VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Van Quang

AN ALGORITHM ADAPTATION MULTI-LABEL CLASSIFICATION METHOD AND EXPERIMENTS ON VIETNAMESE TEXT

Major: Computer Science

HA NOI - 2017

VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Van Quang

AN ALGORITHM ADAPTATION MULTI-LABEL CLASSIFICATION METHOD AND EXPERIMENTS ON VIETNAMESE TEXT

Major: Computer Science

Supervisor: Assoc. Prof. Ha Quang Thuy

Co-Supervisor: MSc. Pham Thi Ngan

HA NOI - 2017

AUTHORSHIP

"I hereby declare that the work contained in this thesis is of my own and our research group and has not been previously submitted for a degree or diploma at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no materials previously published or written by another person except where due reference or acknowledgement is made."

Hanoi, 4th of May, 2017

Nguyen Van Quang

SUPERVISOR'S APPROVAL

"I hereby approve that the thesis in its current form is ready for committee examination as a requirement for the Bachelor of Computer Science degree at the University of Engineering and Technology."

Signature:……...

Assoc.Prof. Ha Quang Thuy
Thesis Supervisor
Faculty of Information Technology
University of Engineering and Technology (UET)
Vietnam National University, Hanoi (VNU)

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincerest gratitude to my supervisor, Assoc. Prof. Ha Quang Thuy who has offered me a precious chance to pursue this research and brought me to the exciting but challenging world of academia. His constant encouragement and mentorship have helped me not only in gaining the understanding of research, but also in learning various aspects of life in general. I am truly indebted to every assistance and opportunity that Prof. Ha Quang Thuy has brought to my life.

I am also deeply grateful to my co-supervisor, MSc. Pham Thi Ngan for her invaluable guidance in making this research possible. Without her advice and support, it would be a lot tougher to the completion of my research. The knowledge and research experience I gained from her will be crucial to my future.

I greatly appreciate Data Science and Knowledge Technology Laboratory of VNU-UET where I set my first foot on academic research, and later my thesis. I have many times taken the invaluable advice, experience and knowledge from the distinguished scholars there: Assoc. Prof. Phan Xuan Hieu, Dr. Tran Trong Hieu and many others.

My friends and lab-mates here at university also deserve my deepest appreciation: Nguyen Thac Thong, Nguyen Tuan Phong, Trinh Thi Thu, Tran Van Hien and others. They have all provided companionship and encouragement during this long journey, both in and out from the schooling and researching.

I acknowledge a forever gratitude to my parents and my siblings for their unconditional love and support throughout the writing of this thesis and my life in general. They are inspiring motivation for every single step I have been walking through life.

.

ABSTRACT

Multi-label classification is one of the fundamental and crucial tasks in data mining. Multi-label classification (MLC) has received considerable attention in recent years, whose goal is to construct a model assigning a proper subset of predefined labels to new unseen samples. Current studies on multi-label classification have mostly concentrated on the improvement of supervised learning approaches, which depend largely on labeled data to train models.

The main aim of this thesis is to investigate some existing MLC approaches and propose a novel algorithm adaptation MLC method leveraging the label correlations as well as unlabeled data to boost the performance of the system. The underlying idea of the algorithm is to adapt the semi-supervised clustering algorithm to multi-label data. In the training phase, the algorithm harnesses the specific features of prominent labels using the greedy idea of LIFT algorithm [1] while exploiting unlabeled data by clustering mechanism from TESC [2]. Moreover, this thesis also concerned the feature enrichment using hidden topic features and examined the contribution of feature dimension reduction to the model. In the predicting phase, the model calculates the similarity distances between an unseen sample and each of the learned clusters, and assigns the label set of the nearest clusters to it.

Several experiments on Vietnamese text were conducted to evaluate the performance and effectiveness of the proposed method. The experimental results on test data show that a reasonably small amount of unlabeled data helps to increase the F1 score which is better than several classical algorithms for multi-label data. Remarkably, a small amount of unlabeled data and LDA features added can make a great contribution towards the performance of the algorithm compared with only a large amount of labeled data.

Keywords: multi-label classification, algorithm adaptation, semi-supervised, specific feature, LDA, dimension reduction.

TÓM TẮT

Phân lớp đa nhãn là một trong những bài toán cơ bản và quan trọng trong khai phá dữ liệu. Mục tiêu của phân lớp đa nhãn là xây dựng một mô hình để gán một tập nhãn được định nghĩa trước cho các đối tượng chưa biết. Các nghiên cứu về phân lớp đa nhãn gần đây phần lớn tập trung vào sự cải tiến theo tiếp cận học có giám sát, phương pháp này thường đòi hỏi một lượng lớn các dữ liệu được gán nhãn để huấn luyện mô hình.

Mục đích chính của khóa luận này là nghiên cứu các phương pháp học phân lớp đa nhãn đã có và đề xuất một giải thuật phân lớp đa nhãn mới dựa trên tiếp cận thích nghi thuật toán. Ý tưởng cơ bản của thuật toán đề xuất là thích nghi phương pháp phân cụm bán giám sát cho dữ liệu đa nhãn. Mô hình đề xuất khai thác sự tương quan giữa các nhãn cũng như sự sẵn có của dữ liệu không nhãn nhằm tăng kết quả gán nhãn của mô hình. Trong pha huấn luyện, thuật toán khai thác các đặc trưng riêng biệt của từng lớp nhãn nổi bật sử dụng ý tưởng của giải thuật LIFT [1], và tận dụng dữ liệu không nhãn bằng cơ chế phân cụm bán giám sát trong thuật toán TESC [2]. Hơn nữa, khóa luận cũng quan tâm đến việc làm giàu đặc trưng bằng cách thêm các đặc trưng ẩn và xem xét sự đóng góp của việc giảm số chiều đặc trưng đến thuật toán. Trong pha dự đoán, mô hình tính toán khoảng cách giữa một mẫu mới chưa gán nhãn đến các cụm đã học được, sau đó gán tập nhãn của cụm có khoảng cách gần nhất cho nó.

Một vài thực nghiệm trên miền dữ liệu tiếng Việt được thực hiện để đánh giá kết quả phân lớp đa nhãn của mô hình đề xuất. Kết quả thực nghiệm trên dữ liệu đánh giá cho thấy một lượng hợp lý dữ liệu không nhãn giúp tăng độ đo F1, giá trị này cao hơn so với một số kết quả thu được thuật toán phân lớp đa nhãn kinh điển. Điều này gợi ý rằng, sự đóng góp của một lượng nhỏ dữ liệu không nhãn và các đặc trưng ẩn được thêm vào có thể cải thiện kết quả phân lớp của mô hình hơn là chỉ sử dụng một lượng lớn dữ liệu được gán nhãn.

Từ khóa: phân lớp đa nhãn, thích nghi thuật toán, bán giám sát, đặc trưng riêng biệt, LDA, giảm chiều đặc trưng.

TABLE OF CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

MLC	Multi-Label Classification
MASS	semi-supervised Multi-label clASSification algorithm with specific features
MLL	Multi-Label Learning
TESC	Text classification using Semi-supervised Clustering
LIFT	multi-label learning with Label specIfic FeaTures
LDA	Latent Dirichlet Allocation
ML-kNN	Multi-Label k-Nearest Neighbor
ML-DT	Multi-Label Decision Tree
SVM	Support Vector Machine

Chapter 1

ALGORITHM ADAPTATION METHODS IN MULTI-LABEL CLASSIFICATION

1.1. Multi-label classification overview

One of the most important tasks in data mining is classification or categorization. The goal of classification is to build a model to assign a class label to a new sample as accurately as possible. This formulation is attributed to a single-label classification where an unseen sample is assigned to a single label or class. However, the current classification problems have become considerably challenging, in which a sample can belong to several labels simultaneously. This kind of problems is categorized into multi-label classification (also called multi-label learning) compared with single-label classification. So far, multi-label classification has appeared in many application domains such as protein classification [3] [4], music categorization [5] [6], medical diagnosis [7], and image annotation [8].

1.1.1. Multi-label classification problem representation

Conventionally, Zhi-Hua Zhou et al [9] formulate the multi-label classification (MLC) problem: Let $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n}}$ denote the n-dimensional instance space, and $\boldsymbol{Y} = \{y_1, y_2, \dots, y_q\}$ denote the label space comprising of q possible class labels. The task

of multi-label learning is to learn a function $f: X \rightarrow 2^Y$ from the multi-label training data $D = \{(\boldsymbol{x_i}, Y_i) \mid 1 \leq i \leq m\}$, in which for each instance $(\boldsymbol{x_i}, Y_i)$, $\boldsymbol{x}_i \in \boldsymbol{X}$ is a n-dimensional feature vector $\boldsymbol{x_i} = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$, and $Y_i \subseteq \boldsymbol{Y}$ is the set of labels it belongs to. For a new instance $\boldsymbol{x} \in \boldsymbol{X}$, the multi-label function $f(\cdot)$ returns $f(\boldsymbol{x})$ as a set of predicted labels for this instance.

Table 1.1 An Example of a Single-label vs. a Multi-label Dataset [9]

Examples	Features	SINGLE-LABEL	SINGLE-LABEL MULTICLASS	MULTI-LABEL OUPUT				
		$Y_i \in \boldsymbol{Y} = \{0, 1\}$	$Y_i \in \boldsymbol{Y} = \{, y_2, y_3, y_4\}$	y1	y2	y3	y4	$Y_i \subseteq \boldsymbol{Y} = \{y_1, y_2, y_3, y_4\}$
1	$\boldsymbol{x_1}$	1	y_2	1	1	0	1	$\{y_1, y_2, y_4\}$
2	$\boldsymbol{x_2}$	0	y_4	0	0	0	1	$\{y_4\}$
3	$\boldsymbol{x_3}$	0	y_3	0	1	1	1	$\{y_2, y_3, y_4\}$
4	$\boldsymbol{x_4}$	1	y_1	1	0	1	0	$\{y_1, y_3\}$
5	$\boldsymbol{x_5}$	0	y_4	0	1	1	0	$\{y_2, y_3\}$

1.1.2. The properties of multi-label classification

As described in [10], a multi-label classification problem consists of several properties as below:

(1) The set of class labels is predefined, meaningful to human.

(2) The number of class labels is not greater than the number of features or attributes, and limited to a certain scope.

(3) Each training instance belongs to a subset of label set.

(4) Feature dimension reduction might be applied although the number of features may be large.

(5) The number of training instances may be large.

(6) There may be the correlations among the labels. For example, the news related to "politics" is likely to be irrelevant to "entertainment" label while in film classification, *action* movies are more related to *crime* label rather than *comedy*.

(7) The imbalance may exist in the dataset. There are some labels that appear more frequently with the relatively higher number than others. Figure 1.1 illustrates the unbalanced distribution of labels in movie dataset from IMDB.

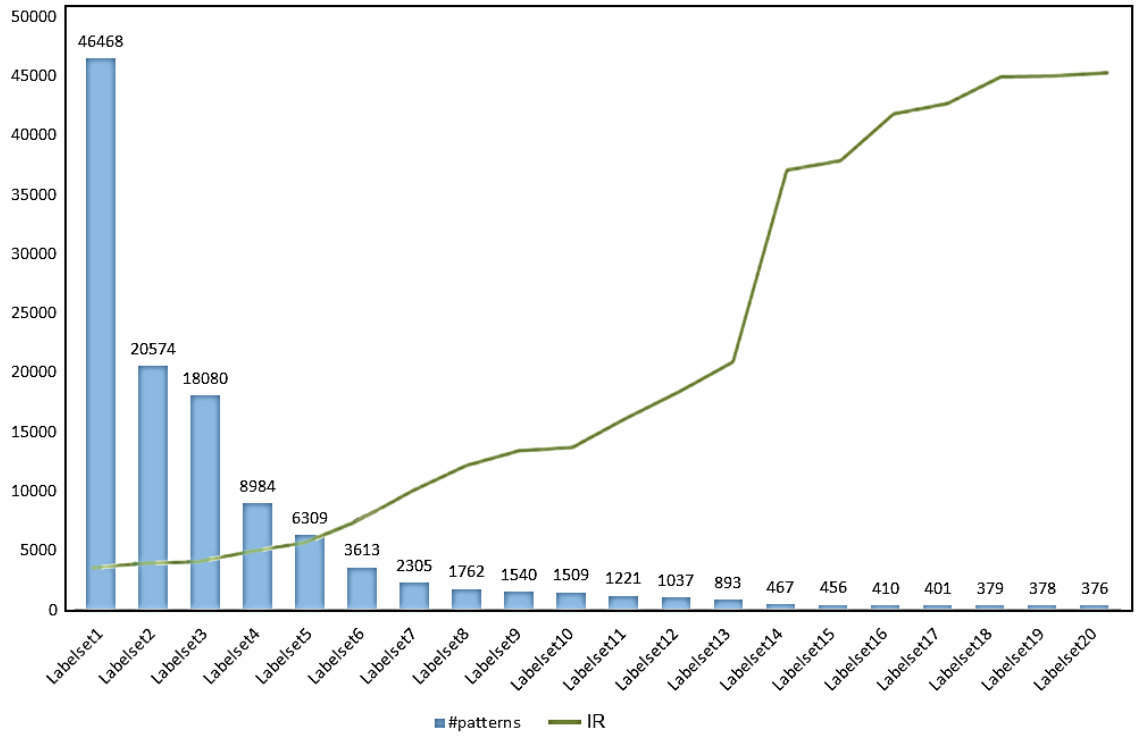


Figure 1.1 Distribution of labels over examples in IMDB dataset [10]

1.2. Multi-label classification methods

According to Eva Gibaja and Sebastian Ventura [10], the existing methods for multi-label classification can be classified into two main groups: a) Problem transformation methods, and b) Algorithm adaptation methods.

Concisely, the idea behind problem transformation methods is to fit data to algorithm. On the other hand, the idea behind algorithm adaptation methods is to fit algorithm to data.

1.2.1. Problem Transformation Methods

Problem transformation methods are those approaches that transform the multi-label classification problems into either one or more single-label classification or regression problems. Following this approach, every single-label classifier is performed to predict each label. After that, the results of each classifier will be combined to obtain the final

multi-label classification result. Those methods take the huge advantage of any existing classical supervised methods including Support Vector Machine (SVM), Naïve Bayes, Logistic regression, and Perceptron, etc. Representative algorithms include Binary Relevance, Adaboost.MH, Stacked Aggregation, Classifier Chains, Calibrated Label Ranking and Random k-labelsets. Most of those algorithms can be implemented parallel processing, which save time to train classifiers. However, their main drawback is that they do not account the existence of label correlations, which might lead to the degradation on the performance of classification system.

1.2.2. Algorithm adaptation methods

We call algorithm adaptation methods, those methods extend specific single-label classification algorithms into a multi-label classification algorithm which can be directly applied to deal with multi-label dataset. Some methods can be grouped into algorithm adaptation methods including Multi-Label Decision Tree (ML-DT), adapting lazy learning techniques Multi-Label k Nearest Neighbor (k-NN), adapting kernel techniques Rank Support Vector Machine (Rank – SVM), adapting information-theoretic techniques Collective Multi-Label (CML), etc. Several methods based on this approach consider the existence of correlations among labels to improve the performance of the system.

1.3. Some representatives of algorithm adaptation methods

Three representatives of algorithm adaptation MLC including ML-kNN, ML-DT, and Rank-SVM will be introduced in detail. Those representatives are selected to present here with some criteria according to [11]: a) Broad spectrum b) primitive impact c) Favorable citations.

Briefly, the main idea of algorithm adaptation methods in multi-label classification is to adapt popular single label learning techniques to handle with multi-label data.

1.3.1. ML-kNN Algorithm

Multi-Label k-Nearest Neighbor (ML-kNN) is an adaptation of traditional k-nearest neighbor lazy learning algorithm which implements k-nearest neighbor technique to

tackle with multi-label data. Min-Ling Zhang and Zhi-Hua Zhou [12] suggest that the maximum a posteriori (MAP) principle is adopted to identify a set of labels for unseen instances.

Let $\boldsymbol{D}$ denote the domain of instances and $\mathrm{N}(\boldsymbol{x}, \boldsymbol{D}, k)$ denote the set of the k nearest neighbors for an unseen instance $\boldsymbol{x} \in \boldsymbol{D}$. Let $Y = \{y_1, y_2, \ldots, y_q\}$ be the set of predefined labels. To classify the labels for $\boldsymbol{x}$, ML-kNN firstly calculates the distances from $\boldsymbol{x}$ to all the instance in the training set to identify the set of k nearest neighbors $\mathrm{N}(\boldsymbol{x}, \boldsymbol{D}, k)$. ML-kNN uses Euclidean metrics to measure the similarity between instances. Given two instances $\boldsymbol{x_1} = (x_{11}, x_{12}, \ldots, x_{1n})$ and $\boldsymbol{x_2} = (x_{21}, x_{22}, \ldots, x_{2n})$, the Euclidean distance between two instances is defined:

$$Dist(\boldsymbol{x_1}, \boldsymbol{x_2}) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2} \tag{1.1}$$

For each label y_i, the algorithm computes the number of neighbors of $\boldsymbol{x}$ belonging to this label as below:

$$C_j = \sum_{(x^*, Y^*) \in N(x, \boldsymbol{D}, k)} [\![y_j \in Y^*]\!] \tag{1.2}$$

Let H_j denote the event that the unseen instance $\boldsymbol{x}$ has label y_j; thereby, deriving two statistical quantities. First, $\mathbb{P}(H_j | C_j)$ denotes the posterior probability that H_j holds given the condition that the instance $\boldsymbol{x}$ has exactly C_j neighbors which have label y_j. On the other hand, $\mathbb{P}(\neg H_j | C_j)$ denotes the posterior probability that H_j doesn't hold given the above condition. The set of predicted labels Y is determined using the following MAP principle:

$$Y = \{y_j | \mathbb{P}(H_j | C_j) / \mathbb{P}(\neg H_j | C_j) > 1, 1 \leq j \leq q\} \tag{1.3}$$

Using the Bayesian rule, the ratio can be written as below:

$$\frac{\mathbb{P}(H_j|C_j)}{\mathbb{P}(\neg H_j|C_j)} = \frac{\mathbb{P}(H_j).\mathbb{P}(C_j|H_j)}{\mathbb{P}(\neg H_j).\mathbb{P}(C_j|\neg H_j)} \tag{1.4}$$

Where $\mathbb{P}(H_j)$ and $\mathbb{P}(\neg H_j)$ denote the prior probability that H_j holds or doesn't hold respectively. Moreover, $\mathbb{P}(C_j|H_j)$ and $\mathbb{P}(C_j|\neg H_j)$ denote the likelihood that $\boldsymbol{x}$ has exactly C_j neighbors which have label y_j. The prior and posterior probabilities can be derived directly from the training data set. The prior probabilities can be obtained using the following formula:

$$\mathbb{P}(H_j) = \frac{s + \sum_{i=1}^{m} [\![y_i \in Y_i]\!]}{s \times 2 + m} \tag{1.5}$$

$$and\ \mathbb{P}(\neg H_j) = 1 - \ \mathbb{P}(H_j)\ (1 \le j \le q) \tag{1.6}$$

Here, *s* is a smoothing parameter to adjust the effect of prior uniform on the estimation and *m* is the number of examples in the training set. Usually, in Laplace smoothing, the value of *s* is equal to 1.

Two frequency arrays $\mathcal{K}_j$ and $\widetilde{\mathcal{K}}_j$ of size $k + 1$ are computed in ML-kNN:

$$\begin{aligned} \mathcal{K}_j[r] &= \sum_{i=1}^{m} [\![y_j \in Y_i]\!].[\![\delta_j(\boldsymbol{x_i}) = r]\!]\ (0 \le r \le k) \\ \widetilde{\mathcal{K}}_j[r] &= \sum_{i=1}^{m} [\![y_j \notin Y_i]\!].[\![\delta_j(\boldsymbol{x}_i) = r]\!]\ (0 \le r \le k) \end{aligned} \tag{1.7}$$

In which, $\mathcal{K}_j$[r] stores the number of training examples which are associated with label y_j and have exactly r neighbors with label y_j while $\widetilde{\mathcal{K}}_j[r]$ stores the number of training examples which are not associated with label y_j and have exactly r neighbors with label y_j. Similar to C_j, $\delta_j(\boldsymbol{x}_i)$ counts the number of $\boldsymbol{x_i}$'s neighbors with label y_j.

$$\delta_j(\boldsymbol{x}_i) = \sum_{(\boldsymbol{x}^*, \boldsymbol{Y}^*) \in N(x, \boldsymbol{D}, k)} [\![y_j \in Y^*]\!] \tag{1.8}$$

Afterwards, ML-kNN estimates the likelihoods based on elements in $\mathcal{K}_j$ and $\widetilde{\mathcal{K}}_j$:

$$\mathbb{P}(C_j|H_j) = \frac{s+\mathcal{K}_j[C_j]}{s\times(k+1)+\sum_{r=0}^{k}\mathcal{K}_j[r]} \quad 1 \le j \le q, 0 \le C_j \le k)$$

$$\mathbb{P}(C_j|\neg H_j) = \frac{s+\widetilde{\mathcal{K}}_j[C_j]}{s\times(k+1)+\sum_{r=0}^{k}\widetilde{\mathcal{K}}_j[r]} \; 1 \le j \le q, 0 \le C_j \le k) \tag{1.9}$$

Replacing equations (1.5) of prior probabilities and equations (1.9) of likelihoods into the equation (1.4), and finally obtaining the result of predicted labels in Eq. (1.3).

$$\mathrm{Y} = \mathrm{ML} - \mathrm{kNN}(\mathcal{D}, \mathrm{k}, \mathrm{x})$$

1. **for** i=1 **to** m **do**
2. Find a set $\mathcal{N}(x)$ of k nearest neighbors for $\boldsymbol{x_i}$
3. **end for**
4. **for** j=1 **to** q **do**
5. Estimate the prior probabilities $\mathbb{P}(H_j)$ và $\mathbb{P}(\neg H_j)$ according to the Eq.:

$$\mathbb{P}(H_j) = \frac{s+\sum_{i=1}^{m}[\![y_j \in Y_i]\!]}{s\times 2+m}$$
$$\mathbb{P}(\neg H_j) = 1 - \mathbb{P}(H_j)\ (1 \le j \le q\ )$$

6. Update frequency arrays $\mathcal{K}_j$ and $\widetilde{\mathcal{K}}_j$ as *Eq.(2.7)*
7. **end for**
8. Find a set $\mathcal{N}(x)$ of k nearest neighbors for $\boldsymbol{x_i}$
9. **for** j=1 **to** q **do**
10. Compute C_j based on the following equation:

$$C_j = \sum_{(x^*,Y^*)\in\mathcal{N}(x)} [\![y_j \in Y^*]\!]$$

11. **end for**
12. **return** Y according to this equation:

$$Y = \{y_j|\mathbb{P}(H_j|C_j)/\mathbb{P}(\neg H_j|C_j) > 1, 1 \le j \le q\}$$

Figure 1.2 Pseudo-code of ML-kNN [11]

Adopting the philosophy of both lazy learning and Bayesian reasoning brings the advantage to ML-kNN: one is that we can adjust the decision boundary flexibly as the

neighbors are identified for each unseen object; second is that the class-imbalance issue can be lessened due to the prior probabilities estimated for each label. On the other hand, the minus point of this method is that ML-kNN requires the large space to store the whole training dataset; and it also ignores the exploitation of label correlations.

ML-kNN has the computational complexity of $\mathcal{O}(m^2 d + qmk)$ for training phase and $\mathcal{O}(md + qk)$ for evaluation phase.

1.3.2. ML-DT Algorithm

The basic concept behind Multi-label decision tree algorithm (MT-DT), which A. Clare and R. D. King [13] chose, is to fit the decision tree algorithm C4.5 to multi-label data. According to the algorithm C4.5 [14], the decision tree is built top down using an information gain criterion based on multi-label entropy. Information gain is one of the highly-used criterion for splitting decision tree, which calculates the difference between the entropy of the whole set of remaining training examples and the weighted sum of the entropy of the subsets resulted from partitioning on the values of that attributes. Let $\mathcal{T}\{(x, y_j)|1 \leq i \leq n\}$ be multi-label set of n training examples, the information gain obtained by partitioning $\mathcal{T}$ on the value ϑ of the *l*-th feature is defined:

$$IG(\mathcal{T}, l, \vartheta) = MLEnt(\mathcal{T}) - \sum_{\rho \in \{-,+\}} \frac{|\mathcal{T}^\rho|}{|\mathcal{T}|} . MLEnt(\mathcal{T}^\rho) \tag{1.10}$$

Where,

$$\begin{aligned} \mathcal{T}^- &= \{(x_i, Y_i)|x_{il} \leq \vartheta, 1 \leq i \leq n\} \\ \mathcal{T}^+ &= \{(x_i, Y_i)|x_{il} > \vartheta, 1 \leq i \leq n\} \end{aligned} \tag{1.11}$$

$\mathcal{T}^-$ and $\mathcal{T}^+$ comprise of examples which have values on the *l*-th feature lower than ϑ and higher than ϑ.

To build a tree from the root note ($\mathcal{T} = \mathcal{D}$), the feature and its splitting value which maximize the information gain according to Eq. (1.10), are determined. Thereby, two child nodes, namely $\mathcal{T}^-$ and $\mathcal{T}^+$, are generated. Similarly, this procedure is called recursively by considering $\mathcal{T}^-$ or $\mathcal{T}^+$ as the new root node. The procedure will terminate

when it reaches the stopping criterion $\mathcal{C}$. The criterion C here can be the size of the child node being smaller than the pre-defined threshold.

To obtain multi-label entropy which is $MLEnt$ appearing in Eq. (1.10), each subset $Y \subseteq \mathcal{Y}$ is considered as a new class then apply the conventional single-label entropy to compute:

$$where\ \mathbb{P}(Y) = \frac{\sum_{i=1}^{n} [\![Y_i = Y]\!]}{n} \tag{1.12}$$

Many of them might, however, not appear in $\mathcal{T}$ as the number of new classes increases exponentially ($2^{|\mathcal{Y}|}$) with regards to the size $|\mathcal{Y}|$. The probability for those is estimated trivially, $\mathbb{P}(Y) = 0$. ML-DT makes an assumption that the labels are independent, and estimates the multi-label entropy:

$$\begin{gathered} MLEnt(\mathcal{T}) = \sum_{j=1}^{q} -p_j log_2 p_j - (1 - p_j) log_2 (1 - p_j) \\ where\ p_j = \frac{\sum_{i=1}^{n} [\![y_j \in Y_i]\!]}{n} \end{gathered} \tag{1.13}$$

Here, p_j denotes the proportion of examples in $\mathcal{T}$ with labels y_j. The Eq. (1.13) can be considered as a simplified version of Eq. (1.10) given the assumption that labels are uncorrelated, and $MLEnt(\mathcal{T}) \geq \widehat{MLEnt}(\mathcal{T})$.

To get the set of predicted labels for an unseen instance $\boldsymbol{x}$, ML-DT absorbs the instance and traverses the learned decision tree till reaching a leaf node associated with a number of training examples $\mathcal{T} \subseteq \mathcal{D}$. Afterwards, the set of predicted labels for an instance $\boldsymbol{x}$ is defined as below:

$$Y = \{ y_j | p_j > 0.5, 1 \leq j \leq q \} \tag{1.14}$$

In other words, if for one leaf node, the majority of training examples falling into it have label y_j, any test instance allocated within the same leaf node will regard y_j as its relevant label.

Figure 1.3 illustrates the pseudo-code of multi-label. The eminent advantage of multi-label decision tree is the high efficiency in adopting the decision tree method from multi-label data.

The computational complexity of ML-DT is $\mathcal{O}(mdq)$ for training phase and $\mathcal{O}(mq)$ for testing phase.

$\mathrm{Y} = \mathrm{ML} - \mathrm{DT}(\mathcal{D}, \mathcal{C}, \mathrm{x})$

1. Create a decision tree with root node $\mathcal{N}$ affiliated with the whole training set ($\mathcal{T} = \mathcal{D}$)
2. **if** stopping criterion $\mathcal{C}$ is met **then**
3. **break** and **go to** step 9
4. **else**
5. Identify the feature-value pair (l, ϑ) *which maximizes Eq.(1.13)*
6. Set $\mathcal{T}^-$ and $\mathcal{T}^+$ according to Eq. [1.11]
7. Set $\mathcal{N}.lsubtree$ and $\mathcal{N}.rsubtree$ to the decision trees recursively constructed with $\mathcal{T}^-$ and $\mathcal{T}^+$ respectively.
8. **end if**
9. Tranverse **x** along the decision tree from the root node until a leaf node node is reached;
10. **return** Y according to Eq. (1.14)

Figure 1.3 Pseudo-code of Multi-label Decision Tree [11]

1.3.3. Rank-SVM Algorithm

Elisseeff and Weston [15] introduce an SVM ranking-based algorithm which improves the performance over the Binary Relevance method with SVMs. This algorithm follows the key idea of SVMs: a set of linear classifiers are optimized to minimize a cost function while maintaining a large margin. The cost function they use is the empirical *ranking loss* which is defined as the average fraction of pairs of labels that are ordered incorrectly. Rank-SVM also enables to deal with nonlinearity using kernel tricks.

Let $\mathcal{W} = \{(\boldsymbol{w_j}, b_j) | 1 \leq j \leq q\}$ denote a set of q linear classifiers, where $\boldsymbol{w_j} \in \mathbb{R}^{\boldsymbol{d}}$ and $b_j \in \mathbb{R}$ are the weight *vector* and *bias* with respect to the *j*-th class label y_j.

Rank-SVM defines the learning system's *margin* on $(\boldsymbol{x_j}, Y_i)$ by considering its ranking ability on the example's relevant and irrelevant labels:

$$\min_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \frac{\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k}{\|\boldsymbol{w_j} - \boldsymbol{w_k}\|} \tag{1.16}$$

Here, $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ will return the dot product $\boldsymbol{u^T v}$. From geometrical perspective, the boundary for each relevant-irrelevant label pair $(y_j, y_k) \in Y_i \times \bar{Y}_i$ corresponds to the hyperplane $\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x} \rangle + b_j - b_k = 0$. The Eq. (1.16) considers the signed L_2-distance of $\boldsymbol{x_i}$ to hyperplanes of every relevant-irrelevant label pair, and then returns the minimum as the margin on $(\boldsymbol{x_i}, Y_i)$. The multi-label margin on the whole training set $\mathcal{D}$ follows the equation below:

$$\min_{(\boldsymbol{x_i}, Y_i) \in \mathcal{D}} \min_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \frac{\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k}{\|\boldsymbol{w_j} - \boldsymbol{w_k}\|} \tag{1.17}$$

When the learning system has the capability to properly rank every relevant-irrelevant label pair for each training example, $(\boldsymbol{x_j}, Y_i)$, in the training set $\mathcal{D}$, Eq. (1.17) will return positive margin. Ideally, Rank-SVM can rescale the linear classifiers to ensure:

a) $\forall\, 1 \le i \le m \; and \; (y_j, y_k) \in Y_i \times \bar{Y}_i, \langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k > 1$;

b) $\exists i^* \in \{1, \ldots, m\} \; and \; (y_{j^*}, y_{k^*}) \in Y_{i^*} \times \bar{Y}_{k^*}, \langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k = 1$;

Therefore, the problem of maximizing the margin in Eq. (2.17) can be represented as:

$$\min_{W} \min_{(\boldsymbol{x_i}, Y_i) \in \mathcal{D}} \min_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \frac{1}{\|\boldsymbol{w_j} - \boldsymbol{w_k}\|^2} \tag{1.18}$$

subject to: $\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k \ge 1 \qquad with\; 1 \le i \le m \; and \; (y_j, y_k) \in Y_i \times \bar{Y}_i$. Given that adequate samples in training set such that for each label pair (y_j, y_k) $(j \ne k)$, there exists $(\boldsymbol{x}, Y) \in \mathcal{D}$ satisfying $(y_j, y_k) \in Y \times \bar{Y}$.

Hence, the aim of Eq. (1.18) corresponds to $\max_{W} \min_{1 \le j < k \le q} \frac{1}{\|\boldsymbol{w_j} - \boldsymbol{w_k}\|^2}$ and the optimization problem can be re-presented as:

$$\max_{W} \min_{1 \le j < k \le q} \left\| \boldsymbol{w_j} - \boldsymbol{w_k} \right\|^2 \tag{1.19}$$

subject to: $\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k \geq 1 \qquad with\ 1 \le i \le m\ and\ (y_j, y_k) \in Y_i \times \bar{Y}_i$

To deal with the difficulty of maximum operator, Rank-SVM tries to simplify Eq. (1.19) by substituting the maximum operator by the sum operator approximately:

$$\min_{W} \sum_{j=1}^{q} \left\| w_j \right\|^2 \tag{1.20}$$

subject to: $\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k \geq 1 \qquad with\ 1 \le i \le m\ and\ (y_j, y_k) \in Y_i \times \bar{Y}_i$

In reality, the constraints in Eq. (1.20) cannot always be met, *slack variables* are introduced to Eq. (1.20):

$$\min_{\{W, \Xi\}} \sum_{j=1}^{q} \left\| w_j \right\|^2 + C \sum_{i=1}^{m} \frac{1}{|Y_i|.|\bar{Y}_i|} \sum_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \xi_{ijk} \tag{1.21}$$

subject to: $\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k \geq 1 - \xi_{ijk}$,

$\xi_{ijk} \geq 0 \ \left(1 \le i \le m\ and\ (y_j, y_k) \in Y_i \times \bar{Y}_i\right)$

Here, $\Xi = \left\{ \xi_{ijk} \,\middle|\, 1 \le i \le m\ and\ (y_j, y_k) \in Y_i \times \bar{Y}_i \right\}$ denotes the set of slack variables. The aim of Eq. (1.21) comprises of two parts balanced by the parameter *C*. To be precise, the first part is equivalent to the *margin* of the learning system, while the second part is equivalent to the *ranking loss* of the learning system adopted in hinge form.

It is worth noting that Eq. (1.21) is the standard *quadratic programming* (QP) problem with convex optimization objective and linear constraints, which can be dealt with any off-the-shelf QP solver. Rank-SVM uses its dual form of kernel tricks to tackle with the nonlinear problems, solving the Eq. (1.21). Dual approach in multi-label classification is introduced in detail by A. Elisseeff and J. Weston [16].

The stacking-style procedure is adopted in Rank-SVM to set the thresholding function $t(\cdot)$,

i.e. $t(\boldsymbol{x}) = \langle \boldsymbol{w}^*, \boldsymbol{f}^*(\boldsymbol{x}) \rangle + b^*$ with $\boldsymbol{f}^*(\boldsymbol{x}) = \left(f(\boldsymbol{x}, y_1), \ldots, f(\boldsymbol{x}, y_q) \right)^T$,

and $f(\boldsymbol{x}, y_j) = \langle \boldsymbol{w_j}, \boldsymbol{x} \rangle + b_j$. For an unseen instance $\boldsymbol{x}$, the set of predicted labels corresponds to:

$$Y = \{y_j | \langle \boldsymbol{w_j}, \boldsymbol{x} \rangle + b_j > \langle \boldsymbol{w}^*, \boldsymbol{f}^*(\boldsymbol{x}) \rangle + b^*, 1 \leq j \leq q \} \quad (1.22)$$

Briefly, Rank-SVM adapts the idea of maximizing margin to multi-label data and takes the advantage of kernel method to deal with nonlinearity problem of multi-label classification.

The computational complexity of Rank-SVM is $\mathcal{O}(\mathcal{F}_{QP}(dp + mq^2, mq^2) + q^2(q + m))$ for training phase and $\mathcal{O}(dq)$ for testing phase. Figure 1.4 presents the pseudo-code of Rank-SVM.

$Y = \text{RankSVM}(\mathcal{D}, \mathcal{C}, \boldsymbol{x})$

1. Create the learning system $\mathcal{W} = \{(w_i, b_j) | 1 \leq j \leq q\}$ by solving the QP problem in Eq. (1.21):

$$\min_{(\mathcal{W},\Xi)} \sum_{j=1}^{q} \|\boldsymbol{w_j}\|^2 + C \sum_{i=1}^{m} \frac{1}{|Y_i||\bar{Y}_i|} \sum_{(y_j, y_k) \in Y_i \times \bar{Y}_i} \xi_{ijk}$$

2. with $\langle \boldsymbol{w_j} - \boldsymbol{w_k}, \boldsymbol{x_i} \rangle + b_j - b_k \geq 1 - \xi_{ijk}$,
3. $\xi_{ijk} \geq 0, (1 \leq i \leq m, (y_j, y_k) \in Y_i \times \bar{Y}_i)$
4. $Determine$ $(\boldsymbol{w}^*, b^*)$ for thresholding function by solving the linear square problem.
5. **return** *Y* according to this equation:

$$Y = \{y_j | \langle w_j, x \rangle + b_j > \langle w^*, f^*(\boldsymbol{x}) \rangle + b^*, 1 \leq j \leq q\}$$

Figure 1.4 Rank-SVM algorithm [11]

1.4. Evaluation metrics

Conventionally, the performance of traditional supervised learning system is evaluated with some common metrics such as accuracy, F-score, recall, area under the ROC curve (AUC), etc. However, to evaluate the performance of multi-label classification methods, we need to use different metrics than those in traditional single-label classification. Therefore, some specific metrics are proposed for evaluating the

performance of multi-label classification methods. M-L Zhang and Z-H Zhou [11] categorize them into two groups: *example-based* metrics and *label-based* metrics.

Let $S = \{(\boldsymbol{x_i}, Y_i) \mid 1 \leq i \leq p\}$ be the test set and $f(\cdot)$ be the learned multi-label classifier. Example-based metrics are employed by evaluating the performance on each test sample separately and then computing the *mean value* through the test set while label-based metrics are used by evaluating the performance on each class label separately, and then returning the *macro/micro-averaged value* across all class labels.

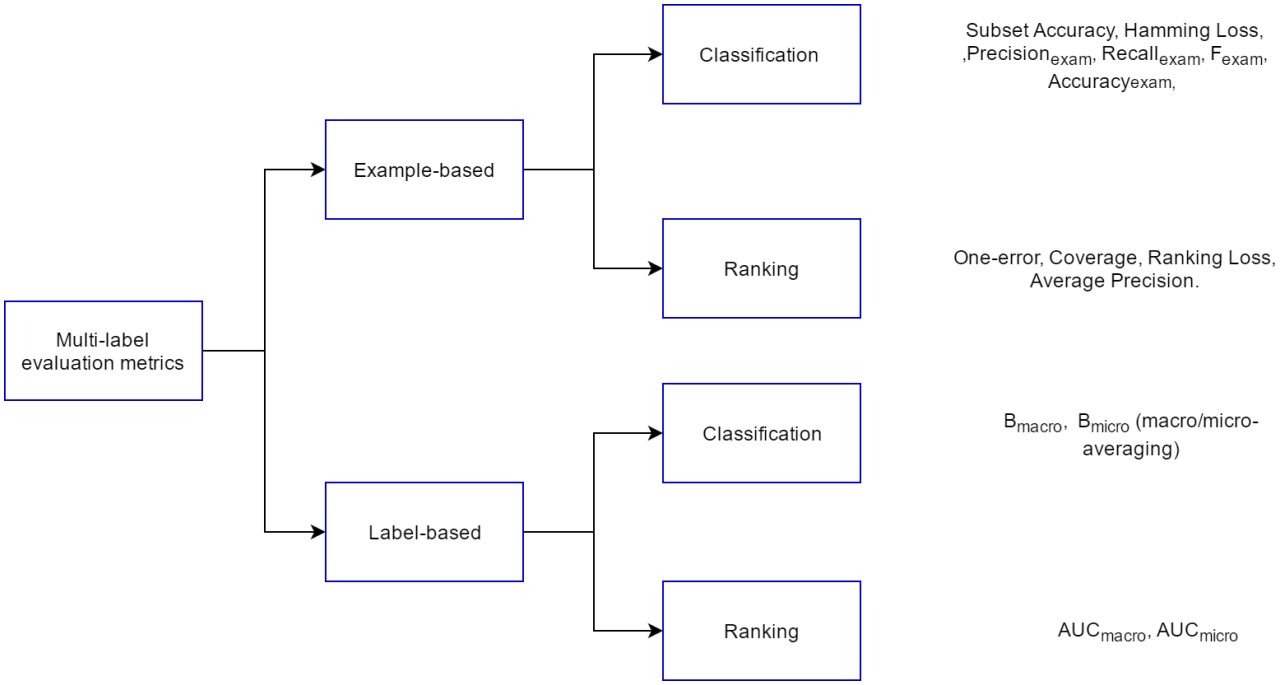


Figure 1.5 Summary of major multi-label evaluation metrics [11]

Figure 1.5 illustrates the summarization of the major evaluation metrics for multi-label classification, in which the thesis will use the *label-based metrics* to evaluate the proposed model. The details of label-based metrics will be introduced as below.

1.4.1. Label-based metrics

According to F.Sebastiani [17], 4 basic terms for performance on binary classification for the *j*-th class label y_i can be defined based on $f(\cdot)$:

$$TP_j = |\{\boldsymbol{x}_i | \, y_i \in Y_i \ \wedge y_i \in h(\boldsymbol{x}_i), 1 \leq i \leq p\}| \quad (1.23)$$

$$FP_j = |\{\boldsymbol{x}_i | \, y_i \notin Y_i \ \wedge y_i \in h(\boldsymbol{x}_i), 1 \leq i \leq p\}| \quad (1.24)$$

$$TN_j = |\{\boldsymbol{x}_i | \, y_i \notin Y_i \ \wedge y_i \notin h(\boldsymbol{x}_i), 1 \leq i \leq p\}| \quad (1.25)$$

$$FN_j = |\{\boldsymbol{x}_i | \, y_i \in Y_i \ \wedge y_i \notin h(\boldsymbol{x}_i), 1 \leq i \leq p\}| \quad (1.26)$$

Where, TP_j, FP_j, TN_j, FN_j stand for the number of *true positive*, *false positive*, *true negative*,
and *false negative* test examples respectively regarding to y_j.

It can be seen that

$$TP_j \ + \ FP_j \ + \ TN_j, + FN_j \ = \ p. \quad (1.27)$$

Table 1.2 An illustration of four basic quantities

		OBSERVED	
		Presence	Absence
PREDICTED	Presence	TP	FN
	Absence	FP	TN

Let $B(TP_j, FP_j, TN_j, FN_j)$ denote binary classification metrics where it may usually be $B \in \{Accuracy, Precision, Recall, F^{\beta}\}$. There are two modes to obtain the label-based metrics for multi label classification:

- *Macro-averaging:*

$$B_{macro}(f) = \frac{1}{q}\sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j) \quad (1.28)$$

- *Micro-averaging:*

$$B_{micro}(f) = B\left(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN,\right) \quad (1.29)$$

Here, *Accuracy*, *Precision*, *Recall*, and F^{β} are well-defined as follows:

$$Accuracy\left(TP_j, FP_j, TN_j, FN_j\right) = \frac{TP_j + TN_j}{TP_j + FP_j + TN_j + FN_j} \tag{1.30}$$

$$Precision\left(TP_j, FP_j, TN_j, FN_j\right) = \frac{TP_j}{TP_j + FP_j} \tag{1.31}$$

$$Recall\left(TP_j, FP_j, TN_j, FN_j\right) = \frac{TP_j}{TP_j + FN_j} \tag{1.32}$$

$$F^{\beta}\left(TP_j, FP_j, TN_j, FN_j\right) = \frac{(1+\beta^2)TP_j}{(1+\beta^2)TP_j + FP_j + \beta^2 FN_j} \tag{1.33}$$

Obviously, the following equations can be derived:

$$\begin{aligned} &Accuracy_{macro}(f) = Accuracy_{micro}(f), \text{ and} \\ &Accuracy_{micro}(f) + hloss(f) = 1. \end{aligned} \tag{1.33}$$

For the above label-based metrics for multi-label classification, the performance of models is better if the metric value is higher, with the optimal value is 1.

1.5. Motivation

In reality, multi-label classification has a wide range of application domains such as text classification, multi-media mining, medical prediction, social mining, etc. It is obvious that single-label classification is much simpler than multi-label classification problems. The key challenge of multi-label learning is the size of label space, i.e. the space complexity of labels grows exponentially as the number of label increases. The label space of a set including q unique single labels is 2^q, for example, it surpasses to more than one million when the number of labels is equal to 20 (2^{20} label combinations). It requires some improvements and solutions dealing with large-scale problems computational cost of algorithms, the imbalance of labels and the multi-dimensionality of data, etc. Therefore, multi-label classification has attracted a huge attention from research communities so far.

The most outstanding point of problem transformation methods is that they inherit some existing machine learning algorithms. Thereby, those types of methods can take the

advantage of parallel computational implementation to cut the cost of time. However, problem transformation methods ignore the existence of label correlations, which decreases the performance of the classification system.

It is also worth noting that most of the existing multi-label classification methods concentrate on the improvement of supervised learning which relies much on the labeled data. To prepare the data for training supervised learning models, we need to label manually samples in the collection of data. That labor-intensive task usually is time-consuming because each sample might be assigned with multiple labels rather than a single label. Meanwhile, the unlabeled data proves its availability and abundance across the tremendous online resources like social media and websites. It is natural to improve the performance of the learning system using unlabeled data.

Along its dynamic line of research on multi-label classification and together with our research group, this thesis introduces an algorithm adaptation method for MLC, which harnesses the correlations among labels and the availability of unlabeled data.

1.6. Contributions

The purpose of this thesis is to propose a semi-supervised multi-label classification algorithm which allows to leverage both the label correlations and unlabeled data to enhance the performance of model. First, the feature enrichment step will be performed by adding hidden topic feature. Second, an idea from LIFT is implemented to exploit the specific features of eminent labels. Two techniques to divide training dataset into smaller subset are inherited from former works. Labeled data and unlabeled will be consumed in the clustering step using the semi-supervised method from TESC. After obtaining a classification model, we will start classifying an unseen instance by adopting 1-Nearest-Neighbor (1NN) a set of predefined labels.

The open source of implementation for this method were developed in both Java and Python.The API is provided to train models and classify new unseen instances.

Regarding scientific values, the proposed methods for multi-label semi-supervised learning were accepted to publish as a paper on the 9th Asian Conference on Intelligent

Information and Database Systems[1] (ACIIDS 2017), and another paper on Journal of Information and Telecommunication[2] as below. In other words, the results of this thesis research has contributed to the diversity of multi-label classification methods of research communities.

1. Thi-Ngan Pham, Van-Quang Nguyen, Duc-Trong Dinh, Tri-Thanh Nguyen, Quang-Thuy Ha (2017). MASS: *a Semi-supervised Multi-label Classification Algorithm With specific Features.* ACIIDS 2017. Studies in Computational Intelligence, Volume 710 (Advanced Topics in Intelligent Information and Database Systems): 37-47. (Scopus).

2. Thi-Ngan Pham, Van-Quang Nguyen, Van-Hien Tran, Tri-Thanh Nguyen, and Quang-Thuy Ha. *A semi-supervised multi-label classification framework with feature reduction and enrichment.* Journal of Information and Telecommunication (Taylor & Francis), in press.

1.7. Thesis Structure

The rest of this thesis is organized as follows.

Chapter 2 provides some basic background on semi-supervised algorithm in multi-label classification. This chapter will introduce the overview of semi-supervised classification for multi-label data, the label space distribution definition and techniques, and a semi-supervised classification algorithm (TESC) which will be adopted in the proposed method.

In chapter 3, the problem is formulated formally and an algorithm adaptation method and framework will be introduced to resolve the multi-label classification problem.

Experiments on Vietnamese text will be presented in chapter 4 to examine the contribution of unlabeled data feature enrichment, and other aspects to the proposed model. Hereby, experimental results are evaluated and discussed as well.

The last part will be dedicated to the conclusions of the thesis and the following-up work in the future to improve the performance of model.

Chapter 2

SEMI-SUPERVISED MULTI-LABEL CLASSIFICATION METHODS

2.1. Semi-supervised multi-label classification

Traditional classification methods consume only labeled data for training models. However, labeled data is difficult to obtain, as it requires manual effort which is expensive and time-consuming. Most previous studies on multi-label classification concentrated on supervised learning.

Meanwhile it is quite easy to collect unlabeled data. Semi-supervised learning addresses this problem by using an amount of labeled data together with unlabeled data to train models.

Let $D = D^L \cup D^U$ be a data collection, where D^L and D^U are the collections of labeled and unlabeled documents respectively. Also, let $\overline{D} = \{\overline{D}^L, \overline{D}^U\}$ be the subset of D being used for training the classification function where $\overline{D}^L \subseteq D^L$ and $\overline{D}^U \subseteq D^U$. The task of semi-supervised MLC is to build the classification function from the data $\overline{D} = \{\overline{D}^L, \overline{D}^U\}$ as Eq. (2.1).

$$f: \overline{D}^L \cup \overline{D}^U \to 2^L \quad f(d^u) \subseteq L \tag{2.1}$$

So far, current semi-supervised methods on multi-label classification adopts the transductive approach. Those methods assume that the samples in the input data which have high score of similarity will be likely to be overlapping in the output space. Xiangnan et al [18] proposed a model of transductive multi-label learning by using label

set propagation TRAM to assign a set of multiple labels to each instance. Firstly, TRAM formulates the transductive multi-label learning as an optimization problem of estimating label concept compositions. Then it derives a closed-form solution to this optimization problem and develops an algorithm to assign label sets to unlabeled instances. Zheng-Jun Zha et al [19] introduces a multi-label classification framework using the graph of label importance. The novel point of this framework is to harness both the correlations among the labels and the consistency of the labels over the graph simultaneously. Furthermore, Y. Liu et al [20] proposed semi-supervised learning methods using constrained non-negative matrix factorization. Firstly, they calculate two kind of scores which are the similarities among the features of samples, and the class label of samples. Then they optimize the class label assignment for unlabeled data that minimizes the difference between two kinds of similarities. The optimization problem here is considered as constrained non-negative Matrix Factorization problem. A novel method is introduced to solve this optimization problem. This method takes the advantage of both unlabeled data and the label correlations simultaneously.

2.2. Label space division in multi-label classification

Let L represent for a set of class labels in multi-label classification problem. Therefore, the number of label combinations in multi-label learning will be exponential, 2^L. In problem transformation methods, there are two basic approaches, namely binary relevance and label powerset.

Binary Relevance assumes that the labels are independent and transforms multi-label problems into $|L|$ single-label problems, each of which learns a single-label single class function $h_i: X \to \{0,1\}$ for each label $L_i \in L$. The result for multi-label classification for an unseen instance $\boldsymbol{x}$ will be the union of the ouput of each classifier:

$$h(\boldsymbol{x}) = \{L_i \in L : h_i(\boldsymbol{x}) = 1\}. \tag{2.2}$$

Label powerset (LP) handles multi-label classification problems with different assumption, in which the label space is *non-divisible label-wise*. LP formulates a bijection $lp: 2^L \rightarrow C$ between each sebset of labels L_i and a new class C_i. Label powerset then learns a single-label multi-class function $h: X \rightarrow C$, and converts its result back to multi-label using an inverse function $lp^{-1}\big(h(\boldsymbol{x})\big)$. It is supposed that each combination of labels can be a unique class, which leads to an exponential boom. Therefore, label space division techniques were proposed to avoid this exponential burgeon

2.2.1. RaKEL

Tsoumakas et al. divide the label space into small subspaces which are called random k-label sets. Then, it adopts label powerset for each subspaces. The notion of RAkEL is that it is easier to perform label powerset on a small label space rather a large label space. There are two variants of RAkEL: *RAkELd* which divide the label space into k random subspaces which disjoins from each other, and the second is *RAkELo* which is a sampling method allows the intersection of label subspaces.

RAkELd conducts dividing randomly the label space L into k subspaces $L_j, 1 \leq j \leq k$. Adopting the idea of Label powerset, *RAkELd* then learns a single-label multi-class for each subset L_j f_j: $X \rightarrow L_j$. To obtain the final result of multi-label classification on an unseen instance $\boldsymbol{x}$, *RAkELd* performs senquentially each f_j for each subspace L_j, and combines all the outputs together. The *RAkELd* classifier h can be formulated as $h(\boldsymbol{x}) = \bigcup_{\boldsymbol{j=0}}^{\boldsymbol{k}} f_j(\boldsymbol{x})$. All the subspaces of the set L are equally probable.

2.2.2. Data-driven approach

While RAkEL divides label space into k subsets randomly, Piotr Szymanski et al. [21] propose a data-driven approach to label space division. Firstly, they establish an undirected label co-occurrence graph G with the label set L and the vertex set from training data as below:

$$E = \{\{l_1, l_2\}: \exists(\boldsymbol{x}, Y) \in \boldsymbol{D_{train}}\ (l_1 \in Y\ and\ l_2 \in Y)\} \quad (2.3)$$

The edges are formulated from all pairs of labels that belong to at least an input instance $\boldsymbol{x}$ with the set of labels Y in the training set. We can also construct the weighted graph by introducing a new function $w: L \rightarrow \mathbb{N}$:

$$w(l_1, l_2) = the\ number\ of\ input\ instance\ \boldsymbol{x}\ that\ contain\ both\ labels =$$
$$= |\{\boldsymbol{x}: (\boldsymbol{x}, Y) \in D_{train}, l_1 \in Y\ and\ l_2 \in Y\}|$$

Using such a graph G, weighted or unweighted, they perform label space division by adopting one of five community detection methods to separate the vertex set of the graph, which is equivalent to the label set. Five community detection methods used include infomap, fast greedy, walktrap, leading eigenvector approximations, and label propagation algorithm. Each of them were investigated and evaluated concretely in [21].

After adopting community detection approaches to separate label set into label subspaces L_i, a new training set D_i is selected from the original training data which contains only the labels of the subspace L_i. For each community L_i, a function h_i is learned on training set D_i.

To predict the set of associated labels for an unseen instance $\boldsymbol{x}$, we perform classification on all subspaces detected in the training phase and obtaining the combination of predicted labels: $h(\boldsymbol{x}) = \bigcup_{\boldsymbol{j=0}}^{\boldsymbol{k}} h_j(\boldsymbol{x})$.

Experimental results show that data-driven approaches to label space division considerably outperform the RAkELd which randomly partitions the label space.

2.3. TESC Algorithm introduction

In 2015, Wen Zhang et al [2] introduced TESC standing for TExt classification using Semi-supervised Clustering. In this work, TESC is a *semi-supervised clustering* method

for *single label classification* in which each instance can be associated with only a single label. TESC consists of 2 phases: the first phase is a clustering process to find the set of partitions based on the labeled and unlabeled data, the second is a predicting process to assign a label for a new unseen text based on the trained clusters. In the clustering process, TESC uses

---

Input:

$\overline{D}$: a collection of labeled and unlabeled texts;

Output:

C: a collection of labeled text clusters.

Procedure:

(1) Initialization

1. **for** each $d_i \in \overline{D}$ **do**:
2. Construct a cluster candidate C_i using each d_i and label C_i with l_{d_i};
3. Set C_i as unidentified and the centroid of C_i as d_i;
4. Add C_i to cluster candidate set S_c;
5. **end for**

(2) Clustering

1. **Loop while** the number of unidentified and labeled cluster candidates in S_C is larger than 1;
2. Find a cluster candidate pair (C_i, C_j) in S_C, which has the smallest distance between centroids among all the possible identified cluster candidate pairs in S_C;
3. **if** C_i and C_j are all labeled cluster candidates and have different cluster labels **do**:
4. Then set C_i and C_j as identified;
5. **else**:
6. Merge C_i and C_j into a new cluster candidate C_k;
7. Remove C_i and C_j from S_C;
8. Add C_k to S_C;
9. **end if**

(3) Output

1. Remove the cluster candidates whose size is smaller than 3 from S_C;
2. **return** each cluster candidate C_i in S_c to C;

Figure 2.1 The pseudo-code of clustering processing in TESC [2]

labeled texts to capture the silhouettes of text clusters, next the unlabeled texts are added to the corresponding clusters to adjust their centroid. These clusters are used for building the model in the classification phase. In the predicting process, for a new unlabeled text, the model returns a predicted label which is the label of the nearest cluster to it.

Assume that we have a document collection $D = \{D^U, D^L\}$, where D^U and D^L are the collection of labeled data and unlabeled data respectively. TESC takes a subset of D, $\overline{D} = \{\overline{D}^L, \overline{D}^U\}$ to find a partition $C = \{C_1, \dots, C_m\}$, where $\overline{D}^L \subseteq D^L, \overline{D}^U \subseteq D^U$ and $C_i = \left\{d_1^{(i)}, \dots, d_{|C_i|}^{(i)}\right\}$ $(1 \le i \le m)$, $\bigcup_{1 \le i \le m} C_i = \overline{D}$, and $C_i \cap C_j = \emptyset$ $(1 \le i \ne j \le m)$.

Figure 2.1 presents pseudo-code of clustering processing in TESC. After this process, the derived cluster set *C* is regarded as the model of the classification function. In the predicting processing, the predicted label of a new document is assigned as the label of its nearest cluster, i.e., given an unseen example, the label of its nearest cluster $C_j \in C$ is used to assign to it. TESC adopts the 1-nearest neighbor (1-NN) technique to search the centroid of predict the label for a new unlabeled text d^u in D^U:

$$l(d^u) = l_{C_j}, \quad C_j = \ argmin_{C_p} \left\| d^u - c_p \right\| \tag{2.4}$$

Here c_p is the centroid of text cluster C_p. TESC is innovative and different from the existing semi-supervised learning techniques. The experimental results [2] indicate that TESC surpass the state-of-the-art algorithm including SVM, BPNN and the DKS method, is comparable with NBEM with better scalability.

2.4. Adapting TESC for multi-label classification

One can extend TESC algorithm to become adaptive with multi-label data. We can make a manifold assumption by assigning a set of labels with a temporary class. The idea of label space division is adopted to separate training dataset into subspaces, in which we perform recursively clustering on each subspace. In transformation problem methods, each classifier will be responsible for a subset of labels. The result on each subspace of

labels will then be combined together to get the overall classification result. However, instead of using those techniques, we propose a new technique to handle with the divided subspaces, in which only one classifier is built, and the result of classifier on each subspace is the final classification result. Besides utilizing the semi-supervised algorithm for single label data (TESC), the thesis also exploits the idea of RAkEL and the data-driven techniques to divide training dataset into subspaces corresponding to their label set.

The technique to adapt TESC to handle multi-label data will be presented more elaborately in chapter 3.

Chapter 3

THE PROPOSED FRAMEWORK

In the previous chapter, the thesis summarizes the overview of semi-supervised learning in multi-label classification and the algorithm TESC as well as the strategies of label space division in multi-label classification. In this chapter, the thesis will introduce a new method which adapts a semi-supervised single label classification algorithm (TESC) to multi-label data using a technique with specific label features. The framework to evaluate the method will also be presented in the last part of this chapter.

The thesis proposes a framework for multi-label classification including: a) a stage of enriching features by using the hidden topic model (LDA) [22] [23], to exploit information of semantic meaning of text representation; c) a novel *semi-supervised Multi-label ClASSification* (called MASS), which can exploit both *unlabeled data* and *specific features* to enhance the performance. By determining the specific label λ in the predefined collection, the dataset is then divided into three different subsets, and the semi-supervised clustering is applied to each subset to extract features specific to each label or label set. The method of extracting specific features is an extension of the LIFT algorithm proposed by Min-Ling Zhang and Lei Wu [1]. In addition, MASS employs the semi-supervised clustering algorithm to exploit both labeled and unlabeled data together at the same time as mentioned in TESC of Zhang et al [2].

3.1. Problem formulation

3.1.1. Semi-supervised multi-label classification

Let $D = D^L \cup D^U$ be a document collection, where D^L and D^U are the collections of labeled and unlabeled documents respectively. Also, let $\overline{D} = \{\overline{D}^L, \overline{D}^U\}$ be the subset of D being used for training the classification function where $\overline{D}^L \subseteq D^L$ and $\overline{D}^U \subseteq D^U$. The task of semi-supervised MLC is to build the classification function from the data $\overline{D} = \{\overline{D}^L, \overline{D}^U\}$ as Eq. (3.2). In our proposed semi-supervised MLC algorithm – MASS, the goal in building the function f is to find a partition C derived from $\overline{D}$, such that $C = \{C_1, \dots, C_m\}$, where $C_i = \left\{d_1^{(i)}, \dots, d_{|C_i|}^{(i)}\right\}$ $(1 \le i \le m)$, $\bigcup_{1 \le i \le m} C_i = \overline{D}$, and $C_i \cap C_j = \emptyset$ $(1 \le i \ne j \le m)$. All documents in cluster C_i are given the same non-empty label set (called cluster-label) l_{C_i}.

$$f: \overline{D}^L \cup \overline{D}^U \to 2^L \quad f(d^u) \subseteq L \tag{3.1}$$

In the traditional unsupervised clustering methods, the number of clusters is often predefined and chosen manually. However, in our algorithm, the number of clusters (m) is automatically identified based on the label set in combination with the labeled and unlabeled dataset.

After we have obtained the partition C, given a new unlabeled document $d^u \in D^U$, f employs the 1-nearest neighbor (1-NN) to get the nearest cluster $C_j = \underset{C_p}{arg\,min}\, dis\left(d^u, c_p\right)$, where c_p is the centroid of the cluster C_p and $dis(.)$ is the distance between two documents, then the cluster label of C_j is assigned to d^u, i.e., $l(d^u) = l_{C_j}$. Our contribution is to harness both labeled and unlabeled documents to find the partition C to form the classification model f, which could predict the associated label(s) of unlabeled documents in D^U.

Before introducing a proposed algorithm, the thesis will present LIFT algorithm as below. Our proposed algorithm adopts partly the idea from LIFT to adapt TESC to multi-label data.

3.2. LIFT Algorithm

In problem transformation approaches, multi-label classification problem is transformed into q single label classifiers corresponding to a set of q predefined labels.

Each classifier is usually obtained by using the *identical* set of *features* from the multi-label data. This kind of approaches is applied in a wide range of multi-label learning algorithms. However, this method might not be optimal because each label is supposed to have specific characteristics of its own.

Zhang [1] proposes multi-label learning with Label specific Features (LIF) for enhancing the performance of supervised multi-label classification using label-specific features. Concretely, LIFT, at the first step, aims at figuring out features with label-specific characteristics so as to provide appropriately discriminative information to facilitate its learning as well as classification. For each class label $l_k \in L$, the set of positive and negative training instances are founded as the set of the training instances with and without label l_k, respectively. After that, clustering analysis is performed on its positive and negative sets to extract the features specific to l_k . In the second step, q binary classifiers, one for each class label l_k using l_k - specific features, are used to check whether a new instance has the label l_k . The approach in LIFT is a supervised method, in which the input is a labeled dataset for the training process and the output is a classification model including the family of q classifiers corresponding to q labels. Given an unseen example, its associated label set is predicted by going through q classifiers to get prediction for each label.

LIFT Algorithm can be improved by some ways: change the configuration of determining the number of clusters on negative and positive data; k-means clustering techniques can be replaced by other clustering methods. The pseudo code of LIFT is presented in Figure 3.1.

Inheriting the flexibility of LIFT algorithm, the thesis proposes a new multi-label classification methods with some improvements: (1) determining a set of labels characterizing for each label or label set in which a semi-supervised clustering technique is introduced to replace the traditional k-means clustering method; (2) dividing a training labeled dataset into subsets by identifying the prominent label based on a greedy technique.

$$Y = \text{LIFT}(\mathcal{D}, r, \mathcal{L}, u)$$

Input:

$\mathcal{D}$: multi-label training set $\{(\boldsymbol{x_i}, Y_i) | \ 1 \leq i \leq m\}$

$(\boldsymbol{x_i} \in \boldsymbol{\mathcal{X}}, Y_i \subseteq \mathcal{Y}, \boldsymbol{\mathcal{X}} = \mathbb{R}^{\boldsymbol{d}}, \mathcal{Y} = \{l_1, l_2, \dots, l_q\})$

r: ratio parameter.

$\mathcal{L}$: binary learner for classifier induction

$\boldsymbol{u}$: unseen example ($\boldsymbol{u} \in \boldsymbol{\mathcal{X}}$)

Output:

Y: predicted label set for $\boldsymbol{u}$ ($Y \subseteq \mathcal{Y}$)

Procedure:

1. **for** k = 1 **to** q **do:**
2. Form $P_k \ and \ N_k$ based on $\mathcal{D}$ as:

$$P_k = \{\boldsymbol{x_i} | (\boldsymbol{x_i}, Y_i) \in \mathcal{D}, l_k \in Y_i\}$$
$$N_k = \{\boldsymbol{x_i} | (\boldsymbol{x_i}, Y_i) \in \mathcal{D}, l_k \notin Y_i\}$$

3. Perform *k*-means clustering on $P_k \ and \ N_k$, each with m_k clusters as defined:

$$m_k = \lceil r.min(|P_k|, |N_k|) \rceil$$

where $P_k and \ N_k$ consists of training examples with and without l_k respectively.

4. Create the mapping $\phi_k \ for \ l_k$ according to:

$$\phi_k(x) = \left[d\left(x, p_1^k\right), \dots, d\left(x, p_{m_k}^k\right), d\left(x, n_1^k\right), \dots, d\left(x, n_{m_k}^k\right)\right]$$

5. **end for**
6. **for** k = 1 **to** q **do**:
7. Form a binary learner $\mathcal{B}_k$ for label l_k:

$$\mathcal{B}_k = \{(\phi_k(x_i), Y_i(k)) | (x_i, Y_i) \in \mathcal{D}\}$$

where $Y_i(k) = +1$ if $l_k \in Y_i$, otherwise $Y_i(k) = -1$

8. Induce g_k by invoking $\mathcal{L}$ on $\mathcal{B}_k$, i.e. $g_k \leftarrow \mathcal{L}(\mathcal{B}_k)$;
9. **end for**
10. **return** $Y = \left\{l_k \mid g_k\left(\phi_k(u)\right) > 0, 1 \leq k \leq q\right\}$

Figure 3.1. The pseudo-code of LIFT algorithm [1].

3.3. The proposed method

The thesis introduces a semi-supervised method for multi-label classification that builds the specific features for each label and label set based on the idea proposed in LIFT

with several improvements. In LIFT, the features specific to each label were built in the same manner.

In our model, the first step is to find the prominent labels in a cluster following the some strategies in which the ideas of RaKEL and the label co-occurrence graph (data driven) are adopted flexibly to select a label named λ. The role of the label λ plays as an indicator to divide the training dataset into smaller subsets based on the label attribute. The details of deployment of RaKEL and the label co-occurrence graph in our algorithm will be introduced later.

The next step in LIFT is to extract features specific to each label by the k-means clustering algorithm on its positive and negative samples. Our model makes some important changes in this stage. As described in chapter 2, we divide the document collection into three different subsets: 1) documents with expansion of the only prominent label λ, 2) documents with a set of label including λ, and 3) documents without λ. After that, we perform semi-clustering analysis on these three subsets to get a partition derived from collection of unlabeled and labeled documents. The semi-supervised clustering technique in TESC is applied in our model to consume unlabeled documents, i.e., an unlabeled document is added to its nearest cluster, and its label set is the same as the cluster label. Finally, the partition derived from both labeled and unlabeled documents is used as the classification model.

No additional classification algorithm is used in our approach. This is different from LIFT, which uses q (i.e., the cardinality of the label set) binary classifiers with label-specific features in the classification phase. Here is the main difference between algorithm adaptation methods and problem transformation methods.

The proposed algorithm consists of two phases: one is the learning phase, which uses clustering to identify the components (i.e., clusters) from both labeled and unlabeled texts based on the label λ; the other is predicting phase, which identifies the nearest text cluster to assign labels for an unseen text in D^U.

In the learning phase, we use the semi-supervised clustering method in [2] to take advantages of the TESC algorithm to get partition derived from the training data collection $\overline{D}$.

Figure 3.2 Pseudo-code of clustering process.

$C \leftarrow MASSLearn\ (\overline{D}, L_1, L_2)$

Inputs:

$\overline{D}$: a collection of labeled and unlabeled texts.

L_1: a set of available default labels for all the instances in $\overline{D}^L$

L_2: a set of possible labels which could be assigned for some instances in $\overline{D}^L$

Output:

C: a collection of labeled text clusters

Procedure:

1. Let λ be the label in L_2 with the maximum occurrences in $\overline{D}^L$
2. Create a new label set L^* including three assistant labels:
   $\lambda_1 \leftarrow L_1 \cup \{\lambda\}$,
   $\lambda_2 \leftarrow L_1 \cup \{\lambda\} \cup \{\beta | \beta \in L_2\}$, and
   $\lambda_3 \leftarrow L_2 \backslash \{\lambda\}$ //label set without λ
   $\overline{D}' \leftarrow Annotate(\overline{D}, \{\lambda_1, \lambda_2, \lambda_3\})$; $C^* \leftarrow TESC(\overline{D}', L^*)$
3. Divide $\overline{D}$ into three subsets of labeled and unlabeled texts regarding to the partition C^*:
   a. $\overline{D}_1$: the subset of unlabeled texts, and labeled texts assigned the label λ_1
   b. $\overline{D}_2$: the subset of unlabeled texts, and labeled texts assigned the label λ_2
   c. $\overline{D}_3$: the subset of unlabeled texts, and labeled texts assigned the label λ_3
4. On $\overline{D}_1$:
   $C \leftarrow C \cup TESC(\overline{D}_1, L_1 \cup \{\lambda\})$
5. On $\overline{D}_2$:
   a. **if** all instances in $\overline{D}_2$ have the same label set L_3 **then:**
      $C \leftarrow C \cup TESC(\overline{D}_2, L_3)$
   b. else:
      // Instances in $\overline{D}_2$ do not have an identical set of labels.
      // Move λ from L_2 to L_1
      $C \leftarrow C \cup MASSLearn(\overline{D}_2, L_1 \cup \{\lambda\}, L_2 \backslash \{\lambda\})$
6. On $\overline{D}_3$:
   a. **if** all instances in $\overline{D}_3$ have the same label set L_4 **then:**
      $C \leftarrow C \cup TESC(\overline{D}_3, L_4)$
   b. else:
      // Instances in $\overline{D}_3$ do not have an identical set of labels.
      // Delete λ from L_2
      $C \leftarrow C \cup MASSLearn(\overline{D}_3, L_1, L_2 \backslash \{\lambda\})$
7. **return** C.

The training procedure is called *MASSLearn*(.) whose pseudo-code is shown in Figure 3.2.

In order to find the partition C (i.e., the model of our classification algorithm), we first initialize $C = \{\}$, then call $MASSLearn(\overline{D}, \{\}, L, C)$. The result set of text clusters *C* is regarded as components and used to predict labels of unlabeled texts in the predicting phase as shown in **Figure 3.3**.

$L^U \leftarrow$ MASS*Classifier*(C, D^U)

Input:

C: a collection of labeled text clusters $C = \{C_1, \dots, C_m\}$

D^U: a collection of unlabeled texts

Output:

L^U: a collection of possible labels to assign the unlabeled texts in D^U

Procedure:

1. **for** each $d^u \in D^U$ **do:**
2. $C_{temp} \leftarrow C_0$ // C_0 is the first cluster in C
3. $l_{d^u} \leftarrow l_{C_{temp}}$ //$l_{C_{temp}}$ is the label set of cluster C_{temp}
4. **for** each $C_j \in C$ **do:**
5. $Dis(d^u, C_j) \leftarrow \|d^u - C_j\|$ //we use Euclidean distance here
6. **If** $Dis(d^u, C_{temp}) > Dis(d^u, C_j)$ **then:**
7. $C_{temp} \leftarrow C_j$
8. **end if**
9. **end for**
10. $l_{d^u} \leftarrow l_{C_{temp}}$
11. Add $l_{C_{temp}}$ to L^U
12. **end for**
13. **return** L^U.

Figure 3.3 Pseudo code of the predicting processing.

In the predicting process, the input includes unlabeled texts needed labeling. The output is the collection of labels corresponding to each text. We compute the distances from unlabeled text to the centroids of all clusters to find out the nearest centroid. Then the unlabeled text will be assigned the label set of its nearest cluster.

3.3.1. The strategies for selecting the label λ

As described briefly in chapter 2 and above, we use the label λ as an indicator to separate dataset into three subsets. The training samples belonging to each subset satisfy some label-constrained conditions which are described in Figure 3.2. Therefore, the strategies of selecting λ must be considered careful because it leads to the general performance of the algorithm. Although taking the idea of RaKEL and the label co-occurrence graph, there are some different points of how we utilize λ to divide subset. At each time of calling $MASSLearn\ (\overline{D}, L_1, L_2)$, we need to find a label λ in the label set L_2.

3.3.1.1. The trivial technique of choosing λ

To be objective, we select the label λ randomly as the idea of RaKEL. In RaKEL, the label set will be divided into k equal subspaces of labels, in which k is a tuning parameter. However, together with L_1 and λ, we divide label set into three subsets of labels and then using an assistant labels $(\lambda_1, \lambda_2, \lambda_3)$ to assign the training samples. In this technique, every label in label set L_2 shares the equal chance of being selecting without any bias.

Figure 3.4 Trivial technique of selecting λ

---

1. Choose the label $\lambda \in L_2$ randomly
2. Create a new label set L^* including three assistant labels:
   a. $\lambda_1 \leftarrow L_1 \cup \{\lambda\}$,
   b. $\lambda_2 \leftarrow L_1 \cup \{\lambda\} \cup \{\beta | \beta \in L_2\}$, and
   c. $\lambda_3 \leftarrow L_2 \backslash \{\lambda\}$ //label set without λ

   $\overline{D}' \leftarrow Annotate(\overline{D}, \{\lambda_1, \lambda_2, \lambda_3\})$; $C^* \leftarrow TESC(\overline{D}', L^*)$
3. Divide $\overline{D}$ into three subsets of labeled and unlabeled texts regarding to the partition C^*:
   a. $\overline{D}_1$: the subset of unlabeled texts, and labeled texts assigned the label λ_1
   b. $\overline{D}_2$: the subset of unlabeled texts, and labeled texts assigned the label λ_2
   c. $\overline{D}_3$: the subset of unlabeled texts, and labeled texts assigned the label λ_3

3.3.1.2. The data driven technique of choosing λ

While the first strategy considers the role of each label in L_2 equal in the context of label space division, this technique focuses on the exploitation of label correlations which lead to better and faster division of label set. The idea here is adopted from *data-driven approach to label space division* proposed by Piotr Szymanski et al [21] which is introduced briefly in chapter 2. However, there also some points which is different from their approach.

Firstly, we establish a directed label co-occurrence graph G among the labels in $L_2 = \{l_{21}, l_{22}, \dots, l_{2n}\}$ and the labels in $L_1 = \{l_{11}, l_{12}, \dots, l_{1m}\}$ and the vertex set from training data as below:

$$E = \left\{\{l_i, l_j\}: \exists(\boldsymbol{x}, Y) \in \boldsymbol{D_{train}} \left(l_i \in L_2 \; and \; l_j \in L_1\right)\right\} \tag{3.2}$$

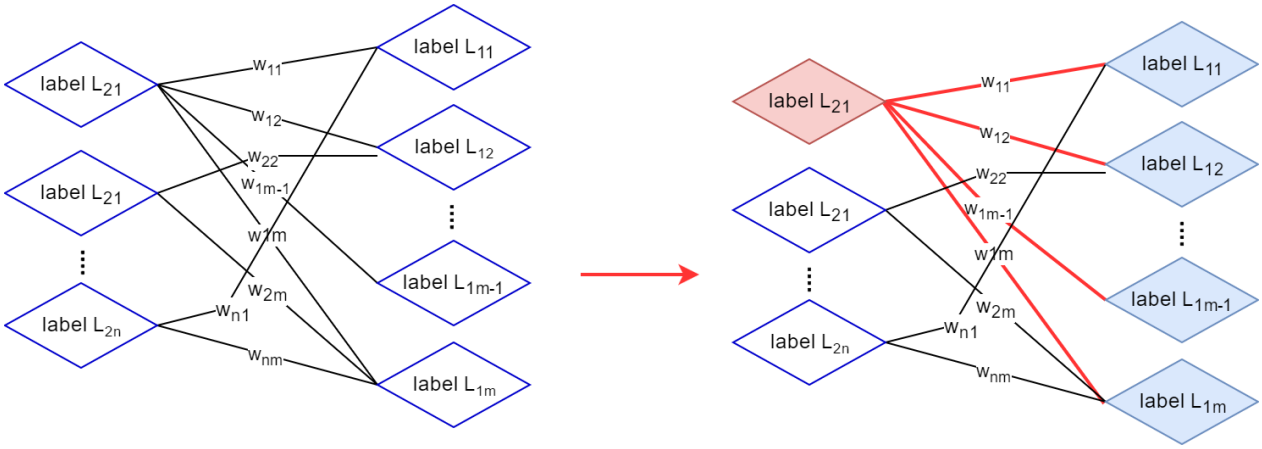


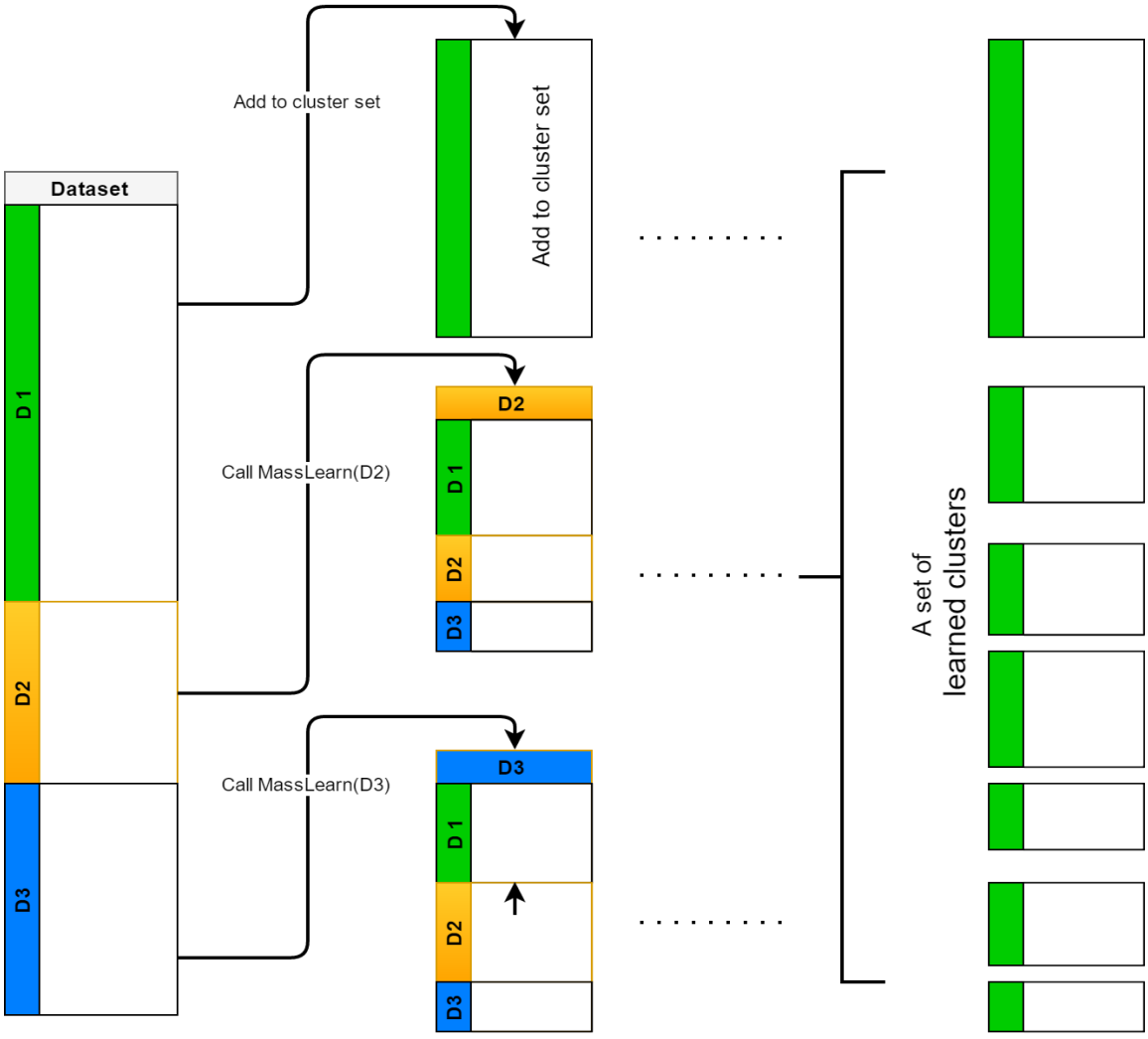
Figure 3.5 The data driven approach to select $\boldsymbol{\lambda}$

The edges are formulated from all pairs of labels that belong to at least an input instance $\boldsymbol{x}$ with the set of labels Y in the training set. We can also construct the weighted graph by introducing a new function $w: L \rightarrow \mathbb{N}$:

$$w(l_i, l_j) = the\ number\ of\ input\ instance\ \boldsymbol{x}\ that\ contain\ both\ labels =$$
$$= \left|\{\boldsymbol{x}: (\boldsymbol{x}, Y) \in D_{train}, l_i \in L_2 \; and \; l_j \in L_1\}\right|$$

We perform constructing the weighted graph for each label in L2 to all labels in the set L1. Then, we find the label that construct the with the L1 the most highly-traited community. Using this approach, we exploit the correlations among the labels to better divide label space into subspaces. The performance comparison of this approach against the former trivial technique will be illustrated as below while the experimental results of performance will be later re-confirmed in chapter 4.

Figure 3.6. Visualization of the learning phase.



When we select λ with the criterion which the label must the most high-traited community with labels in L_1, it is highly probable to obtain the subset D_1 of larger size

than the first random technique after performing TESC clustering. As described in the algorithm, this subset D_1 will be added to the set of learned clusters. Therefore, we only need to run recursively this procedure on the rest subset D_2 and D_3. Compared to the random technique, the size of D_2 and D_3 will smaller. Firstly, this leads to better performance of TESC because the complexity of TESC is proportional to the number of sample in training dataset. Furthermore, the *data-driven* technique also yields less overfitting than the former technique, or in other words, make a better performance of prediction on new unseen samples.

3.4. The proposed model of semi-supervised multi-label classifier

To examine MASS performance, we proposed a model for MLC which is described in Figure 3.7.

In our approach, we combine several techniques to enhance the performance of classifier. First of all, we make use of the hidden topic model of LDA to build the features of hidden topic probabilities for each document in *Feature Enrichment* step. This kind of features provides a much richer semantic meaning of text representation.

Next, a *feature selection* method based on Mutual Information is applied to improve the features for the classifier.

In the last step, a *multi-label classifier* is built based on the MASS algorithm. This classifier will be used to classify new documents.

3.4.1. Building features by applying the LDA model

The hidden topic model of LDA has been determined by applying the GibbsLDA++ tool[1] on a universal dataset with different numbers of topics. Due to the number of classes, we select the hidden topic numbers of 10, 15, 25, 50 and 100 to evaluate. These models

[1] Xuan-Hieu Phan, Cam-Tu Nguyen, GibbsLDA++, *http://gibbslda.sourceforge.net*, 2007.

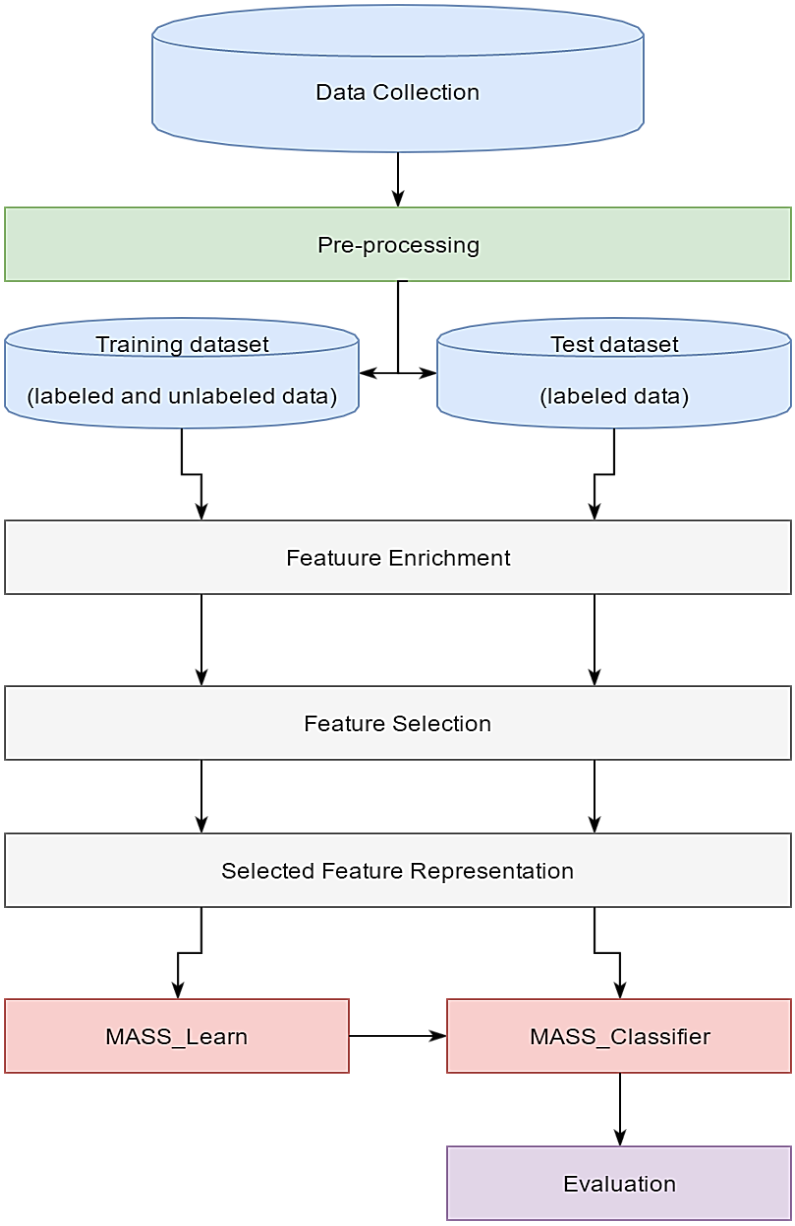


Figure 3.7. The proposed framework for MLC

will be applied on the training/testing data to generate probabilities of assigning hidden topics for each document.

Let $p(d, j)$ be the probability that a review d belongs to the hidden topic j (i.e., $j=1...k$, where k is the number of hidden topics). The vector ($p(d,1)$, $p(d,2)$, …., $p(d,k)$) is called the hidden topic feature vector of the review. These features will be combined with other features of documents to build the feature set for the classifier.

3.4.2. Selecting features based on the mutual information

Feature selection is one of the fundamental steps in machine learning and data mining research area for reducing dimensionality, choosing a small subset of the relevant features from the original ones, which usually leads to better learning performance, lower computational cost and better model interpretability.

We apply the method of mutual information (MI) [24] to perform feature selection in multi-label classification problems. MI measures the amount of information contained in variable X in order to predict variable Y in *any* relation, not only linear ones. In addition, the MI concept is directly applicable to groups of variables. The MI is given in [25] as below:

$$I(X,Y) = \int\int p_{x,y}(x,y) log \frac{p_{x,y}(x,y)}{p_x(x)p_y(y)} \tag{3.3}$$

where $p_x(x)$ and $p_y(y)$ are the marginal probability density function of X and Y respectively.

The method of achieving feature selection based on MI [24] includes the following steps. First, the multi-label problem is transformed using the pruned problem transformation method [26]. Then, for each single-label classification task, a forward/backward selection algorithm based on mutual information is employed to choose the "optimal" feature subset. The search strategy used in this model is considered as backward elimination, which starts with the set of all features and recursively removes them one at a time. The procedure ends when the predefined number of features has been reached. Another search strategy called a greedy forward feature selection algorithm that begins with an empty set of features and first selects the feature whose MI to the class vector is the highest. Then, the algorithm sequentially selects the feature not yet selected whose addition to the current subset of selected features leading to the set having the highest MI to the output.

Chapter 4

RESULTS AND DISCUSSIONS

4.1. The dataset and its representation

4.1.1. Data sources

The thesis applies the proposed framework to performance experiments on the set of customer's reviews about Vietnamese hotels retrieved from several famous Vietnamese websites on tourism and hotels. We build labeled, unlabeled and testing datasets with different numbers of documents to evaluate the effectiveness of labeled and unlabeled data on the model.

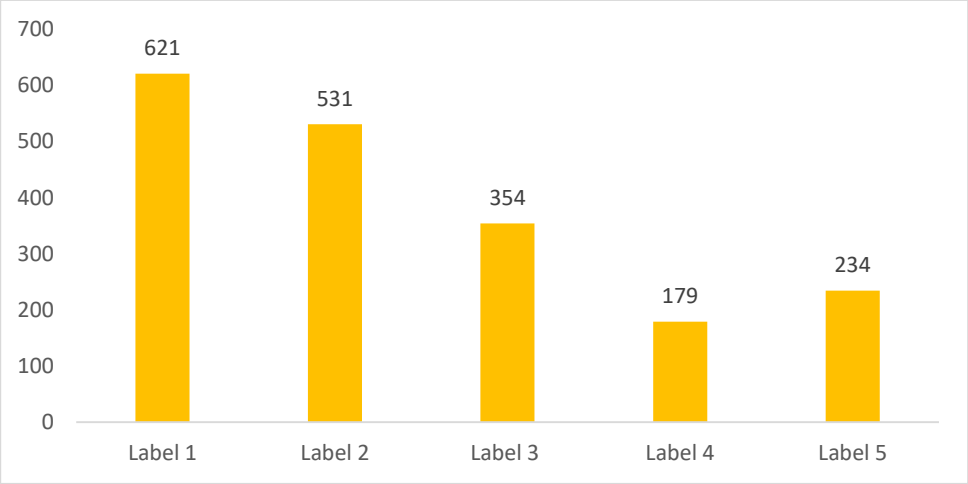


Figure 4.1 The distribution of labels in 1500 labelled reviews.

After some pre-processing steps on these datasets, i.e., main text context extraction, word segmentation, and some string manipulation, we got about 1800 reviews. 1500

reviews were manually tagged to create the labelled set of 1250 reviews, and the testing set of 250 reviews. The remaining 300 reviews were left intact to create the unlabeled set. We considered reviews on five aspects: a) *location and price*, b) *staff service*, c) *facilities*, d) *room standard*, and e) *food*.

Table 4.1 Information of predefined labels

	Label 1	**Label 2**	**Label 3**	**Label 4**	**Label 5**
Attribute	Location & price	Staff service	Facilities	Room standard	food
No.	621	531	354	179	234

In order to train the LDA model for generating the hidden topic models, the universal dataset of 24000 articles, introductions, comments about hotels in Vietnam (from the above sources) were also crawled. The preprocessing step is applied to all datasets for LDA construction and classification.

4.1.2. Data representation

Let $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots, \boldsymbol{x_m}\}$ be a feature set of samples, in which each element represents for a feature vector of a sample i.e. $\boldsymbol{x_i} = \{x_{i1}, x_{i2}, x_{i3}, \dots, x_{in}\}$. Also, let $\boldsymbol{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_m}\}$ be a label set of samples, in which each element represents a label vector or a set of labels corresponding to its sample. Therefore, $\boldsymbol{X}$ and $\boldsymbol{Y}$ can be defined as a matrix $\boldsymbol{X_{m\times n}}$ and $\boldsymbol{Y_{m\times l}}$, in which m, n, l are the number of samples, features and labels.

$$\boldsymbol{X_{m\times n}} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix} \text{ and } \boldsymbol{Y_{m\times l}} \begin{pmatrix} y_{11} & \cdots & y_{1l} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{ml} \end{pmatrix}$$

The features of text sentences usually are the binary type representing for the presence of a word in that sentence, or TF-IDF type, etc. When we observe the text dataset, the number of words appearing in a text review normally from 5 – 30 words while the number of features in a vector can increase to thousands (here is approximately 2400 in our problem). Mostly, the features vector will be very sparse which is computationally

costly. Therefore, we choose sparse matrices for data representation instead of feature matrix. Every row in a matrix can be rewritten as a list of valued features: $(i, k) - val$ where i, k and val indicate at the row i^{th} and column k^{th} the value of this feature is val. This representation can reduce the time of computation to hundreds time.

4.2. The environment configuration and tools used

Table 4.2 The details of system environment used in the thesis.

No.	Components	Value
1	Processor CPU	Intel Core i5-2430M 2.4 GHz
2	Installed RAM	8 GB
3	System Type	Windows 10
4	HDD Memory	300 GB

Table 4.3 The list of softwares and tools used in this thesis.

No.	Name	Author	Functionality	Source
1	Eclipse-SDK-3.5-win64	Eclipse Foundation	Development evironment	https://eclipse.org/
2	Visual Studio Code	Microsoft	Text Editor for Python	code.visualstudio.com
2	Vn.vitk	Le Hong Phuong	Sentence, word Segmentation	github.com/phuonglh/vn.vitk
3	GibbsLDA++	Xuan-Hieu Phan, Cam-Tu Nguyen	Train LDA models	jgibblda.sourceforge.net/
4	MULAN	Grigorios Tsoumakas	Feature Selection	mulan.sourceforge.net
5	Sklearn	Mathieu Blonde et al	Data handling Baseline	http://scikit-learn.org
6	Scipy, numpy	Python library	Computation	https://www.python.org/
7	MASS	Van-Quang Nguyen	A multi-label classification package	The thesis

The detailed information about the system environment configuration is described in Table 4.2. Furthermore, the list of softwares and tools the author used in this thesis is also presented in Table 4.3. The framework were both developed in Java and Python under the latest version Python 3.5.3, and Java 8.

In Python, we employed several open sources including Scipy, Pandas and Numpy to optimize the computational calculations of the system.

4.3. Experimental Tests

We took several experiments with different settings to evaluate the effect of the proposed algorithm. Each of the series experimental test and its results will be conducted and presented as below to analyze a different aspect of algorithm. Along with the configuration settings and experiment results, some discussion is also featured.

To evaluate the performance of the algorithms, we use label-based metrics which are described in chapter 1. These evaluation metrics are micro-averaging including $Precision, Recall, F1score$.

$$B_{micro}(f) = B\left(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN,\right) \qquad (4.1)$$

where q is the total number of labels. For these metrics, the bigger value, the better classification performance.

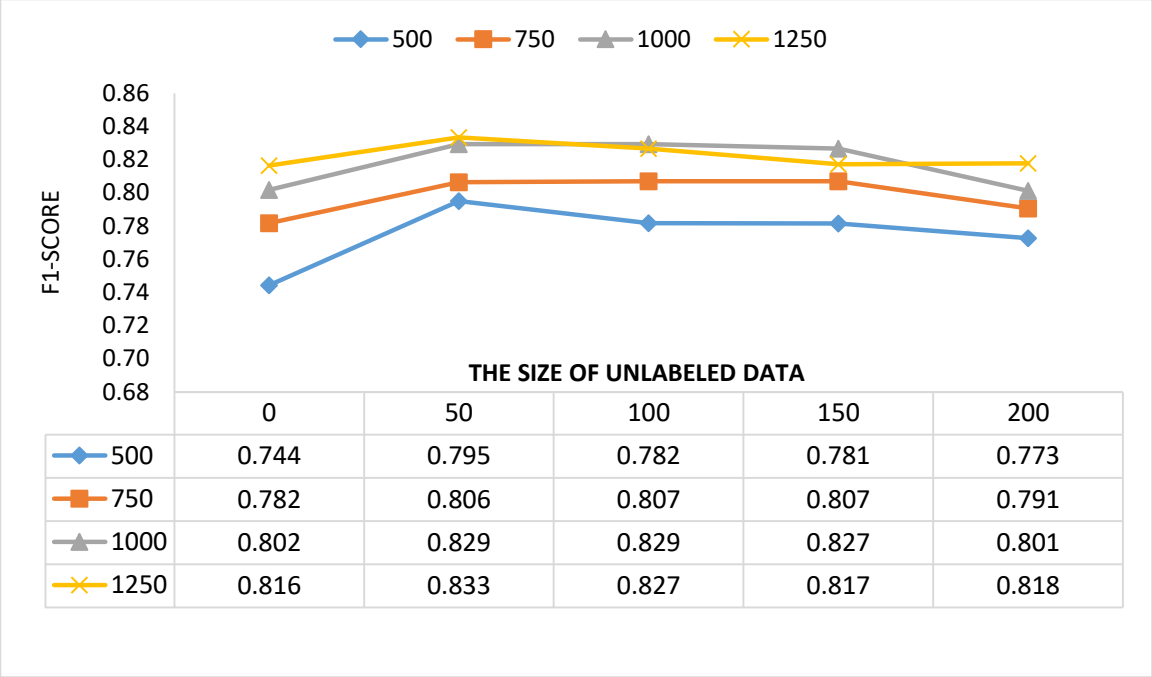
4.3.1. Experiments to examine the contribution of unlabeled data

In these experiments, we perform four main tests, in which the labeled data is fixed at 500, 750, 1000, 1250. To examine the contribution of unlabeled data in each test, the size of unlabeled data will be set at 0, 50, 100, 150, 200. The result of these experiment is shown in Figure 4.2.

It can be seen easily that in most cases with the size of unlabeled data different from zero, the results are higher than those with none of unlabeled samples. This confirms that unlabeled data, assumedly with same distribution as labeled data, will yield better results by adjusting the clusters of labeled data. However, four lines in Figure 4.2 have a tendency

of decreasing slightly the values of F1-score as the number of unlabeled data increases. Zhang [2] suggests the best fraction of unlabeled data over labeled data in TESC should be 1/10.

Figure 4.2 The result of experiments on the contribution of unlabeled data



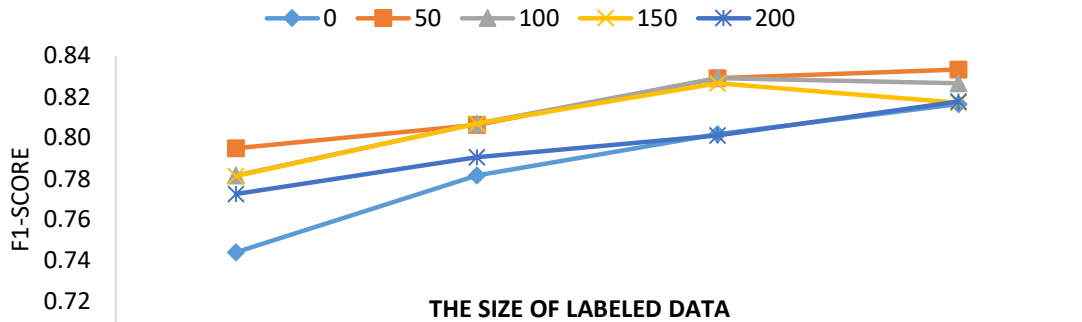
	0	50	100	150	200
500	0.744	0.795	0.782	0.781	0.773
750	0.782	0.806	0.807	0.807	0.791
1000	0.802	0.829	0.829	0.827	0.801
1250	0.816	0.833	0.827	0.817	0.818

4.3.2. Experiments to examine the contribution of labeled data

In these experiments, we repeat performing main tests as labeled. However we plot each line as the results of tests with fixed unlabeled data at 0, 50, 100, 150, 200. To examine the contribution of labeled data in each test, the size of labeled data will be set at 500, 750, 1000, 1250. The result of these experiment is shown in Figure 4.4.

Obviously, when we increase the number of labeled data while keeping the unlabeled data unchanged, the performance of classification on test set also sees an upward tendency. Also with the contribution of unlabeled data, when the amount of labeled data increases, the model gets better results.

Figure 4.3 The result of experiments on the contribution of labeled data



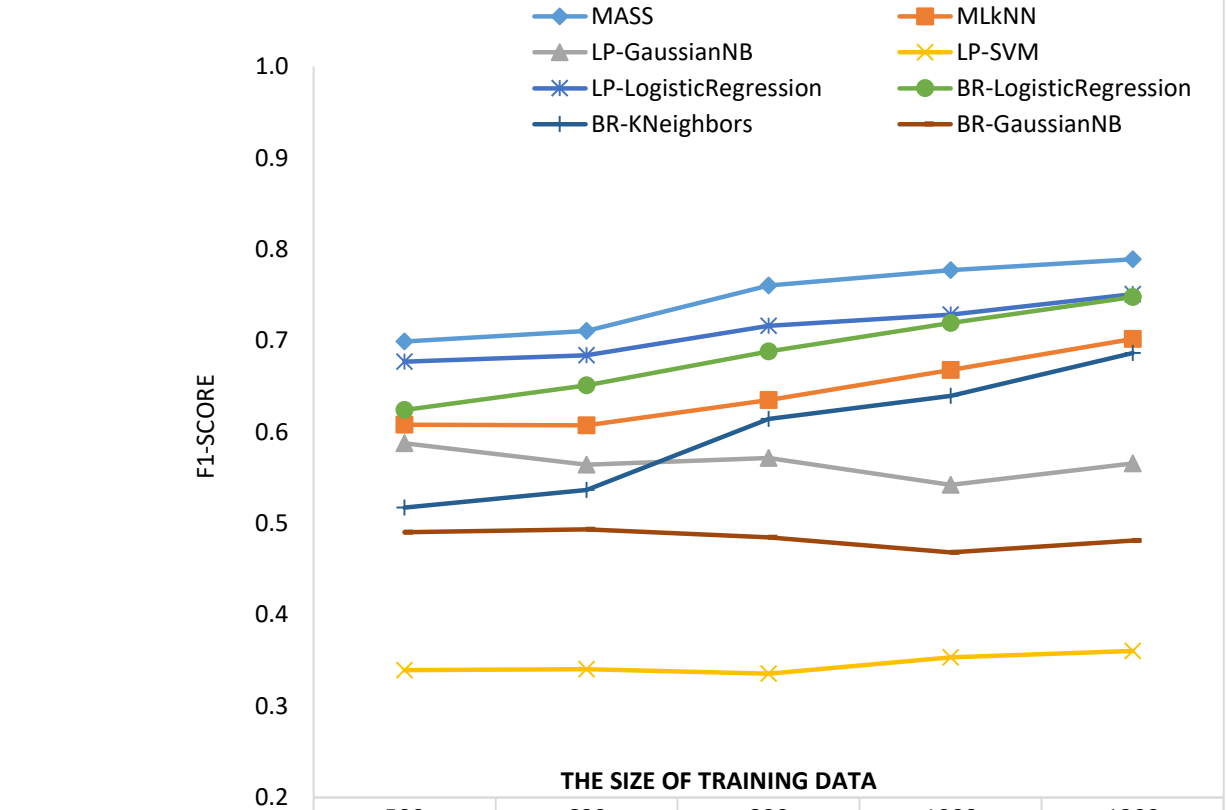
	500	750	1000	1250
0	0.744	0.782	0.802	0.816
50	0.795	0.806	0.829	0.833
100	0.782	0.807	0.829	0.827
150	0.781	0.807	0.827	0.817
200	0.773	0.791	0.801	0.818

4.3.3. Experiments to compare our method with some baselines

The thesis performs the 5-fold cross validation on our model and other outstanding algorithm baselines. The thesis chooses some classical multi-label classification methods including Binary Relevance (BR) Label Powerset (LP) and an algorithm adaptation ML-kNN against with our method (with fixed unlabeled data of 100). Several classical single-label classifiers are selected in Binary Relevance and Label Powerset including SVM, Logistic Regression, K-Nearest Neighbors, Gaussian Naïve Bayes. The result is presented in Figure 4.4.

The experimental results show that our method yield better performance on Vietnamese dataset. BR-Logistic Regression and LP-Logistic Regression also give us the promising results which are approximate to our method performance. Besides, the performance of an algorithm adaptation method ML-kNN only lies on the 4^{th} position which is less than about 10%.

Figure 4.4 The result of experiments to compare our method with baseliness



	500	600	800	1000	1200
MASS	0.699	0.711	0.760	0.777	0.789
MLkNN	0.608	0.607	0.635	0.668	0.702
LP-GaussianNB	0.588	0.564	0.572	0.542	0.566
LP-SVM	0.339	0.341	0.335	0.353	0.361
LP-LogisticRegression	0.677	0.684	0.716	0.728	0.751
BR-LogisticRegression	0.624	0.651	0.688	0.719	0.748
BR-KNeighbors	0.517	0.537	0.614	0.640	0.686
BR-GaussianNB	0.490	0.494	0.485	0.468	0.481

4.3.4. Experiments to compare different label space division techniques

The thesis re-performs the experimental tests with the same configuration for k-fold cross validation on our method. The parameter for selecting the label λ will be changed to examine two strategies namely *random* and *data driven* approach. In chapter

3, the thesis already explain how *data driven* approach can be adopted to increase the performance of our method as well as decrease the time of training models.

Overall, Figure 4.5 shows that the result of *data driven* approach is somehow better than *random* approach where the F1-score is higher and the training time is lower. However, the difference here is not remarkably visible. It is because the label space capacity of our dataset contains only five labels, which is not large enough to acknowledge the outperformance of *data driven* approach over *random* approach.

Figure 4.5 The result of experiments to compare different label space division approaches: (a) Comparison of F1-score, (b) Comparision of training time consumption.

4.3.5. Experiments to compare different feature representations

In these experiments, the thesis performs the evaluation on different of feature representations including binary feature and TF-IDF features and compare the results. The number of unlabeled data is fixed at 100. The cross-validation of 5-folds is conducted to compare the performance of both feature representations.

The result in Figure 4.6 shows that TF-IDF features seem to be better in case of vectorizing the text data rather binary features for text data.

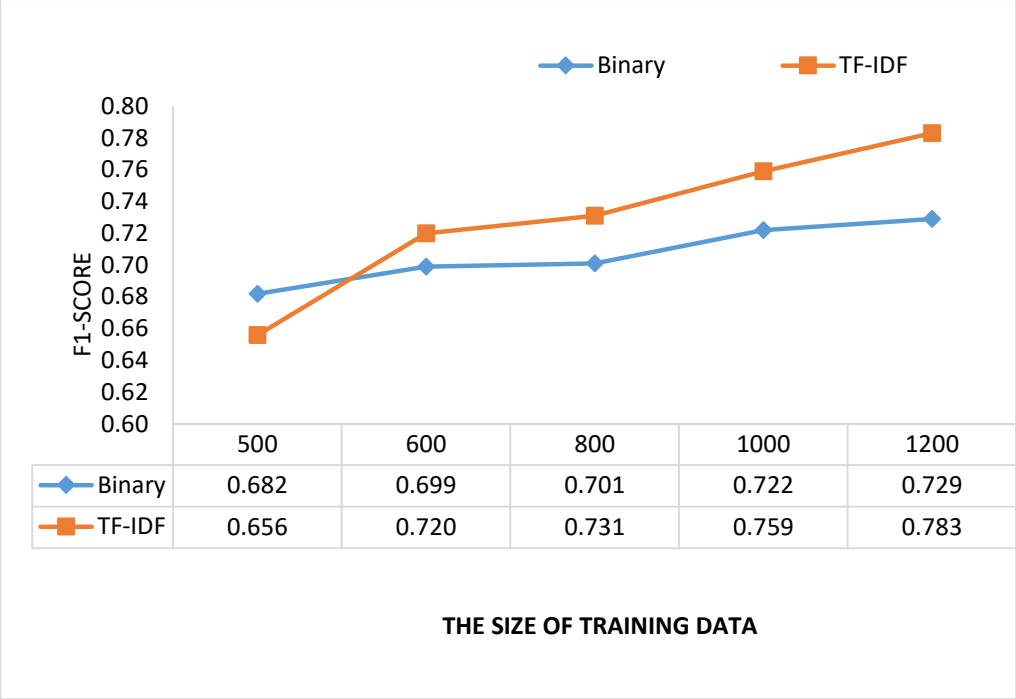
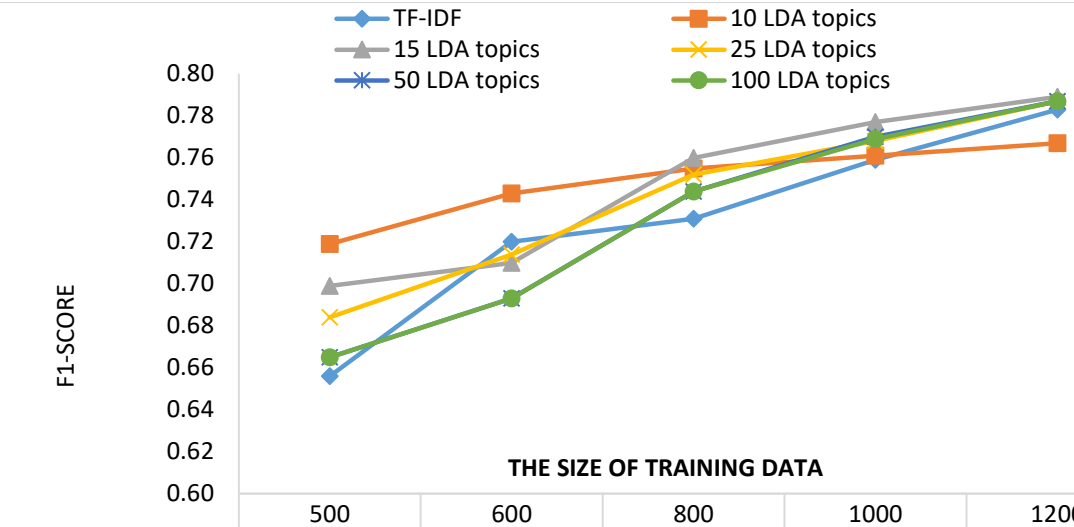


Figure 4.6 The result of experiments to compare different features representation.

4.3.6. Experiments to examine the contribution of feature enrichment

To examine the effectiveness of adding hidden topic features into current feature set, we perform some experiments with different configurations of hidden topic features: 10, 15, 25, 50, 100. Binary features, whose values are 1 or 0, seem to be incompatible with LDA features, whose value is continuous. Therefore, we combine the features of TF.IDF with the features of the hidden topic probabilities to build continuous features. K-fold cross validation (k = 5) is performed on each test.

The results in Figure 4.7 shows that with in most cases, the results of the tests without adding LDA topic features are somehow lower than the other. With the contribution of LDA topic features, the performance of the system can increase from about 2% to 8%.



	500	600	800	1000	1200
TF-IDF	0.656	0.720	0.731	0.759	0.783
10 LDA topics	0.719	0.743	0.755	0.761	0.767
15 LDA topics	0.699	0.710	0.760	0.777	0.789
25 LDA topics	0.684	0.714	0.752	0.768	0.787
50 LDA topics	0.665	0.693	0.744	0.770	0.787
100 LDA topics	0.665	0.693	0.744	0.769	0.787

Figure 4.7 The result of experiments to examine the contribution of feature enrichment.

CONCLUSIONS

Conclusions

In this thesis, we introduce an algorithm adaptation method using semi-supervised learning for multi-label classification. The author also builds a framework for MLC adopting a new algorithm adaptation method. The framework includes the process of enriching features with the hidden topic probabilities, the process of feature selection with mutual information, and the process of classification of MASS – an approach for semi-supervised MLC to exploit label-specific features. Using two basic assumptions including the effect of label-specific features in the learning phase and the multiple components in each label which can be identified by clustering, our proposed model brings major contribution in building label-specific features for multi-label learning with an approach of semi-supervised clustering technique. The experiments show that MASS gives a promising results for MLC even higher than several classical MLC methods. Regarding to the feature enrichment on Vietnamese text, the combination with the hidden topic probability features also contributes better performance.

Future Works

For future direction, we are making some improvements including the method to effectively select unlabeled instances, or post-process to prune the result clusters to remove outliers, to evaluate the proposed approach, and the technique to select the prominent label rather than the greedy idea. Here are the main follow-ups that can be done in right after the thesis.

1. **Parallel computing**

Currently, the method is implemented sequentially. Even though adopting the most efficient calculation on vectors and matrices, the algorithm will consume quite a lot time

with a larger set of labels. However, the underlying point of MASSLearn is that we can performance clustering simultaneously on 3 subsets $D_1, D_2,$ and D_3. It means that if we implement parallel programming for this task, we can save a huge amount of time while dealing with large-scale data of labels. Since the performance of the model is quite promising on different dataset including Vietnamese texts and other baseline dataset, its open source with improvement on computational time will be valuable to public community.

2. Label space division techniques

Although taking the ideas of label space division in RaKEL and *data-driven* approach, we only select a label λ, and perform dividing the dataset mainly based on λ. There is a lot room to deploy the techniques of label space division into the algorithm to boost the performance. One can figure out that we can divide the dataset into k subsets rather exactly three subsets. How to determine the optimal values of k is still open for further research and experiments.

3. Experiments on different dataset domains

The experiments were conducted on Vietnamese text dataset to the scope of the thesis. The baseline dataset of multi-label classification usually is quite large in both samples space and label space aspects. Therefore, after making some improvements on computational performance as mentioned above, the author will perform some experiments on other baseline dataset and extend the scope of the thesis to different domains of data instead of only text.

References

[1] Min-Ling Zhang, "LIFT: Multi-Label Learning with Label-Specific Features," *IJCAI 2011*, pp. 1609-1611, 2011.

[2] Wen Zhang, Xijin Tanga, and Taketoshi Yoshida, "TESC: An approach to text classification using semi-supervised clustering," *Knowledge-Based Systems*, pp. 152-160, 2015.

[3] Sotiris Diplaris, Grigorios Tsoumakas and Pericles A., "Protein Classification with Multiple Algorithms," *PCI 2005*, vol. 3746, pp. 448-456, 2005.

[4] Volker Roth and Bernd Fischer, "Improved functional prediction of proteins by," *BMC Bioinformatics2007*, vol. 8, p. S12, 2006.

[5] Tao. Li, Mitsunori Ogihara and George Tzanetakis, "Detecting emotion in music," *Proceedings of the International Symposium on Music Information Retrieval*, pp. 239-240, 2003.

[6] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris and Ioannis P. Vlahavas, "Multi-Label Classification of Music into Emotions," *ISMIR 2008*, pp. 325-330, 2008.

[7] Carlos Nascimento Silla Jr., Alex Alves Freitas, "A survey of hierarchical classification across different application domains," *Data Min. Knowl. Discov. (DATAMINE)*, vol. 22, pp. 31-72, 2011.

[8] Nguyen Cam-Tu, Xuan-Hieu Phan, Natsuda Kaothanthon, Takeshi Tokuyama, "A feature-word-topic model for image annotation," *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1481-1484, 2010.

[9] M.-L. Z. S.-J. H. a. Y.-F. L. Zhi-Hua Zhou, "Multi-Instance Multi-Label Learning with Application to Scene Classification," *National Key Laboratory for Novel Software Technology*, pp. 2291-2320, 2012.

[10] Eva Gibaja and Sebastián Ventura, "A Tutorial on Multi-Label Learning," *ACM Computing Surveys (CSUR),* vol. 3, p. 52, 2015.

[11] Min-Ling Zhang and Zhi-Hua Zhou, "A Review on Multi-Label Learning Algorithms," *IEEE Trans. Knowl. Data Eng.*, pp. 1819-1837, 2014.

[12] Min-Ling Zhang and Zhi-Hua Zhou, "ML-KNN: A Lazy Learning Approach to Multi-Label Learning," *Pattern Recogn,* vol. 40, pp. 2038-2048, 2007.

[13] Amanda Clare and Ross D. King, "Knowledge Discovery in Multi-Label Phenotype Data," *Lecture Notes in Computer Science 2168,* no. Berlin: Springer, pp. 42-53, 2001.

[14] J. R. Quinlan, C4.5: programs for Machine Learning, San Francisco: Morgan Kaufmann, 1993.

[15] Andre´ Elisseeff and Jason Weston, "A kernel method for multi-labelled classification," *NIPS'01 Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic,* pp. 681-687 , 2001.

[16] André Elisseeff and Jason Weston, "Kernel Methods for Multi-Labelled Classification and Categorical Regression Problems," *BIOwulf Technologies,* no. Tech. Rep. , 2001.

[17] Fabrizio Sebastiani, "Machine Learning in Automated Text Categorization,"," *ACM Computing Survey,* vol. 34, pp. 1-47, 2012.

[18] Kong, Xiangnan, Michael K. Ng, and Zhi-Hua Zhou, "Transductive multilabel learning via label set propagation," *IEEE Transactions on Knowledge and Data Engineering,* vol. 25.3, pp. 704-719, 2013.

[19] Zhen-Jun Zha, Tao Mei, Jingdong Wang, and Xian-Sheng Hua, "Graph-based semi-supervised learning with multiple labels," *Journal of Visual Communication and Image Representation,* vol. 2, pp. 61-74, 2009.

[20] Y.Liu, R.Jin and L.Yang, "Semi-supervised multi-label learning by constrained non-negative matrix factorization," *Proceedings of the 21st AAAI Conference on Artificial Intelligence,* pp. 421-426, 2006.

[21] Piotr Szymanski, Tomasz Kajdanowicz and Kristian Kersting, "How is a data-driven approach better than random choice in Label space division for multi-label classification," *Entropy,* 2016.

[22] David M. Blei, Andrew Y. Ng, Michael I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research,* vol. 3, pp. 993-1022, 2003.

[23] D. M. Blei, "Probabilistic topic models," *Communications of the ACM,* vol. 55.4, pp. 77-84, 2012.

[24] Gauthier Doquire, Michel Verleysen, "Feature Selection for Multi-label Classification Problems," *IWANN* , pp. 9-16, 2011.

[25] Gómez-Verdejo, Vanessa, Michel Verleysen, and Jérôme Fleury, "Information-theoretic feature selection for the classification of hysteresis curves," *International Work-Conference on Artificial Neural Networks,* pp. 522-529, 2007.

[26] D. M. Blei, "Probabilistic topic models," *Communications of the ACM,* vol. 55.4, pp. 77-84, 2012.

[27] M.-L. Z. S.-J. H. a. Y.-F. L. Zhi-Hua Zhou, "Multi-instance multi-label learning," *Artif. Intell,* vol. 176, pp. 2291-2320, 2012.

[28] Min-Ling Zhang, Zhi-Hua Zhou, "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization," *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, no. 10, pp. 1338-1351 , 2006.

Appendix A

Currently, MassClassifier is hosted by Github[2]. It contains necessary external resources and a pre-trained model for multi-label classification for Vietnamese hotel's review text.

A.1 How to call MassClassifier from command line

The following command is used to run the toolkit, the system needs python version 3.0 or newer:

```
python massclassifier.py <resources_dir> <input_file_dir> <output_file_dir> <options>
```

The arguments can be described as follows:

- <resources_dir> is the directory containing a learned model (The default model is for multi-label classification for Vietnamese hotel).
- <input_file_dir> is the directory containing the input file. In the input file, each line represents for the list of features of an instance.
- <output_file_dir> is the directory containing the output file. Each line contains a label indicator of binary value {0, 1}.
- <options> the type of label space division technique "*random*" for *random approach*, and "*dd"* for data driven approach.

A.2 How to use APIs

[2] https://github.com/quanguet/massclassifier

1. Initialize a classifier

Firstly, we have to create a classifier:

```
mClassifier =  MassClassifier(learned_model_path, lspace_type="random",
                                        required_dense=True)
```

- **learned_model_path: i**f we want to use our trained model, we have to pass a path of the learned model for the argument "learned_model_path. Otherwise, we will skip this argument and the default value for this: learned_model_path = None.
- **lspace_type:** This argument defines the strategy this classifier will adopt to train a model. "*random*" for random approach, and *"dd"* for data driven approach.
- **required_dense:** The representation of features would be dense matrix (True) or sparse matrix (False).

2. Train a new model

After initializing a classifier, we can use the following method to train a new model from our data.

```
mClassifier.fit(X_train, Y_train, X_unlabel)
```

- **X_train:** A matrix of features representing for a set of labeled instances <n1_samples x n_features>
- **Y_train:** A matrix of labels corresponding to each of X_train <n1_samples x n_labels>

- **X_unlabel:** A matrix of features representing for a set of unlabeled instances. <n2_samples x n_features>

X_train, Y_train, X_unlabel can be represented by dense matrices or sparse matrices. The classifier will ultimately transform dense matrices to sparse matrices.

Then the learned model will be exported into a file at the directory declared at the former step.

3. Predict a set of labels for unseen instances

```
Y_pred = mClassifier.predict(X_test)
```

- **X_test:** A feature matrix of instances we need to predicts their labels.
- This method will return a predicted matrix of labels for **X_test,** here we store it to the variable **Y_pred.**

4. Evaluate the performance of a trained model.

```
result = Evaluation.eval(Y_test, Y_pred)
print(result)
```

- **Y_test** here is a label matrix of instances for testing or their truly observed labels.
- **Y_pred,** as predicted in step 3, are a matrix of predicted labels for testing instances.

The result should display the value as “Precision = a %, Recall = b % and F1-score = c%.