresholding effect and the difficulties it imposes, we give a simple illustration. For a data set BlogCatalog (details in Section 5.1), we apply the one-vs-rest training on embedding vectors generated by the method DeepWalk (Perozzi, Al-Rfou, and Skiena 2014). Then the unrealistic prediction of knowing the number of labels in each test instance is performed. Results (Micro-F1 = 0.41, Macro-F1 = 0.27) are similar to those reported in some past works.

In contrast, when using the one-vs-rest-basic setting as in Faerman et al. (2018); Liu and Kim (2018), results are very poor (Micro-F1 = 0.33 and Macro-F1 = 0.19). We see that many instances are predicted to have no label at all. A probable cause of this situation is the class imbalance of each binary classification problem. That is, in problem (1), few training instances have $y_i = 1$, and so the decision function tends to predict everything as negative. Many multi-label techniques are available to address such difficulties, and an important one is the thresholding method (e.g., Yang 2001; Fan and Lin 2007). Via a constant Δ to adjust the decision value, in (2) we can replace

$$\boldsymbol{w}^T\boldsymbol{x} \quad \text{with} \quad \boldsymbol{w}^T\boldsymbol{x} + \Delta. \tag{3}$$

A positive Δ can make the binary problem produce more positive predictions. Usually Δ is decided by a cross-validation (CV) procedure. Because each label needs one Δ, the overall procedure is more complicated than one-vs-rest-basic. Moreover, the training time is significantly longer. Therefore, past works may not consider such a technique.

3 Analysis of the Unrealistic Predictions

We analyze the effect of using the unrealistic predictions. To facilitate the discussion, in this section we consider

$$i : \text{index of test instances, and } j : \text{index of labels.}$$

We further assume that for test instance i,

$$\begin{aligned} K_i &: \text{true number of labels,} \\ \hat{K}_i &: \text{predicted number of labels.} \end{aligned} \tag{4}$$

In multi-label classification, two types of evaluation metrics are commonly used (Wu and Zhou 2017).

- Ranking measures: examples include precision@K, nDCG@K, ranking loss, etc. For each test instance, all we need to predict is a ranked list of labels.
- Classification measures: examples include Hamming loss, Micro-F1, Macro-F1, Instance-F1, etc. For each test instance, several labels are chosen as the predictions.

Among these metrics, Macro-F1 and Micro-F1 are used in most works on representation learning. We first define Macro-F1, which is the average of F1 over labels:

$$\text{Macro-F1} = \text{Label-F1} = \frac{\sum \text{F1 of label } j}{\#\text{labels}}, \tag{5}$$

where

$$\text{F1 of label } j = \frac{2 \times \text{TP}_j}{\text{TP}_j + \text{FP}_j + \text{TP}_j + \text{FN}_j}.$$

Note that TP_j, FP_j, and FN_j are respectively the number of true positives, false positives and false negatives on the prediction of a given label j. Then Micro-F1 is the F1 by considering all instances (or all labels) together:

$$\text{Micro-F1} = \frac{2 \times \text{TP sum}}{\text{TP sum} + \text{FP sum} + \text{TP sum} + \text{FN sum}}, \tag{6}$$

where "sum" indicates the accumulation of prediction results over all binary problems. Next we prove an upper bound of Micro-F1.

Theorem 1. *With the definition in* (4)*, we have*

$$\textit{Micro-F1} \le \frac{2 \times \sum_{i=1}^{l} \min\left(\hat{K}_i, K_i\right)}{\sum_{i=1}^{l}\left(K_i + \hat{K}_i\right)} \le 1, \tag{7}$$

where l is the number of test instances. Moreover, when $\hat{K}_i = K_i$, the bound in (7) *achieves the maximum (i.e., 1).*

The proof is in supplementary materials. For the upper bound of Micro-F1 proved in Theorem 1, we see that knowing K_i "pushes" the bound to its maximum. If a larger upper bound leads to a larger Micro-F1, then Theorem 1 indicates the advantage of knowing K_i.

While Theorem 1 proves only an upper bound, by some assumption on the decision values,[5] we can exactly obtain Micro-F1 for analysis. The following theorem shows that if all binary models are good enough, the upper bound in (7) is attained. Further, if K_i is known, we achieve the best possible Micro-F1 $= 1$.

Theorem 2. *Assume for each test instance i, decision values are properly ranked so that*

$$\begin{aligned} &\textit{decision values of its } K_i \textit{ labels} \\ &> \textit{decision values of other labels.} \end{aligned} \tag{8}$$

Under specified $\hat{K}_i$, $\forall$ i, the best Micro-F1 is obtained by predicting labels with the largest decision values. Moreover, the resulting Micro-F1 is the same as the upper bound in (7)*. That is,*

$$\textit{Micro-F1} = \frac{2 \times \sum_{i=1}^{l} \min\left(\hat{K}_i, K_i\right)}{\sum_{i=1}^{l}\left(K_i + \hat{K}_i\right)}. \tag{9}$$

If $\hat{K}_i = K_i$, the best possible Micro-F1 $= 1$ is attained.

The proof is in supplementary materials. Theorem 2 indicates that even if the classifier can output properly ranked decision values, without the true number of labels K_i, optimal Micro-F1 still may not be obtained. Therefore, using K_i gives predictions an inappropriate advantage and may cause the performance to be over-estimated as a result.

Next, we investigate why unrealistic predictions were commonly considered and point out several possible reasons in the current and subsequent sections. The first one is the relation to multi-class problems. Some popular node classification benchmarks are close to multi-class problems because

[5] Wu and Zhou (2017) also assumed (8) for analyzing Micro-F1. However, their results are not suited for our use here because of various reasons. In particular, they made a strong assumption that Micro-F1 is equal to Instance-F1.

many of their instances are single-labeled with $K_i = 1$. See the data statistics in Table 1. For multi-class problems, the number of labels (i.e., one) for each instance is known. Thus in prediction, we simply find the most probable label. In this situation, Theorem 3 shows that the accuracy commonly used for evaluating multi-class problems is the same as Micro-F1. The proof is in supplementary materials.

Theorem 3. *For multi-class problems,*

$$\textit{accuracy} = \textit{Micro-F1}.$$

Therefore, using Micro-F1 with prior knowledge on the number of labels is entirely valid for multi-class classification. Some past studies may conveniently but erroneously extend the setting to multi-label problems.

Based on the findings so far, in Section 3.1 we explain that the unrealistic prediction roughly works if a multi-label problem contains mostly single-labeled instances.

3.1 Predicting at Least One Label per Instance

The discussion in Theorem 3 leads to an interesting issue on whether in multi-label classification, at least one label should be predicted for each instance. In contrast to multi-class classification, for multi-label scenarios, we may predict that an instance is associated with no label. For the sample experiment on one-vs-rest-basic in Section 2, we mentioned that this "no label" situation occurs on many test instances and results in poor performance. A possible remedy by tweaking the simple one-vs-rest-basic method is:

- one-vs-rest-no-empty: The method is the same as one-vs-rest-basic, except that for instances predicted to have no label, we predict the label with the highest decision value.

For the example considered in Section 2, this new setting greatly improves the result to 0.39 Micro-F1 and 0.24 Macro-F1. If we agree that each instance is associated with at least a label (i.e., $K_i \geq 1$), then the method one-vs-rest-no-empty does not take any unknown information in the prediction stage. In this regard, the method of unrealistic predictions is probably usable for single-labeled instances. However, it is definitely inappropriate for multi-labeled instances. For some benchmark sets in Section 5, the majority of instances are multi-labeled. Thus there is a need to develop effective prediction methods without using unrealistic information. This subject will be discussed in Section 4.

4 Appropriate Methods for Training and Prediction

Multi-label classification is a well-developed area, so naturally we may criticize researchers in representation learning for not applying suitable techniques. However, this criticism may not be entirely fair: what if algorithms and/or tools on the multi-label side are not quite ready for them? In this section, we discuss the difficulties faced by researchers on representation learning and explain why simple and effective settings are hard to obtain.

The first challenge faced by those handling multi-label problems is that they must choose from a myriad of methods according to the properties of their applications. Typically two considerations are

- number of labels, and
- evaluation metrics.

For example, some problems have extremely many labels, and the corresponding research area is called "eXtreme Multi-label Learning (XML);" see the website (Bhatia et al. 2016) containing many such sets. For this type of problems it is impossible to train and store the many binary models used by the one-vs-rest setting, so advanced methods that organize labels into a tree structure are needed (e.g., You et al. 2019; Khandagale, Xiao, and Babbar 2020; Chang et al. 2021). With a huge number of tail labels (i.e., labels that rarely occur), the resulting Macro-F1, which is the average F1 over all labels, is often too low to be used. In practice, a short ranked list is considered in the prediction stage, so precision@K or nDCG@K commonly serve as the evaluation metrics.

Nevertheless, the focus now is on node classification problems in past studies on representation learning. The number of labels is relatively small, and some even contain many single-labeled instances. From the predominant use of Micro-F1 and Macro-F1 in past works it seems that a subset of labels instead of a ranked list is needed for node classification. Therefore, our considerations are narrowed to

- methods that are designed for problems without too many labels, and
- methods that can predict a subset of labels (instead of just ranks) and achieve a high classification measure such as Micro-F1, Macro-F1, and Instance-F1.

In addition to one-vs-rest, other methods are applicable for our scenario (e.g., Tai and Lin 2012; Read et al. 2011; Read, Pfahringer, and Holmes 2008; Tsoumakas and Vlahavas 2007). Because one-vs-rest does not consider label correlation, this aspect is the focus of some methods. For simplicity we stick with the one-vs-rest setting here and prioritize achieving good Macro-F1. Macro-F1 in (5) is the average of F1 results over labels, so under the one-vs-rest framework, all we need is to design a method that can give satisfactory F1 on each single label. In contrast, optimizing Micro-F1 is more difficult because it couples all labels and all instances together; see the definition in (6).[6] Therefore, we mainly focus on techniques to optimize Macro-F1 in the following sections.

4.1 Extending One-vs-rest to Incorporate Parameter Selection

If we examine the one-vs-rest-basic method more closely, it is easy to see that a crucial process is missing – parameter selection of the regularization parameter C. While the importance of parameter selection is well recognized, this step is easily forgotten in many places (Liu et al. 2021). For example, out of the works that criticized the unrealistic setting

[6] See, for example, "... is the most challenging measure, since it does not decompose over instances nor over labels." in Pillai, Fumera, and Roli (2017)

(see Section 2), Faerman et al. (2018) used a fixed regularization parameter for comparing with past works, but Liu and Kim (2018) conducted cross-validation in their one-vs-rest implementation. Therefore, a more appropriate baseline should be the following extension of one-vs-rest-basic:

- one-vs-rest-basic-C: For each binary problem, cross-validation is performed on the training data by checking a grid of C values. The one yielding the best F1 score is chosen to train the binary model of the label for future prediction.

CV is so standard in machine learning that the above procedure seems to be extremely simple. Surprisingly, several issues may hamper its wide use.

- We learned in Section 2 that some binary problems may not predict any positives in the prediction process. Thus cross-validation F1 may be zero under all C values. In this situation, which C should we choose?
- To improve robustness, should the same splits of data for CV be used throughout all C values?
- If C is slightly changed from one value to another, solutions of the two binary optimization problems may be similar. Thus a warm-start implementation of using the solution of one problem as the initialization for training the other can effectively reduce the running time. However, the implementation, together with CV, can be complicated.

The discussion above shows that even for a setting as simple as one-vs-rest-basic-C, off-the-shelf implementations may not be directly available to users.[7]

4.2 Thresholding Techniques

While the basic concept of thresholding has been discussed in Section 2, the actual procedure is more complicated and several variants exist (Yang 2001). From early works such as Lewis et al. (1996); Yang (1999), a natural idea is to use decision values of validation data to decide Δ in (3). For each label, the procedure is as follows.

- For each CV fold, sort validation decision values.

  Sequentially assign Δ as the midpoint of two adjacent decision values and select the one achieving the best F1 as the threshold of the current fold.
- Solve a binary problem (1) using all training data. The average of Δ values over all folds is then used to adjust the decision function.

However, Yang (2001) showed that this setting easily overfits data if the binary problem is unbalanced. Consequently, the same author proposed the fbr heuristic to reduce the overfitting problem. Specifically, if the F1 of a label is smaller than a pre-defined fbr value, then the threshold is set to the largest decision value of the validation data. This method requires a complicated two-level CV procedure. The outer level uses CV to check that among a list of given fbr candidates, which one leads to the best F1. The inner CV checks if the validation F1 is better than the given fbr.

The above fbr heuristic was further studied in an influential paper (Lewis et al. 2004). An implementation from Fan and Lin (2007) as a LIBLINEAR extension has long been publicly available. Interestingly, our survey seems to indicate that no one in the field of representation learning ever tried it. One reason may be that the procedure is complicated. If we also select the parameter C, then a cumbersome outer-level CV to sweep some (C, fbr) pairs is needed. Furthermore, it is difficult to use the same data split, especially in the inner CV. Another reason may be that as a heuristic, people are not confident about the method. For example, Tang and Liu (2009) stated that because "thresholding can affect the final prediction performance drastically (Fan and Lin 2007; Tang, Rajan, and Narayanan 2009)," they decided that "For evaluation purpose, we assume the number of labels of unobserved nodes is already known."

4.3 Cost-sensitive Learning

We learned in Section 2 that because of class imbalance, one-vs-rest-basic suffers from the issue of predicting very few positives. While one remedy is the thresholding technique to adjust the decision function, another possibility is to conduct cost-sensitive learning. Namely, by using a higher loss on positive training instances (usually through a larger regularization parameter), the resulting model may predict more positives. For example, Parambath, Usunier, and Grandvalet (2014) give some theoretical support showing that the F1 score can be optimized through cost-sensitive learning. They extend the optimization problem (1) to

$$\min_{\boldsymbol{w}} \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C^+ \sum_{i:y_i=1} \xi(y_i \boldsymbol{w}^T \boldsymbol{x}_i) + C^- \sum_{i:y_i=-1} \xi(y_i \boldsymbol{w}^T \boldsymbol{x}_i),$$

where

$$C^+ = C(2-t),\ C^- = Ct, \text{ and } t \in [0,1].$$

Then we can check cross-validation F1 on a grid of (C, t) pairs. The best pair is then applied to the whole training set to get the final decision function.

An advantage over the thresholding method (fbr heuristic) is that only a one-level CV is needed. However, if many (C, t) pairs are checked, the running time can be long. In Section 5.2 we discuss two implementations for this approach.

5 Experiments

In this section we experiment with training/prediction methods discussed in Sections 2-4 on popular node classification benchmarks. Embedding vectors are generated by some well-known methods and their quality is assessed.

5.1 Experimental Settings

We consider the following popular node classification problems:

BlogCatalog, Flickr, YouTube, PPI.

[7] LIBLINEAR supports warm-start and same CV folds for parameter selection after their work in Chu et al. (2015). However, the purpose is to optimize CV accuracy. Our understanding is that an extension to check F1 scores is available only very recently.

Data	#instances		#labels	avg. #labels
	single-labeled	multi-labeled		per instance
BlogCatalog	7,460	2,852	39	1.40
Flickr	62,521	17,992	195	1.34
YouTube	22,374	9,329	46	1.60
PPI	85	54,873	121	38.26

Table 1: Data statistics.

From data statistics in Table 1, some have many single-labeled instances, but some have very few. We generate embedding vectors by the following influential works.

- DeepWalk (Perozzi, Al-Rfou, and Skiena 2014).
- Node2vec (Grover and Leskovec 2016).
- LINE (Tang et al. 2015).

Since we consider representation learning independent of the downstream task, the embedding-vector generation is unsupervised. As such, deciding the parameters for each method can be tricky. We reviewed many past works and selected the most used values.

In past studies, Node2vec often had two of its parameters p, q selected based on the results of the downstream task. This procedure is in effect a form of supervised learning. Therefore, in our experiments, the parameters p, q are fixed to the same values for all data sets.

For training each binary problem, logistic regression is solved by the software LIBLINEAR (Fan et al. 2008). We follow many existing works to randomly split each set to 80% for training and 20% for testing. This process is repeated five times and the average score is presented. The same training/testing split is used across the different graph representations. More details on experimental settings are given in the supplementary materials.

5.2 Multi-label Training and Prediction Methods for Comparisons

We consider the following methods. Unless specified, for binary problems (1), we mimic many past works to set $C = 1$.

- unrealistic: After the one-vs-rest training, the unrealistic prediction of knowing the number of labels is applied.
- one-vs-rest-basic: After the one-vs-rest training, each binary classifier predicts labels that have positive decision values.
- one-vs-rest-basic-C: The method, described in Section 4.1, selects the parameter C by cross-validation. We use a LIBLINEAR parameter-selection functionality that checks dozens of automatically selected C values. It applies a warm-start technique to save the running time. An issue mentioned in Section 4.1 is that CV F1=0 for every C may occur. We checked a few ways to choose C in this situation, but find results do not differ much.
- one-vs-rest-no-empty: This method slightly extends one-vs-rest-basic so that if all decision values of a test instance are negative, then we predict the label with the highest decision value; see Section 3.1.
- thresholding: The method was described in Section 4.2.

For the approach in Section 4.3 we consider two variants.

- cost-sensitive: A dense grid of (C, t) is used. The range of t is $\{0.1, 0.2, \dots, 1\}$. For each t, we follow one-vs-rest-basic-C to use a LIBLINEAR functionality that checks dozens of automatically selected C values. In this variant, we do not ensure that CV folds are the same across different t.
- cost-sensitive-simple: We check fewer parameter settings by considering $t = \{1/7, 2/7, \dots, 1\}$ and $C = \{0.01/t, 0.1/t, 1/t, 10/t, 100/t\}$. We ensure the same data split is applied on the CV for every pair. The implementation is relatively simple if all parameter pairs are independently trained without time-saving techniques such as warm-start.

Similar to one-vs-rest-basic, for thresholding or cost-sensitive approaches, an instance may be predicted to have no labels. Therefore, we check the following extension.

- cost-sensitive-no-empty: This method extends cost-sensitive by the same way from one-vs-rest-basic to one-vs-rest-no-empty.

5.3 Results and Analysis

In Table 2 we compare the unrealistic method and representative methods in Section 4. Other variants are investigated in Table 3 later. Due to the space limit, we omit the YouTube data set, though results follow similar trends. Observations from Table 2 are as follows.

- As expected, unrealistic is the best in nearly all situations. It significantly outperforms others on Micro-F1, a situation confirming not only the analysis in Theorem 3 but also that unrealistic may over-estimate performance.
- In Section 2 we showed an example that one-vs-rest-basic performs poorly because of the thresholding issue. Even with the parameter selection, one-vs-rest-basic-C still suffers from the same issue and performs the worst.
- Both thresholding and cost-sensitive effectively optimize Macro-F1 and achieve similar results to unrealistic. Despite Micro-F1 not being the optimized metric, the improvement over one-vs-rest-basic-C is still significant.

In Table 3 we study the variations of one-vs-rest-basic and cost-sensitive. We only present the results of embedding vectors generated by DeepWalk, while complete results with similar trends are in supplementary materials. Some observations from Table 3 are as follows.

- Even with parameter selection, one-vs-rest-basic-C is only marginally better than one-vs-rest-basic. This result is possible because for binary logistic regression, it is proved that after C is sufficiently large, the decision function is about the same (Theorem 3 in Chu et al. 2015). The result shows that conducting parameter selection is not enough to overcome the thresholding issue.
- Following the analysis in Section 3.1, one-vs-rest-no-empty significantly improves upon one-vs-rest-basic for problems that have many single-labeled instances. However, it has no visible effect on the set PPI, in which most instances are multi-labeled.

Training and prediction methods	BlogCatalog			Flickr			PPI		
	DeepWalk	Node2vec	LINE	DeepWalk	Node2vec	LINE	DeepWalk	Node2vec	LINE
Macro-F1 (avg. of five; std. in supplementary)									
unrealistic	0.276	0.294	0.239	0.304	0.306	0.258	0.483	0.442	0.504
one-vs-rest-basic-C	0.208	0.220	0.195	0.209	0.208	0.188	0.183	0.150	0.243
thresholding	0.269	0.283	0.221	**0.299**	**0.302**	0.264	**0.482**	0.457	**0.498**
cost-sensitive	**0.270**	**0.283**	**0.250**	0.297	0.301	**0.279**	0.482	**0.461**	0.495
Micro-F1 (avg. of five; std. in supplementary)									
unrealistic	0.417	0.426	0.406	0.416	0.420	0.409	0.641	0.626	0.647
one-vs-rest-basic-C	0.344	0.355	0.335	0.291	0.296	0.289	0.458	0.441	0.489
thresholding	**0.390**	**0.396**	**0.353**	**0.370**	**0.376**	**0.364**	**0.535**	0.482	**0.553**
cost-sensitive	0.366	0.371	0.341	0.352	0.358	0.354	0.533	**0.495**	0.548

Table 2: Results of representative training/prediction methods applied to various embedding vectors. Each value is the average of five 80/20 training/testing splits. The score of the best training/prediction method (excluding unrealistic) is bold-faced.

Training and prediction methods on DeepWalk vectors	BlogCatalog		Flickr		YouTube		PPI	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
one-vs-rest-basic	0.190	0.334	0.195	0.283	0.213	0.287	0.181	0.449
one-vs-rest-basic-C	0.208	0.344	0.209	0.291	0.217	0.290	0.183	0.458
one-vs-rest-no-empty	0.241	**0.390**	0.256	**0.377**	0.263	**0.382**	0.181	0.449
cost-sensitive	**0.270**	0.366	0.297	0.352	**0.360**	0.374	**0.482**	**0.533**
cost-sensitive-no-empty	0.268	0.351	**0.298**	0.343	0.359	0.372	**0.482**	**0.533**
cost-sensitive-simple	0.266	0.353	0.297	0.358	0.357	0.372	0.481	0.529

Table 3: Ablation study on variations of one-vs-rest-basic and cost-sensitive applied to embedding vectors generated by DeepWalk. Each value is the average of five 80/20 training/testing splits. The best training/prediction method is bold-faced.

- However, cost-sensitive-no-empty shows no such improvement over cost-sensitive because cost-sensitive mitigates the issue of predicting no labels for a large portion of instances. Further, for the remaining instances with no predicted labels, the label with the highest decision value may be an incorrect one, resulting in worse Micro-F1 in some cases. This experiment shows the importance to have techniques that allow empty predictions.
- cost-sensitive-simple is generally competitive with cost-sensitive and thresholding.

An issue raised in Section 4 is whether the same split of data (i.e., CV folds) should be used in the multiple CV procedures ran by, for example, cost-sensitive-simple. We have conducted some analysis, but leave details in supplementary materials due to the space limitation.

Regarding methods for representation learning, we have the following observations.

- Our results of the unrealistic method are close to those in the recent comparative study (Khosla, Setty, and Anand 2021). This outcome supports the validity of our experiments.
- Among the three methods to generate representations, there is no clear winner, indicating that the selection may be application dependent. DeepWalk and Node2vec are closer to each other because they are both based on random walks. In contrast, LINE is based on edge modeling.
- DeepWalk is a special case of Node2vec under some parameter values, though here Node2vec is generated by other commonly suggested values. Because DeepWalk is generally competitive and does not require the selection of some Node2vec's parameters, DeepWalk may be a better practical choice.
- The relative difference between the three representation learning methods differs from what unrealistic suggests. Even though in our comparisons such effects are not large enough to change their relative ranking, an unfair comparison diminishes the utility of benchmark results.

6 Conclusions

We summarize the results on training/prediction methods. The two methods thresholding and cost-sensitive are effective and can be applied in future studies. They are robust without the concerns mentioned in some papers. Further, if an easy implementation is favored, then the simple yet competitive cost-sensitive-simple can be a pragmatic choice. The implementations are available in an easy-to-use package

https://github.com/ASUS-AICS/LibMultiLabel

Thus, researchers in the area of representation learning can easily apply appropriate prediction settings.

In the well-developed world of machine learning, it may be hard to believe that unrealistic predictions were used in almost an entire research area. However, it is not the time to blame anyone. Instead, the challenge is to ensure that appropriate settings are used in the future. In this work, we analyze how and why unrealistic predictions were used in the past. We then discuss suitable replacements. Through our investigation hopefully unrealistic predictions will no longer be used.

7 Acknowledgments

This work was supported by MOST of Taiwan grant 110-2221-E-002-115-MY3 and ASUS Intelligent Cloud Services.

References

Bhatia, K.; Dahiya, K.; Jain, H.; Kar, P.; Mittal, A.; Prabhu, Y.; and Varma, M. 2016. The extreme classification repository: Multi-label datasets and code.

Chang, W.-C.; Jiang, D.; Yu, H.-F.; Teo, C.-H.; Zhang, J.; Zhong, K.; Kolluri, K.; Hu, Q.; Shandilya, N.; Ievgrafov, V.; Singh, J.; and Dhillon, I. S. 2021. Extreme Multi-label Learning for Semantic Matching in Product Search. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Chanpuriya, S.; and Musco, C. 2020. InfiniteWalk: Deep Network Embeddings as Laplacian Embeddings with a Nonlinearity. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1325–1333.

Chu, B.-Y.; Ho, C.-H.; Tsai, C.-H.; Lin, C.-Y.; and Lin, C.-J. 2015. Warm Start for Parameter Selection of Linear Classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Faerman, E.; Borutta, F.; Fountoulakis, K.; and Mahoney, M. W. 2018. LASAGNE: Locality and Structure Aware Graph Node Embedding. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 246–253.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874.

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

Grover, A.; and Leskovec, J. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 855–864.

Khandagale, S.; Xiao, H.; and Babbar, R. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109: 2099–2119.

Khosla, M.; Setty, V.; and Anand, A. 2021. A Comparative Study for Unsupervised Network Representation Learning. *IEEE Transactions on Knowledge and Data Engineering*, 33(5): 1807–1818.

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

Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5: 361–397.

Li, J.; Zhu, J.; and Zhang, B. 2016. Discriminative Deep Random Walk for Network Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 1004–1013.

Liu, J.-J.; Yang, T.-H.; Chen, S.-A.; and Lin, C.-J. 2021. Parameter Selection: Why We Should Pay More Attention to It. In *Proceedings of the 59th Annual Meeting of the Association of Computational Linguistics (ACL)*. Short paper.

Liu, X.; and Kim, K.-S. 2018. A Comparative Study of Network Embedding Based on Matrix Factorization. In *International Conference on Data Mining and Big Data*, 89–101.

Liu, Y.; Jin, R.; and Yang, L. 2006. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, 421–426.

Parambath, S. A. P.; Usunier, N.; and Grandvalet, Y. 2014. Optimizing F-Measures by Cost-Sensitive Classification. In *Advances in Neural Information Processing Systems*, volume 27.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 701–710.

Pillai, I.; Fumera, G.; and Roli, F. 2017. Designing multi-label classifiers that maximize F measures: State of the art. *Pattern Recognition*, 61: 394–404.

Qiu, J.; Dong, Y.; Ma, H.; Li, J.; Wang, K.; and Tang, J. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and Node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, 459–467.

Read, J.; Pfahringer, B.; and Holmes, G. 2008. Multi-label Classification Using Ensembles of Pruned Sets. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 995–1000.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2011. Classifier chains for multi-label classification. *Machine learning*, 85: 333–359.

Schlötterer, J.; Wehking, M.; Rizi, F. S.; and Granitzer, M. 2019. Investigating Extensions to Random Walk Based Graph Embedding. In *Proceedings of IEEE International Conference on Cognitive Computing*, 81–89.

Tai, F.; and Lin, H.-T. 2012. Multilabel Classification with Principal Label Space Transformation. *Neural Computation*, 24: 2508–2542.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international Conference on World Wide Web (WWW)*, 1067–1077.

Tang, L.; and Liu, H. 2009. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and Knowledge Management (CIKM)*, 1107–1116.

Tang, L.; Rajan, S.; and Narayanan, V. K. 2009. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, 211–220.

Tsoumakas, G.; and Vlahavas, I. 2007. Random k-labelsets: An ensemble method for multilabel classification. In *European conference on machine learning*, 406–417.

Wu, X.-Z.; and Zhou, Z.-H. 2017. A Unified View of Multi-Label Performance Measures. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 3780–3788.

Yang, Y. 1999. An Evaluation of Statistical Approaches to Text Categorization. *Information Retrieval*, 1(1/2): 69–90.

Yang, Y. 2001. A Study on Thresholding Strategies for Text Categorization. In Croft, W. B.; Harper, D. J.; Kraft, D. H.; and Zobel, J., eds., *Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval*, 137–145. New Orleans, US: ACM Press, New York, US.

You, R.; Zhang, Z.; Wang, Z.; Dai, S.; Mamitsuka, H.; and Zhu, S. 2019. AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. In *Advances in Neural Information Processing Systems*, volume 32.