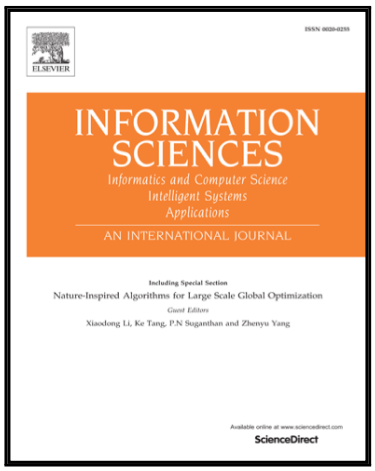


Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec

Donghwa Kim, Deokseong Seo, Suhyoun Cho, Pilsung Kang

PII: S0020-0255(18)30802-8
DOI: https://doi.org/10.1016/j.ins.2018.10.006
Reference: INS 13987

To appear in: *Information Sciences*

Received date: 14 November 2017
Revised date: 28 September 2018
Accepted date: 7 October 2018

Please cite this article as: Donghwa Kim, Deokseong Seo, Suhyoun Cho, Pilsung Kang, Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec, *Information Sciences* (2018), doi: https://doi.org/10.1016/j.ins.2018.10.006

Multi-co-training for document classification using various document representations: TF–IDF, LDA, and Doc2Vec

Donghwa Kim[a], Deokseong Seo[a], Deokseong Seo[a], Suhyoun Cho[a], Pilsung Kang[a,*]

[a]*School of Industrial Management Engineering, Korea University, Seoul, Republic of Korea*

Abstract

The purpose of document classification is to assign the most appropriate label to a specified document. The main challenges in document classification are insufficient label information and unstructured sparse format. A semi-supervised learning (SSL) approach could be an effective solution to the former problem, whereas the consideration of multiple document representation schemes can resolve the latter problem. Co-training is a popular SSL method that attempts to exploit various perspectives in terms of feature subsets for the same example. In this paper, we propose multi-co-training (MCT) for improving the performance of document classification. In order to increase the variety of feature sets for classification, we transform a document using three document representation methods: term frequency–inverse document frequency (TF–IDF) based on the bag-of-words scheme, topic distribution based on latent Dirichlet allocation (LDA), and neural-network-based document embedding known as document to vector (Doc2Vec). The experimental results demonstrate that the proposed MCT is robust to parameter changes and outperforms benchmark methods under various conditions.

Keywords: Document classification, Semi-Supervised Learning, TF–IDF, LDA, Doc2Vec, Co-Training

1. Introduction

Document classification is one of the main tasks of text mining and has been used in several applications [6] such as spam filtering [3] and sentiment analysis [10, 11, 20]. There are two main challenges for document classification: insufficient label information [19] and

*Corresponding author, Tel: +82-2-970-7286, Fax: +82-2-979-3377

Email addresses: donghwa89@korea.ac.kr (Donghwa Kim), heyhi16@korea.ac.kr (Deokseong Seo), heyhi16@korea.ac.kr (Deokseong Seo), jogly91@korea.ac.kr (Suhyoun Cho), pilsung_kang@korea.ac.kr (Pilsung Kang)

absence of an optimal representation method [12]. For document classification, no systematic and automated process is available that can be used to assign class labels to a large number of documents and then update the classification model simultaneously. Meanwhile, in numerous other classifications, once a new dataset is obtained, its class label is automatically determined. For example, in the customer churn classification of the telecommunication industry, the class labels, i.e., to stay or to leave, can be automatically determined because the customer status is periodically updated [1]. In addition, for the daily stock market prediction in the financial industry, the fluctuating price of a certain equity, which is the class label for the task, is also automatically determined when the market is closed on that day. However, the label assignment for documents is human-labor intensive, time consuming, and cost ineffective. Moreover, because a document is a list of words with a variable length, for further analysis, it should be transformed into a fixed size of numerical vector. Although document representation methods are available, such as term frequency and inverse document frequency (TF–IDF) [23] and the recently proposed neural network-based distributed representation [16], no document representation method performs higher than the other methods, for all text analytics tasks.

When there are only a few labeled examples but a large number of unlabeled examples are also available, one can consider employing semi-supervised learning (SSL) approaches for improving the classification performance [8]. In SSL, it is assumed that examples belonging to a single class are generated from a single distribution. Hence, although unlabeled examples cannot be explicitly used to help classification models to learn the discrimination function, they can be used to help estimate the data distribution for various classes; this, in turn, helps in obtaining an improved class boundary as compared with that obtained based on only labeled examples. Several strategies have realized the SSL concept in learning algorithms. Self-training (ST) constructs a classifier using only the labeled examples and unlabeled test examples with the current classifier [24]. If the likelihood of an unlabeled instance for a class is sufficiently high, it is included in the labeled dataset with the predicted class label. Then, a new classifier is trained based on the extended labeled dataset. This procedure is repeated until all the unlabeled examples are classified into one of the available classes. In contrast, generative models attempt to estimate the underlying data-generation function based on the labeled and unlabeled examples [9]. They can be used to determine the most appropriate distribution parameters by maximizing the posterior probability of both the labeled and

unlabeled examples. Graph-based SSL is based on the assumption that a specified dataset can be expressed using a set of nodes (examples) and edges (relations between nodes, e.g., similarity or distance) [32]. Once the graph is constructed, the label information is propagated throughout the graph in order to assign appropriate class labels to the unlabeled examples.

Among the SSL approaches, co-training is highly noteworthy in that it considers data from various perspectives [5]. The premise of the co-training approach is that not all significant characteristics of data can be observed by a single view. A few characteristics can be conveniently understood using one view, whereas others can be captured using another view. Therefore, if a feature set of a specified dataset can be split into two subsets, two classifiers are trained independently based only on one feature set. Let us assume that these classifiers are `Model A` and `Model B`. The classification models evolve by teaching each other as follows: If `Model A` is highly confident about the prediction for an unlabeled example whereas `Model B`'s confidence is low, this instance is added to the training set of `Model B` with the label predicted by `Model A`. The reverse case is also likely to occur. With the aid of the other classification model, each model can learn the characteristics of the dataset that cannot be learned independently. The key indicator of success of the co-training is whether the features of the dataset can be split into the independent subsets. If the features of a specified dataset are originally generated from a single view, dividing the feature set cannot aid in improving the classification performance. In contrast, if the features are naturally generated using different views, such as the text description of an object (one view) and its image (another view), the co-training approach can effectively utilize unlabeled examples to enhance the classification performance.

We have derived inspiration from the manner in which co-training algorithms are trained, and as different document representation methods have demonstrated their effectiveness in various text-mining tasks, we propose a multi-co-training (MCT) method for document classification. In our method, documents are expressed using three representation schemes: TF–IDF, latent Dirichlet allocation (LDA) [4], and document to vector (Doc2Vec) [16]. TF–IDF representation is based on the bag-of-words philosophy, which involves the assumption that a document is simply a collection of words, and thus, the document can be vectorized by computing the relative importance of each word, i.e., by considering the word's frequency in the document and its popularity in the corpus. LDA was originally developed for topic modeling,

the main purpose of which is to discover latent themes that permeate the corpus. Once the LDA is trained, two outputs are generated: word distribution per topic and topic distribution per document. The latter can be regarded as another document representation in which both the word frequencies and semantic information (topic constitution) is considered. Doc2Vec is the newest among the three document representation schemes, and it is an extension of the word-to-vector (Word2Vec) representation. A word is regarded as a single vector, the element values of which are real numbers in the Word2Vec representation. The assumption of Word2Vec is that the element values of a word are affected by those of other words surrounding the target word. This assumption is encoded as a neural network structure, e.g., continuous bag-of-words or skip-gram, and the network weights are adjusted by learning observed examples [17]. Doc2Vec extends Word2Vec from the word level to the document level [16]. Each document has its own vector values in the same space as that for words. Thus, the distributed representation for both words and documents are learned simultaneously. Once the documents in a corpus were expressed using the three representation methods, we trained three classification models based on each representation method. As in co-training, a document with a prediction confidence that is significantly high for one of the three models is added to the training set of the other two models with the confidently predicted label. In order to verify the proposed MCT method, we conduct experiments by varying the rate of labeled training examples, representation dimensions, and data-dependent parameters.

The rest of this paper is organized as follows: In Section 2, we briefly review previous studies on document classification with the three document representation methods (TF–IDF, LDA, Doc2Vec) and their variants. In Section 3, the proposed MCT method is demonstrated. In Section 4, we explain the experimental design with the data description, parameter settings, benchmarked methods, and performance measure. The experimental results are discussed in Section 5. Finally, in Section 6, we conclude the current work with a few future research directions.

2. Literature Review

As document classification is one of the main text mining tasks, a large number of related studies have exhibited significant progress to date. In this study, we briefly review a few representative studies while focusing on document representation methods.

To date, TF–IDF has been the most commonly adopted document representation method for various document-processing tasks. It provides each word in a document a weight according to the following two criteria: (1) the frequency of its usage in the specified document (TF) and (2) the rarity of its appearance in the other documents in the corpus (IDF) [13]. Zhang et al. [31] performed a comparative study of text classification using TF–IDF, latent semantic indexing (LSI), and multi-words for text representation. Ranjan et al. [22] used TF–IDF to assign weights to the words and classified documents using long short-term memory (LSTM) neural networks. Moreover, a few studies have used variants or extensions of TF–IDF as the main document representation method to perform classification tasks [25, 30].

Another popular document representation approach is based on topic modeling. The main purpose of topic modeling is to identify underlying themes across the corpus based on the assumption that the document is generated by a certain stochastic process. LDA is the most fundamental and popular realization of topic modeling [4]. Once LDA is trained based on a corpus, it produces two outputs: term distribution per topic and topic distribution per document. In document classification, the latter output is used as a document representation method. Blei et al. [4] applied LDA to topic modeling as well as to other document-processing tasks such as document classification and collaborative filtering. Bíró et al. [3] employed LDA for spam filtering and achieved higher accuracies as compared with the traditional document representation methods. Although LDA is an unsupervised learning, which does not use label information, Wang et al. [27] attempted to improve the performance of document classification by using label information in addition to LDA-based document representation.

Recently, Doc2Vec has been successfully used for document classification tasks in various domains [16, 28]. Doc2Vec is a natural extension of Word2Vec, the main task of which is to determine an appropriate distributed representation for a single document by learning a neural network with the information of a target word and the words surrounding it in the document. Previous studies have demonstrated that Doc2Vec yields higher classification accuracy than other document representation methods in various domains, such as sentiment classification with the IMDB dataset [16], news categorization [28], and forum question duplication [15].

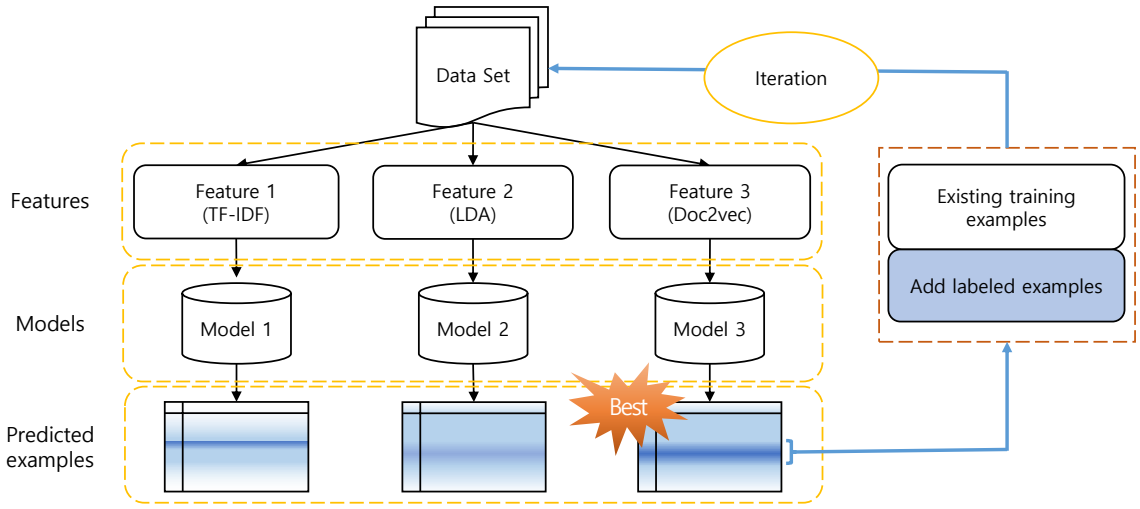


Figure 1: Framework of the proposed method

3. Method: Multi-Co-Training

The proposed MCT method is illustrated in Figure 1. Each document is converted into three numerical vectors (three feature sets) based on three document representation methods: TF–IDF, LDA, and Doc2Vec. Then, three learning schemes are applied: supervised learning (SL), ST, and MCT. SL-based algorithms use only labeled documents, whereas ST-based algorithms use unlabeled data although they are trained based only on one of the three feature sets. In contrast, in MCT, each model is initially trained based on a single feature set; however, it is aided by other models with various feature sets that provide highly confident predictions for originally unlabeled examples.

3.1. Document Representation (Feature Extraction)

Three document representation methods are employed to transform an unstructured document into a structured numerical vector: TF–IDF [23], LDA [4], and Doc2Vec [16].

3.1.1. Term Frequency–Inverse Document Frequency (TF–IDF)

TF–IDF is the most fundamental form of document representation and has the longest history among the three adopted representation methods [23]. It is based on the bag-of-word scheme in which a document can be represented by a collection of words used in the document. TF–IDF also assumes that if a word is important for a document, it should repeatedly appear

in that document whereas it should rarely appear in other documents. The TF is associated with the former assumption whereas the IDF is associated with the latter assumption. The parameter tf_{ij} is defined as the number of times word i appears in document j; the larger the value, the more important the word is. The parameter df_i is the number of documents in which word i appears at least once; the larger the value, the more common the word is. If word i can be considered important for document j, it should have a large TF (tf_{ij}) and a small DF (df_i). For example, articles such as 'a' and 'the' and pronouns such as 'it,' 'this,' 'that,' and 'those' would frequently appear in a document. However, they cannot be considered important words for that document because they are prevalent in all documents. Hence, TF–IDF is defined by Eq.(1):

$$\text{TF–IDF}_{ij} = tf_{ij} \times \log\left(\frac{N}{df_i + 1}\right), \tag{1}$$

where the IDF takes the logarithm of the ratio of the number of total documents in the corpus to the document frequency of word i with IDF smoothing to prevent it from being divided by zero, i.e., $idf_i = \log\left(\frac{N}{df_i+1}\right)$. We select those words with high TF–IDF scores on an average for the corpus.

3.1.2. Latent Dirichlet Allocation (LDA)

The main purpose of LDA is to identify latent themes or topics penetrating the specified corpus based on the word frequency of each document assuming that a single document is generated by a pre-defined probabilistic process [4]. Thus, it can be considered as an unsupervised generative topic model. LDA assumes that a document is characterized by topic distributions, whereas a topic is characterized by word distributions. Hence, the topic probability distribution θ per document and word probability distribution ϕ per topic are estimated using the specified corpus assuming that each document is generated by the procedure illustrated in Figure 2. The document generation process of LDA is as follows: First, the word distribution per topic ϕ_k is sampled from the Dirichlet distribution with the hyperparameter β.

$$p(\phi|\beta) = \prod_{k=1}^{K} \frac{\Gamma(\beta_{k,\cdot})}{\prod_{v=1}^{V}\Gamma(\beta_{k,v})} \prod_{v=1}^{V} \phi_{k,v}^{\beta_{k,v}-1}, \text{for} \quad \phi_k \sim Dir(\beta). \tag{2}$$

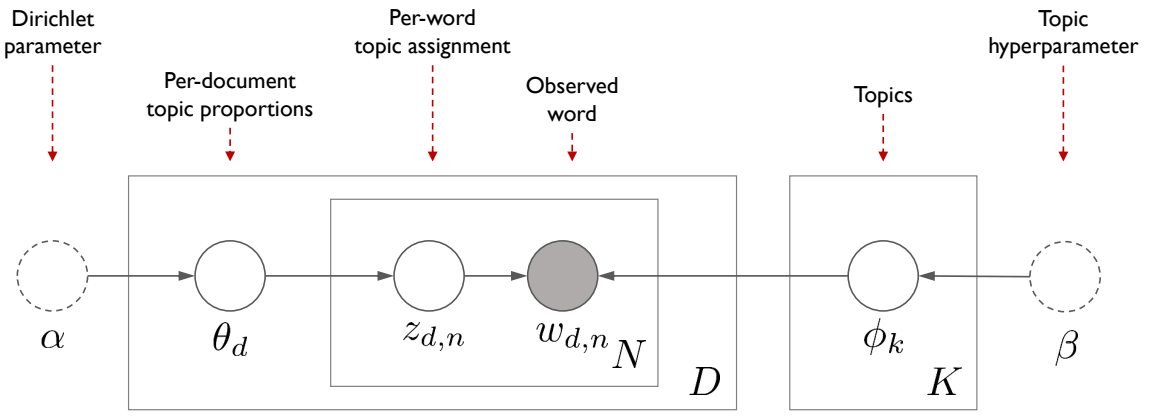


Figure 2: Document generation process of LDA [4]. Plates denote repetitions (K: number of topics, D: number of documents in the corpus, and N: number of words in a document). The dotted, solid white, and solid gray circles represent the hyperparameters, latent variables, and observed variables, respectively.

Secondly, the topic distribution θ of a document is sampled from the Dirichlet distribution with the hyperparameter α, as expressed by Eq.(3):

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k}\alpha_i)}{\prod_{i=1}^{k}\Gamma(\alpha_i)}\prod_{i=1}^{K}\theta_i^{\alpha_i-1} \quad \text{for} \quad \theta_d \sim Dir(\alpha). \tag{3}$$

and the latent variable z_n, which is the topic assignment per word, follows a multinomial distribution with the parameter θ, as expressed by Eq.(4):

$$p(z|\theta) = \prod_{i=1}^{K}\theta^{z_i}, \quad \text{for} \quad z \sim Multi(\theta), \quad \sum_{i=1}^{K}\theta_i = 1, \quad \theta_i \geq 0. \tag{4}$$

and ϕ_k in Eq.(5) follows the Dirichlet distribution with the parameter β.

$$p(\phi_k|\beta) = \frac{\Gamma(\sum_{i=1}^{k}\beta_{k,.})}{\prod_{i=1}^{k}\Gamma(\beta_{k,v})}\prod_{v=1}^{K}\phi_{k,v}^{\beta_{k,v}-1} \tag{5}$$

Finally, word w_n is also obtained from the multinomial distribution with z_n and β, as expressed by Eq.(6):

$$p(w_n|z_n,\beta) = \prod_{j=1}^{V}\left(\sum_{i=1}^{k}z_{n,i}\beta_{i,j}\right)^{w_{n,j}} \quad \text{for} \quad w_n \sim Multi(z_n,\beta) \tag{6}$$

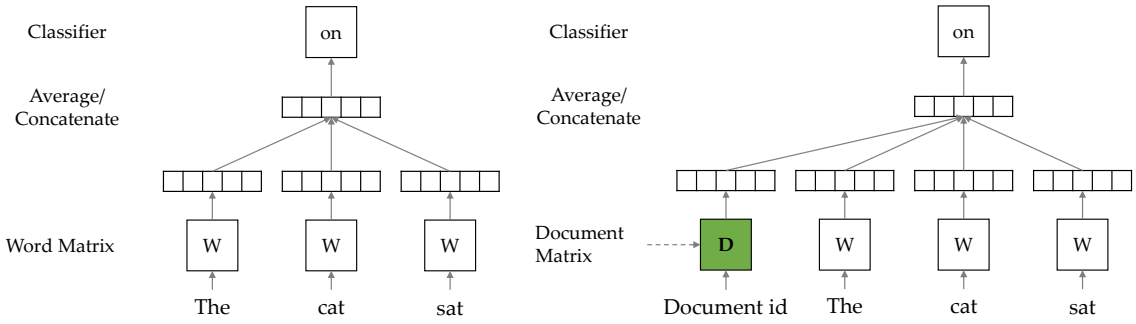


Figure 3: Conceptual framework of Word2Vec (a) and Doc2Vec (b). In Word2Vec, the context of three words ("the," "cat," and "sat") is used to predict the fourth word ("on"), whereas the context of the document is used to predict the fourth word in Doc2Vec.

Given α and β, the joint distribution with θ, z, and d is first formed to obtain the d distribution, as expressed by Eq.(7).

$$p(\theta, z, d|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{7}$$

The posterior probability in Eq.(7) of the latent variables can be obtained by variational expectation-maximization or Collapsed Gibbs Sampling [21]. Provided that the features for a document, i.e., the proportion of the K topics in the document, are extracted from relationships among the documents and words according to the LDA formula, LDA can yield a higher performance of document classification than that by other document representation methods such as TF–IDF and bigram [4].

3.1.3. Document to Vector (Doc2Vec)

One-hot encoding, which is the simplest method to transform a word into a fixed-sized numerical vector, exhibits two major drawbacks. First, one-hot encoding requires a large amount of memory space as the dimensionality of the word vector in the one-hot encoding is equal to the number of unique words in the corpus, which is generally over hundreds of thousands. Secondly, one-hot encoding cannot preserve the semantic relationships between two words as the inner products of any specified two words become zero, i.e., all words are independent of each other. In order to overcome these limitations, Mikolov et al. [17] proposed a neural-network-based word representation method called Word2Vec (Figure 3 (a)). Given a

sequence of training words $w_1, w_2, ..., w_T$, the goal of Word2Vec is to maximize the predicted log probability.

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t|w_{t-k},\ldots,w_{t+k}), \tag{8}$$

where k is the window size for preserving the contextual information. The prediction is generally performed via a multiclass classification using the softmax function as follows:

$$p(w_t|w_{t-k},\ldots,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}, \tag{9}$$

where each y_i is the i-th output value of a feed-forward neural network computed using

$$y = b + Uh(w_{t-k},\ldots,w_{t+k};W), \tag{10}$$

where b, U, h, and W are the bias terms between the hidden and output layer, weight matrix between the hidden and output layer, average or concatenation for context words, and word embedding matrix, respectively.

Doc2Vec is an extension of Word2Vec that attempts to determine an adequate continuous vector for a paragraph or even a larger document in order to preserve the semantic relationship among various documents [16]. In a manner similar to Word2Vec, each word is represented by a d-dimensional continuous vector ($d << |V|$, which is the size of the vocabulary in the corpus). In addition, the document itself is also represented by a continuous vector in the same space of word vectors, as shown in Figure 3 (b). In Doc2Vec, each document is mapped to a unique vector that is represented by a column in matrix D, whereas each word is mapped to a unique vector that is represented by a column in matrix W. Therefore, the only alteration in the network formulation is the addition of D in Eq. (10) as follows:

$$y = b + Uh(w_{t-k},\ldots,w_{t+k};W,D) \tag{11}$$

Once the network is sufficiently trained, we can obtain a distributed representation of each document in the corpus; this, in turn, can be used for the subsequent text-mining tasks such as clustering or classification. In this study, we trained both the PV-DM and PV-DBOW,

Algorithm 1: Co-training

$\mathbf{X}^L = \{\mathbf{x}_n \in \mathbb{R}^d,\ y_n \in \{0,1\}\}_{m=1}^{M}$: labeled examples
$\mathbf{X}^U = \{\mathbf{x}_n \in \mathbb{R}^d\}_{n=1}^{N}$: unlabeled examples
$\mathbf{y}^U = \{y_n \in \{0,1\}\}_{n=1}^{N}$: predicted labels for $\mathbf{X}^U$
Function co-training $(\mathbf{X}_1^L, \mathbf{X}_2^L, \mathbf{X}_1^U, \mathbf{X}_2^U)$
Input : Two feature sets $\mathbf{X}_1 = (\mathbf{X}_1^L \cup \mathbf{X}_1^U)$ and $\mathbf{X}_2 = (\mathbf{X}_2^L \cup \mathbf{X}_2^U)$ where the specified joint labels y correspond with both x_1 and x_2
Output: (1) Predicted labels $\mathbf{y}^U$ for both $\mathbf{X}_1^U$ and $\mathbf{X}_2^U$
(2) A best classifier learned using co-training
repeat
  $f_A \leftarrow$ Model A$(\mathbf{X}_1^L)$
  $f_B \leftarrow$ Model B$(\mathbf{X}_2^L)$
  $\mathbf{y}_1^U \leftarrow f_A(\mathbf{X}_1^U)$
  $\mathbf{y}_2^U \leftarrow f_B(\mathbf{X}_2^U)$
  $\mathbf{U} = \{(\mathbf{X}_1^U, \mathbf{y}_1^U), (\mathbf{X}_2^U, \mathbf{y}_2^U)\}$
  $\mathbf{S} = \underset{\{(\mathbf{x}_n,\ y_n)\}}{\arg\max}\ C(\mathbf{U})$ // C is a confidence function
  $\mathbf{U} \leftarrow (\mathbf{U} - \mathbf{S})$
  $\mathbf{X}^L \leftarrow (\mathbf{X}^L \cup \mathbf{S})$
  $\mathbf{X}^U \leftarrow (\mathbf{X}^U - \mathbf{S})$
until $|\mathbf{U}| = 0$
return $(\mathbf{X}_1^L, \mathbf{X}_2^L, f^*)$ // f^* is the better model between f_A and f_B

both of which are the primary structures of the Doc2Vec model, and used their concatenated vector to improve the classification performance, as proposed by the authors in the original paper [17].

3.2. Multi-Co-Training

3.2.1. Co-Training: Theoretical Overview

Co-training is based on the assumption that the entire feature set can be divided into two mutually exclusive sets in the data space, i.e., $\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2$, and each example has an ordered pair $x = (x_1 \cup x_2)$ with a non-zero probability related to the distribution $\mathbf{D}$ according to $\mathbf{X}$. It also has a set of target functions C_1 and C_2 corresponding to $\mathbf{X} = \mathbf{X}_1$ and $\mathbf{X} = \mathbf{X}_2$, respectively [5]. The procedure used for co-training is explained using Algorithm 1. The main concept underlying co-training is that if we can examine an example from two independent views, e.g., view A and view B, and one view (A) can classify certain unlabeled examples more confidently than the other view (B), these examples can be provided to the classifier that is trained using data from view B in order to improve its classification

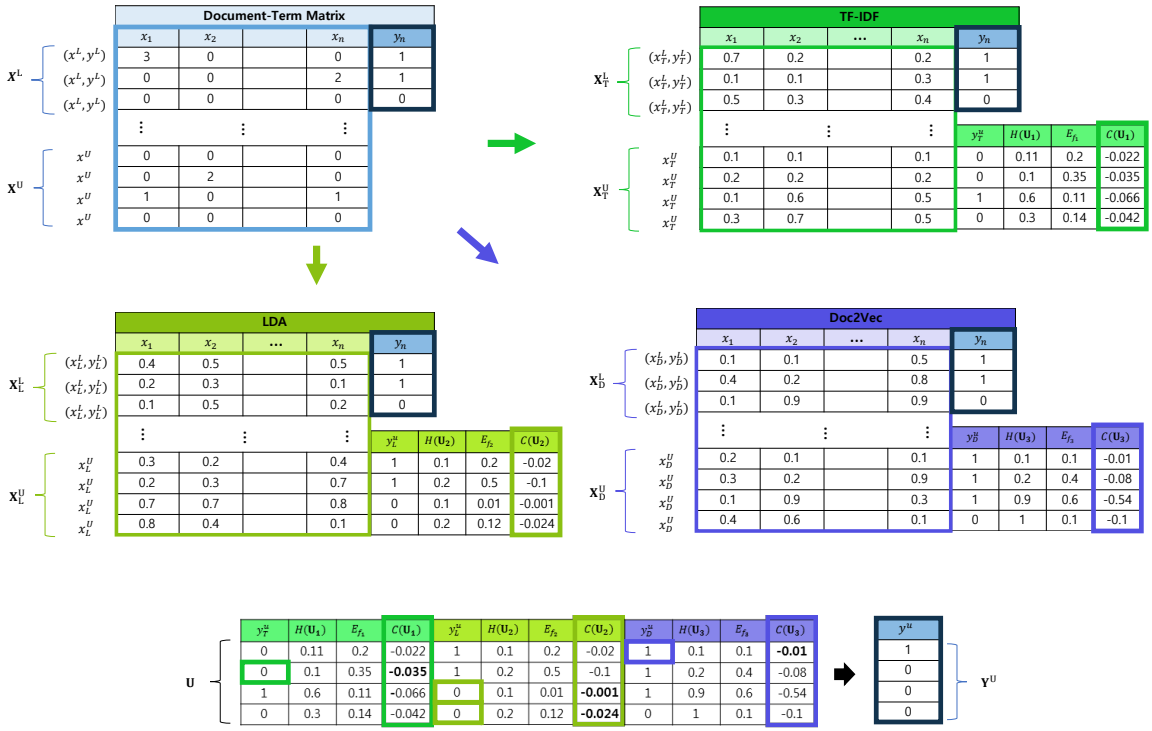


Figure 4: Illustrated example of proposed MCT framework

performance. Furthermore, the contrary case is also permissible. This concept is realized by constructing the models f_A and f_B based only on the specified initial labeled feature sets $\mathbf{X}_1^L$ and $\mathbf{X}_2^L$, respectively (Steps 6–7 in Algorithm 1) and predicting the unlabeled example $\mathbf{X}^U$ by employing each classifier f (Steps 8–9 in Algorithm 1). Finally, the examples that are predicted confidently $\mathbf{U}$ (lines 10–11 in Algorithm 1) by each classifier are added to the labeled feature sets $\mathbf{X}^L$ for the other classifier with the predicted label (lines 12–14 in Algorithm 1).

Co-training is based on two assumptions regarding feature sets in order for it to perform effectively [5]. First, the instance distribution D must be compatible with the target function $f = (f_1, f_2)$. That is, each of the feature sets must be sufficient for learning a classifier. Secondly, the feature sets must be independent of each other. For example, in web page classification, contents and hyperlinks are used as two feature sets for co-training Nigam and Ghani [18].

Algorithm 2: MULTI-CO-TRAINING

1 $\mathbf{X}^L = \{\mathbf{x}_m \in \mathbb{R}^d,\ y_m \in \{0,1\}^K\}_{m=1}^M$: labeled examples
2 $c_k \in \{1, ..., K\}$
3 $y_{m,k} = \begin{cases} 1 & \text{if } \mathbf{x}_m \in c_k \\ 0 & \text{otherwise} \end{cases}$
4 $\mathbf{X}^U = \{\mathbf{x}_n \in \mathbb{R}^d\}_{n=1}^N$: unlabeled examples
5 $\mathbf{y}^U = \{y_n \in \{0,1\}^K\}_{n=1}^N$: predicted labels for $\mathbf{X}^U$
6 Function multi-co-training $(\mathbf{X}^L, \mathbf{X}^U, \gamma)$
Input : Vectorized documents $\mathbf{X} = (\mathbf{X}^L \cup \mathbf{X}^U)$ based on bag-of-words
Output: (1) Predicted labels $\mathbf{y}^U$ for $\mathbf{X}^U$
(2) Documents represented as latent variables
(3) Better classifier learned using MCT
7 $\mathbf{X}_j \leftarrow \mathrm{R}(\mathbf{X})$ `// R: document representation`
8 $T = \lceil N/\gamma \rceil$
9 $j \in$ {TF–IDF, LDA, Doc2Vec}
10 **repeat**
11 $f_j \leftarrow \mathrm{Model}(\mathbf{X}_j^L)$, $\mathbf{X}_j^L$ comprises labeled examples based on j
12 $\mathbf{y}_j^U \leftarrow f_j(\mathbf{X}_j^U)$, $\mathbf{X}_j^U$ comprises unlabeled examples based on j
13 $\mathbf{U}_j = \{(\mathbf{x}_{nj}^U, y_{nj}^U)\}_{n=1}^N$
14 $P(\mathbf{U}_j) = \{p(y_{nj} \mid \mathbf{x}_{nj})\}_{n=1}^N$
15 $H(\mathbf{U}_j) = -\sum_{i=1}^k P(\mathbf{U}_j) \log\left(P(\mathbf{U}_j)\right)$ `// H is an entropy function`
16 $C(\mathbf{U}_j) = -E_{f_j} \times H(\mathbf{U}_j)$ `// E_{f_j} is a training error from f_j`
17 $\mathbf{S} = \underset{\{(\mathbf{x}_{nj},\ y_{nj})\}_{n=1}^T}{\arg\max} C(\mathbf{U}_j)$ `// C is the confidence function`
18 $N = N - T$
19 $\mathbf{U}_j \leftarrow (\mathbf{U}_j - \mathbf{S})$
20 $\mathbf{X}_j^L \leftarrow (\mathbf{X}_j^L \cup \mathbf{S})$
21 $\mathbf{X}_j^U \leftarrow (\mathbf{X}_j^U - \mathbf{S})$
22 **until** $N = 0$
23 $f^* = \underset{f_j}{\arg\max}\, B(f_j)$ `// B is our performance measure`
24 **return** $(\mathbf{X}_j^L, f^*)$

3.2.2. Proposed Method

In a manner similar to co-training, we used the three document representation methods: TF–IDF, LDA, and Doc2Vec. The process of MCT is demonstrated using Algorithm 2, and an illustrative example is provided in Figure 4. Initially, each document is transformed into the three feature vectors based on TF–IDF, LDA, and Doc2Vec (line 7 in Algorithm 2). As TF–IDF is a simple and count-based method, a representation model is not required to be learnt, whereas the LDA and Doc2Vec models are trained based on a specified corpus. Once

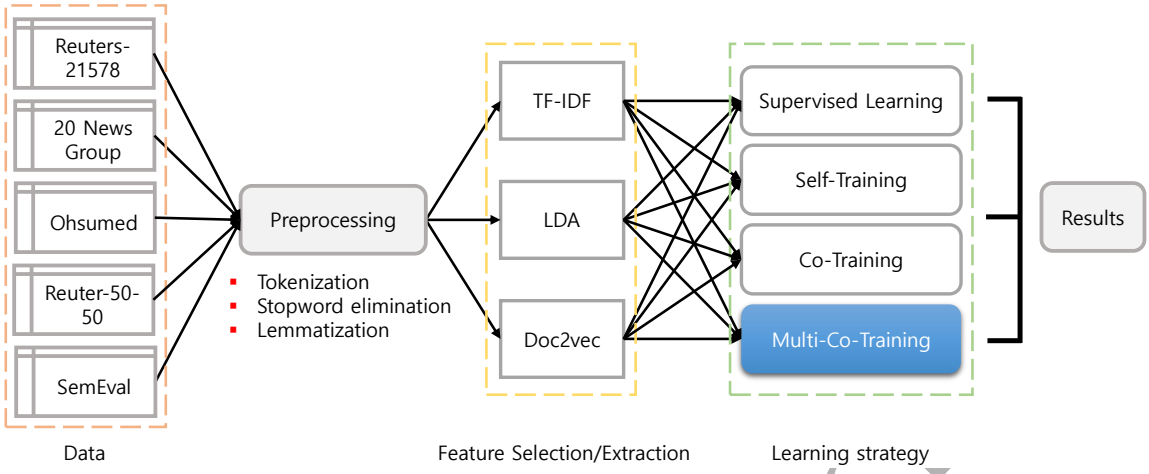


Figure 5: Procedure of comparative experiments for various document representation and classification methods.

the document representation is completed, the three classifiers are trained based only on the labeled examples using the corresponding feature set (line 11 in Algorithm 2). Then, the three classifiers predict the unlabeled documents and compute the confidence levels for their predictions (line 12 in Algorithm 2). The confidence level of a classifier for an example increases when the classifier is more accurate and the predicted class probability is not uniformly distributed (lines 13–17 in Algorithm 2). Once these highly confident unlabeled examples are identified, they are added to the labeled training set with the predicted class labels (lines 18–21 in Algorithm 2). We adjusted the number of label-assigned examples $\mathbf{N}$ to gradually improve the performance of the classifiers with regard to the iteration of γ (lines 11–21 in Algorithm 2).

4. Experiments

In order to verify the proposed MCT method, we conducted a comparative experiment as described in Figure 5. The same text-preprocessing techniques are applied to the five selected datasets. Each document in the datasets is transformed into three sets of vectors based on TF–IDF, LDA, and Doc2Vec. Based on the three document representation methods, four learning strategies are employed: pure SL, ST-based SSL, co-training, and the proposed MCT. Consequently, 10 strategies are compared as presented in Table 1. We also varied the dimensions of the document representation and the ratio of the labeled data.

Table 1: Learning strategies

Strategy	Description
SL–1	Purely supervised learning with TF–IDF
SL–1	Purely supervised learning with LDA
SL–3	Purely supervised learning with Doc2Vec
ST–1	Self-training with TF–IDF
ST–2	Self-training with LDA
ST–3	Self-training with Doc2Vec
CT–12	Co-training with TF–IDF and LDA
CT–13	Co-training with TF–IDF and Doc2Vec
CT–23	Co-training with LDA and Doc2Vec
MCT	Multi-co-training with TF–IDF, LDA, and Doc2Vec

Table 2: Data descriptions

Data	Description	No. of classes	No. of training/test set
Reuters-21587	21,578 documents obtained from the Reuters news data	10	77,690/30,180
Newsgroup	Data of 20,000 messages collected from 20 news categories	20	11,313/7,532
Ohsumed	Article-related abstracts of medical data corresponding to 23 diseases	23	10,433/12,733
Reuter-50-50	Dataset for author identification, consisting of the top 50 authors	50	2,500/2,500
SemEval	Tweets which contain positive, negative, or neutral sentiment	3	7,868/20,632

4.1. Data Description

Five popular datasets for document classification are selected: Reuters-21578[1], 20 Newsgroup[2], Ohsumed[3], Reuters-50-50[4] and SemEval[5], as described in Table 2. The length of the documents in the corpus varies from short sentences (SemEval) to long documents (Reuter-21587). All the datasets except Reuter-21587 were originally used for multi-class classification with predefined partition sets (training/test sets). For the Reuter-21587 dataset, we selected the ten most-frequently used categories, as used in [26].

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578
[2] http://qwone.com/~jason/20Newsgroups
[3] http://disi.unitn.it/moschitti/corpora.htm
[4] https://archive.ics.uci.edu/ml/machine-learning-databases/00217/C50.zip
[5] http://alt.qcri.org/semeval2017/task4/index.php?id=data-and-tools

Table 3: Experimental parameters

Parameters	Candidate values
Labeled examples	2%, 5%, 10%, 20%, 50%
Dimensions (No. of features)	8, 16, 32, 64, 100, 200, 300

4.2. Classification Algorithms and Parameter Settings

In this study, the Naïve Bayesian (NB) classification and random forests (RF) are adopted as the base classifiers. Both have been widely used for the text classification tasks [2, 7, 14] because of their computational efficiency and convenient implementation in real-world systems [29]. In contrast to other machine-learning-based classification algorithms such as artificial neural network and support vector machines, NB does not contain algorithm-specific parameters, which generally results in extensive computational burden for optimization. In NB, we assumed that each variable follows a normal distribution; we estimated the mean and standard deviation of each variable to infer the test documents. We also adopted Laplace smoothing. RF contains certain algorithm-specific parameters, e.g., the number of trees and the number of variables to be considered for each split. However, its performance is reasonably robust to changes in such parameters provided that a sufficiently large size of ensemble population is secured. We set the ensemble population to 100 in this study.

There are two significant data-dependent parameters that can affect the performance of classifiers: the ratio of the labeled examples and number of features (dimensions) for a document. In order to exploit the effect of SSL approaches under various labeling ratios, we tested five labeling ratios between 2% and 50%. Purely supervised classifiers are trained using only the labeled data, whereas SSL approaches use the unlabeled examples during training. In addition, we tested the effect of the representation dimension (number of features) by varying it from eight to 300. For LDA, we set $\alpha = 0.1$ and $\beta = 1$. The data-specific parameter settings are provided in Table 3.

4.3. Performance Measurement

We used two types of performance measurements, i.e., average F1 score of the positive and negative classes for the SemEval dataset and the break-even point of precision and recall of each class for the other datasets, to compare the results with those of other studies. The average F1 measure was adopted because it was the primary performance measure used by

Table 4: The confusion matrix. P, U, and N denote positive, neutral, and negative, respectively

		Predicted class		
		Positive	Neutral	Negative
True class	Positive	PP	PU	PN
	Neutral	UP	UU	UN
	Negative	NP	NU	NN

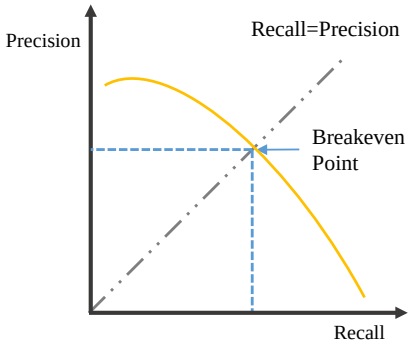


Figure 6: Break-even point of recall and precision

the competition, with the SemEval dataset. The class-wise break-even point of precision and recall was used in [26] for the other datasets. It is noteworthy that the former is a dataset-level performance measure, whereas the latter is a class-level performance measure.

For the SemEval dataset, three classes, i.e., positive, neutral, and negative, are present. Thus, the confusion matrix of a classifier can be summarized as in Table 4. The average F1 measure for the positive and negative classes can be computed using Eq. (12–15):

$$Recall^P = \frac{PP}{PP + PU + PN} \quad , \quad Precision^P = \frac{PP}{PP + UP + NP}, \tag{12}$$

$$Recall^N = \frac{NN}{NP + NU + NN} \quad , \quad Precision^N = \frac{NN}{PN + UN + NN}, \tag{13}$$

$$F_1^P = \frac{2 \times Recall^P \times Precision^P}{Recall^P + Precision^P} \quad , \quad F_1^N = \frac{2 \times Recall^N \times Precision^N}{Recall^N + Precision^N}, \tag{14}$$

$$F_1^{PN} = \frac{1}{2}(F_1^P + F_1^N). \tag{15}$$

Figure 6 illustrates the break-even point of precision and recall. Because a trade-off exists between recall and precision, we obtained the cut-off for each class wherein the precision and recall becomes equivalent: the higher the value is, the more effective the classifier is.

In order to obtain statistically reliable results, we repeated the process 10 times, as il-

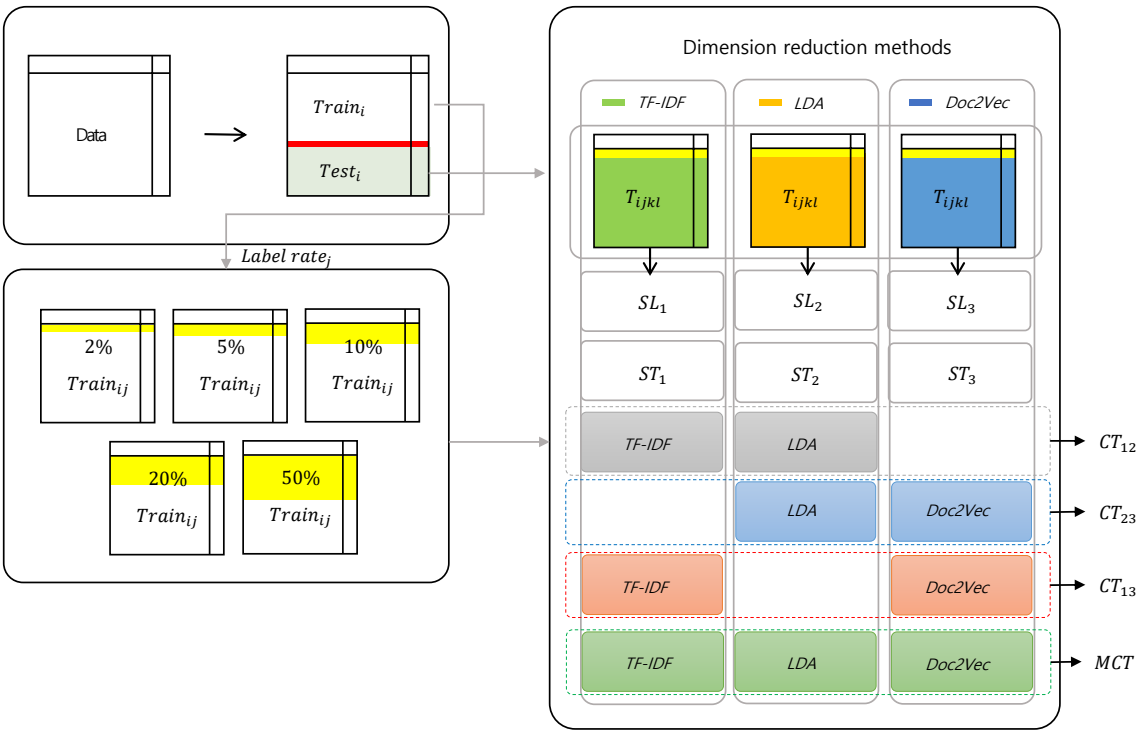


Figure 7: Experimental process for measuring the average and standard deviation of the classification performance for each case. $\mathbf{SL}_{ijkl}$, $\mathbf{ST}_{ijkl}$, and $\mathbf{MCT}_{ijk}$ denote the classification model for the i^{th} random partition, j^{th} labeling rate, k^{th} feature dimension, and l^{th} feature-representation method for SL, ST, and the proposed MCT, respectively.

lustrated in Figure 7, and computed the average and standard deviation of the performance measures. First, we obtained the entire dataset that is already divided into the predefined partition set. In the second step, only a portion of the labels are preserved (2%, 5%, 10%, 20%, and 50%). In the third step, each document in the training set is transformed into different feature vectors according to the dimensions optimized by the 10-fold cross-validation with each classifier. It should be noted that TF–IDF, LDA, and Doc2Vec are constructed or trained based on the training dataset before sampling the labeled examples. That is, the representation vector of a single document for a specific representation method and dimension is identical during the repeated process. Finally, classification models based on SL, ST, and the proposed MCT are trained, and their classification performances are evaluated using the predefined test set.

Table 5: Optimized document representation dimensions for each dataset-classifier pair

Data	NB			RF		
	TF-IDF	LDA	Doc2Vec	TF-IDF	LDA	Doc2Vec
Reuters-21587	200	100	32	200	100	16
Newsgroup	300	64	100	300	64	16
Ohusumed	200	32	64	200	100	16
Reuters-50-50	200	32	64	200	100	16
SemEval	300	100	32	200	100	32

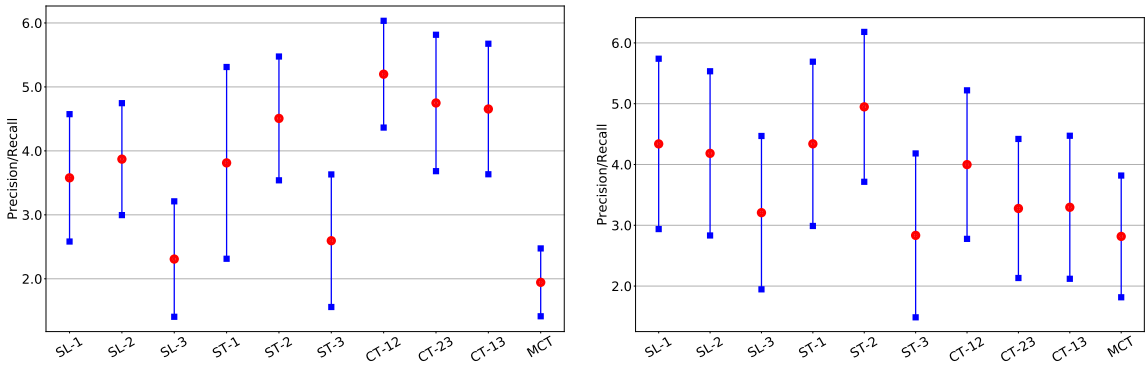
Table 6: Ratio of hypotheses rejected at the significance level of $\alpha = 0.05$ for correlation between two feature sets

Features	Reuter-50-50	Reuters-21578	SemEval	Newsgroups	ohsumed
(TF-IDF, LDA)	0.004	0.046	0.008	0.008	0.013
(LDA, Doc2Vec)	0.000	0.009	0.000	0.001	0.001
(TF-IDF, Doc2Vec)	0.000	0.001	0.000	0.000	0.000

5. Result

Table 5 summarizes the number of feature dimensions optimized by the 10-fold cross-validation. Generally, TF–IDF requires the highest dimension, followed by LDA and Doc2Vec. This result is straightforward in that although TF–IDF selected the most significant terms for classification tasks, it has a more sparse representation than the other two document representation methods: numerous feature values can be zero in TF–IDF. Meanwhile, LDA and Doc2Vec learn the distributed representation, and they can contain more information with less number of features. The optimal dimensions of LDA and Doc2Vec are dependent on the dataset and classification algorithms. RF generally requires a lower dimension of Doc2Vec compared to LDA.

One of the primary premises of co-training and MCT is that the different feature sets are independent of each other. Therefore, certain useful information obtained by a feature set is delivered to the other feature sets to improve the overall performance. To verify this assumption, we conducted the hypothesis test for the Pearson correlation adjusted by the Bonferroni correction between two feature sets, as presented in Table 6. The null hypothesis (H_0) is that the correlation between the two feature sets is zero, i.e., the two feature sets are not correlated. The alternative hypothesis (H_1) is that the correlation between the two feature sets is not zero. i.e., the two feature sets are more or less correlated. The number in



(a) Naive Bayesian Classifier

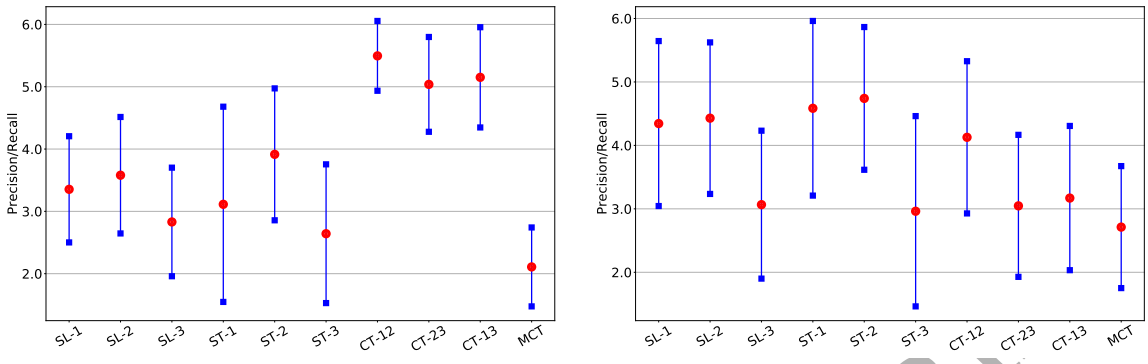
(b) Random Forest

Figure 8: Average rank and its standard deviation for all learning strategies with different classification algorithms.

each cell is the ratio of the rejected hypotheses for each dataset with the two feature sets. The lower the value is, the more strongly the feature independence is supported. TF–IDF and Doc2Vec can be considered as independent because the rejection ratio is smaller than 0.001 for the five datasets. Although a few rejection rates between LDA and Doc2Vec and between TF–IDF and LDA are relatively higher than those of the other cases, they are smaller than 0.05; this indicates the independence between the two feature sets.

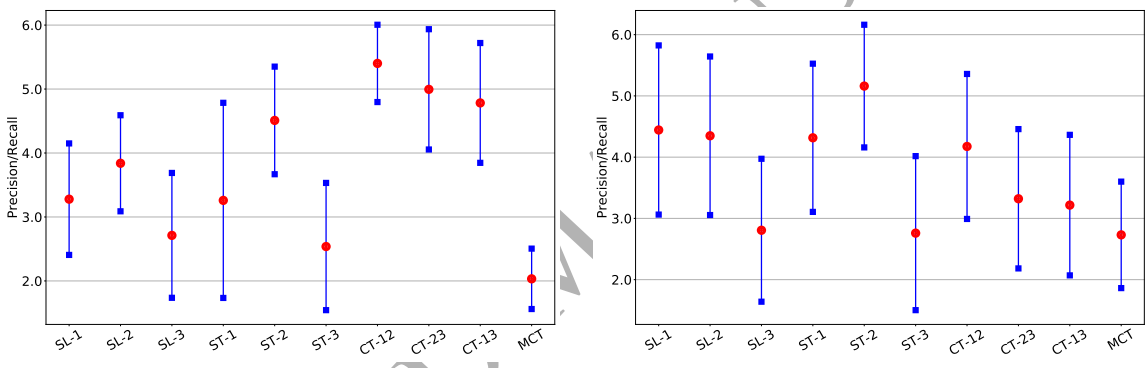
For each learning strategy, we trained the 10 classification models by varying the classification algorithms (NB and RF) and label ratios (2%, 5%, 10%, 20%, and 50%). In addition, the class-level performance metrics are computed for the four datasets apart from the SemEval dataset. For example, we have 10 break-even points of recall and precision for Reuters-21587, which results in 100 performance metric values for each training strategy for the dataset. Because these performance values vary across classes, a simple average of other summary statistics would not be appropriate to compare the MCT with other learning strategies. Hence, we compared the rank of the performance metric values of each strategy under a similar condition. The raw data to draw the following figures are provided in the supplementary material.

Figure 8 shows the average rank and the standard deviation for each learning strategy for NB and RF. As demonstrated above, these rank plots are drawn based on 520 cases for each learning strategy. The lower the rank average value is, the higher is the classification performance. In addition, the narrower the standard deviation bar, the more stable is the classification model. Among the feature representation methods, Doc2Vec resulted in the



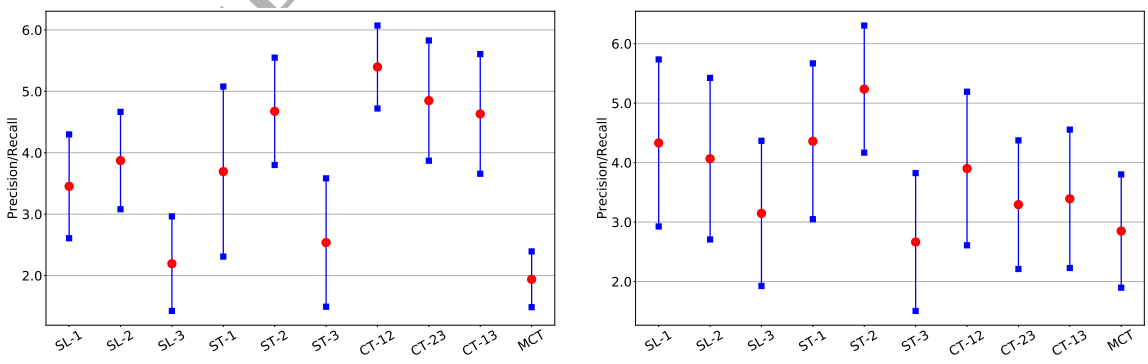
(a) Naive Bayesian Classifier (b) Random Forest

Figure 9: Average rank and its standard deviation for all learning strategies with `0.02` label ratio and classifiers.



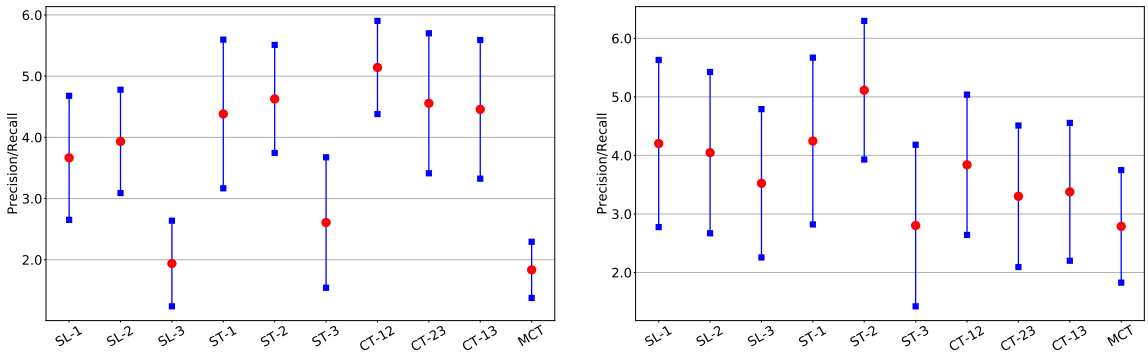
(a) Naive Bayesian Classifier (b) Random Forest

Figure 10: Average rank and its standard deviation of all learning strategies with `0.05` label ratio and classifiers.



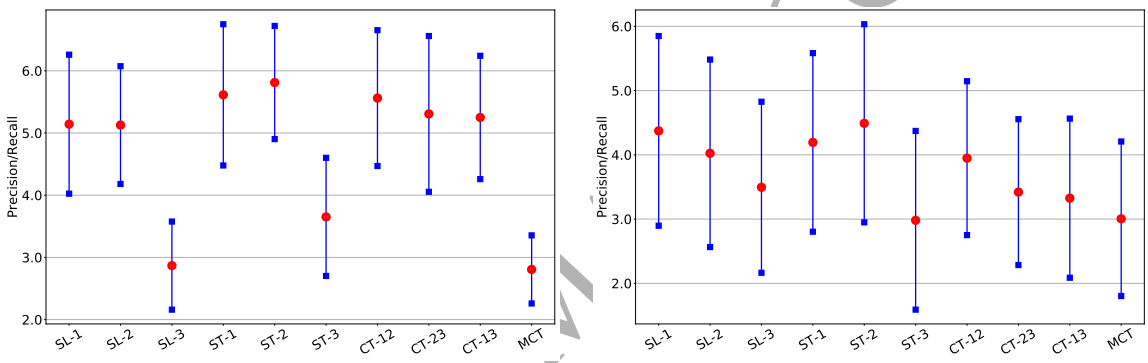
(a) Naive Bayesian Classifier (b) Random Forest

Figure 11: Average rank and its standard deviation for all learning strategies with `0.1` label ratio and classifiers.



(a) Naive Bayesian Classifier (b) Random Forest

Figure 12: Average rank and its standard deviation for all learning strategies with `0.2` label ratio and classifiers.



(a) Naive Bayesian Classifier (b) Random Forest

Figure 13: Average rank and its standard deviation of all learning strategies with `0.5` label ratio and classifiers.

highest classification performance, whereas TF–IDF and LDA yielded similar performances. A noteworthy observation is that when a simple classification algorithm is employed, conventional SSL methods did not improve the classification performance in numerous cases. For example, with the NB classifier, the average rank of SL–1 (self-training with TF–IDF features) is marginally lower than that of ST–1, and the average rank of CT–12 is higher than those of SL–1 and SL–2. Similar relationships are observed for the other features. Meanwhile, with a relatively sophisticated classification algorithm, co-training facilitated in either improving the classification performance or rendering the performance more stable.

Figure 8 illustrates that the proposed MCT enhanced the classification performance of conventional SSL approaches. The performance improvement is more apparent with the NB classifier: its average rank is the second among those of the 10 learning strategies, and the

standard deviation is approximately one. Although the performance improvement of the MCT for RF is not as significant as that for NB, its average rank was the lowest among those of the learning strategies and had the smallest rank variations.

Because Figure 8 provides the most abstracted information, we compared the performance ranks with respect to the class label ratio, as shown in Figures 9, 10, 11, 12, and 13. For both NB and RF, the proposed MCT achieved the highest classification performance for all the class label ratios. Other observations are as follows: when the class label ratio is very low, e.g., 2% or 5%, the co-training degenerated the classification performance when NB was used as the base classifier. In addition, with the NB classifier, self-training with TF–IDF representation becomes unstable because the standard deviation of ST–1 is larger than that of SL–1. When RF was used as the base classifier, the performance ranking does not significantly change with respect to the class labeling ratio. MCT resulted in the highest performance for most labeling ratios, followed by ST–3. In addition, the performance rankings change more frequently using RF; therefore, the average rank of a certain learning strategy is typically higher compared to that of NB and the standard deviations are also larger compared to that of NB.

6. Conclusion

In this paper, we propose MCT for improving document classification accuracy. Three document representation methods, i.e., TF–IDF, LDA, and Doc2Vec, are employed for transforming an unstructured document into a real-valued vector. As a larger number of unlabeled text documents than labeled documents are present in real-world problems, we adopt an SSL scheme for altering the classification model by using available albeit unlabeled data. The experimental results verify that the proposed MCT can achieve a superior and more robust classification performance than the traditional SL- or ST-based SSL, particularly under harsh circumstances, e.g., when a document is transformed into a very-low-dimensional vector and the labeled documents are very few.

The current study has certain limitations, which guides us in identifying future research directions. Firstly, although the classes in the datasets are not uniformly distributed, the class imbalance ratios are not very high. In certain applications, there are only a small number of documents for a certain class, whereas abundant examples are available for other classes. It is likely to be effective to test the proposed MCT under these challenging circumstances. Next,

the iterative algorithm used in our experiment can be burdensome in terms of computational complexity. Hence, efficient learning methods such as early stopping or removing redundant examples should be explored.

Acknowledgment

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A1B03930729) and Institute for Information & communications Technology Promotion (IITP) grant funded by the Korean government (MSIP) (No. 2017-0-00349, Development of Media Streaming system with Machine Learning using QoE (Quality of Experience))

References

[1] A. Amin, C. Khan, I. Ali, S. Anwar, Customer churn prediction in telecommunication industry: With and without counter-example, in: 2014 European Network Intelligence Conference, pp. 134–137.

[2] W.T. Aung, Y. Myanmar, K.H.M.S. Hla, Random forest classifier for multi-category classification of web pages, in: Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific, IEEE, pp. 372–376.

[3] I. Bíró, J. Szabó, A.A. Benczúr, Latent dirichlet allocation in web spam filtering, in: Proceedings of the 4th international workshop on Adversarial information retrieval on the web, ACM, pp. 29–32.

[4] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, Journal of machine Learning research 3 (2003) 993–1022.

[5] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proceedings of the eleventh annual conference on Computational learning theory, ACM, pp. 92–100.

[6] H. Borko, M. Bernick, Automatic document classification, Journal of the ACM (JACM) 10 (1963) 151–162.

[7] M.R. Bouguelia, Y. Belaïd, A. Belaïd, A stream-based semi-supervised active learning approach for document classification, in: Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, IEEE, pp. 611–615.

[8] O. Chapelle, B. Schlkopf, A. Zien, Semi-Supervised Learning, The MIT Press, 1st edition, 2010.

[9] G. Druck, C. Pal, A. McCallum, X. Zhu, Semi-supervised classification with hybrid generative/discriminative methods, in: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07, ACM, New York, NY, USA, 2007, pp. 280–289.

[10] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in: Proceedings of the 28th international conference on machine learning (ICML-11), pp. 513–520.

[11] A. Go, R. Bhayani, L. Huang, Twitter sentiment classification using distant supervision, CS224N Project Report, Stanford 1 (2009) 12.

[12] B.S. Harish, D.S. Guru, S. Manjunath, Representation and classification of text documents: A brief review, IJCA, Special Issue on RTIPPR (2010) 110–119. Published By Foundation of Computer Science.

[13] A. Khan, B. Baharudin, L.H. Lee, K. Khan, A review of machine learning algorithms for text-documents classification, Journal of advances in information technology 1 (2010) 4–20.

[14] S.B. Kim, K.S. Han, H.C. Rim, S.H. Myaeng, Some effective techniques for naive bayes text classification, IEEE transactions on knowledge and data engineering 18 (2006) 1457–1466.

[15] J.H. Lau, T. Baldwin, An empirical evaluation of doc2vec with practical insights into document embedding generation, arXiv arXiv:1607.05368 (2016).

[16] Q.V. Le, T. Mikolov, Distributed representations of sentences and documents., in: ICML, volume 14, pp. 1188–1196.

[17] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).

[18] K. Nigam, R. Ghani, Analyzing the effectiveness and applicability of co-training, in: Proceedings of the ninth international conference on Information and knowledge management, ACM, pp. 86–93.

[19] K. Nigam, A.K. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using em, Machine learning 39 (2000) 103–134.

[20] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up?: sentiment classification using machine learning techniques, in: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics, pp. 79–86.

[21] Z. Qiu, B. Wu, B. Wang, C. Shi, L. Yu, Collapsed gibbs sampling for latent dirichlet allocation on spark, Journal Machine Learning Research 36 (2014) 17–28.

[22] M.N.M. Ranjan, Y.R. Ghorpade, G.R. Kanthale, A.R. Ghorpade, A.S. Dubey, Document classification using lstm neural network, Journal of Data Mining and Management 2 (2017).

[23] S. Robertson, Understanding inverse document frequency: on theoretical arguments for idf, Journal of documentation 60 (2004) 503–520.

[24] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-supervised self-training of object detection models., in: WACV/MOTION, IEEE Computer Society, 2005, pp. 29–36.

[25] T. Sabbah, A. Selamat, M.H. Selamat, F.S. Al-Anzi, E.H. Viedma, O. Krejcar, H. Fujita, Modified frequency-based term weighting schemes for text classification, Applied Soft Computing 58 (2017) 193–206.

[26] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, Journal of machine learning research 2 (2001) 45–66.

[27] D. Wang, M. Thint, A. Al-Rubaie, Semi-supervised latent dirichlet allocation and its application for document classification, in: Proceedings of the The 2012 IEEE/WIC/ACM

International Joint Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03, IEEE Computer Society, pp. 306–310.

[28] C. Xing, D. Wang, X. Zhang, C. Liu, Document classification with distributions of word vectors, in: Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific, IEEE, pp. 1–5.

[29] S. Xu, Bayesian naïve bayes classifiers to text classification, Journal of Information Science (2016) 0165551516677946.

[30] Z. Yun-tao, G. Ling, W. Yong-cheng, An improved tf-idf approach for text classification, Journal of Zhejiang University Science A 6 (2005) 49–55.

[31] W. Zhang, T. Yoshida, X. Tang, A comparative study of tf* idf, lsi and multi-words for text classification, Expert Systems with Applications 38 (2011) 2758–2765.

[32] X. Zhu, Semi-supervised Learning with Graphs, Ph.D. thesis, Pittsburgh, PA, USA, 2005. AAI3179046.