TFIDF meets deep document representation : a re-visit of co-training for text classification

Chen, Zhiwei

2020

https://hdl.handle.net/10356/138643

TFIDF meets Deep Document Representation: A Re-Visit of Co-Training for Text Classification

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor of Computer Science of the Nanyang Technological University

by

Chen Zhiwei

Abstract

Many text classification tasks face the challenge of lack of sufficient labelled data. Co-training algorithm is a candidate solution, which learns from both labeled and unlabelled data for better classification accuracy. However, two sufficient and redundant views of an instance are often not available to fully facilitate co-training in the past. With the recent development of deep learning, we now have both traditional TFIDF representation and deep representation for documents. In this paper, we conduct experiments to evaluate the effectiveness of co-training with different combinations of document representations (*e.g.,* TFIDF, Doc2vec, ELMo, BERT) and classifiers (*e.g.,* SVM, Random Forest, XGBoost, MLP, and CNN) on two benchmark datasets (20 Newsgroup and Ohsumed). Our results show that co-training with TFIDF and deep contextualised representation offers improvement to classification accuracy.

Acknowledgements

I would like to express my sincere gratitude towards Associate Professor Sun Aixin for his kind guidance and encouragement which helped me in the completion of this project.

Contents

List of Tables

List of Figures

1 Introduction

We often face the situation of inadequate labeled data to learn accurate text classification models. It is often expensive to obtain more labeled data. Among many possible solutions, the co-training algorithm is a solution designed to utilize both labeled and unlabeled data. Co-training assumes that each instance is described by two "views" (or two feature sets) Blum and Mitchell [1998]. The two views are expected to be conditionally independent and each view is sufficient to predict an instance's class label. With a small number of labeled examples, two weak classifiers are first trained, each trained with one view. Both weak classifiers are used to predict the labels of unlabeled instances and the instances with most confident predictions are added as labeled instances to learn better classifiers. The co-training algorithm shows effectiveness in web page classification where a web page can be described by its content and also the anchor words pointing to the web page.

In practice, it is however often hard to find two views of an instance which are conditionally independent and both are sufficient for classification. Traditionally, a document is typically represented by Bag-of-Words (BoW) model in a vector space. Each word (*i.e.,* a feature) is weighted by TFIDF scheme (*i.e.,* term frequency $\times$ inverse document frequency). For easy presentation, we refer to this document representation as TFIDF. Recently, with the rapid development of representation learning, we are now offered multiple different views of a document in low dimensional embedding spaces. Examples range from Word2Vec based representations to deep contextualized representations like Embeddings from Language Models (ELMo) Peters et al. [2018] and Bidirectional Encoder Representations from Transformers (BERT) Devlin et al. [2019].

Both TFIDF and word embedding based representations are sufficient to represent a document and they are from different feature spaces. This motivate us to re-visit co-training algorithm, to explore whether the two sets of document representations facilitate us to utilize unlabeled documents in text classification. To this end, we conduct experiments on two benchmark datasets, namely 20Newsgroup and Ohsumed, using five different document representations: TFIDF, Doc2Vec Le and Mikolov [2014] which is based on pre-trained word2vec embeddings, and three contextualized representations, *i.e.,* ELMo, BERT, and Universal Sentence Encoder (USE) Cer et al. [2018]. The two weak classifiers for each co-training setting, are selected from Support Vector Machine (SVM), Random Forest (RF), XGBoost (XGB), Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN).

Experimental results suggest that deep contextualised representations together with the right choice of classification models is likely to achieve the best results. Nevertheless, TFIDF+SVM remains a strong baseline. Combination of different document representations and classification models give very diverse results, particularly when there is a lacking of sufficient training data. The different document representations do facilitate co-training to achieve significantly better classification performance when both weak classifiers could give relatively good performance with limited labeled data. On the other hand, on challenging dataset, when the weak classifiers could not achieve reasonable accuracy with small amount of training data, co-training leads to degradation in final predictions.

2 Related Work

Co-training has been applied to web page classification in the original paper (see Figure 1). The algorithm has also been applied to classify email using header and body as the two views Kiritchenko and Matwin [2011]. Co-training has also been extended to tri-training Zhou and Li [2005] (see Figure 2). Compared to co-training, tri-training does not requires the assumption of sufficient and redundant views of the data, instead, it trains its initial three weak-classifiers using bootstrapped labeled data.

Figure 1: Co-training

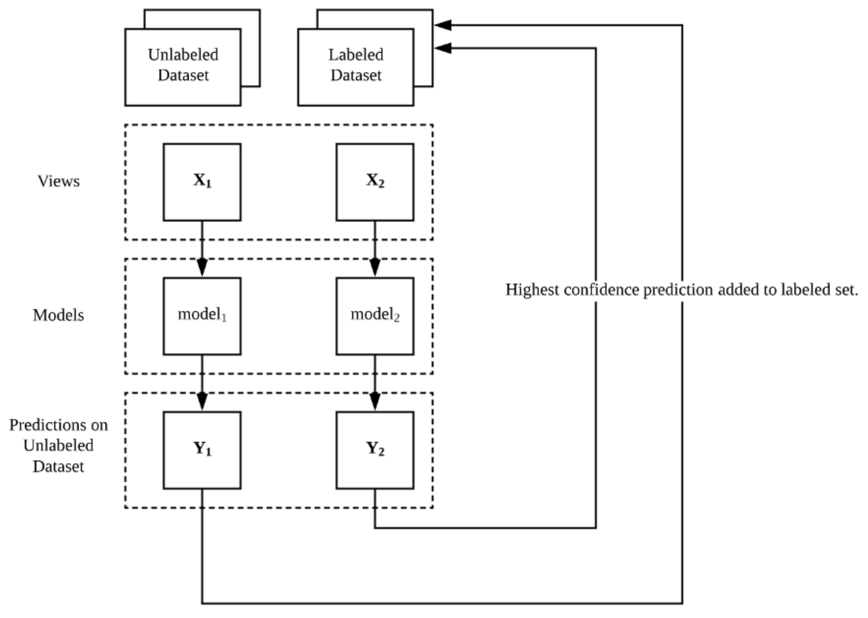
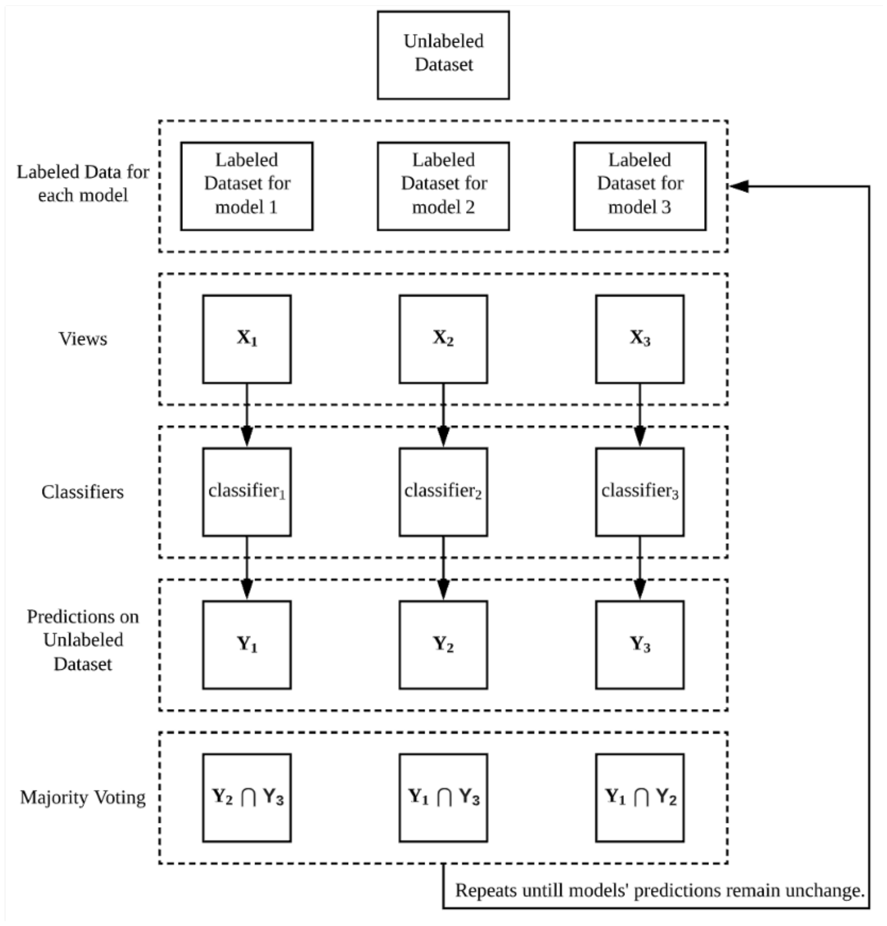
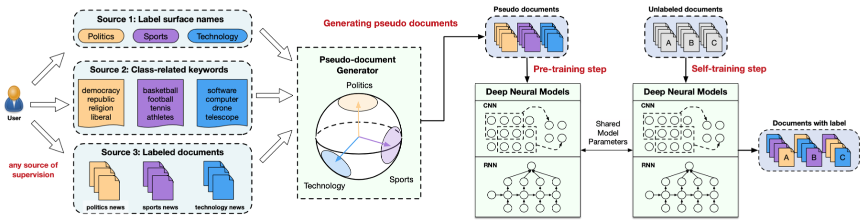

Unlabeled Dataset
Labeled Dataset
Views
X_1
X_2
Models
$model_1$
$model_2$
Predictions on Unlabeled Dataset
Y_1
Y_2
Highest confidence prediction added to labeled set.

Figure 2: Tri-training


Unlabeled Dataset
Labeled Data for each model
Labeled Dataset for model 1
Labeled Dataset for model 2
Labeled Dataset for model 3
Views
X_1
X_2
X_3
Classifiers
$classifier_1$
$classifier_2$
$classifier_3$
Predictions on Unlabeled Dataset
Y_1
Y_2
Y_3
Majority Voting
$Y_2 \cap Y_3$
$Y_1 \cap Y_3$
$Y_1 \cap Y_2$
Repeats untill models' predictions remain unchange.

Another semi-supervised learning algorithm that is related to co-training is named the **WeSTClass**, by Meng et al. [2018] (see Figure 3). WeSTClass uses seed information (such as label surface names, class-related keywords, or labeled documents themselves) to generate pseudo documents, and then use the pseudo documents to train the initial weak-classifier. Unlike the aforementioned tri-training and co-training algorithm, the WeSTClass uses only the unlabelled data to self-train the initial weak classifier.

Figure 3: WeSTClass

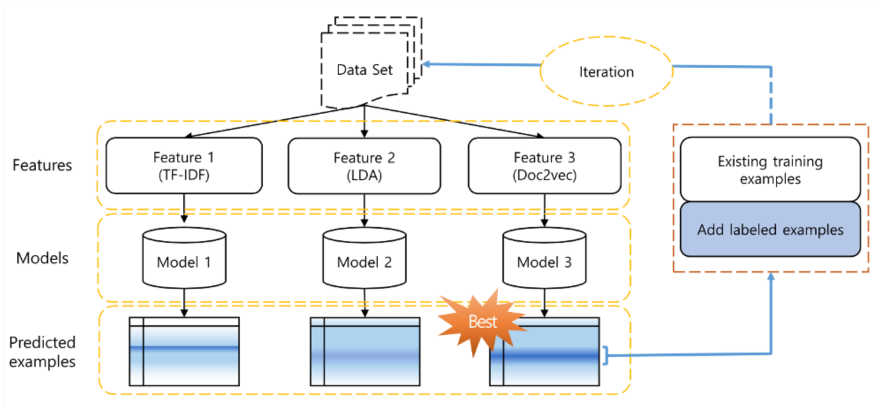

Source 1: Label surface names
Politics
Sports
Technology
Source 2: Class-related keywords
democracy republic religion liberal
basketball football tennis athletes
software computer drone telescope
Source 3: Labeled documents
politics news
sports news
technology news
User
any source of supervision
Generating pseudo documents
Pseudo-document Generator
Politics
Technology
Sports
Pseudo documents
Pre-training step
Deep Neural Models
CNN
RNN
Shared Model Parameters
Unlabeled documents
A
B
C
Self-training step
Deep Neural Models
CNN
RNN
Documents with label
A
B
C

For the text classification task, co-training has also been applied. For instance, in **email classification with co-training** Kiritchenko and Matwin [2011], the header and body of a email are used to represent the two sufficient and redundant representations of the email text. In addition, GhaniGhani [2002] had also proposed a algorithm than combined Error Correcting Output Coding**?** and co-trainingBlum and Mitchell [1998] for multi-class text classification. Clustering and co-trainingRaskutti et al. [2002] used body text and features derived from clustering labeled and unlabeled data as the two sufficient and redundant views.

There is also a work named **vertical ensemble solution** Katz et al. [2018] that performs binary text classification. The authors of vertical ensemble solution proposed a vertical ensemble solution that uses the models created from earlier training iterations, whereby for each training iteration, the earlier models trained with less labeled data will also be used in the current iteration to rank the predictions on the unlabeled data. The intuition of this vertical ensemble solution is to provide more variation among different models, so as to ensure more information is used when labeling unlabeled data.

More relevant to our work is multi-co-training Kim et al. [2019] (see Figure 4). The authors exploit three document representations (TFIDF, LDA, and Doc2Vec) for multi-class text classification. The proposed method is similar to tri-training, where three weak classifiers are trained each with one document representation, by the same classification model. The authors evaluated Naïve Bayes and Random Forest as base classifiers. However, the objective of multi-co-training is to maximise the benefits of different document representations. Our work is to systematically evaluate the effectiveness of co-training with combinations of 5 document representations and 5 classification models.

Figure 4: Multi-co-training



3 Project Schedule

The project schedule served the purpose of project planning and milestone tracking. The schedule is presented in table 1.

Table 1: Project Schedule

Week	**Plan**	**Remarks**
Week 3	- Submit interim report. - Incorporate Universal Sentence Encoder, ELMo, BERT.	- Monday - A deep learning way to embed a sentence.
Week 4	Experiment on Ohsumed dataset	
Week 5	Consolidate all findings, and start on report writing	
Week 6	Continue report writing	
Week 7	Continue report writing	
Recess	Report/poster/video	
Week 8	Report/poster/video	
Week 9	Report/poster/video	
Week 10	Submit final report/poster/video	Monday
Week 11	Continue report writing	
Week 12	Continue report writing	
Week 13	Submit amended final report	Friday
Week 14 and beyond	Prepare for oral presentation	Presentation on 12th May

4 Methodology

In the original paper Blum and Mitchell [1998], co-training was introduced for a binary classification task. In our experiments, both datasets are for multi-class text classification. Before reporting our results, we briefly describe the co-training setting in the original paper Blum and Mitchell [1998] as Algorithm 1, as well as the modified version in our experiments as Algorithm 2.

The original co-training algorithm is as shown in Algorithm 1. In the experiment of Blum and Mitchell [1998], both h_1 and h_2 were trained using a naive Bayes algorithm, and $p = 1$, $n = 3$, $k = 30$ and $u = 75$.

Algorithm 1: Original Co-training Algorithm

```
Given:
    • a set $L$ of labeled training examples
    • a set $u$ of unlabeled training examples
Create a pool $U'$ of examples by choosing $u$ examples at random from $u$
  Loop for $k$ iterations:
    • Use $L$ to train a classifier $h_1$ that considers the $x_1$ portion of x
    • Use $L$ to train a classifier $h_2$ that considers the $x_2$ portion of x
    • Allow $h_1$ to label $p$ positive and $n$ negative examples from $U'$
    • Allow $h_2$ to label $p$ positive and $n$ negative examples from $U'$
    • Add these self-labeled examples to L
    • Randomly choose 2p + 2n examples from $u$ to replenish $U'$
Define classifier $h3$ by multiplying outputs of $h_1$ and $h_2$
```

Let L denote the set of labeled data; let U denotes the set of unlabeled data. Let $U' \in U$ denote a subset of U with size $|U'|$. Recall that we evaluate co-training with 5 different document representations and 5 classifiers. In each co-training setting, we choose two document representations x_1 and x_2, and two classification

Algorithm 2: Modified Co-training Algorithm

```
Given:
    • a set $L$ of labeled training examples
    • a set $u$ of unlabeled training examples
Create a pool $U'$ of examples by choosing $u$ examples at random from $u$
Loop for $k$ iterations:
    • Use $L$ to train a classifier $h_1$ that considers the $x_1$ portion of x
    • Use $L$ to train a classifier $h_2$ that considers the $x_2$ portion of x
    • Allow $h_1$ to make predictions on $U'$, and label top $p$ examples from $U'$
      according to the confidence score
    • Randomly choose $p$ examples from $u$ to replenish $U'$
    • Allow $h_2$ to make predictions on $U'$, and label top $p$ examples from $U'$
      according to the confidence score
    • Randomly choose $p$ examples from $u$ to replenish $U'$
    • Add these self-labeled examples to $L$
Define classifier $h_3$ by multiplying outputs of $h_1$ and $h_2$
```

models. Then all labeled instances in L are used to train two weak classifiers h_1 and h_2, using representations x_1 and x_2 respectively. The trained weak classifier h_1 then predicts the category labels of documents in U', and the top p documents with the highest confidence scores from U' are added to L as labeled data. The same applies to h_2. The instances that have been added to L will be removed from U', and U' is replenished with remaining data from U. This is one iteration. The process repeats for k iterations.

After k iterations, h_1 and h_2 are trained with the updated L. Then both h_1 and h_2 make predictions on the test data. A combined classifier h_3 is then defined, which makes prediction basing the outputs of h_1 and h_2. In our implementation h_3 estimates probability $P(c_j|x)$ of class c_j given the instance $x = (x_1, x_2)$ by multiplying the probabilities derived from outputs h_1 and h_2.

4.1 Document Representation

In our evaluation, a document can have the following representations: $x \in \{TFIDF, Doc2Vec, ELMo_s, ELMo_p, USE, BERT_s, BERT_p\}$. Among them, TFIDF is traditional vector space representation, Doc2Vec is based on pre-trained word2vec, where the embedding for a word is fixed. ELMo, USE, and BERT are deep contextualize representations, where the embeddings of the same word could be different depending on the context of the word.

TFIDF represents a document as a feature vector where each word is one feature and is weighted by its TFIDF value (see Figure 5). The essence of TFIDF is to weight the n-grams in a document according two criteria. First is Term Frequency (TF), which is the frequency of the n-gram appearing within the document. The higher the TF of the n-gram, the higher the weight is given. Second is Inverse Document Frequency (IDF), which is referring to the frequency of the n-gram appearing across the corpus of documents. As suggested by the term "Inverse", the higher the document frequency of a n-gram, the lower the weight is given. We use scikit-learn's TFIDFVectorizer[1] to derive TFIDF vectors for documents.

Figure 5: Term Frequency Inverse Document Frequency

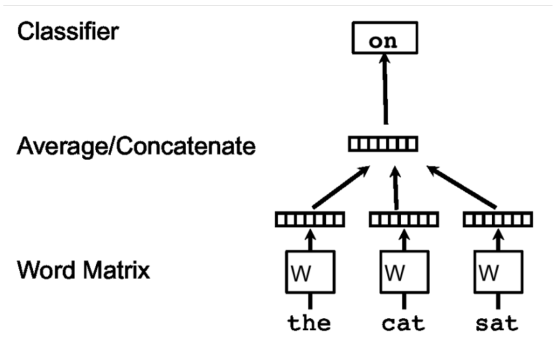
$$\mathbf{tfidf}_{i,j} = \mathbf{tf}_{i,j} \times \log\left(\frac{\mathbf{N}}{\mathbf{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = total number of occurences of i in j
df_i = total number of documents (speeches) containing i
N = total number of documents (speeches)

Doc2Vec is an extension of Word2Vec Le and Mikolov [2014]; Mikolov et al. [2013] (see Figure 6). A word vector intends to capture the concept of a word in a vector, while a document vector intends to capture the concept of a document in a vector. Although Doc2Vec was designed to make use of some context information during training, Doc2Vec is still differentiated from deep contextualised representations. The reason for such differentiation is that during inference, Doc2Vec embed words using fixed word vectors, which does not demonstrate the ability to differentiate a word's context in different scenarios. We adopt pre-trained Doc2Vec embeddings[2] where the word vectors are learned from Wikipedia. Document vector dimension is 300.

[1]`https://scikit-learn.org/` Class: sklearn.feature_extraction.text.TfidfVectorizer
[2]`https://github.com/jhlau/Doc2Vec`

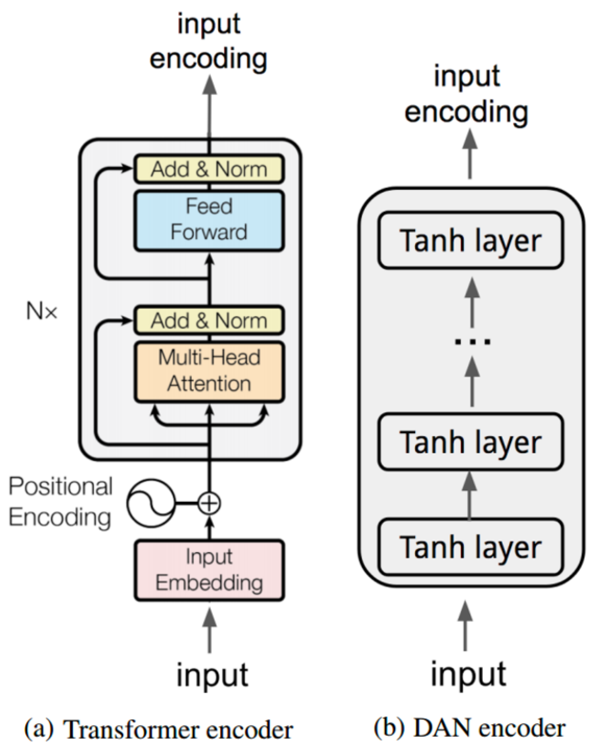
Figure 6: Doc2Vec



USE. For Universal-sentence-encoder there are two different models: transformer which extends from Vaswani et al. [2017] and deep averaging network (DAN) (see Figure 7). Both models take English strings as input and gives a fixed 512-dimension embedding as the output. Transformer has a more complex structure as it uses the encoding sub-graph of the transformer architecture for sentence embeddings. It involves more parameters and is more time-consuming for training. As return, high accuracy is achieved. On the other hand, DAN has a simpler structure and only has two hidden layers. It involves fewer parameters and is more resource-efficient, but the result is generally not as good as the transformer. We adopt pre-trained USE model with transformer architectrure.[3] Each document is represented in a 512 dimension vector.

[3]`https://tfhub.dev/google/universal-sentence-encoder/4`

Figure 7: Universal Sentence Encoder

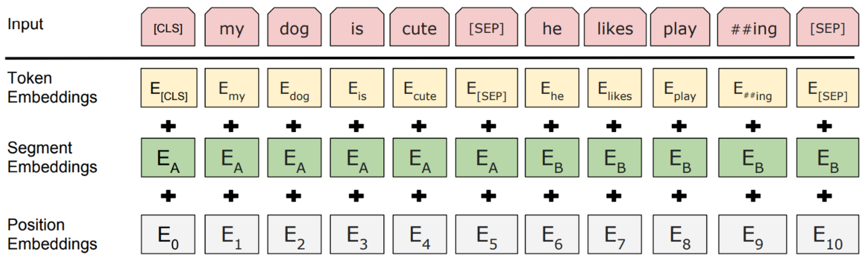


BERT applies bi-directional training of transformer to language modelling Devlin et al. [2019]; Vaswani et al. [2017] (see Figure 8). We used pre-trained BERT-base[4] for 20 Newsgroup. BERT-base generates two kinds of embeddings: one is *sequence* output, a 128 x 768 matrix for each document (document length is capped at 128), the other is *pooled* output, where a document is represented by a 768 dimension vector. Because Ohsumed dataset is in medical domain, we use pre-trained **BioBERT** model[5]. BioBERT is pre-trained on large-scale biomedical corpora Lee et al. [2019].

[4]`https://tfhub.dev/google/bert\_uncased\_L-12\_H-768\_A-12/1`

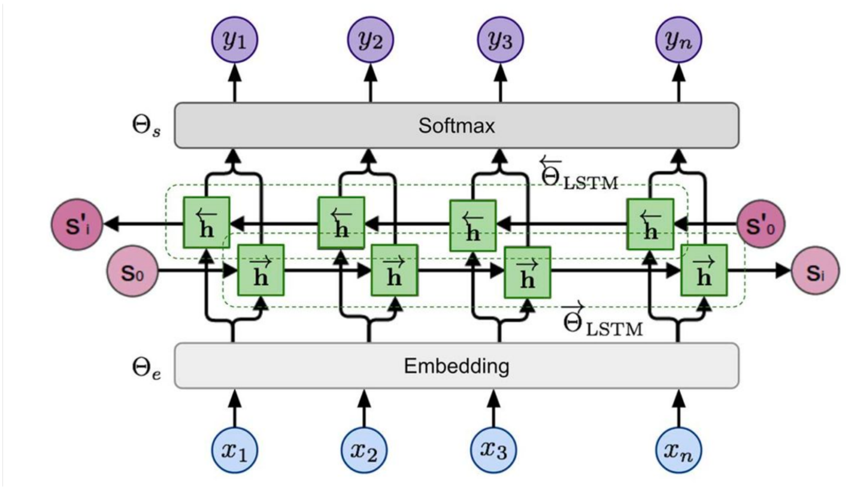
[5]`https://github.com/naver/biobert-pretrained`

Figure 8: Bi-directional Encoder Representations from Transformers


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

ELMo is also a language model that gives contextualised word embeddings Peters et al. [2018] (see Figure 9). It also uses bi-directional LSTM to assign each word with an embedding after looking at the entire input sequence. We use pre-trained ELMo model[6]. Similar to BERT, this model also generates two kinds of embeddings (sequence and pooled), except that the word dimension is 1028 instead of 768.

Figure 9: Embeddings from Language Models



Every document in 20 News Group and Ohsumed is first featurized into different

[6]`https://tfhub.dev/google/elmo/3`

document representations, as summarised in Table 2.

Table 2: Document Representations and Dimensions

Representation	**Dimension**
TFIDF	10,000
Doc2Vec	300
USE	512
$BERT_s$ / $BioBERT_{seq}$	128×768
$BERT_p$ / $BioBERT_{pool}$	768
$ELMo_s$	128×1028
$ELMo_p$	1028

4.2 Classification Model and Setting

To learn weak classifiers (h_1 and h_2) in co-training, we evaluated the following classifiers: Support Vector Machines (SVM), Random Forest classifier (RF), XG-Boost (XGB), Multi-Layer Perceptron (MLP), and Convolutional Neural Network (CNN). These models are commonly used as baselines in many classification tasks and we used the implementations available in mainstream packages.[7] Note that, CNN is only used for sequential word embeddings generated by BERT/ELMo, so as to learn the dependencies between words. All combinations are summarised in Table 3.

[7]Code will be released upon acceptance.

Table 3: Combinations of Document Representations and Model Architectures

Representation	**Classifier**
TFIDF, Doc2Vec, USE, BERT$_p$, ELMo$_p$	SVM, MLP, RF, XGB
BERT$_s$, ELMo$_s$	CNN

In our experiments, we simply adopt the pre-trained embeddings as detailed earlier, without finetuning the embeddings for the classification task. For classification models in our experiments, we adopt either default or commonly adopted parameters. The reason is, in co-training setting, we only has a small number of labeled examples to start with, which makes parameter finetuning less effective. Further, during each iteration, two new classifiers are learned.

5 Experiments and Results

Datasets. We conduct experiments on two benchmark datasets: 20 Newsgroup and Ohsumed. The 20 NG dataset contains 11,314 labeled documents for training and 7,532 documents for test, in 20 classes. Ohsumed dataset contains abstracts of medical research papers. Some of its documents has multiple class labels. We follow the same preprocessing as in Yao et al. [2019] to exclude all documents belonging to multiple classes. As the results, we 3,357 training documents and 4,043 test documents, in 23 classes. It is worth noting that the processed Ohsumed dataset is highly imbalance, such that a few of the classes only have less than 10 supports. We did not implement any of the special techniques such as upsampling to handle the class imbalance problem. The main reason was that there are many combinations to experiment with, so we wanted to keep training cost to the minimum by keeping the model architectures simple and avoid fine-tuning.

Co-Training Setting. To evaluate co-training, we randomly sample 10% of labeled documents from each class to form the labeled set L. The remaining 90% of training data becomes U. We set the size of U' to be 400. After each iteration, the top p most confident documents from U' are added to L. We set $p = 1$ for the first 10 iterations, and increase the value of p by 1 after every 10 iterations. After $k = 40$ iterations, we have in total 100 documents added to L. Then we stop co-training and evaluate the combined prediction of the two weak classifiers.[8]

5.1 Results with 100% and 10% Labeled Data

To provide reference performances for co-training, we first evaluate the different classifiers trained on different document representations, using labeled documents

[8]We have also evaluated another setting $p = 1$ and $k = 100$, and the results are comparable.

Table 4: Performance of different combinations of document representation and classification model, on 20Newsgroup. Two sets of results are presented one is trained with 10% of training data, *i.e.,* documents in L, and the other is trained with all training data, *i.e.,* documents in $L + U$ in the co-training setting.

Dataset		20NG 100% labeled data				20NG 10% labeled data			
DocRep.	**Model**	**Pre**	**Rec**	F_1	**Acc**	**Pre**$_L$	**Rec**$_L$	$F1_L$	**Acc**$_L$
TFIDF	SVM	0.78	0.77	**0.77**	**0.78**	0.69	0.68	**0.68**	**0.69**
	MLP	0.78	0.77	**0.77**	0.77	0.69	0.67	0.67	0.68
	RF	0.62	0.59	0.59	0.60	0.57	0.44	0.43	0.45
	XGB	0.64	0.62	0.63	0.63	0.39	0.35	0.35	0.36
Doc2Vec	SVM	0.53	0.53	0.52	**0.54**	0.41	0.41	0.40	0.42
	MLP	0.54	0.53	**0.53**	**0.54**	0.49	0.48	**0.46**	**0.49**
	RF	0.44	0.44	0.42	0.45	0.35	0.33	0.31	0.34
	XGB	0.43	0.43	0.42	0.44	0.31	0.31	0.29	0.31
USE	SVM	0.70	0.70	**0.69**	**0.71**	0.66	0.67	**0.66**	**0.68**
	MLP	0.70	0.69	**0.69**	0.70	0.67	0.66	0.65	**0.68**
	RF	0.68	0.68	0.67	0.69	0.64	0.64	0.62	0.65
	XGB	0.43	0.43	0.42	0.44	0.58	0.58	0.57	0.59
BERT$_s$	CNN	0.80	0.80	**0.80**	**0.81**	0.68	0.67	**0.67**	**0.68**
BERT$_p$	SVM	0.66	0.65	0.65	0.66	0.54	0.54	0.53	0.55
	MLP	0.58	0.51	0.50	0.53	0.37	0.29	0.24	0.30
	RF	0.38	0.38	0.36	0.39	0.51	0.50	0.48	0.51
	XGB	0.42	0.43	0.42	0.44	0.29	0.29	0.27	0.29
ELMo$_s$	CNN	0.76	0.76	**0.76**	**0.76**	0.62	0.62	**0.61**	**0.63**
ELMo$_p$	SVM	0.64	0.64	0.64	0.65	0.57	0.57	0.57	0.58
	MLP	0.68	0.66	0.66	0.67	0.61	0.55	0.54	0.56
	RF	0.58	0.58	0.57	0.66	0.51	0.50	0.48	0.51
	XGB	0.58	0.58	0.58	0.59	0.47	0.47	0.46	0.48

in L (*i.e.,* 10% of training data in the original dataset) and documents in L+U (*i.e.,* 100% training data in the original dataset) respectively. We report the classification performance by macro-averaged Precision/Recall/F_1 and Accuracy. The results on the two benchmark datasets are reported in Table 4 and Table 5. In our discussion, we mainly focus on F_1 score.

Performance on both datasets share similar trend. All classifiers show significant drop of performance when only 10% of labeled data are used for train-

Table 5: Performance of different combinations of document representation and classification model, on Ohsumed. Two sets of results are presented one is trained with 10% of training data, *i.e.,* documents in L, and the other is trained with all training data, *i.e.,* documents in $L + U$ in the co-training setting. BERT refers to BioBERT on Ohsumed dataset.

Dataset		Ohsumed 100% labeled data				Ohsumed 10% labeled data			
DocRep.	**Model**	**Pre**	**Rec**	F_1	**Acc**	**Pre**$_L$	**Rec**$_L$	$F1_L$	**Acc**$_L$
TFIDF	SVM	0.68	0.56	**0.59**	**0.67**	0.49	0.26	**0.28**	**0.47**
	MLP	0.69	0.52	0.57	0.66	0.43	0.20	0.21	0.42
	RF	0.40	0.26	0.28	0.49	0.23	0.10	0.08	0.29
	XGB	0.42	0.33	0.34	0.52	0.04	0.06	0.03	0.02
Doc2Vec	SVM	0.39	0.37	0.37	0.48	0.30	0.21	**0.21**	0.38
	MLP	0.51	0.38	**0.38**	**0.54**	0.28	0.20	0.20	**0.40**
	RF	0.29	0.13	0.11	0.34	0.15	0.10	0.07	0.29
	XGB	0.26	0.18	0.18	0.38	0.10	0.10	0.09	0.27
USE	SVM	0.47	0.35	**0.36**	**0.50**	0.30	0.21	**0.20**	**0.38**
	MLP	0.42	0.33	0.34	0.48	0.25	0.20	0.19	**0.38**
	RF	0.48	0.24	0.25	0.44	0.20	0.15	0.14	0.33
	XGB	0.34	0.28	0.29	0.45	0.16	0.15	0.14	0.33
BERT$_s$	CNN	0.67	0.55	**0.59**	**0.68**	0.25	0.20	0.18	0.40
BERT$_p$	SVM	0.55	0.52	0.52	0.61	0.40	0.34	**0.35**	**0.49**
	MLP	0.58	0.53	0.54	0.62	0.41	0.28	0.29	0.46
	RF	0.61	0.25	0.27	0.47	0.18	0.14	0.12	0.34
	XGB	0.40	0.30	0.31	0.49	0.15	0.14	0.12	0.32
ELMo$_s$	CNN	0.49	0.39	**0.40**	**0.57**	0.21	0.17	0.16	**0.37**
ELMo$_p$	SVM	0.36	0.33	0.34	0.47	0.20	0.20	**0.20**	0.36
	MLP	0.37	0.33	0.32	0.48	0.20	0.19	0.18	**0.37**
	RF	0.25	0.33	0.18	0.40	0.17	0.14	0.12	0.32
	XGB	0.27	0.23	0.22	0.41	0.13	0.13	0.12	0.32

ing, compared to the version using all training documents. Overall, Ohsumed is much more challenging to get high classification results, compared to 20NG. In terms of classifiers' performance, when all training data are used, BERT$_s$+CNN achieves the highest F_1 on 20NG, followed by TFIDF+SVM and ELMo$_s$+CNN.[9] With 10% of training data, TFIDF+SVM becomes the best performer followed by BERT$_s$+CNN. On Ohsumed, with all training data are available, both TFIDF+SVM and BERT$_s$+CNN achieve the best F_1. Both outperform the rest by a large margin. When only 10% of data is used for training, BERT$_p$+SVM is the best performer with F_1 score 0.35. TFIDF+SVM is in the second position with F_1 score 0.28. We note that, the F_1 scores by the best performers are fairly low.

Our results suggest that deep contextualised representations together with the right choice of classification models likely to achieve the best results. On the other hand, TFIDF+SVM remains a strong baseline. There is also an observation that the combination of different document representations and classification models give very diverse results, particularly when there is a lacking of sufficient training data.

5.2 Results of Co-Training

As results of different classifiers on the same dataset are very diverse (see Table 4 and Table 5). Some of them give very poor results, hence it is not necessary to conduct co-training on every possible combination. Instead, on each dataset, for each kind of document representation, we choose the best performing classification model. Then we conduct co-training on this set of classifiers. With 5 document

[9]The best performance reported in our experiments is not as good as SOTA, as we do not finetune parameters and document representations. Further, our focus is to study the performance of co-training and not to achieve the new SOTA. On the other hand, the best results obtained by using all training data are not too far from SOTA results.

Table 6: Co-training results of combined classifier h_3. Results are in bold if the performance of h_3 is better than either h_1 or h_2 in the evaluated combination. Paired t-test is conducted on F_1 scores only (not on Acc) and * denotes statistically significant.

	$\boldsymbol{h_1}$	$\boldsymbol{ER_{h1}}$	$\boldsymbol{h_2}$	$\boldsymbol{ER_{h2}}$	**Pre**	**Rec**	F_1	$\Delta\mathbf{F1}_L$	ΔF_1	**Acc**	$\Delta\mathbf{Acc}_L$	$\Delta\mathbf{Acc}$
20Newsgroup	TFIDF+SVM	0.05	Doc2Vec+MLP	0.09	0.60	0.56	0.55	-0.13	-0.22	0.57	-0.12	-0.21
		0.05	USE+SVM	0.01	0.73	0.72	**0.72***	+0.04	-0.05	**0.74**	+0.05	-0.04
		0.03	BERT$_s$+CNN	0.02	0.72	0.72	**0.72***	+0.04	-0.08	**0.73**	+0.04	-0.08
		0.07	ELMo$_s$+CNN	0.02	0.66	0.65	0.65	-0.03	-0.12	0.66	-0.03	-0.12
	Doc2Vec+MLP	0.07	USE+SVM	0.00	0.59	0.57	0.56	-0.12	-0.13	0.58	-0.10	-0.13
		0.11	BERT$_s$+CNN	0.01	0.71	0.69	**0.69***	+0.01	-0.11	**0.71**	+0.03	-0.10
		0.15	ELMo$_s$+CNN	0.01	0.63	0.61	0.60	-0.06	-0.16	0.62	-0.06	-0.14
	USE+SVM	0.00	BERT$_s$+CNN	0.02	0.73	0.72	**0.71***	+0.04	-0.09	**0.73**	+0.06	-0.08
		0.00	ELMo$_s$+CNN	0.01	0.67	0.66	0.65	-0.03	-0.11	0.67	-0.01	-0.09
	BERT$_s$+CNN	0.03	ELMo$_s$+CNN	0.03	0.72	0.71	**0.71***	+0.03	-0.09	**0.72**	+0.04	-0.09
Ohsumed	TFIDF+SVM	0.12	Doc2Vec+MLP	0.33	0.27	0.23	0.21	-0.07	-0.38	0.44	-0.03	-0.23
		0.13	USE+SVM	0.26	0.39	0.22	0.21	-0.07	-0.38	0.42	-0.05	-0.25
		0.11	BERT$_p$+SVM	0.19	0.45	0.36	**0.37**	+0.09	-0.22	**0.52**	+0.03	-0.15
		0.10	ELMo$_s$+CNN	0.25	0.20	0.19	0.17	-0.11	-0.11	0.41	-0.06	-0.26
	Doc2Vec+MLP	0.24	USE+SVM	0.23	0.24	0.21	0.20	0.00	-0.18	0.42	+0.02	-0.12
		0.19	BERT$_p$+SVM	0.22	0.37	0.28	0.27	-0.08	-0.25	0.48	-0.01	-0.13
		0.29	ELMo$_s$+CNN	0.20	0.22	0.20	0.19	-0.01	-0.38	0.42	+0.02	-0.15
	USE+SVM	0.26	BERT$_p$+SVM	0.18	0.42	0.31	0.32	-0.03	-0.20	0.49	0.00	-0.12
		0.26	ELMo$_s$+CNN	0.22	0.25	0.21	0.18	-0.02	-0.39	0.41	+0.03	-0.16
	BERT$_p$+SVM	0.31	ELMo$_s$+CNN	0.32	0.24	0.24	0.22	-0.13	-0.30	0.44	-0.05	-0.17

representations, we have 10 combinations of h_1 and h_2 for co-training.

Table 6 reports the performance of 10 co-training settings on both datasets. In this table, we list the two classifiers (h_1 and h_2), and report macro-averaged Precision/Recall/F_1 and accuracy of the combined classifier h_3, after co-training. Again, we mainly focus on F_1. For comparison, we compute the change of F_1 obtained by h_3, compared to the lower bound, denoted by $\Delta F1_L$. The lower bound is determined by the $F1_L$ of the better classifiers among h_1 and h_2, trained by using 10% of training data (*i.e.,* set L). Accordingly ΔF_1 denotes the difference between h_3's F_1 and the upper bound, which is the F_1 of the better classifier among h_1 and h_2, training by using all training data $(L + U)$. We also list the error rate of h_1 and h_2, which is the rate of wrong predictions made by the weak classifier when adding documents to L. From the results, we make the following observations.

On 20Newsgroup datasets, five classifier combinations in co-training setting lead to significant improvement in F_1 score. In particular $\langle$ TFIDF+SVM, USE+SVM$\rangle$ and $\langle$ TFIDF+SVM, BERT$_s$+CNN$\rangle$ and both manage to improve the F_1 score to 0.72. And another two combinations achieve F_1 score 0.71. Comparing to the F_1 score of a single classifier obtained by using all labeled documents, the improvement is promising. We note that the error rate is fairly low for both h_1 and h_2 on 20Newsgroup dataset.

On Ohsumed, only one combination $\langle$ TFIDF+SVM, BERT$_p$+SVM$\rangle$ manages to get a positive $\Delta F1_L$. And the improvement is not statistically significant by paired t-test with two-tails. In other words, co-training results on this dataset are in general worse than the weak classifiers h_1 and h_2 trained on limited labeled data L. We also note that the error rate for both classifiers are fairly high.

In summary, the evaluated co-training combinations show very different results on the two datasets. On a typical text classification task like 20Newsgroup, we observe improvements on 5 out of 10 combinations of classifiers after co-training. The classification models in each combination leading to improvement, could be the same or different. However, in all the 5 combinations leading to improvement, the weak classifiers h_1 and h_2 are both among the better ones, compared to the other classifiers in the comparison (see Table 4). On a challenging classification task like Ohsumed. We do not observe the benefit of co-training, probably due to the weak performance of weak classifiers h_1 and h_2. With a relatively high error rate, the quality of documents added to L is not ensured, leading to the classifiers in the following iterations to learn from noisy labeled data.

6 Conclusion

In this study, we conduct a systematic evaluation of co-training by using different document representations and classification models. Our study show that traditional TFIDF representation and deep learning based representation both are sufficient to represent documents for classification task, making co-training applicable. Co-training achieves significant improvement if the classification task is not very challenging and weak classifiers are able to get reasonable good predictions with limited training data. The traditional setting of TFIDF+SVM remains a strong baseline.

7 Recommendations

One obvious area that can be explored further is effect of co-training and the various document representations on Ohsumed dataset. As mentioned in Section 5, the processed Ohsumed dataset is highly imbalanced, with a few classes having less than 10 supports. We did not specifically implemented techniques, such as upsampling or assigning classes with respective weights, which can potentially mitigate data imbalance problem. This could be one key thing that we can do to enhance the benefits of co-training on Ohsumed. Moreover, we adopted macro-average scores for precision, recall and F_1 for this work, which consider all classes having equal weights, it might undermined the actual performance of co-training with document representations on Ohsumed dataset. Thus, a more balanced metric such as micro-average or weighted-average should also be used in parallel, so as to provide a more comprehensive assessment of the experimental results.

Besides, due to limitation on compute resources, we did not finetune the pre-trained embeddings such as USE, BERT and ELMo for the down stream classification task. If we could finetune the pretrained embeddings, the individual h_1 and h_2 will probably show better performance, thereby enhancing the performance of the combined h_3.

References

Avrim Blum and Tom M. Mitchell. 1998. Combining Labeled and Unlabeled Data with Co-Training. In *COLT*. 92–100.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. *CoRR* abs/1803.11175 (2018).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

Rayid Ghani. 2002. Combining Labeled and Unlabeled Data for MultiClass Text Categorization. In *ICML*. 187–194.

Gilad Katz, Cornelia Caragea, and Asaf Shabtai. 2018. Vertical Ensemble Co-Training for Text Classification. *ACM TIST* 9, 2 (2018), 21:1–21:23.

DongHwa Kim, Deokseong Seo, Suhyoun Cho, and Pilsung Kang. 2019. Multi-co-training for document classification using various document representations: TF-IDF, LDA, and Doc2Vec. *Inf. Sci.* 477 (2019), 15–29.

Svetlana Kiritchenko and Stan Matwin. 2011. Email classification with co-training. In *Proc CASCON*. 301–312.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*, Vol. 32. 1188–1196.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *CoRR* abs/1901.08746 (2019).

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-Supervised Neural Text Classification. In *CIKM*. 983–992.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NeurIPS*. 3111–3119.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL-HLT*. 2227–2237.

Bhavani Raskutti, Herman L. Ferrá, and Adam Kowalczyk. 2002. Combining clustering and co-training to enhance text classification using unlabelled data. In *KDD*. 620–625.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph Convolutional Networks for Text Classification. In *AAAI*. 7370–7377.

Zhi-Hua Zhou and Ming Li. 2005. Tri-Training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE TKDE* 17, 11 (2005), 1529–1541.