Label Propagation Using Amendable Clamping

Tatsurou Miyazaki and Yasunobu Sumikawa
Depart. of Information Sciences, Tokyo University of Science, Japan
tatsumiya05@gmail.com, ysumikawa@acm.org

ABSTRACT
Assigning several labels to digital data is becoming easier because we can perform it in a collaborative manner with Internet users. However, some suitable labels may be missed and may not be attached to the data leading to inaccuracies in classification. In this paper, we propose a novel graph-based multi-label classifier to support the multi-labeling task. The core process of our algorithm is to update label weights of labeled data from their top-k similar data in each label propagation step. We report that our algorithm is more stable for F-scores compared to the state-of-the-art ones even though the some correct labels are missed.

ACM Classification Keywords
I.5.2. Pattern Recognition: Design Methodology

Author Keywords
Label propagation; multi-label classification

INTRODUCTION
Defining categories and dividing digital data into these categories play key processes. For example, Wikipedia has numerous articles and categories. Wikipedia editors can not only edit them but can also add new articles and categories using the discussions on them. Thus, when some Wikipedia categories are updated, some articles can be assigned new categories. As another example, if we want to train classifiers on historical news articles or on heterogeneous dataset, e.g., the New York Times and Japan Times, we sometimes reassign their labels.

Thanks to the increasing number of Internet users, this task has becoming easier because many people can work on such tasks following the same policies used on sites such as Wikipedia. Nevertheless, this task is still quite challenging. For example, some Wikipedia articles report mass murder by bombing [1] [2]. Although there are two common Wikipedia categories, "Mass murder" and "Terrorist incidents", in these articles, another category "suicide bombing" is assigned to only the latter Wikipedia article. Moreover, these three categories are not assigned to another article [3]. Thus, even though these articles report the same topic, the attached categories are not the same.

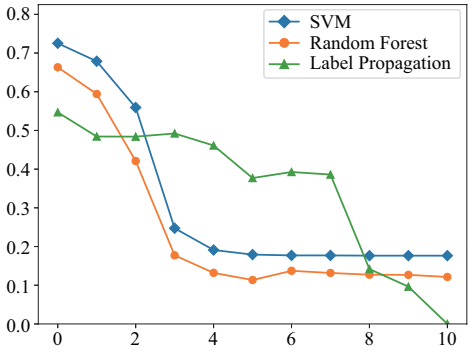

Figure 1. Effect of missing labels in multi-label classification. The x and y axes represent the number of missing labels and the micro-averaged F-scores of SVM, random forest and label propagation, respectively.

The missing of labels is a very serious issue to achieve good accuracy because almost all classifiers assume that labeled data prepared by people are correct. We show how the missing correct labels worsens the accuracies of SVM, random forest and label propagation on a multi-label dataset called the SIAM 2007 Text Mining Competition dataset [4], whose documents are assigned 3.4 labels on average in Fig. 1. We can see that if only one label is missed, all their micro-averaged F-scores worsen by about 5%. If more than two labels are missing, the scores can decrease by about 10%.

In this paper, we propose a novel graph-based algorithm for multi-label classification, label propagation using amendable clamping (LPAC), in order to decrease the impact of missing labels on accuracy. Our algorithm enhances *cluster assumption* [7], which means that similar nodes tend to have common labels, during label propagation.

Contributions: The core contribution of this paper is to update label weights of labeled data in order to make label propagations for missed labels in each label propagation step. This process takes cluster assumption in clamping; in other words, even though some suitable labels in labeled data are missed,

[1] https://en.wikipedia.org/wiki/2016_Davao_City_bombing
[2] https://en.wikipedia.org/wiki/2016_Brussels_bombings
[3] https://en.wikipedia.org/wiki/Bombing_of_Sandhurst_Road_School
[4] https://catalog.data.gov/dataset/siam-2007-text-mining-competition-dataset

WII'18, March 11, 2018, Tokyo, Japan

our algorithm adds the weight from their similar data after propagating labels.

The original LP algorithm also takes the cluster assumption, but we enhance it by adding weights of top-k nearest neighbors. Some researchers improve the original LP algorithm to take label correlation [3, 5], to allow transductive learning [4], and to smooth affects of incorrect labels [2, 7, 6]. The closest studies to our study is the smoothing problem of incorrect labels because these studies solve the problems by taking cluster assumption. [2] takes the assumption to train kernels. [7] propagates label values as well as the original LP. Then, this algorithm adds the initial label values to the current label values at the end of each iteration. [6] applies KNN to incorporate similarity into probability matrix; thus, it does not update label values directly. The core difference between our paper and these previous studies is the objective. These three past works assume that some labels are wrongly assigned whereas we assume that all the attached labels are correct.

TRADITIONAL LABEL PROPAGATION

In this section, we describe the well-known LP algorithm [8]. We first present a formal definition of a graph $\mathscr{G} = (N, E)$, as LP invokes classification on a graph. Let $(x_1, y_1), ..., (x_l, y_l)$ represent labeled data, and $x_{l+1}, ..., x_n$ represent unlabeled data. For each data x_i, we create a node $n_i \in N$. We assign a positive value $w_{i,j}$ to an edge $(n_i, n_j) \in E$ as a similar value between x_i and x_j. The value of $w_{i,j}$ is determined as follows:

$$w_{i,j} = \exp(-\frac{\|v_i - v_j\|^2}{\alpha^2})$$

where v_i is a vector of x_i and α is a hyper-parameter.

Because a graph can be represented as a matrix, LP is usually invoked by multiplying a probabilistic transition matrix P by a label matrix Y to propagate label. The probabilistic transition matrix P represents the probability of transition from n_i to n_j, and is defined by normalizing the similarity metrics as follows:

$$P(i,j) = \frac{w_{i,j}}{\sum_{k=1}^{n} w_{i,k}}$$

The label matrix Y is defined as $[Y^l : Y^u] \in \mathbb{R}^{|D| \times |NC|}$, where $|D|$ is the number of data, $|NC|$ is the number of classes, Y^l is the label matrix for labeled data, and Y^u is the label matrix for unlabeled data. The value of $Y^l(i,c)$ is 1 if x_i is in class c; otherwise, it is 0.

The entire algorithm for LP is as follows: first, the two initial matrices, P and Y, are constructed. Then, the multiplication of them is repeated until all elements are fixed, or the iterative process achieves a certain number. During the iteration, values of Y^l are reset before invoking the next iteration–this is known as *clamping*. This clamping assumes that all the labeled data are perfectly assigned suitable labels.

LABEL PROPAGATION USING AMENDABLE CLAMPING

We extend the traditional LP to make label propagations for missed labels by adding the following two steps:

1. Enhancing propagating label values of similar documents.
2. Updating Y^l from labels of similar documents.

For the first extension, we add $(M \odot P) * Y_t$ in matrix multiplication where $\odot$ is the Hadamard product and $M \in \mathbb{N}^{|N| \times |N|}$ is a matrix representing KNN of the data; M_{ij} is 1 if the ith data is in the top-k similar ones of the jth data, otherwise it is 0. We call this extension *local-propagation* because $M \odot P$ allows propagating label values only for top-k similar data. In contrast, we call $P * Y_t$, which is used in traditional LP, *global-propagation*. To adjust the balance between local- and global-propagations, we introduce a hyper-parameter β as follows:

$$Prop_t = \beta P \cdot Y_t + (1-\beta)(M \odot P) \cdot Y_t$$

We then take cluster assumption, which means that *the data tend to form discrete clusters, and points in the same cluster are more likely to share a label*. For this, we calculate the averages from the top-k similar documents, and set them as the result of label propagation. The following equation invokes this process.

$$\frac{M \cdot Prop_t}{k}$$

For the second extension (updating the values of labeled data by similar data), we perform KNN again. Then, we calculate the average values of the top-k similar data for each label. These values are set before invoking the next iteration as follows:

$$Y^l_{t+1} = \frac{M \cdot Y^l_{t+1}}{k}$$

Finally, we show our entire algorithm in Alg. 1. Lines 4 and 6 correspond to the above two extensions, respectively.

Algorithm 1 Label propagation using amendable clamping

1: Construct a probablistic transition matrix P and M defined.
2: Let $Y_0 = [Y^l_0 : \mathbf{0}]$
3: **for** $t = 1$ to $T-1$ **do**
4: $\quad Prop_t = \beta P \cdot Y_t + (1-\beta)(M \odot P) \cdot Y_t$
5: $\quad Y_{t+1} = M \cdot Prop_t / k$
6: $\quad Y^l_{t+1} = M \cdot Y^l_{t+1} / k$
7: **end for**
8: **return** Y_T

EXPERIMENTAL RESULTS

Experimental Design

Dataset. We used the SIAM 2007 Text Mining Competition dataset, which is a subset of the Aviation Safety Reporting System (ASRS) dataset. It provides various types of aviation safety events reported by pilots, controllers, mechanics, flight attendants, and dispatchers. In this dataset, there are 4,819 labeled data, 4,819 unlabeled data, and 22 classes. On average, a labeled data is assigned 3.41 labels.

We first extract some data from this dataset to remove labels; we then remove labels for the data. Both the extraction ratio

and removal ratio are increased from 10% to 100%, in 10% increments, and are selected at random. For all the cases, we check the Micro-averaged F-scores measured by the 10-fold cross-validation of classifiers that are trained on the dataset.

As our demonstration focuses on document classification, we apply latent dirichlet allocation (LDA) [1] to all our data in order to assign weights between edges.

Algorithms. We compared LPAC with five baselines: LP, dynamic LP (DLP) [5], LP through Linear Neighborhoods (LNP) [6], random forest and SVM. DLP, which is the state-of-the-art LP algorithm, uses label correlation in the label propagation phase. LNP, which is an extension of the LP algorithm, discovers the structure of the entire data set through the linear neighborhoods of each data.

Parameters. We set T (the iteration number of LP), the dimension of the LDA and β to 1,000, 1,000 and 0.1 respectively. All these values were empirically chosen based on analyzing the results on the small held-out development dataset. Note here that DLP uses two additional parameters (λ and α). We set them to the same values used in [5]. Finally, we set a threshold for label assignment after the iteration of LP-based algorithms because these algorithms only assign a score for each label to each data point. To determine this threshold, we used 10% of unlabeled data, and selected 0.2 as the threshold because using this value was the best for F-scores of our algorithm. We used the same threshold for other LP-based algorithms.

Discussions of Accuracies

We show the results in Fig. 2. We can see that LPAC outperforms the comparative algorithms. Figs. 2 (a) $\sim$ (f) show the accuracies of each classifier trained on datasets in which 0% $\sim$ 100% of the documents do not have correct labels. The accuracies of the LPAC almost match the result achieved when the classifier is trained on a dataset in which 10% of the documents have missing labels. In contrast, all baselines tend to provide decreased accuracy once the ratio of missing-label documents exceeds 30%. In particular, once the missing-label ratio reaches 50%, accuracies of DLP, Random Forest and SVM rapidly decrease from about 10% to 50%. Note that LNP and LP also tend to keep their accuracies, but the scores are lower than LPAC.

To better enable comparison between all classifiers, Fig. 3 shows their F-scores in the cases that we extracted 50% and 70% of documents for removing labels, respectively. When the ratio of documents had missing labels reaches 40%, the F-scores of three baselines (DLP, Random Forest and SVM) begin to more early decrease compared with 0 $\sim$ 40%. Moreover, when the ratio is 70%, SVM and random forest sharply fall accuracies. In the case of 70% of documents extracted, the accuracies of almost all baselines begin to decrease when the ratio of missing labels exceeds 20%. Conversely, we can see that LPAC maintains a stable accuracy for all ratios of missing labels.

Next, we show the correlation between F-scores and the number of labeled documents for LPAC in Fig. 4. We plot results of three cases where 0%, 50% and 70% of documents had

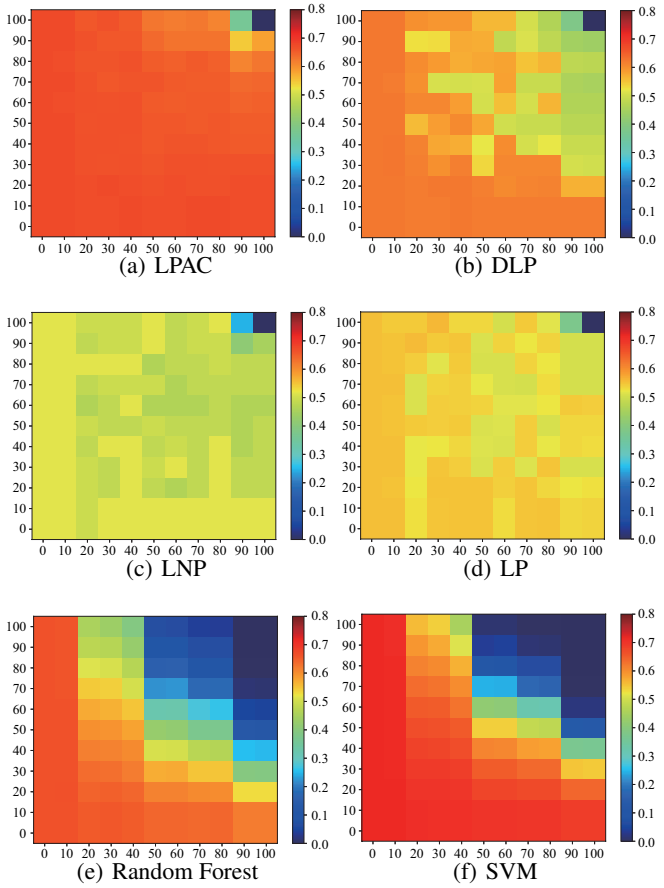

Figure 2. Each sub-figure shows the Micro-averaged F-scores for six classifiers. The x axis represents the ratio of documents that has missing labels. The y axis represents the ratio of missing labels. The color of each cell represents the F-score. A red cell indicates better accuracy than blue.

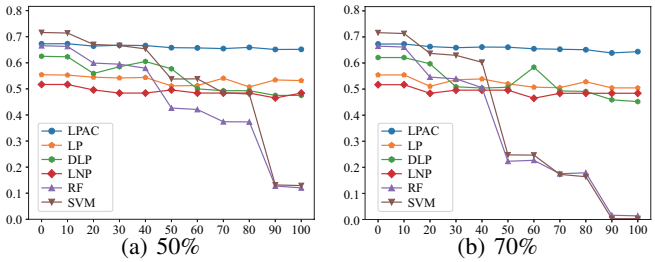

Figure 3. Each sub-figure shows the results achieved when 50% and 70% of documents had missing labels, respectively. In the two sub-figures, the x axis represents the ratio of missing labels whereas the y axis represents the F-score.

missing labels. Fig. 4 shows that there is a positive correlation between them. Indeed, the correlation coefficiencies of them are 0.76, 0.72 and 0.75, respectively. Thus, we can say that the more there are correct labels are assigned to training data, the better accuracy we can obtain.

Next, we show how many iterations each LP-based algorithm needs to convergence the propagating labels in Fig. 5. As this figure shows, LPAC is the lowest of the four LP-based algo-

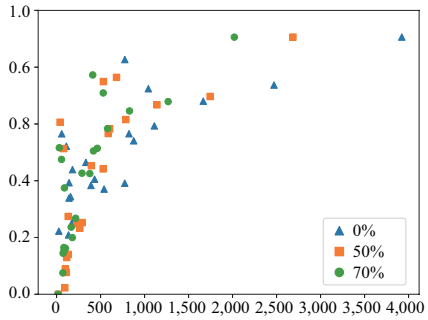

Figure 4. Correlation between the number of training label and F-scores of LPAC. The x axis represents the number of training data in each category. The y axis represents the value of F-scores. The blue, yellow and green points represent the F-score when 0%, 50% and 70% of documents had missing labels, respectively.

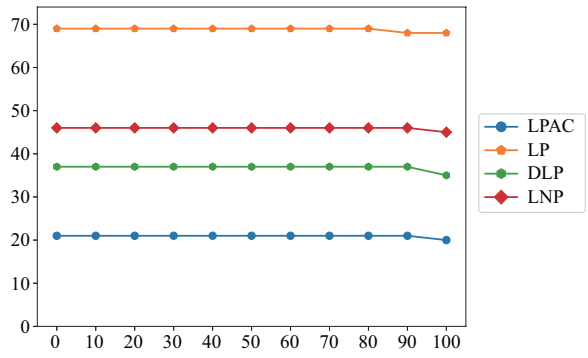

Figure 5. The number of iterations of LP-based algorithms.

rithms. This is because that LPAC makes label propagation by averaging label values of not only all data but also ones of similar documents in order to enhance take cluster assumption during propagating labels.

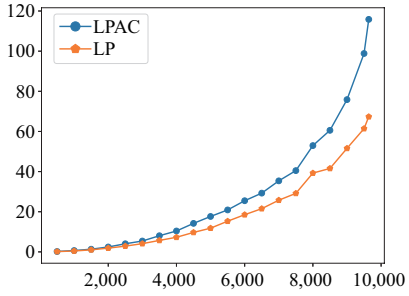

Figure 6. The analyzing time of LP and LPAC. The x axis represents the size of training datasets. The y axis represents the analyzing time (seconds) of each method.

Finally, we show the analyzing time of LP and LPAC in Fig. 6. We can see that both of the two lines draw exponential growth. The analyzing time of LPAC is about twice as slow as the one of LP if we use 10,000 data; however, it is not problem for many objectives since the LPAC's analyzing time is about two minutes.

CONCLUSION

In this paper, we proposed a novel graph-based multi-label classification (LPAC) to apply to a moderately challenging multi-labeling task. LPAC enforces label propagations by two extensions: propagating labels according to top-k similar data and updating labeled data by taking cluster assumption. As demonstrated in our experiments, these two extensions make the F-score of LPAC stable compared to those of algorithms proposed by previous studies.

Future work will identify (a) *the effective utilization of label correlation*. Once our algorithm is extended to take care of label correlation, we can implement label recommendations. This suggestion should be helpful especially in the case that there are many labels such as Wikipedia category system. Future work will also identify (b) *how effectively our algorithm works on a real dataset that contains missing labels data like Wikipedia* as discussed in Introduction. Finally, as a future work we will study (c) *establishing algorithm that can be trained on dataset including both of wrong and missing*. Although this paper assumes that there are no data attached any wrong labels, real datasets might contain both of the two kinds of labels at the same time. We will explore how to reduce the noises.

Acknowledgments. This work was supported in part by MEXT Grant-in-Aid (#17K12792).

REFERENCES

1. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3 (March 2003), 993–1022.
2. Olivier Chapelle, Jason Weston, and Bernhard Schölkopf. 2002. Cluster Kernels for Semi-supervised Learning *(NIPS'02)*. MIT Press, Cambridge, MA, USA, 601–608.
3. Feng Kang, Rong Jin, and Rahul Sukthankar. 2006. Correlated Label Propagation with Application to Multi-label Learning *(CVPR'06)*. New York, NY, USA, 1719–1726.
4. Xiangnan Kong, Michael K. Ng, and Zhi-Hua Zhou. 2013. Transductive Multilabel Learning via Label Set Propagation. *IEEE Trans. Knowl. Data Eng.* 25, 3 (2013), 704–719.
5. Bo Wang and John Tsotsos. 2016. Dynamic label propagation for semi-supervised multi-class multi-label classification. *Pattern Recognition* 52 (2016), 75 – 84.
6. Fei Wang and Changshui Zhang. 2006. Label propagation through linear neighborhoods *(ICML'06)*. ACM, New York, NY, USA, 985–992.
7. Denny Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard SchÃűlkopf. 2004. Learning with Local and Global Consistency *(NIPS'04)*. MIT Press, 321–328.
8. Xiaojin Zhu. 2005. *Semi-supervised Learning with Graphs*. Ph.D. Dissertation. Pittsburgh, PA, USA.