WEAKLY-SUPERVISED TEXT CLASSIFICATION

BY

YU MENG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Professor Jiawei Han

ABSTRACT

Deep neural networks are gaining increasing popularity for the classic text classification task, due to their strong expressive power and less requirement for feature engineering. Despite such attractiveness, neural text classification models suffer from the lack of training data in many real-world applications. Although many semi-supervised and weakly-supervised text classification models exist, they cannot be easily applied to deep neural models and meanwhile support limited supervision types. In this work, we propose a weakly-supervised framework that addresses the lack of training data in neural text classification. Our framework consists of two modules: (1) a pseudo-document generator that leverages seed information to generate pseudo-labeled documents for model pre-training, and (2) a self-training module that bootstraps on real unlabeled data for model refinement. Our framework has the flexibility to handle different types of weak supervision and can be easily integrated into existing deep neural models for text classification. Based on this framework, we propose two methods, WeSTClass and WeSHClass, for flat text classification and hierarchical text classification, respectively. We have performed extensive experiments on real-world datasets from different domains. The results demonstrate that our proposed framework achieves inspiring performance without requiring excessive training data and outperforms baselines significantly.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I would like to first thank my adviser Professor Jiawei Han of the Department of Computer Science at University of Illinois at Urbana-Champaign. Professor Han's continuous encouragement, inspiring guidance, and perspective advising have always steered me in the right direction towards good research topics and quality research outputs that constitute this thesis work. I would also like to thank all Data Mining Group members, especially Jiaming Shen and Chao Zhang, for the inspiring discussions that greatly polished this thesis work.

This research work is sponsored in part by U.S. Army Research Lab. under Cooperative Agreement No. W911NF-09-2-0053 (NSCTA), DARPA under Agreement No. W911NF-17-C-0099, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, DTRA HDTRA11810026, and grant 1U54GM114838 awarded by NIGMS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov).

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Text classification plays a fundamental role in a wide variety of applications, ranging from sentiment analysis [37] to document categorization [42] and query intent classification [39]. Recently, deep neural models—including convolutional neural networks (CNNs) [16, 44, 15, 45] and recurrent neural networks (RNNs) [33, 32, 42]—have demonstrated superiority for text classification. The attractiveness of these neural models for text classification is mainly two-fold. First, they can largely reduce feature engineering efforts by automatically learning distributed representations that capture text semantics. Second, they enjoy strong expressive power to better learn from the data and yield better classification performance.

Despite the attractiveness and increasing popularity of neural models for text classification, *the lack of training data* is still a key bottleneck that prohibits them from being adopted in many practical scenarios. Indeed, training a deep neural model for text classification can easily consume million-scale labeled documents. Collecting such training data requires domain experts to read through millions of documents and carefully label them with domain knowledge, which is often too expensive to realize.

To address the label scarcity bottleneck, we study the problem of learning neural models for text classification *under weak supervision*. In many scenarios, while users cannot afford to label many documents for training neural models, they can provide a small amount of seed information for the classification task. Such seed information may arrive in various forms: either a set of representative keywords for each class, or a few (less than a dozen) labeled documents, or even only the surface names of the classes. Such a problem is called *weakly-supervised* text classification.

There have been many studies related to weakly-supervised text classification. However, training neural models for text classification under weak supervision remains an open research problem. Several semi-supervised neural models have been proposed [25, 41], but they still require hundreds or even thousands of labeled training examples, which are not available in the weakly supervised setting [27]. Along another line, there are existing methods that perform weakly-supervised text classification, including latent variable models [18] and embedding-based methods [38, 19]. These models have the following limitations: (1) *supervision inflexibility*: they can only handle one type of seed information, either a collection of labeled documents or a set of class-related keywords, which restricts their applicabilities; (2) *seed sensitivity*: the "seed supervision" from users completely controls the model training process, making the learned model very sensitive to the initial seed information; (3) *limited extensibility*: these methods are specific to either latent variable models or embedding methods,

and cannot be readily applied to learn deep neural models based on CNN or RNN.

In this thesis, we will introduce a neural approach for weakly-supervised text classification. Our framework can utilize various neural models as classifier and meanwhile support different types of weak supervisions (both word-level and document-level). We will introduce how to apply our framework to flat classification tasks in Chapter 2 and to hierarchical classification tasks in Chapter 3. Finally, we will conclude the thesis by summarizing the methods and proposing potential future work in Chapter 4.

CHAPTER 2: WEAKLY-SUPERVISED FLAT TEXT CLASSIFICATION

In this chapter, we present a method that addresses flat text classification under weak supervision. Then in the next chapter, we will extend our method to support hierarchical text classification.

2.1 OVERVIEW

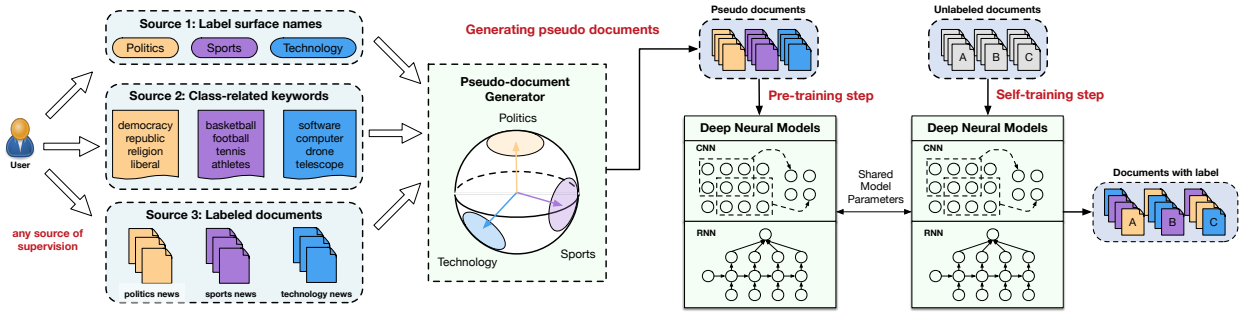


Figure 2.1: WeSTClass consists of two key modules: (1) a pseudo-document generator that leverages seed information to generate pseudo-labeled documents for model pre-training, and (2) a self-training module that bootstraps on real unlabeled data for model refinement.

In this chapter, we present a new method, named WeSTClass [22], for **We**akly-**S**upervised **T**ext **Class**ification. As shown in Figure 2.1, WeSTClass contains two modules to address the above challenges. The first module is a pseudo-document generator, which leverages seed information to generate pseudo documents as synthesized training data. By assuming word and document representations reside in the same semantic space, we generate pseudo documents for each class by modeling the semantics of each class as a high-dimensional spherical distribution [10], and further sampling keywords to form pseudo documents. The pseudo document generator can not only expand user-given seed information for better generalization, but also handle different types of seed information (*e.g.*, label surface names, class-related keywords, or a few labeled documents) flexibly.

The second key module of our method is a self-training module that fits real unlabeled documents for model refinement. First, the self-training module uses pseudo documents to pre-train either CNN-based or RNN-based models to produce an initial model, which serves as a starting point in the subsequent model refining process. Then, it applies a self-training procedure, which iteratively makes predictions on real unlabeled documents and leverages high-confidence predictions to refine the neural model.

Below is an overview of this chapter:

1. We design the WeSTClass method for addressing the label scarcity bottleneck of neural text classification. To the best of our knowledge, WeSTClass is the first weakly-supervised text classification method that can be applied to most existing neural models and meanwhile handle different types of seed information.

2. We present a novel pseudo document generator by modeling the class semantic as a spherical distribution. The generator is able to generate pseudo documents that are highly correlated to each class, and meanwhile effectively expands user-provided seed information for better generalization.

3. We present a self-training algorithm for training deep neural models by leveraging pseudo documents. The self-training algorithm can iteratively bootstrap the unlabeled data to obtain high-quality deep neural models, and is generic enough to be integrated into either CNN-based or RNN-based models.

4. We conduct a thorough evaluation of our method on three real-world datasets from different domains. The experiment results show that our method can achieve inspiring text classification performance even without excessive training data and outperforms various baselines.

2.2 RELATED WORK

In this section, we review existing studies for weakly-supervised text classification, which can be categorized into two classes: (1) latent variable models; and (2) embedding-based models.

2.2.1 Latent Variable Models

Existing latent variable models for weakly-supervised text classification mainly extend topic models by incorporating user-provided seed information. Specifically, semi-supervised PLSA [21] extends the classic PLSA model by incorporating a conjugate prior based on expert review segments (topic keywords or phrases) to force extracted topics to be aligned with provided review segments. [12] encodes prior knowledge and indirect supervision in constraints on posteriors of latent variable probabilistic models. Descriptive LDA [8] uses an LDA model as the describing device to infer Dirichlet priors from given category labels

and descriptions. The Dirichlet priors guides LDA to induce the category-aware topics. Seed-guided topic model [18] takes a small set of seed words that are relevant to the semantic meaning of the category, and then predicts the category labels of the documents through two kinds of topic influence: category-topics and general-topics. The labels of the documents are inferred based on posterior category-topic assignment. Our method differs from these latent variable models in that it is a weakly-supervised neural model. As such, it enjoys two advantages over these latent variable models: (1) it has more flexibility to handle different types of seed information which can be a collection of labeled documents or a set of seed keywords related to each class; (2) it does not need to impose assumptions on document-topic or topic-keyword distributions, but instead directly uses massive data to learn distributed representations to capture text semantics.

2.2.2 Embedding-based Models

Embedding-based weakly supervised models use seed information to derive vectorized representations for documents and label names for the text classification task. Dataless classification [7, 34] takes category names and projects each word and document into the same semantic space of Wikipedia concepts. Each category is represented with words in the category label. The document classification is performed based on the vector similarity between a document and a category using explicit semantic analysis [11]. Unsupervised neural categorization [19] takes category names as input and applies a cascade embedding approach: First the seeded category names and other significant phrases (concepts) are embedded into vectors for capturing concept semantics. Then the concepts are embedded into a hidden category space to make the category information explicit. Predictive text embedding [38] is a semi-supervised algorithm that utilizes both labeled and unlabeled documents to learn text embedding specifically for a task. Labeled data and different levels of word co-occurrence information are first represented as a large-scale heterogeneous text network and then embedded into a low dimensional space that preserves the semantic similarity of words and documents. Classification is performed by using one-vs-rest logistic regression model as classifier and the learned embedding as input. Compared with our method, these embedding-based weakly supervised methods cannot be directly applied to deep neural models (CNN, RNN) for the text classification task. Furthermore, while they allow the seed information to directly control the model training process, we introduce a pseudo document generation paradigm which is generalized from the seed information. Hence, our model is less prone to seed information overfitting and enjoys better generalization ability.

2.3 PRELIMINARIES

In this section, we formulate the problem of weakly-supervised text classification, and give an overview of our proposed method.

2.3.1 Problem Formulation

Given a text collection $\mathcal{D} = \{D_1, \ldots, D_n\}$ and m target classes $\mathcal{C} = \{C_1, \ldots, C_m\}$, text classification aims to assign a class label $C_j \in \mathcal{C}$ to each document $D_i \in \mathcal{D}$. To characterize each class, traditional supervised text classification methods rely on large amounts of labeled documents. In this chapter, we focus on the text classification under weakly-supervised setting where the supervision signal comes from one of the following sources: (1) *label surface names*: $\mathcal{L} = \{L_j\}|_{j=1}^m$, where L_j is the surface name for class C_j, (2) *class-related keywords*: $\mathcal{S} = \{S_j\}|_{j=1}^m$, where $S_j = \{w_{j,1}, \ldots, w_{j,k}\}$ represents a set of k keywords in class C_j, and (3) *labeled documents*: $\mathcal{D}^L = \{\mathcal{D}_j^L\}|_{j=1}^m$, where $\mathcal{D}_j^L = \{D_{j,1}, \ldots, D_{j,l}\}$ denotes a set of l ($l \ll n$) labeled documents in class C_j. In many scenarios, the above weak supervision signals can be easily obtained from users. Finally, we define our problem as follows:

Definition 2.1 (Problem Formulation) *Given a text collection $\mathcal{D} = \{D_1, \ldots, D_n\}$, target classes $\mathcal{C} = \{C_1, \ldots, C_m\}$, and weak supervision from either $\mathcal{L}$, $\mathcal{S}$ or $\mathcal{D}^L$, the weakly-supervised text classification task aims to assign a label $C_j \in \mathcal{C}$ to each $D_i \in \mathcal{D}$.*

2.3.2 Method Overview

Our proposed weakly-supervised text classification method contains two key modules. The first one is a pseudo-document generator that unifies seed information and outputs pseudo documents for model training. We assume words and documents share a joint semantic space which provides flexibility for handling different types of seed information. Then, we model each class as a high-dimensional spherical distribution from which keywords are sampled to form pseudo documents as training data. The second key module of our method is a self-training module that can be easily integrated into existing deep neural models, either CNN-based or RNN-based. It first uses the generated pseudo documents to pre-train neural models, which allows the model to start with a good initialization. Then, a self-training procedure is applied to iteratively refine the neural model using unlabeled real documents based on the model's high-confidence predictions. We show the entire process of our method in Figure 2.1.

2.4 PSEUDO DOCUMENT GENERATION

In this section, we describe the details of the pseudo-document generator, which leverages seed information to generate a bunch of pseudo documents that are correlated to each class. Below, we first introduce how to model class distributions in a joint semantic space with words and documents, and then describe the pseudo document generation process.

2.4.1 Modeling Class Distribution

To effectively leverage user-provided seed information and capture the semantic correlations between words, documents and classes, we assume words and documents share a joint semantic space, based on which we learn a generative model for each class to generate pseudo documents.

Specifically, we first use the Skip-Gram model [24] to learn p-dimensional vector representations of all the words in the corpus. Furthermore, since directional similarities between vectors are more effective in capturing semantic correlations [35, 2, 17], we normalize all the p-dimensional word embeddings so that they reside on a unit sphere in $\mathbb{R}^p$, which is the joint semantic space. We call it "joint" because we assume pseudo document vectors reside on the same unit sphere as well, which we will explain in Section 2.4.2. We retrieve a set of keywords in the semantic space that are correlated to each class based on the seed information. We describe how to handle different types of seed information as follows:

- **Label surface names**: When only label surface names $\mathcal{L}$ are given as seed information, for each class j we use the embedding of its surface name L_j to retrieve top-t nearest words in the semantic space. We set t to be the largest number that does not results in shared words across different classes.
- **Class-related keywords**: When users provide a list of related keywords S_j for each class j, we use the embeddings of these seed keywords to find top-t keywords in the semantic space, by measuring the average similarity to the seed keywords.
- **Labeled documents**: When users provide a small number of documents $\mathcal{D}_j^L$ that are correlated with class j, we first extract t representative keywords in $\mathcal{D}_j^L$ using tf-idf weighting, and then consider them as class-related keywords.

After obtaining a set of keywords that are correlated with each class, we model the semantic of each class as a von Mises Fisher (vMF) distribution [2, 13], which models word embeddings on a unit sphere in $\mathbb{R}^p$ and has been shown effective for various tasks [3, 43]. Specifically, we

define the probability distribution of a class as:

$$f(\boldsymbol{x}; \boldsymbol{\mu}, \kappa) = c_p(\kappa) e^{\kappa \boldsymbol{\mu}^T \boldsymbol{x}},$$

where $\kappa \geq 0$, $\|\boldsymbol{\mu}\| = 1$, $p \geq 2$ and the normalization constant $c_p(\kappa)$ is given by

$$c_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)},$$

where $I_r(\cdot)$ represents the modified Bessel function of the first kind at order r. We justify our choice of the vMF distribution as follows: the vMF distribution has two parameters—the mean direction $\boldsymbol{\mu}$ and the concentration parameter κ. The distribution of keywords on the unit sphere for a specific class concentrates around the mean direction $\boldsymbol{\mu}$, and is more concentrated if κ is large. Intuitively, the mean direction $\boldsymbol{\mu}$ acts as a semantic focus on the unit sphere, and produces relevant semantic embeddings around it, where concentration degree is controlled by the parameter κ.

Now that we have leveraged the seed information to obtain a set of keywords for each class on the unit sphere, we can use these correlated keywords to fit a vMF distribution $f(\boldsymbol{x}; \boldsymbol{\mu}, \kappa)$. Specifically, let X be a set of vectors for the keywords on the unit sphere, i.e.,

$$X = \{\boldsymbol{x}_i \in \mathbb{R}^p \mid \boldsymbol{x}_i \text{ drawn from } f(\boldsymbol{x}; \boldsymbol{\mu}, \kappa), 1 \leq i \leq t\},$$

then we use the maximum likelihood estimates [2] for finding the parameters $\hat{\boldsymbol{\mu}}$ and $\hat{\kappa}$ of the vMF distribution:

$$\hat{\boldsymbol{\mu}} = \frac{\sum_{i=1}^t \boldsymbol{x}_i}{\|\sum_{i=1}^t \boldsymbol{x}_i\|},$$

and

$$\frac{I_{p/2}(\hat{\kappa})}{I_{p/2-1}(\hat{\kappa})} = \frac{\|\sum_{i=1}^t \boldsymbol{x}_i\|}{t}.$$

Obtaining an analytic solution for $\hat{\kappa}$ is infeasible because the formula involves an implicit equation which is a ratio of Bessel functions. We thus use a numerical procedure based on Newton's method [2] to derive an approximation of $\hat{\kappa}$.

2.4.2 Generating Pseudo Documents

To generate a pseudo document D_i^* (we use D_i^* instead of D_i to denote it is a pseudo document) of class j, we propose a generative mixture model based on class j's distribution $f(\boldsymbol{x}; \boldsymbol{\mu}_j, \kappa_j)$. The mixture model repeatedly generates a number of terms to form a pseudo

document; when generating each term, the model chooses from a background distribution with probability α ($0 < \alpha < 1$) and from the class-specific distribution with probability $1 - \alpha$.

The class-specific distribution is defined based on class j's distribution $f(\boldsymbol{x}; \boldsymbol{\mu}_j, \kappa_j)$. Particularly, we first sample a document vector $\boldsymbol{d}_i$ from $f(\boldsymbol{x}; \boldsymbol{\mu}_j, \kappa_j)$, then build a keyword vocabulary V_{d_i} for $\boldsymbol{d}_i$ that contains the top-γ words with most similar word embedding with $\boldsymbol{d}_i$. These γ words in V_{d_i} are highly semantically relevant with the topic of pseudo document D_i^* and will appear frequently in D_i^*. Each term of a pseudo document is generated according to the following probability distribution:

$$p(w \mid \boldsymbol{d}_i) = \begin{cases} \alpha p_B(w) & w \notin V_{d_i} \\ \alpha p_B(w) + (1-\alpha)\frac{\exp(\boldsymbol{d}_i^T \boldsymbol{v}_w)}{\sum_{w' \in V_{d_i}} \exp(\boldsymbol{d}_i^T \boldsymbol{v}_{w'})} & w \in V_{d_i} \end{cases} \tag{2.1}$$

where $\boldsymbol{v}_w$ is the word embedding for w and $p_B(w)$ is the background distribution for the entire corpus.

Note that we generate document vectors from $f(\boldsymbol{x}; \boldsymbol{\mu}_j, \kappa_j)$ instead of fixing them to be $\boldsymbol{\mu}_j$. The reason is that some class (*e.g.*, Sports) may cover a wide range of topics (*e.g.*, athlete activities, sport competitions, etc.), but using $\boldsymbol{\mu}_j$ as the pseudo document vector will only attract words that are semantically similar to the centroid direction of a class. Sampling pseudo document vectors from the distribution, however, allows the generated pseudo documents to be more semantically diversified and thus cover more information about the class. Consequently, models trained on such more diversified pseudo documents are expected to have better generalization ability.

Algorithm 2.1 shows the whole process of generating a collection of β pseudo documents per class. For each class j, given the learned class distributions and the average length of pseudo documents dl[1], we draw a document vector $\boldsymbol{d}_i$ from class j's distribution $f(\boldsymbol{x}; \boldsymbol{\mu}_j, \kappa_j)$. After that, we generate dl words sequentially based on $\boldsymbol{d}_i$ and add the generated document into the pseudo document collection $\mathcal{D}_j^*$ of class j. After the above process repeats β times, we finally obtain $\mathcal{D}_j^*$ which contains β pseudo documents for class j.

2.5 NEURAL MODELS WITH SELF-TRAINING

In this section, we present the self-training module that trains deep neural models with the generated pseudo documents. The self-training module first uses the pseudo documents to pre-train a deep neural network, and then iteratively refines the trained model on the

[1]The length of each pseudo document can be either manually set or equal to the average document length in the real document collection.

Algorithm 2.1: Pseudo Documents Generation.

Input: Class distributions $\{f(\boldsymbol{x}; \boldsymbol{\mu}_j, \kappa_j)\}|_{j=1}^{m}$; average document length dl; number of pseudo documents β to generate for each class.

Output: A set of $m \times \beta$ pseudo documents $\mathcal{D}^*$.

```
Initialize D* ← ∅
for class index j from 1 to m do
  Initialize D*_j ← ∅
  for pseudo document index i from 1 to β do
    Sample document vector d_i from f(x; μ_j, κ_j)
    D*_i ← empty string
    for word index k from 1 to dl do
      Sample word w_{i,k} ~ p(w | d_i) based on Eq. (2.1)
      D*_i = D*_i ⊕ w_{i,k} // concatenate w_{i,k} after D*_i
    end for
    D*.append(D*_i)
  end for
  D* ← D* ∪ D*_j
end for
Return D*
```

real unlabeled documents in a bootstrapping fashion. In the following, we first present the pre-training and the self-training steps in Section 2.5.1 and 2.5.2, and then demonstrate how the framework can be instantiated with CNN and RNN models in Section 2.5.3.

2.5.1 Neural Model Pre-training

As we have obtained pseudo documents for each class, we use them to pre-train a neural network M[2]. A naive way of creating the label for a pseudo document D_i^* is to directly use the associated class label that D_i^* is generated from, *i.e.* using one-hot encoding where the generating class takes value 1 and all other classes are set to 0. However, this naive strategy often causes the neural model to overfit to the pseudo documents and have limited performance when classifying real documents, due to the fact that the generated pseudo documents do not contain word ordering information. To tackle this problem, we create pseudo labels for pseudo documents. In Equation (2.1), we design pseudo documents to be generated from a mixture of background and class-specific word distributions, controlled by a balancing parameter α. Such a process naturally leads to our design of the following procedure

[2]When the supervision source is **labeled documents**, these seed documents will be used to augment the pseudo document set during the pre-training step.

for pseudo label creation: we evenly split the fraction of the background distribution into all m classes, and set the pseudo label $\boldsymbol{l}_i$ for pseudo document D_i^* as

$$l_{ij} = \begin{cases} (1-\alpha) + \alpha/m & D_i^* \text{ is generated from class } j \\ \alpha/m & \text{otherwise} \end{cases}$$

After creating the pseudo labels, we pre-train a neural model M by generating β pseudo documents for each class, and minimizing the KL divergence loss from the neural network outputs Y to the pseudo labels L, namely

$$loss = KL(L\|Y) = \sum_i \sum_j l_{ij} \log \frac{l_{ij}}{y_{ij}}$$

We will detail how we instantiate the neural model M shortly in Section 2.5.3.

2.5.2 Neural Model Self-training

While the pre-training step produces an initial neural model M, the performance of the M is not the best one can hope for. The major reason is that the pre-trained model M only uses the set of pseudo documents but fails to take advantage of the information encoded in the real unlabeled documents. The self-training step is designed to tackle the above issues. Self-training [26, 30] is a common strategy used in classic semi-supervised learning scenarios. The rationale behind self-training is to first train the model with labeled data, and then bootstrap the learning model with its current highly-confident predictions.

After the pre-training step, we use the pre-trained model to classify all unlabeled documents in the corpus and then apply a self-training strategy to improve the current predictions. During self-training, we iteratively compute pseudo labels based on current predictions and refine model parameters by training the neural network with pseudo labels. Given the current outputs Y, the pseudo labels are computed using the same self-training formula as in [40]:

$$l_{ij} = \frac{y_{ij}^2 / f_j}{\sum_{j'} y_{ij'}^2 / f_{j'}}$$

where $f_j = \sum_i y_{ij}$ is the soft frequency for class j.

Self-training is performed by iteratively computing pseudo labels and minimizing the KL divergence loss from the current predictions Y to the pseudo labels L. This process terminates when less than $\delta\%$ of the documents in the corpus have class assignment changes.

Although both pre-training and self-training create pseudo labels and use them to train

neural models, it is worth mentioning the difference between them: in pre-training, pseudo labels are paired with generated pseudo documents to distinguish them from given labeled documents (if provided) and prevent the neural models from overfitting to pseudo documents; in self-training, pseudo labels are paired with every unlabeled real documents from corpus and reflect current high confidence predictions.

2.5.3 Instantiating with CNNs and RNNs

As mentioned earlier, our method for text classification is generic enough to be applied to most existing deep neural models. In this section, we instantiate the framework with two mainstream deep neural network models: convolution neural networks (CNN) and recurrent neural networks (RNN), by focusing on how they are used to learn document representations and perform classification.

CNN-Based Models

CNNs have been explored for text classification [16]. When instantiating our framework with CNN, the input to a CNN is a document of length dl represented by a concatenation of word vectors, i.e.,

$$\boldsymbol{d} = \boldsymbol{x}_1 \oplus \boldsymbol{x}_2 \oplus \cdots \oplus \boldsymbol{x}_{dl},$$

where $\boldsymbol{x}_i \in \mathbb{R}^p$ is the p dimensional word vector of the ith word in the document. We use $\boldsymbol{x}_{i:i+j}$ to represent the concatenation of word vectors $\boldsymbol{x}_i, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_{i+j}$. For window size of h, a feature c_i is generated from a window of words $\boldsymbol{x}_{i:i+h-1}$ by the following convolution operation

$$c_i = f(\boldsymbol{w} \cdot \boldsymbol{x}_{i:i+h-1} + b),$$

where $b \in \mathbb{R}$ is a bias term, $\boldsymbol{w} \in \mathbb{R}^{hp}$ is the filter operating on h words. For each possible size-h window of words, a feature map is generated as

$$\boldsymbol{c} = [c_1, c_2, \ldots, c_{dl-h+1}].$$

Then a max-over-time pooling operation is performed on $\boldsymbol{c}$ to output the maximum value $\hat{c} = \max(\boldsymbol{c})$ as the feature corresponding to this particular filter. If we use multiple filters, we will obtain multiple features that are passed through a fully connected softmax layer whose output is the probability distribution over labels.

RNN-Based Models

Besides CNNs, we also discuss how to instantiate our framework with RNNs. We choose the Hierarchical Attention Network (HAN) [42] as an exemplar RNN-based model. HAN consists of sequence encoders and attention layers for both words and sentences. In our context, the input document is represented by a sequence of sentences $s_i, i \in [1, L]$ and each sentence is represented by a sequence of words $w_{it}, t \in [1, T]$. At time t, the GRU [1] computes the new state as

$$\boldsymbol{h}_t = (\mathbf{1} - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t,$$

where the update gate vector

$$\boldsymbol{z}_t = \sigma(W_z \boldsymbol{x}_t + U_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z),$$

the candidate state vector

$$\tilde{\boldsymbol{h}}_t = \tanh(W_h \boldsymbol{x}_t + \boldsymbol{r}_t \odot (U_h \boldsymbol{h}_{t-1}) + \boldsymbol{b}_h),$$

the reset gate vector

$$\boldsymbol{r}_t = \sigma(W_r \boldsymbol{x}_t + U_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r),$$

and $\boldsymbol{x}_t$ is the sequence vector (word embedding or sentence vector) at time t. After encoding words and sentences, we also impose the attention layers to extract important words and sentences with the attention mechanism, and derive their weighted average as document representations.

2.6 EXPERIMENTS

In this section, we evaluate the empirical performance of our method for weakly supervised text classification.

2.6.1 Datasets

We use three corpora from different domains to evaluate the performance of our proposed method: (1) **The New York Times (NYT)**: We crawl 13,081 news articles using the New York Times API[3]. This corpus covers 5 major news topics; (2) **AG's News (AG)**: We use

[3] `http://developer.nytimes.com/`

the same AG's News dataset from [45] and take its training set portion (120,000 documents evenly distributed into 4 classes) as the corpus for evaluation; (3) **Yelp Review (Yelp)**: We use the Yelp reviews polarity dataset from [45] and take its testing set portion (38,000 documents evenly distributed into 2 classes) as the corpus for evaluation.

2.6.2 Baselines

We compare WeSTClass with a wide range of baseline models, described as follows.

- **IR with tf-idf**: this method accepts either **label surface name** or **class-related keywords** as supervision. We treat the label name or keyword set for each class as a query, and score the relevance of document to this class using the tf-idf model. The class with highest relevance score is assigned to the document.
- **Topic Model**: this method accepts either **label surface name** or **class-related keywords** as supervision. We first train the LDA model [4] on the entire corpus. Given a document, we compute the likelihood of observing label surface names or the average likelihood of observing class-related keywords. The class with maximum likelihood will be assigned to the document.
- **Dataless** [7, 34]: this method [4] accepts only **label surface name** as supervision. It leverages Wikipedia and uses Explicit Semantic Analysis [11] to derive vector representations of both labels and documents. The final document class is assigned based on the vector similarity between labels and documents.
- **UNEC** [19]: this method takes **label surface name** as its weak supervision. It categorizes documents by learning the semantics and category attribution of concepts inside the corpus. We use the authors' original implementation of this model.
- **PTE** [38]: this method [5] uses **labeled documents** as supervision. It first utilizes both labeled and unlabeled data to learn text embedding and then applies logistic regression model as classifier for text classification.
- **CNN** [16]: the original CNN model is a supervised text classification model and we extend it to incorporate all three types of supervision sources. If **labeled documents** are given, we directly train CNN model on the given labeled documents and then apply it on all unlabeled documents. If **label surface names** or **class-related keywords**

[4] `https://cogcomp.org/page/software_view/Descartes`
[5] `https://github.com/mnqu/PTE`

are given, we first use the above "IR with tf-idf" or "Topic Modeling" method (depending on which one works better) to label all unlabeled documents. Then, we select β labeled documents per class to pre-train CNN. Finally, we apply the same self-training module as described in Section 2.5 to obtain the final classifier.

- **HAN** [42]: similar to the above CNN model, we extend the original HAN model [6] to incorporate all three types of supervision sources.
- **NoST-(CNN/HAN)**: this is a variant of WeSTClass without the self-training module, *i.e.*, after pre-training CNN or HAN with pseudo documents, we directly apply it to classify unlabeled documents.
- WeSTClass-**(CNN/HAN)**: this is the full version of our proposed framework, with both pseudo-document generator and self-training module enabled.

2.6.3 Experiment Settings

We first describe our parameter settings as follows. For all datasets, we use the Skip-Gram model [24] to train 100-dimensional word embeddings on the corresponding corpus. We set the background word distribution weight $\alpha = 0.2$, the number of pseudo documents per class for pre-training $\beta = 500$, the size of class-specific vocabulary $\gamma = 50$ and the self-training stopping criterion $\delta = 0.1$.

We apply our proposed framework on two types of state-of-the-art text classification neural models: (1) CNN model, whose filter window sizes are $2, 3, 4, 5$ with 20 feature maps each. (2) HAN model, which uses a forward GRU with 100 dimension output for both word and sentence encoding. Both the pre-training and the self-training steps are performed using SGD with batch size 256.

The seed information we use as weak supervision for different datasets are described as follows: (1) When the supervision source is **label surface name**, we directly use the label surface names of all classes; (2) When the supervision source is **class-related keywords**, we manually choose 3 keywords which do not include the class label name for each class. The selected keywords are shown in Tables 2.1, and we evaluate how our model is sensitive to such seed keyword selection in Section 2.6.6; (3) When the supervision source is **labeled documents**, we randomly sample c documents of each class from the corpus ($c = 10$ for **The New York Times** and **AG's News**; $c = 20$ for **Yelp Review**) and use them as the

[6] `https://github.com/richliao/textClassifier`

given labeled documents. To alleviate the randomness, we repeat the document selection process 10 times and show the performances with average and standard deviation values.

Table 2.1: Seed keywords on NYT, AG, and Yelp.

Dataset	Class	Keyword List
NYT	Politics	{democracy, religion, liberal}
	Arts	{music, movie, dance}
	Business	{investment, economy, industry}
	Science	{scientists, biological, computing}
	Sports	{hockey, tennis, basketball}
AG	Politics	{government, military, war}
	Sports	{basketball, football, athletes}
	Business	{stocks, markets, industries}
	Technology	{computer, telescope, software}
AG	Good	{terrific, great, awesome}
	Bad	{horrible, disappointing, subpar}

2.6.4 Experiment Results

In this subsection, we report our experimental results and our findings.

Overall Text Classification Performance

In the first set of experiments, we compare the classification performance of our method against all the baseline methods on the three datasets. Both macro-F1 and micro-F1 metrics are used to quantify the performance of different methods.

As shown in Table 2.2, our proposed framework achieves the overall best performances among all the baselines on three datasets with different weak supervision sources. Specifically, in almost every case, WeSTClass-**CNN** yields the best performance among all methods; WeSTClass-**HAN** performs slightly worse than WeSTClass-**CNN** but still outperforms other baselines. We discuss the effectiveness of WeSTClass from the following aspects:

1. When **labeled documents** are given as the supervision source, the standard deviation values of WeSTClass-**CNN** and WeSTClass-**HAN** are smaller than those of **CNN** and **HAN**, respectively. This shows that WeSTClass can effectively reduce the seed sensitivity and improve the robustness of CNN and HAN models.

Table 2.2: Macro-F1 scores (upper table) and Micro-F1 scores (lower table) for all methods on three datasets. **LABELS**, **KEYWORDS**, and **DOCS** means the type of seed supervision is label surface name, class-related keywords, and labeled documents, respectively.

Methods	**The New York Times**			**AG's News**			**Yelp Review**		
	LABELS	**KEYWORDS**	**DOCS**	**LABELS**	**KEYWORDS**	**DOCS**	**LABELS**	**KEYWORDS**	**DOCS**
IR with tf-idf	0.319	0.509	-	0.187	0.258	-	0.533	0.638	-
Topic Model	0.301	0.253	-	0.496	0.723	-	0.333	0.333	-
Dataless	0.484	-	-	0.688	-	-	0.337	-	-
UNEC	0.690	-	-	0.659	-	-	0.602	-	-
PTE	-	-	0.834 (0.024)	-	-	0.542 (0.029)	-	-	0.658 (0.042)
HAN	0.348	0.534	0.740 (0.059)	0.498	0.621	0.731 (0.029)	0.519	0.631	0.686 (0.046)
CNN	0.338	0.632	0.702 (0.059)	0.758	0.770	0.766 (0.035)	0.523	0.633	0.634 (0.096)
NoST-HAN	0.515	0.213	0.823 (0.035)	0.590	0.727	0.745 (0.038)	0.731	0.338	0.682 (0.090)
NoST-CNN	0.701	0.702	0.833 (0.013)	0.534	0.759	0.759 (0.032)	0.639	0.740	0.717 (0.058)
WeSTClass-HAN	0.754	0.640	0.832 (0.028)	0.816	0.820	0.782 (0.028)	**0.769**	0.736	0.729 (0.040)
WeSTClass-CNN	**0.830**	**0.837**	**0.835** (**0.010**)	**0.822**	**0.821**	**0.839** (**0.007**)	0.735	**0.816**	**0.775** (**0.037**)

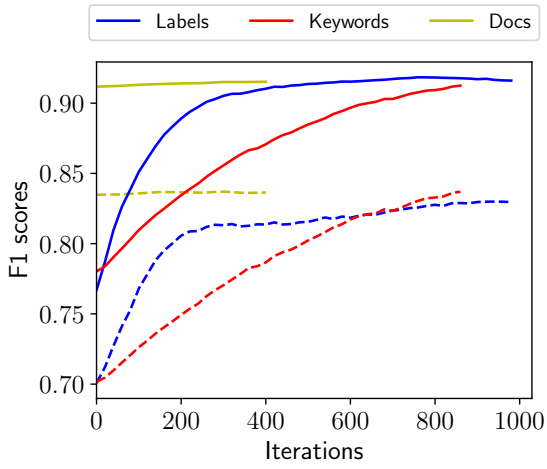
Methods	**The New York Times**			**AG's News**			**Yelp Review**		
	LABELS	**KEYWORDS**	**DOCS**	**LABELS**	**KEYWORDS**	**DOCS**	**LABELS**	**KEYWORDS**	**DOCS**
IR with tf-idf	0.240	0.346	-	0.292	0.333	-	0.548	0.652	-
Topic Model	0.666	0.623	-	0.584	0.735	-	0.500	0.500	-
Dataless	0.710	-	-	0.699	-	-	0.500	-	-
UNEC	0.810	-	-	0.668	-	-	0.603	-	-
PTE	-	-	0.906 (0.020)	-	-	0.544 (0.031)	-	-	0.674 (**0.029**)
HAN	0.251	0.595	0.849 (0.038)	0.500	0.619	0.733 (0.029)	0.530	0.643	0.690 (0.042)
CNN	0.246	0.620	0.798 (0.085)	0.759	0.771	0.769 (0.034)	0.534	0.646	0.662 (0.062)
NoST-HAN	0.788	0.676	0.906 (0.021)	0.619	0.736	0.747 (0.037)	0.740	0.502	0.698 (0.066)
NoST-CNN	0.767	0.780	0.908 (0.013)	0.553	0.766	0.765 (0.031)	0.671	0.750	0.725 (0.050)
WeSTClass-HAN	0.901	0.859	0.908 (0.019)	0.816	0.822	0.782 (0.028)	**0.771**	0.737	0.729 (0.040)
WeSTClass-CNN	**0.916**	**0.912**	**0.911** (**0.007**)	**0.823**	**0.823**	**0.841** (**0.007**)	0.741	**0.816**	**0.776** (0.037)

2. When the supervision source is **label surface name** or **class-related keywords**, we can see that WeSTClass-**CNN** and WeSTClass-**HAN** outperform **CNN** and **HAN**, respectively. This demonstrates that pre-training with generated pseudo documents results in a better neural model initialization compared to pre-training with documents that are labeled using either **IR with tf-idf** or **Topic Modeling**.

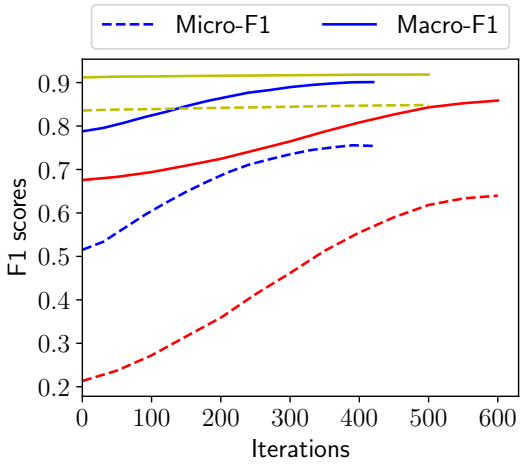
3. WeSTClass-**CNN** and WeSTClass-**HAN** always outperform **NoST-CNN** and **NoST-HAN**, respectively. Note that the only difference between WeSTClass-**CNN**/WeSTClass-**HAN** and **NoST-CNN**/**NoST-HAN** is that the latter two do not include the self-training module. The performance gaps between them thus clearly demonstrate the effectiveness of our self-training module.

Effect of self-training module

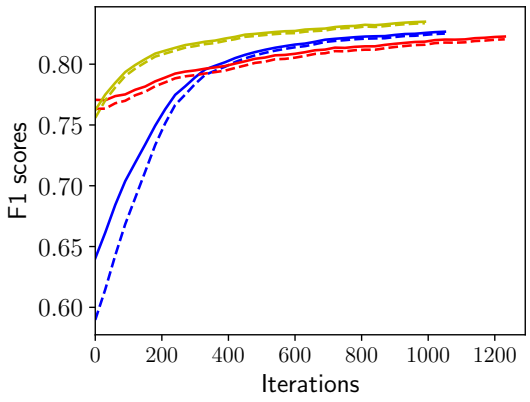
In this set of experiments, we conduct more experiments to study the effect of self-training module in WeSTClass, by investigating the performance of difference models as the number of iterations increases. The results are shown in Figure 2.2. We can see that the self-training module can effectively improve the model performance after the pre-training step. Also, we find that the self-training module generally has the least effect when supervision comes from labeled documents. One possible explanation is that when labeled documents are given, we will use both pseudo documents and provided labeled documents to pre-train the neural models. Such mixture training can often lead to better model initialization, compared to using pseudo documents only. As a result, there is less room for self-training module to make huge improvements.



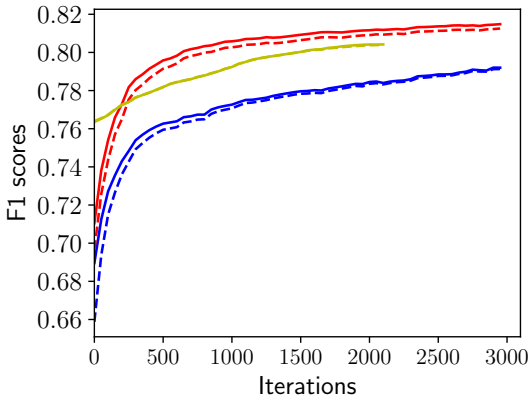
(a) WeSTClass-CNN – New York Times



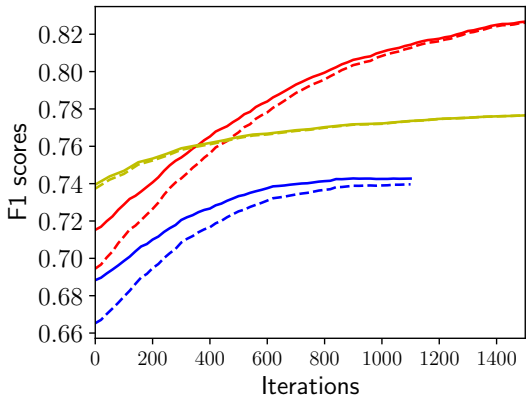
(b) WeSTClass-HAN – New York Times



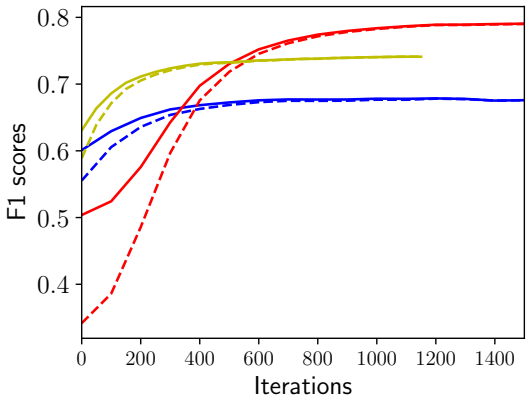
(c) WeSTClass-CNN – AG's News



(d) WeSTClass-HAN – AG's News



(e) WeSTClass-CNN – Yelp Review



(f) WeSTClass-HAN – Yelp Review

Figure 2.2: Effect of self-training modules on three datasets.

Effect of the number of labeled documents

When weak supervision signal comes from labeled documents, the setting is similar to semi-supervised learning except that the amount of labeled documents is very limited. In this set of experiments, we vary the number of labeled documents per class and compare the performances of five methods on the AG's News dataset: **CNN**, **HAN**, **PTE**, WeSTClass-**CNN** and WeSTClass-**HAN**. Again, we run each method 10 times with different sets of labeled documents, and report the average performances with standard deviation (represented as error bars) in Figure 2.3. We can see that when the amount of labeled documents is relatively large, the performances of the five methods are comparable. However, when fewer labeled documents are provided, **PTE**, **CNN** and **HAN** not only exhibit obvious performance drop, but also become very sensitive to the seed documents. Nevertheless, WeSTClass-based models, especially WeSTClass-**CNN**, yield stable performance with varying amount of labeled documents. This phenomenon shows that our method can more effectively take advantage of the limited amount of seed information to achieve better performance.

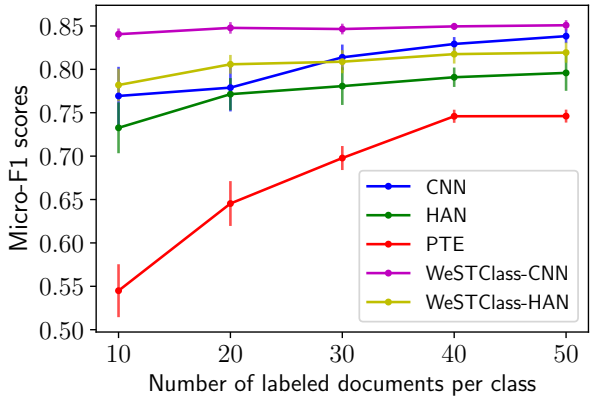


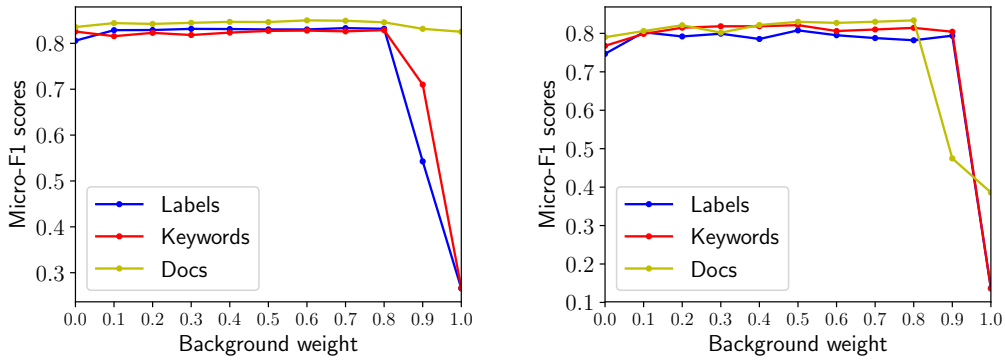
Figure 2.3: The performances of different methods on AG's News dataset when the number of labeled documents varies.

2.6.5 Parameter Study

In this section, we study the effects of different hyperparameter settings on the performance of WeSTClass with CNN and HAN models, including (1) background word distribution weight α, (2) number of generated pseudo documents β for pre-training and (3) keyword vocabulary size γ used in equation (2.1) where $\gamma = |V_{d_i}|$. When studying the effect of one parameter, the other parameters are set to their default values as described in Section 2.6.3 . We conduct all the parameter studies on the **AG's News** dataset.

Background Word Distribution Weight

The background word distribution weight α is used in both the language model for pseudo documents generation and pseudo-labels computation. When α becomes smaller, the generated pseudo documents contain more topic-related words and fewer background words, and the pseudo-labels become similar to one-hot encodings. We vary α from 0 to 1 with interval equal to 0.1. The effect of α is shown in Figure 2.4. Overall, different α values result in comparable performance, except when α is close to 1, pseudo documents and pseudo-labels become uninformative: pseudo documents are generated directly from background word distribution without any topic-related information, and pseudo-labels are uniform distributions. We notice that when $\alpha = 1$, **labeled documents** as supervision source results in much better performance than **label surface name** and **class-related keywords**. This is because pre-training with **labeled documents** is performed using both pseudo documents and labeled documents, and the provided labeled documents are still informative. When α is close to 0, the performance is slightly worse than other settings, because pseudo documents only contain topic-related keywords and pseudo-labels are one-hot encodings, which can easily lead to model overfitting to pseudo documents and behaving worse on real documents classification.



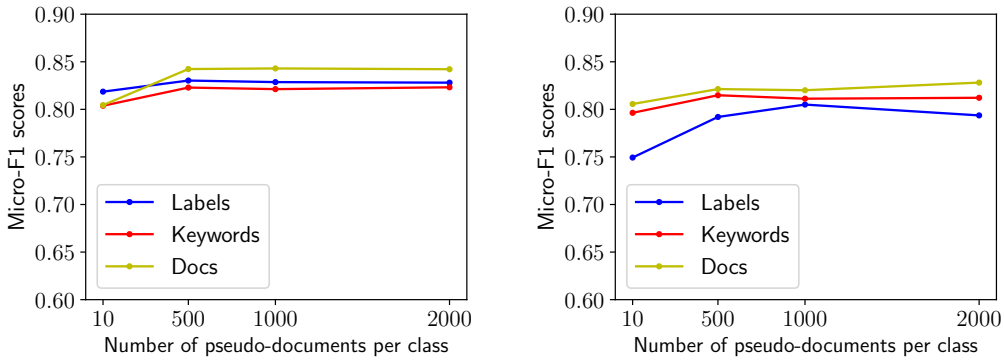
(a) WeSTClass-CNN (b) WeSTClass-HAN

Figure 2.4: Effect of background word distribution weight α.

Number of pseudo documents for pre-training

The effect of pseudo documents amount β is shown in Figure 2.5. We have the following findings from Figure 2.5: On the one hand, if the amount of generated pseudo documents is too small, the information carried in pseudo documents will be insufficient to pre-train a good model. On the other hand, generating too many pseudo documents will make the pre-training

process unnecessarily long. Generating 500 to 1000 pseudo documents of each class for pre-training will strike a good balance between pre-training time and model performance.

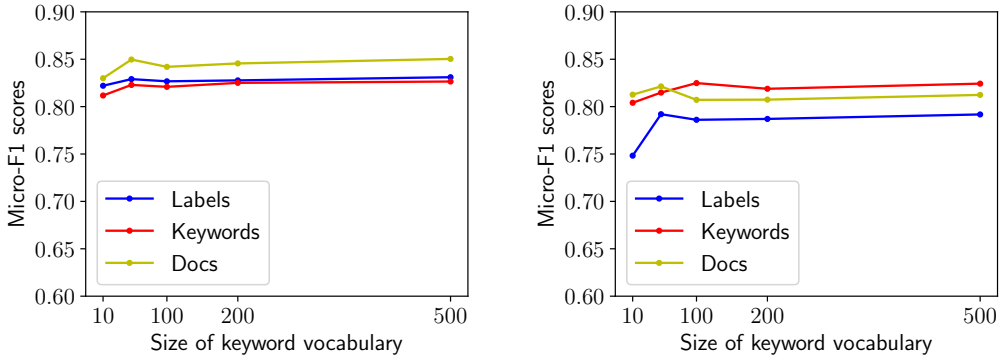


(a) WeSTClass-CNN (b) WeSTClass-HAN

Figure 2.5: Effect of pseudo documents amount per class β for pre-training.

Size of Keyword Vocabulary

Recall the pseudo document generation process in Section 2.4.2, after sampling a document vector $\boldsymbol{d}_i$, we will first construct a keyword vocabulary V_{d_i} that contains the top-γ words with most similar word embedding with $\boldsymbol{d}_i$. The size of the keyword vocabulary γ controls the number of unique words that appear frequently in the generated pseudo documents. If γ is too small, only a few topical keywords will appear frequently in pseudo documents, which will reduce the generalization ability of the pre-trained model. As shown in Figure 2.6, γ can be safely set within a relatively wide range from 50 to 500 in practice.



(a) WeSTClass-CNN (b) WeSTClass-HAN

Figure 2.6: Effect of keyword vocabulary size γ.

2.6.6 Case Study

In this subsection, we perform a set of case studies to further understand the properties of our proposed method.

Table 2.3: Keyword Lists at Top Percentages of Average Tf-idf.

Class	1%	5%	10%
Politics	{government, president, minister}	{mediators, criminals, socialist}	{suspending, minor, lawsuits}
Sports	{game, season, team}	{judges, folks, champagne}	{challenging, youngsters, stretches}
Business	{profit, company, sales}	{refunds, organizations, trader}	{winemaker, skilling, manufactured}
Technology	{internet, web, microsoft}	{biologists, virtually, programme}	{demos, microscopic, journals}

Choice of Seed Keywords

In the first set of case studies, we are interested in how sensitive our model is to the selection of seed keywords. In Section 2.6.3, we manually select class-related keywords, which could be subjective. Here we explore the sensitivity of WeSTClass-**CNN** and WeSTClass-**HAN** to different sets of seed keywords. For each class j of **AG's News** dataset, we first collect all documents belonging to class j, and then compute the tf-idf weighting of each word in each document of class j. We sort each word's average tf-idf weighting in these documents from high to low. Finally we form the seed keyword lists by finding words that rank at top 1% (most relevant), 5% and 10% based on the average tf-idf value. The keywords of each class at these percentages are shown in Table 2.3; the performances of WeSTClass-**CNN** and WeSTClass-**HAN** are shown in Figure 2.7. At top 5% and 10% of the average tf-idf weighting, although some keywords are already slightly irrelevant to their corresponding class semantic, WeSTClass-**CNN** and WeSTClass-**HAN** still perform reasonably well, which shows the robustness of our proposed framework to different sets of seed keywords.

Figure 2.7: Performances on AG's News dataset under different sets of seed keywords.

Self-training Corrects Misclassification

In the second set of case studies, we are interested in how the self-training module behaves to improve the performance of our model. Figure 2.8 shows WeSTClass-**CNN**'s prediction with **label surface name** as supervision source on a sample document from **AG's News** dataset: *The national competition regulator has elected not to oppose Telstra's 3G radio access network sharing arrangement with rival telco Hutchison.* We notice that this document is initially misclassified after the pre-training procedure, but it is then corrected by the subsequent self-training step. This example shows that neural models have the ability of self-correcting by learning from its high-confidence predictions with appropriate pre-training initialization.

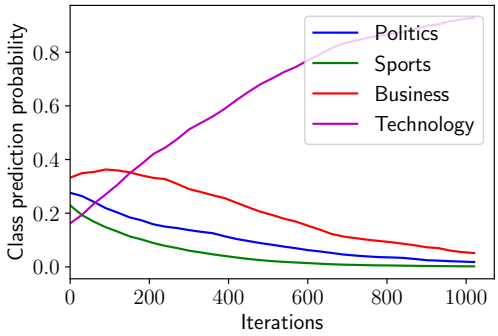


Figure 2.8: Class prediction probability during self-training procedure for a sample document.

2.7 SUMMARY

In this chapter, we have proposed a weakly-supervised text classification method built upon neural classifiers. With (1) a pseudo document generator for generating pseudo training data and (2) a self-training module that bootstraps on real unlabled data for model refining, our method effectively addresses the key bottleneck for existing neural text classifiers—the lack of labeled training data. Our method is not only flexible in incorporating difference sources of weak supervision (class label surface names, class-related keywords, and labeled documents), but also generic enough to support different neural models (CNN and RNN). Our experimental results have shown that our method outperforms baseline methods significantly, and it is quite robust to different settings of hyperparameters and different types of user-provided seed information. An interesting finding based on the experiments in Section 2.6 is that different types of weak supervision are all highly helpful for the good performances of neural models. In the future, it is interesting to study how to integrate different types of seed information to further boost the performance of our method.

CHAPTER 3: WEAKLY-SUPERVISED HIERARCHICAL TEXT CLASSIFICATION

When the categories of a target corpus exhibit hierarchical structures, it is highly desirable to organize text documents into the class taxonomy instead of performing flat classification. In this chapter, we introduce how to extend our previous weakly-supervised text classification approach to hierarchical settings, for enabling multi-granular document classification.

3.1 OVERVIEW

Hierarchical text classification aims at classifying text documents into classes that are organized into a hierarchy. Traditional flat text classifiers (*e.g.*, SVM, logistic regression) have been tailored in various ways for hierarchical text classification. Early attempts [6] disregard the relationships among classes and treat hierarchical classification tasks as flat ones. Later approaches [9, 20, 5] train a set of local classifiers and make predictions in a top-down manner, or design global hierarchical loss functions that regularize with the hierarchy. Most existing efforts for hierarchical text classification rely on traditional text classifiers. Recently, deep neural networks have demonstrated superior performance for flat text classification. Compared with traditional classifiers, deep neural networks [16, 42] largely reduce feature engineering efforts by learning distributed representations that capture text semantics. Meanwhile, they provide stronger expressive power over traditional classifiers, thereby yielding better performance when large amounts of training data are available.

Motivated by the enjoyable properties of deep neural networks, we explore using deep neural networks for hierarchical text classification. Despite the success of deep neural models in flat text classification and their advantages over traditional classifiers, applying them to hierarchical text classification is nontrivial because of two major challenges. The first challenge is that *the training data deficiency prohibits neural models from being adopted.* Neural models are data hungry and require humans to provide tons of carefully-labeled documents for good performance. In many practical scenarios, however, hand-labeling excessive documents often requires domain expertise and can be too expensive to realize. The second challenge is to *determine the most appropriate level for each document in the class hierarchy.* In hierarchical text classification, documents do not necessarily belong to leaf nodes and may be better assigned to intermediate nodes. However, there are no simple ways for existing deep neural networks to automatically determine the best granularity for a given document.

We present a neural approach named WeSHClass [23], for **We**akly-**S**upervised **H**ierarchical Text **Class**ification and address the above two challenges. Our approach is built upon

deep neural networks, yet it requires only a small amount of weak supervision instead of excessive training data. Such weak supervision can be either a few (*e.g.*, less than a dozen) labeled documents or class-correlated keywords, which can be easily provided by users. To leverage such weak supervision for effective classification, our approach employs a novel pretrain-and-refine paradigm. Specifically, in the pre-training step, we leverage user-provided seeds to learn a spherical distribution for each class, and then generate pseudo documents from a language model guided by the spherical distribution. In the refinement step, we iteratively bootstrap the global model on real unlabeled documents, which self-learns from its own high-confident predictions.

WeSHClass automatically determines the most appropriate level during the classification process by explicitly modeling the class hierarchy. Specifically, we pre-train a local classifier at each node in the class hierarchy, and aggregate the classifiers into a global one using self-training. The global classifier is used to make final predictions in a top-down recursive manner. During recursive predictions, we introduce a novel blocking mechanism, which examines the distribution of a document over internal nodes and avoids mandatorily pushing general documents down to leaf nodes.

Below is an overview of this chapter:

1. We design a method for hierarchical text classification using neural models under weak supervision. WeSHClass does not require large amounts of training documents but just easy-to-provide word-level or document-level weak supervision. In addition, it can be applied to different classification types (*e.g.*, topics, sentiments).

2. We present a pseudo document generation module that generates high-quality training documents only based on weak supervision sources. The generated documents serve as pseudo training data which alleviate the training data bottleneck together with the subsequent self-training step.

3. We present a hierarchical neural model structure that mirrors the class taxonomy and its corresponding training method, which involves local classifier pre-training and global classifier self-training. The entire process is tailored for hierarchical text classification, which automatically determines the most appropriate level of each document with a novel blocking mechanism.

4. We conduct a thorough evaluation on three real-world datasets from different domains to demonstrate the effectiveness of WeSHClass. We also perform several case studies to understand the properties of different components in WeSHClass.

3.2 RELATED WORK

3.2.1 Weakly-Supervised Text Classification

There exist some previous studies that use either word-based supervision or limited amount of labeled documents as weak supervision sources for the text classification task. WeSTClass [22] leverages both types of supervision sources. It applies a similar procedure of pre-training the network with pseudo documents followed by self-training on unlabeled data. Descriptive LDA [8] applies an LDA model to infer Dirichlet priors from given keywords as category descriptions. The Dirichlet priors guide LDA to induce the category-aware topics from unlabeled documents for classification. [12] propose to encode prior knowledge and indirect supervision in constraints on posteriors of latent variable probabilistic models. Predictive text embedding [38] utilizes both labeled and unlabeled documents to learn text embedding specifically for a task. Labeled data and word co-occurrence information are first represented as a large-scale heterogeneous text network and then embedded into a low dimensional space. The learned embedding are fed to logistic regression classifiers for classification. None of the above methods are specifically designed for hierarchical classification.

3.2.2 Hierarchical Text Classification

There have been efforts on using SVM for hierarchical classification. [9, 20] propose to use local SVMs that are trained to distinguish the children classes of the same parent node so that the hierarchical classification task is decomposed into several flat classification tasks. [5] define hierarchical loss function and apply cost-sensitive learning to generalize SVM learning for hierarchical classification. A graph-CNN based deep learning model is proposed in [28] to convert text to graph-of-words, on which the graph convolution operations are applied for feature extraction. FastXML [29] is designed for extremely large label space. It learns a hierarchy of training instances and optimizes a ranking-based objective at each node of the hierarchy. The above methods rely heavily on the quantity and quality of training data for good performance, while WeSHClass does not require much training data but only weak supervision from users.

Hierarchical dataless classification [34] uses class-related keywords as class descriptions, and projects classes and documents into the same semantic space by retrieving Wikipedia concepts. Classification can be performed in both top-down and bottom-up manners, by measuring the vector similarity between documents and classes. Although hierarchical dataless classification does not rely on massive training data as well, its performance is highly influenced by the

text similarity between the distant supervision source (Wikipedia) and the given unlabeled corpus.

3.3 PROBLEM FORMULATION

We study hierarchical text classification that involves tree-structured class categories. Specifically, each category can belong to at most one parent category and can have arbitrary number of children categories. Following the definition in [31], we consider non-mandatory leaf prediction, wherein documents can be assigned to both internal and leaf categories in the hierarchy.

Traditional supervised text classification methods rely on large amounts of labeled documents for each class. In this chapter, we focus on text classification under weak supervision. Given a class taxonomy represented as a tree $\mathcal{T}$, we ask the user to provide weak supervision sources (*e.g.*, a few class-related keywords or documents) only for each leaf class in $\mathcal{T}$. Then we propagate the weak supervision sources upwards in $\mathcal{T}$ from leaves to root, so that the weak supervision sources of each internal class are an aggregation of weak supervision sources of all its descendant leaf classes. Specifically, given M leaf node classes, the supervision for each class comes from one of the following:

1. *Word-level supervision*: $\mathcal{S} = \{S_j\}|_{j=1}^{M}$, where $S_j = \{w_{j,1}, \ldots, w_{j,k}\}$ represents a set of k keywords correlated with class C_j;
2. *Document-level supervision*: $\mathcal{D}^L = \{\mathcal{D}_j^L\}|_{j=1}^{M}$, where $\mathcal{D}_j^L = \{D_{j,1}, \ldots, D_{j,l}\}$ denotes a small set of l ($l \ll$ corpus size) labeled documents in class C_j.

Now we are ready to formulate the hierarchical text classification problem. Given a text collection $\mathcal{D} = \{D_1, \ldots, D_N\}$, a class category tree $\mathcal{T}$, and weak supervisions of either $\mathcal{S}$ or $\mathcal{D}^L$ for each leaf class in $\mathcal{T}$, the weakly-supervised hierarchical text classification task aims to assign the most likely label $C_j \in \mathcal{T}$ to each $D_i \in \mathcal{D}$, where C_j could be either an internal or a leaf class.

3.4 PSEUDO DOCUMENT GENERATION

To break the bottleneck of lacking abundant labeled data for model training, we leverage user-given weak supervision to generate pseudo documents, which serve as pseudo training data for model pre-training. In this section, we first introduce how to leverage weak supervision sources to model class distributions in a spherical space, and then explain how to generate class-specific pseudo documents based on class distributions and a language model.

3.4.1 Modeling Class Distribution

We model each class as a high-dimensional spherical probability distribution which has been shown effective for various tasks [43]. We first train Skip-Gram model mikolov2013distributed to learn d-dimensional vector representations for each word in the corpus. Since directional similarities between vectors are more effective in capturing semantic correlations [2, 17], we normalize all the d-dimensional word embeddings so that they reside on a unit sphere in $\mathbb{R}^d$. For each class $C_j \in \mathcal{T}$, we model the semantics of class C_j as a mixture of von Mises Fisher (movMF) distributions [2, 13] in $\mathbb{R}^d$:

$$f(\boldsymbol{x} \mid \Theta) = \sum_{h=1}^{m} \alpha_h f_h(\boldsymbol{x} \mid \boldsymbol{\mu}_h, \kappa_h) = \sum_{h=1}^{m} \alpha_h c_d(\kappa_h) e^{\kappa_h \boldsymbol{\mu}_h^T \boldsymbol{x}},$$

where $\Theta = \{\alpha_1, \ldots, \alpha_m, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m, \kappa_1, \ldots, \kappa_m\}$, $\forall h \in \{1, \ldots, m\}$, $\kappa_h \geq 0$, $\|\boldsymbol{\mu}_h\| = 1$, and the normalization constant $c_d(\kappa_h)$ is given by

$$c_d(\kappa_h) = \frac{\kappa_h^{d/2-1}}{(2\pi)^{d/2} I_{d/2-1}(\kappa_h)},$$

where $I_r(\cdot)$ represents the modified Bessel function of the first kind at order r. We choose the number of components in movMF for leaf and internal classes differently:

- For each leaf class C_j, we set the number of vMF component $m = 1$, and the resulting movMF distribution is equivalent to a single vMF distribution, whose two parameters, the mean direction $\boldsymbol{\mu}$ and the concentration parameter κ, act as semantic focus and concentration for C_j.
- For each internal class C_j, we set the number of vMF component m to be the number of its children classes. Recall that we only ask the user to provide weak supervision sources at the leaf classes, and the weak supervision source of C_j are aggregated from its children classes. The semantics of a parent class can thus be seen as a mixture of the semantics of its children classes.

We first retrieve a set of keywords for each class given the weak supervision sources, then fit movMF distributions using the embedding vectors of the retrieved keywords. Specifically, the set of keywords are retrieved as follows: (1) When users provide related keywords S_j for each class j, we use the average embedding of these seed keywords to find top-n closest keywords in the embedding space; (2) When users provide documents $\mathcal{D}_j^L$ that are correlated with class j, we extract n representative keywords from $\mathcal{D}_j^L$ using tf-idf weighting. The parameter n

above is set to be the largest number that does not result in shared words across different classes. Compared to directly using weak supervision signals, retrieving relevant keywords for modeling class distributions has a smoothing effect which makes our model less sensitive to the weak supervision sources.

Let X be the set of embeddings of the n retrieved keywords on the unit sphere, i.e.,

$$X = \{\boldsymbol{x}_i \in \mathbb{R}^d \mid \boldsymbol{x}_i \text{ drawn from } f(\boldsymbol{x} \mid \Theta), 1 \leq i \leq n\},$$

we use the Expectation Maximization (EM) framework [2] to estimate the parameters Θ of the movMF distributions:

- E-step:
$$p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)}) = \frac{\alpha_h^{(t)} f_h(\boldsymbol{x}_i \mid \boldsymbol{\mu}_h^{(t)}, \kappa_h^{(t)})}{\sum_{h'=1}^{m} \alpha_{h'}^{(t)} f_{h'}(\boldsymbol{x}_i \mid \boldsymbol{\mu}_{h'}^{(t)}, \kappa_{h'}^{(t)})},$$
where $\mathcal{Z} = \{z_1, \ldots, z_n\}$ is the set of hidden random variables that indicate the particular vMF distribution from which the points are sampled;

- M-step:
$$\begin{aligned}
\alpha_h^{(t+1)} &= \frac{1}{n} \sum_{i=1}^{n} p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)}), \\
\boldsymbol{r}_h^{(t+1)} &= \sum_{i=1}^{n} p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)}) \boldsymbol{x}_i, \\
\boldsymbol{\mu}_h^{(t+1)} &= \frac{\boldsymbol{r}_h^{(t+1)}}{\|\boldsymbol{r}_h^{(t+1)}\|}, \\
\frac{I_{d/2}(\kappa_h^{(t+1)})}{I_{d/2-1}(\kappa_h^{(t+1)})} &= \frac{\|\boldsymbol{r}_h^{(t+1)}\|}{\sum_{i=1}^{n} p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)})}.
\end{aligned}$$
where we use the approximation procedure based on Newton's method [2] to derive an approximation of $\kappa_h^{(t+1)}$ because the implicit equation makes obtaining an analytic solution infeasible.

3.4.2 Language Model Based Document Generation

After obtaining the distributions for each class, we use an LSTM-based language model [36] to generate meaningful pseudo documents. Specifically, we first train an LSTM language model on the entire corpus. To generate a pseudo document of class C_j, we sample an

embedding vector from the movMF distribution of C_j and use the closest word in embedding space as the beginning word of the sequence. Then we feed the current sequence to the LSTM language model to generate the next word and attach it to the current sequence recursively [1]. Since the beginning word of the pseudo document comes directly from the class distribution, the generated document is ensured to be correlated to C_j. By virtue of the mixture distribution modeling, the semantics of every children class (if any) of C_j gets a chance to be included in the pseudo documents, so that the resulting trained neural model will have better generalization ability.

3.5 THE HIERARCHICAL CLASSIFICATION MODEL

In this section, we introduce the hierarchical neural model and its training method under weakly-supervised setting.

3.5.1 Local Classifier Pre-Training

We construct a neural classifier M_p (M_p could be any text classifier such as CNNs or RNNs) for each class $C_p \in \mathcal{T}$ if C_p has two or more children classes. Intuitively, the classifier M_p aims to classify the documents assigned to C_p into its children classes for more fine-grained predictions. For each document D_i, the output of M_p can be interpreted as $p(D_i \in C_c \mid D_i \in C_p)$, the conditional probability of D_i belonging to each children class C_c of C_p, given D_i is assigned to C_p.

The local classifiers perform local text classification at internal nodes in the hierarchy, and serve as building blocks that can be later ensembled into a global hierarchical classifier. We generate β pseudo documents per class and use them to pre-train local classifiers with the goal of providing each local classifier with a good initialization for the subsequent self-training step. To prevent the local classifiers from overfitting to pseudo documents and performing badly on classifying real documents, we use pseudo labels instead of one-hot encodings in pre-training. Specifically, we use a hyperparameter α that accounts for the "noises" in pseudo documents, and set the pseudo label $\boldsymbol{l}_i^*$ for pseudo document D_i^* (we use D_i^* instead of D_i to denote a pseudo document) as

$$l_{ij}^* = \begin{cases} (1-\alpha) + \alpha/m & D_i^* \text{ is generated from class } j \\ \alpha/m & \text{otherwise} \end{cases} \tag{3.1}$$

[1] In case of long pseudo documents, we repeatedly generate several sequences and concatenate them to form the entire document.

where m is the total number of children classes at the corresponding local classifier. After creating pseudo labels, we pre-train each local classifier M_p of class C_p using the pseudo documents for each children class of C_p, by minimizing the KL divergence loss from outputs $\mathcal{Y}$ of M_p to the pseudo labels $\mathcal{L}^*$, namely

$$loss = KL(\mathcal{L}^* \| \mathcal{Y}) = \sum_i \sum_j l^*_{ij} \log \frac{l^*_{ij}}{y_{ij}}.$$

3.5.2 Global Classifier Self-Training

At each level k in the class taxonomy, we need the network to output a probability distribution over all classes. Therefore, we construct a global classifier G_k by ensembling all local classifiers from root to level k. The ensemble method is shown in Figure 3.1. The multiplication operation conducted between parent classifier output and children classifier output can be explained by the conditional probability formula:

$$\begin{aligned} p(D_i \in C_c) &= p(D_i \in C_c \cap D_i \in C_p) \\ &= p(D_i \in C_c \mid D_i \in C_p) p(D_i \in C_p), \end{aligned}$$

where D_i is a document; C_c is one of the children classes of C_p. This formula can be recursively applied so that the final prediction is the multiplication of all local classifiers' outputs on the path from root to the destination node.

Greedy top-down classification approaches will propagate misclassifications at higher levels to lower levels, which can never be corrected. However, the way we construct the global classifier assigns documents soft probability at each level, and the final class prediction is made by jointly considering all classifiers' outputs from root to the current level via multiplication, which gives lower-level classifiers chances to correct misclassifications made at higher levels.

At each level k of the class taxonomy, we first ensemble all local classifiers from root to level k to form the global classifier G_k, and then use G_k's prediction on all unlabeled real documents to refine itself iteratively. Specifically, for each unlabeled document D_i, G_k outputs a probability distribution y_{ij} of D_i belonging to each class j at level k, and we set pseudo labels to be [40]:

$$l^{**}_{ij} = \frac{y^2_{ij}/f_j}{\sum_{j'} y^2_{ij'}/f_{j'}}, \tag{3.2}$$

where $f_j = \sum_i y_{ij}$ is the soft frequency for class j.

The pseudo labels reflect high-confident predictions, and we use them to guide the fine-

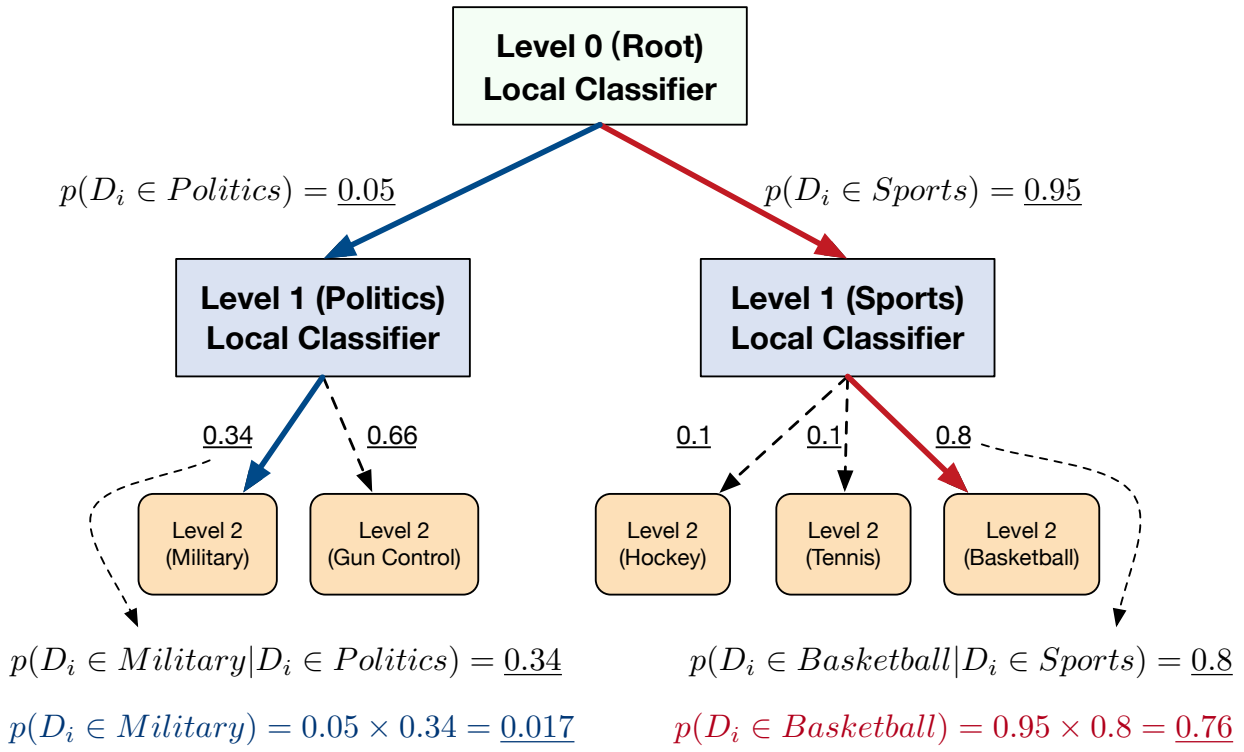


Figure 3.1: Ensemble of local classifiers.

tuning of G_k, by iteratively (1) computing pseudo labels $\mathcal{L}^{**}$ based on G_k's current predictions $\mathcal{Y}$ and (2) minimizing the KL divergence loss from $\mathcal{Y}$ to $\mathcal{L}^{**}$. This process terminates when less than $\delta\%$ of the documents in the corpus have class assignment changes. Since G_k is the ensemble of local classifiers, they are fine-tuned simultaneously via back-propagation during self-training. We will demonstrate the advantages of using global classifier over greedy approaches in the experiments.

3.5.3 Blocking Mechanism

In hierarchical classification, some documents should be classified into internal classes because they are more related to general topics rather than any of the more specific topics, which should be blocked at the corresponding local classifier from getting further passed to children classes.

When a document D_i is classified into an internal class C_j, we use the output $\boldsymbol{q}$ of C_j's local classifier to determine whether or not D_i should be blocked at the current class: if $\boldsymbol{q}$ is close to a one-hot vector, it strongly indicates that D_i should be classified into the corresponding child; if $\boldsymbol{q}$ is close to a uniform distribution, it implies that D_i is equally

relevant or irrelevant to all the children of C_j and thus more likely a general document. Therefore, we use normalized entropy as the measure for blocking. Specifically, we will block D_i from being further passed down to C_j's children if

$$-\frac{1}{\log m}\sum_{i=1}^{m} q_i \log q_i > \gamma, \tag{3.3}$$

where $m \geq 2$ is the number of children of C_j; $0 \leq \gamma \leq 1$ is a threshold value. When $\gamma = 1$, no documents will be blocked and all documents are assigned into leaf classes.

3.5.4 Inference

The hierarchical classification model can be directly applied to classify unseen samples after training. When classifying an unseen document, the model will directly output the probability distribution of that document belonging to each class at each level in the class hierarchy. The same blocking mechanism can be applied to determine the appropriate level that the document should belong to.

3.5.5 Algorithm Summary

Algorithm 3.1 puts the above pieces together and summarizes the overall model training process for hierarchical text classification. As shown, the overall training is proceeded in a top-down manner, from root to the final internal level. At each level, we generate pseudo documents and pseudo labels to pre-train each local classifier. Then we self-train the ensembled global classifier using its own predictions in an iterative manner. Finally we apply blocking mechanism to block general documents, and pass the remaining documents to the next level.

3.6 EXPERIMENTS

3.6.1 Experiment Settings

Datasets and Evaluation Metrics

We use three corpora from three different domains to evaluate the performance of our proposed method:

Algorithm 3.1: Overall Network Training.

Input: A text collection $\mathcal{D} = \{D_i\}|_{i=1}^{N}$; a class category tree $\mathcal{T}$; weak supervisions $\mathcal{W}$ of either $\mathcal{S}$ or $\mathcal{D}^L$ for each leaf class in $\mathcal{T}$.

Output: Class assignment $\mathcal{C} = \{(D_i, C_i)\}|_{i=1}^{N}$, where $C_i \in \mathcal{T}$ is the most specific class label for D_i.

```
Initialize C ← ∅
for k ← 0 to max_level − 1 do
    N ← all nodes at level k of T
    for node ∈ N do
        D* ← Pseudo document generation
        L* ← Equation (3.1)
        pre-train node.classifier with D*, L*
    end for
    G_k ← ensemble all classifiers from level 0 to k
    while not converged do
        L** ← Equation (3.2)
        self-train G_k with D, L**
    end while
    D_B ← documents blocked based on Equation (3.3)
    C_B ← D_B's current class assignments
    C ← C ∪ (D_B, C_B)
    D ← D − D_B
end for
C' ← D's current class assignments
C ← C ∪ (D, C')
Return C
```

- **The New York Times (NYT):** We crawl 13,081 news articles using the New York Times API [2]. This news corpus covers 5 super-categories and 25 sub-categories.

- **arXiv:** We crawl paper abstracts from arXiv website[3] and keep all abstracts that belong to only one category. Then we include all sub-categories with more than 1,000 documents out of 3 largest super-categories and end up with 230,105 abstracts from 53 sub-categories.

- **Yelp Review:** We use the Yelp Review Full dataset [45] and take its testing portion as our dataset. The dataset contains 50,000 documents evenly distributed into 5 sub-categories, corresponding to user ratings from 1 star to 5 stars. We consider 1 and 2 stars as "negative", 3 stars as "neutral", 4 and 5 stars as "positive", so we end up with

[2] `http://developer.nytimes.com/`

[3] `https://arxiv.org/`

3 super-categories.

Table 3.1 provides the statistics of the three datasets. We use Micro-F1 and Macro-F1 scores as metrics for classification performances.

Table 3.1: Dataset Statistics.

Corpus	# classes (level 1 + level 2)	# docs	Avg. doc length
NYT	$5 + 25$	$13,081$	778
arXiv	$3 + 53$	$230,105$	129
Yelp Review	$3 + 5$	$50,000$	157

Baselines

We compare our proposed method with a wide range of baseline models, described as below:

- **Hier-Dataless** [34]: Dataless hierarchical text classification [4] can only take **word-level** supervision sources. It embeds both class labels and documents in a semantic space using Explicit Semantic Analysis [11] on Wikipedia articles, and assigns the nearest label to each document in the semantic space. We try both the top-down approach and bottom-up approach, with and without the bootstrapping procedure, and finally report the best performance.
- **Hier-SVM** [9, 20]: Hierarchical SVM can only take **document-level** supervision sources. It decomposes the training tasks according to the class taxonomy, where each local SVM is trained to distinguish sibling categories that share the same parent node.
- **CNN** [16]: The CNN text classification model [5] can only take **document-level** supervision sources.
- **WeSTClass** [22]: Weakly-supervised neural text classification can take both **word-level** and **document-level** supervision sources. It first generates bag-of-words pseudo documents for neural model pre-training, then bootstraps the model on unlabeled data.
- **No-global**: This is a variant of WeSHClass without the global classifier, i.e., each document is pushed down with local classifiers in a greedy manner.

[4] `https://github.com/CogComp/cogcomp-nlp/tree/master/dataless-classifier`
[5] `https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras`

- **No-vMF**: This is a variant of WeSHClass without using movMF distribution to model class semantics, i.e., we randomly select one word from the keyword set of each class as the beginning word when generating pseudo documents.
- **No-selftrain**: This is a variant of WeSHClass without self-training module, i.e., after pre-training each local classifier, we directly ensemble them as a global classifier at each level to classify unlabeled documents.

Parameter Settings

For all datasets, we use Skip-Gram model [24] to train 100-dimensional word embeddings for both movMF distributions modeling and classifier input embeddings. We set the pseudo label parameter $\alpha = 0.2$, the number of pseudo documents per class for pre-training $\beta = 500$, and the self-training stopping criterion $\delta = 0.1$. We set the blocking threshold $\gamma = 0.9$ for **NYT** dataset where general documents exist and $\gamma = 1$ for the other two.

Although our proposed method can use any neural model as local classifiers, we empirically find that CNN model always results in better performances than RNN models, such as LSTM [14] and Hierarchical Attention Networks [42]. Therefore, we report the performance of our method by using CNN model with one convolutional layer as local classifiers. Specifically, the filter window sizes are $2, 3, 4, 5$ with 20 feature maps each. Both the pre-training and the self-training steps are performed using SGD with batch size 256.

Weak Supervision Settings

The seed information we use as weak supervision for different datasets are described as follows: (1) When the supervision source is **class-related keywords**, we select 3 keywords for each leaf class; (2) When the supervision source is **labeled documents**, we randomly sample c documents of each leaf class from the corpus ($c = 3$ for **NYT** and **arXiv**; $c = 10$ for **Yelp Review**) and use them as given labeled documents. To alleviate the randomness, we repeat the document selection process 10 times and show the performances with average and standard deviation values.

We list the keyword supervisions of some sample classes for **NYT** dataset as follows: **Immigration** (immigrants, immigration, citizenship); **Dance** (ballet, dancers, dancer); **Environment** (climate, wildlife, fish).

3.6.2 Quantitative Comparision

We show the overall text classification results in Table 3.2. WeSHClass achieves the overall best performance among all the baselines on the three datasets. Notably, when the supervision source is **class-related keywords**, WeSHClass outperforms **Hier-Dataless** and **WeSTClass**, which shows that WeSHClass can better leverage word-level supervision sources in hierarchical text classification. When the supervision source is **labeled documents**, WeSHClass has not only higher average performance, but also better stability than the supervised baselines. This demonstrates that when training documents are extremely limited, WeSHClass can better leverage the insufficient supervision for good performances and is less sensitive to seed documents.

Comparing WeSHClass with several ablations, **No-global**, **No-vMF** and **No-self-train**, we observe the effectiveness of the following components: (1) ensemble of local classifiers, (2) modeling class semantics as movMF distributions, and (3) self-training. The results demonstrate that all these components contribute to the performance of WeSHClass.

3.6.3 Component-Wise Evaluation

In this subsection, we conduct a series of breakdown experiments on **NYT** dataset using **class-related keywords** as weak supervision to further investigate different components in our proposed method. We obtain similar results on the other two datasets.

Pseudo Documents Generation

The quality of the generated pseudo documents is critical to our model, since high-quality pseudo documents provide a good model initialization. Therefore, we are interested in which pseudo document generation method gives our model best initialization for the subsequent self-training step. We compare our document generation strategy (movMF + LSTM language model) with the following two methods:

- Bag-of-words [22]: The pseudo documents are generated from a mixture of background unigram distribution and class-related keywords distribution.
- Bag-of-words + reordering: We first generate bag-of-words pseudo documents as in the previous method, and then use the globally trained LSTM language model to reorder the pseudo documents by greedily putting the word with the highest probability at the end of the current sequence. The beginning word is randomly chosen.

We showcase some generated pseudo document snippets of class "politics" for **NYT** dataset using different methods in Table 3.3. Bag-of-words method generates pseudo documents without word order information; bag-of-words method with reordering generates text of high quality at the beginning, but poor near the end, which is probably because the "proper" words have been used at the beginning, but the remaining words are crowded at the end implausibly; our method generates text of high quality.

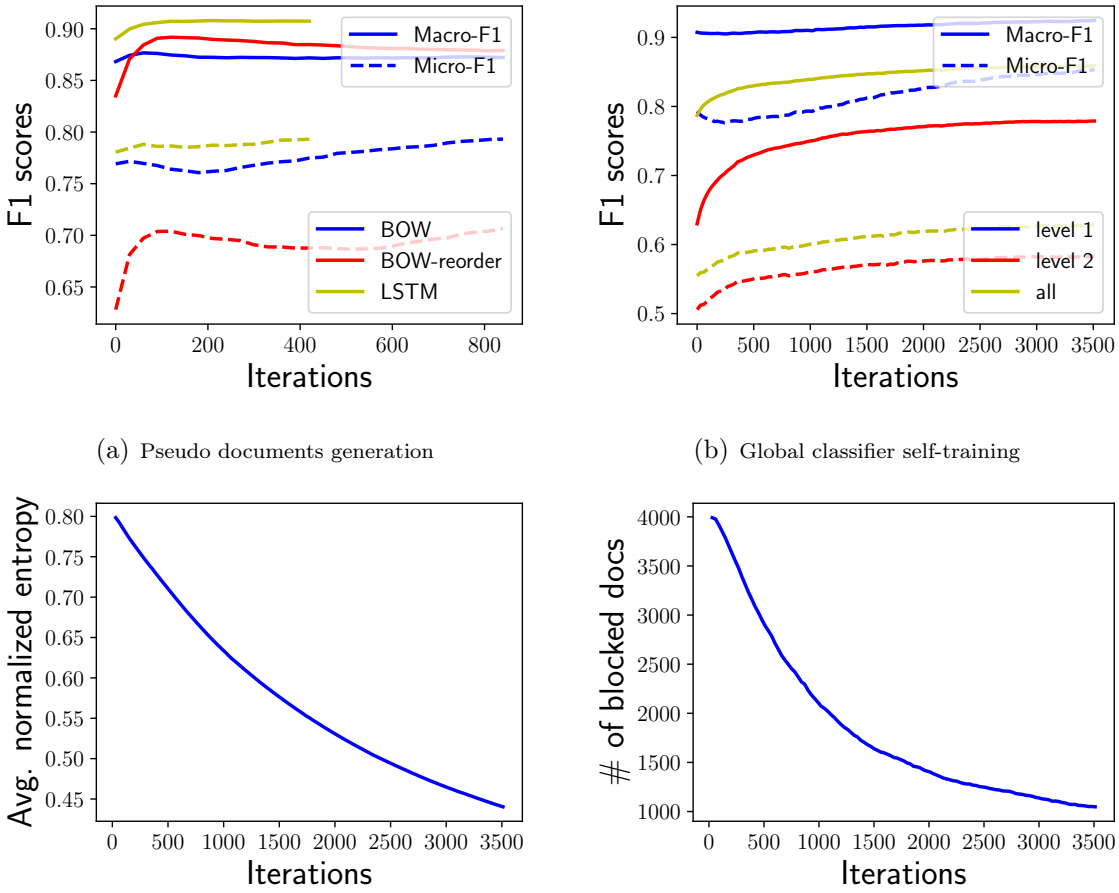
To compare the generalization ability of the pre-trained models with different pseudo documents, we show their subsequent self-training process (at level 1) in Figure 3.2(a). We notice that our strategy not only makes self-training converge faster, but also has better final performance.

Global Classifier and Self-training

We proceed to study why using self-trained global classifier on the ensemble of local classifiers is better than greedy approach. We show the self-training procedure of the global classifier at the final level in Figure 3.2(b), where we demonstrate the classification accuracy at level 1 (super-categories), level 2 (sub-categories) and of all classes. Since at the final level, all local classifiers are ensembled to construct the global classifier, self-training of the global classifier is the joint training of all local classifiers. The result shows that the ensemble of local classifiers for joint training is beneficial for improving the accuracy at all levels. If a greedy approach is used, however, higher-level classifiers will not be updated during lower-level classification, and misclassification at higher levels cannot be corrected.

Blocking During Self-training

We demonstrate the dynamics of the blocking mechanism during self-training. Figure 3.2(c) shows the average normalized entropy of the corresponding local classifier output for each document in **NYT** dataset, and Figure 3.2(d) shows the total number of blocked documents during the self-training procedure at the final level. Recall that we enhance high-confident predictions to refine our model during self-training. Therefore, the average normalized entropy decreases during self-training, implying there is less uncertainty in the outputs of our model. Correspondingly, fewer documents will be blocked, resulting in more available documents for self-training.



(a) Pseudo documents generation

(b) Global classifier self-training

(c) Average normalized entropy

(d) Number of blocked documents

Figure 3.2: Component-wise evaluation on **NYT** dataset.

3.7 SUMMARY

In this chapter, we propose a weakly-supervised hierarchical text classification method WeSHClass. Our designed hierarchical network structure and training method can effectively leverage (1) different types of weak supervision sources to generate high-quality pseudo documents for better model generalization ability, and (2) class taxonomy for better performances than flat methods and greedy approaches. WeSHClass outperforms various supervised and weakly-supervised baselines in three datasets from different domains, which demonstrates the practical value of WeSHClass in real-world applications. In the future, it is interesting to study what kinds of weak supervision are most effective for the hierarchical text classification task and how to combine multiple sources together to achieve even better performance.

Table 3.2: Macro-F1 and Micro-F1 scores for all methods on three datasets, under two types of weak supervisions.

Methods	**NYT**				**arXiv**				**Yelp Review**			
	KEYWORDS		**DOCS**		**KEYWORDS**		**DOCS**		**KEYWORDS**		**DOCS**	
	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Hier-Dataless	0.593	0.811	-	-	0.374	0.594	-	-	0.284	0.312	-	-
Hier-SVM	-	-	0.142 (0.016)	0.469 (**0.012**)	-	-	0.049 (**0.001**)	0.443 (**0.006**)	-	-	0.220 (0.082)	0.310 (0.113)
CNN	-	-	0.165 (0.027)	0.329 (0.097)	-	-	0.124 (0.014)	0.456 (0.023)	-	-	0.306 (0.028)	0.372 (0.028)
WeSTClass	0.386	0.772	0.479 (0.027)	0.728 (0.036)	0.412	0.642	0.264 (0.016)	0.547 (0.009)	0.348	0.389	0.345 (0.027)	0.388 (0.033)
No-global	0.618	0.843	0.520 (0.065)	0.768 (0.100)	0.442	0.673	0.264 (0.020)	0.581 (0.017)	0.391	0.424	0.369 (0.022)	0.403 (0.016)
No-vMF	0.628	0.862	0.527 (0.031)	0.825 (0.032)	0.406	0.665	0.255 (0.015)	0.564 (0.012)	0.410	0.457	0.372 (0.029)	0.407 (0.015)
No-self-train	0.550	0.787	0.491 (0.036)	0.769 (0.039)	0.395	0.635	0.234 (0.013)	0.535 (0.010)	0.362	0.408	0.348 (0.030)	0.382 (0.022)
WeSHClass	**0.632**	**0.874**	**0.532** (**0.015**)	**0.827** (**0.012**)	**0.452**	**0.692**	**0.279** (0.010)	**0.585** (0.009)	**0.423**	**0.461**	**0.375** (**0.021**)	**0.410** (**0.014**)

Table 3.3: Sample generated pseudo document snippets of class "politics" for **NYT** dataset.

Doc #	Bag-of-words	Bag-of-words + reordering	movMF + LSTM language model
1	he's cup abortion bars have pointed use of lawsuits involving smoothen bettors rights in the federal exchange, limewire ...	the clinicians pianists said that the legalizing of the profiling of the ... abortion abortion abortion identification abortions ...	abortion rights is often overlooked by the president's 30-feb format of a moonjock period that offered him the rules to ...
2	first tried to launch the agent in immigrants were in a lazar and lakshmi definition of yerxa riding this we get very coveted as ...	majorities and clintons legalization, moderates and tribes lawfully ... lawmakers clinics immigrants immigrants immigrants ...	immigrants who had been headed to the united states in benghazi, libya, saying that mr. he making comments describing ...
3	the september crew members budget security administrator lat coequal representing a federal customer, identified the bladed ...	the impasse of allowances overruns pensions entitlement ... funding financing budgets budgets budgets budgets taxpayers ...	budget increases on oil supplies have grown more than a ezio of its 20 percent of energy spaces, producing plans by 1 billion ...

CHAPTER 4: CONCLUSION AND FUTURE WORK

In this thesis, we study the problem of text classification with minimal human supervisions to save human efforts from extensive hand-labeling when training neural models for text classification. Our framework generates pseudo training documents based on weak supervisions for pre-training the neural networks and performs self-training by bootstrapping on unlabeled data to refine the classifier. WeSTClass and WeSHClass do not require excessive labeled documents as training data but only weak supervisions such as surface label names, relevant keywords or very few labeled documents, yet they achieve inspiring classification performance on various benchmarks.

In the future, it will be interesting to study (1) how to combine different types of weak supervisions for even better classification performances; (2) how to simultaneously train the pseudo document generation module and the classification module for end-to-end weakly-supervised text classification; (3) how to incorporate another type of weak supervision—potentially noisy-labeled documents—to refine the neural classification model.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[2] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 2005.

[3] K. Batmanghelich, A. Saeedi, K. Narasimhan, and S. Gershman. Nonparametric spherical topic modeling with word embeddings. In *ACL*, 2016.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(1):993–1022, 2003.

[5] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, 2004.

[6] M. Ceci and D. Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28:37–78, 2006.

[7] M. Chang, L. Ratinov, D. Roth, and V. Srikumar. Importance of semantic representation: Dataless classification. In *AAAI*, pages 830–835, 2008.

[8] X. Chen, Y. Xia, P. Jin, and J. A. Carroll. Dataless text classification with descriptive LDA. In *AAAI*, pages 2224–2231, 2015.

[9] S. T. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR*, 2000.

[10] R. Fisher. Dispersion on a sphere. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 1953.

[11] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, 2007.

[12] K. Ganchev, J. Gillenwater, B. Taskar, et al. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.

[13] S. Gopal and Y. Yang. Von mises-fisher clustering models. In *ICML*, 2014.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[15] R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. In *HLT-NAACL*, 2015.

[16] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[17] O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 2015.

[18] C. Li, J. Xing, A. Sun, and Z. Ma. Effective document labeling with very few seed words: A topic model approach. In *CIKM*, 2016.

[19] K. Li, H. Zha, Y. Su, and X. Yan. Unsupervised neural categorization for scientific publications. In *SDM*, 2018.

[20] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7:36–43, 2005.

[21] Y. Lu and C. Zhai. Opinion integration through semi-supervised topic modeling. In *WWW*, pages 121–130, 2008.

[22] Y. Meng, J. Shen, C. Zhang, and J. Han. Weakly-supervised neural text classification. In *CIKM*, 2018.

[23] Y. Meng, J. Shen, C. Zhang, and J. Han. Weakly-supervised hierarchical text classification. In *AAAI*, 2019.

[24] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[25] T. Miyato, A. M. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. 2016.

[26] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[27] A. Oliver, A. Odena, C. Raffel, E. D. Cubuk, and I. J. Goodfellow. Realistic evaluation of semi-supervised learning algorithms. 2018.

[28] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *WWW*, 2018.

[29] Y. Prabhu and M. Varma. Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning. In *KDD*, 2014.

[30] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *WACV/MOTION*, 2005.

[31] C. N. Silla and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22:31–72, 2010.

[32] R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, 2011.

[33] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, 2011.

[34] Y. Song and D. Roth. On dataless hierarchical text classification. In *AAAI*, pages 1579–1585, 2014.

[35] S. Sra. Directional statistics in machine learning: a brief review. *arXiv preprint arXiv:1605.00316*, 2016.

[36] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, 2012.

[37] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, 2015.

[38] J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*, pages 1165–1174, 2015.

[39] G. Tsur, Y. Pinter, I. Szpektor, and D. Carmel. Identifying web queries with question intent. In *WWW*, 2016.

[40] J. Xie, R. B. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, 2016.

[41] W. Xu, H. Sun, C. Deng, and Y. Tan. Variational autoencoder for semi-supervised text classification. In *AAAI*, 2017.

[42] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489, 2016.

[43] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *KDD*, pages 595–604, 2017.

[44] X. Zhang and Y. LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015.

[45] X. Zhang, J. J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.