# Weakly-Supervised Hierarchical Text Classification

**Yu Meng, Jiaming Shen, Chao Zhang, Jiawei Han**
University of Illinois at Urbana-Champaign, Urbana, IL, USA
{yumeng5, js2, czhang82, hanj}@illinois.edu

## Abstract

Hierarchical text classification, which aims to classify text documents into a given hierarchy, is an important task in many real-world applications. Recently, deep neural models are gaining increasing popularity for text classification due to their expressive power and minimum requirement for feature engineering. However, applying deep neural networks for hierarchical text classification remains challenging, because they heavily rely on a large amount of training data and meanwhile cannot easily determine appropriate levels of documents in the hierarchical setting. In this paper, we propose a weakly-supervised neural method for hierarchical text classification. Our method does not require a large amount of training data but requires only easy-to-provide weak supervision signals such as a few class-related documents or keywords. Our method effectively leverages such weak supervision signals to generate pseudo documents for model pre-training, and then performs self-training on real unlabeled data to iteratively refine the model. During the training process, our model features a hierarchical neural structure, which mimics the given hierarchy and is capable of determining the proper levels for documents with a blocking mechanism. Experiments on three datasets from different domains demonstrate the efficacy of our method compared with a comprehensive set of baselines.

## Introduction

Hierarchical text classification, which aims at classifying text documents into classes that are organized into a hierarchy, is an important text mining and natural language processing task. Unlike flat text classification, hierarchical text classification considers the interrelationships among classes and allows for organizing documents into a natural hierarchical structure. It has a wide variety of applications such as semantic classification (Tang, Qin, and Liu 2015), question answering (Li and Roth 2002), and web search organization (Dumais and Chen 2000).

Traditional flat text classifiers (*e.g.*, SVM, logistic regression) have been tailored in various ways for hierarchical text classification. Early attempts (Ceci and Malerba 2006) disregard the relationships among classes and treat hierarchical classification tasks as flat ones. Later approaches (Dumais and Chen 2000; Liu et al. 2005; Cai and Hofmann 2004) 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 (Kim 2014; Yang et al. 2016) 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.

In this work, we propose a neural approach named WeSHClass, 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.

Our contributions are summarized as follows:

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 propose 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 propose 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.

## 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 (Silla and Freitas 2010), 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 work, 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.

## 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.

**Modeling Class Distribution** We model each class as a high-dimensional spherical probability distribution which has been shown effective for various tasks (Zhang et al. 2017). We first train Skip-Gram model (Mikolov et al. 2013) to learn $d$-dimensional vector representations for each word in the corpus. Since directional similarities between vectors are more effective in capturing semantic correlations (Banerjee et al. 2005; Levy, Goldberg, and Dagan 2015), 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 (Banerjee et al. 2005; Gopal and Yang 2014) 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 $\kappa$, 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 (Banerjee et al. 2005) to estimate the parameters $\Theta$ 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 $\mathcal{Z} = \{z_1, \dots, z_n\}$ is the set of hidden random variables that indicate the particular vMF distribution from which the points are sampled;

- M-step:

$$\alpha_h^{(t+1)} = \frac{1}{n}\sum_{i=1}^{n} p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)}),$$

$$\boldsymbol{r}_h^{(t+1)} = \sum_{i=1}^{n} p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)})\boldsymbol{x}_i,$$

$$\boldsymbol{\mu}_h^{(t+1)} = \frac{\boldsymbol{r}_h^{(t+1)}}{\|\boldsymbol{r}_h^{(t+1)}\|},$$

$$\frac{I_{d/2}(\kappa_h^{(t+1)})}{I_{d/2-1}(\kappa_h^{(t+1)})} = \frac{\|\boldsymbol{r}_h^{(t+1)}\|}{\sum_{i=1}^{n} p(z_i = h \mid \boldsymbol{x}_i, \Theta^{(t)})}.$$

where we use the approximation procedure based on Newton's method (Banerjee et al. 2005) to derive an approximation of $\kappa_h^{(t+1)}$ because the implicit equation makes obtaining an analytic solution infeasible.

**Language Model Based Document Generation** After obtaining the distributions for each class, we use an LSTM-based language model (Sundermeyer, Schlüter, and Ney 2012) 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.

## The Hierarchical Classification Model

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

### 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 $\beta$ 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 $\alpha$ 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

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

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

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

labels $\mathcal{L}^*$, namely

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

## 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 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.

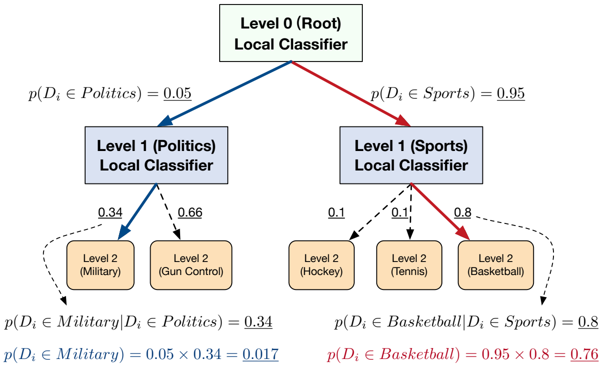

Level 0 (Root) Local Classifier
$p(D_i \in Politics) = \underline{0.05}$
$p(D_i \in Sports) = \underline{0.95}$
Level 1 (Politics) Local Classifier
Level 1 (Sports) Local Classifier
0.34
0.66
0.1
0.1
0.8
Level 2 (Military)
Level 2 (Gun Control)
Level 2 (Hockey)
Level 2 (Tennis)
Level 2 (Basketball)
$p(D_i \in Military | D_i \in Politics) = \underline{0.34}$
$p(D_i \in Basketball | D_i \in Sports) = \underline{0.8}$
$p(D_i \in Military) = 0.05 \times 0.34 = \underline{0.017}$
$p(D_i \in Basketball) = 0.95 \times 0.8 = \underline{0.76}$

Figure 1: Ensemble of local classifiers.

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 (Xie, Girshick, and Farhadi 2016):

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

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-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.

## 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

$$-\frac{1}{\log m} \sum_{i=1}^{m} q_i \log q_i > \gamma, \tag{3}$$

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.

## 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.

## Algorithm Summary

Algorithm 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.

# Experiments

## Experiment Settings

**Datasets and Evaluation Metrics** We use three corpora from three different domains to evaluate the performance of our proposed method:

**Algorithm 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 $\mathcal{C} \leftarrow \emptyset$;
for $k \leftarrow 0$ to $max\_level - 1$ do
    $\mathcal{N} \leftarrow$ all nodes at level $k$ of $\mathcal{T}$;
    foreach $node \in \mathcal{N}$ do
        $\mathcal{D}^* \leftarrow$ Pseudo document generation;
        $\mathcal{L}^* \leftarrow$ Equation (1);
        pre-train $node.classifier$ with $\mathcal{D}^*, \mathcal{L}^*$;
    $G_k \leftarrow$ ensemble all classifiers from level 0 to $k$;
    while not converged do
        $\mathcal{L}^{**} \leftarrow$ Equation (2);
        self-train $G_k$ with $\mathcal{D}, \mathcal{L}^{**}$;
    $\mathcal{D}_B \leftarrow$ documents blocked based on Equation (3);
    $\mathcal{C}_B \leftarrow \mathcal{D}_B$'s current class assignments;
    $\mathcal{C} \leftarrow \mathcal{C} \cup (\mathcal{D}_B, \mathcal{C}_B)$;
    $\mathcal{D} \leftarrow \mathcal{D} - \mathcal{D}_B$;
$\mathcal{C}' \leftarrow \mathcal{D}$'s current class assignments;
$\mathcal{C} \leftarrow \mathcal{C} \cup (\mathcal{D}, \mathcal{C}')$;
Return $\mathcal{C}$;
```

Table 1: Dataset Statistics.

| Corpus name | # 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 |

- **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 (Zhang, Zhao, and LeCun 2015) 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 3 super-categories.

Table 1 provides the statistics of the three datasets; Table 2 and 3 show some sample sub-categories of **NYT** and **arXiv** datasets. We use Micro-F1 and Macro-F1 scores as metrics for classification performances.

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

Table 2: Sample subcategories of NYT Dataset.

| Super-category (# children) | Sub-category |
|---|---|
| Politics (9) | abortion, surveillance, immigration, ... |
| Arts (4) | dance, television, music, movies |
| Business (4) | stocks, energy companies, economy, ... |
| Science (2) | cosmos, environment |
| Sports (7) | hockey, basketball, tennis, golf, ... |

Table 3: Sample subcategories of arXiv Dataset.

| Super-category (# children) | Sub-category |
|---|---|
| Math (25) | math.NA, math.AG, math.FA, ... |
| Physics (10) | physics.optics, physics.flu-dyn, ... |
| CS (18) | cs.CV, cs.GT, cs.IT, cs.AI, cs.DC, ... |

- **Hier-Dataless** (Song and Roth 2014): 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 (Gabrilovich and Markovitch 2007) 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** (Dumais and Chen 2000; Liu et al. 2005): 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** (Kim 2014): The CNN text classification model [5] can only take **document-level** supervision sources.
- **WeSTClass** (Meng et al. 2018): 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.
- **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 (Mikolov et al. 2013) to train 100-dimensional word embeddings for both movMF distributions modeling and classifier input embeddings. We set the pseudo label parameter

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

[3] https://arxiv.org/

[4] https://github.com/CogComp/cogcomp-nlp/tree/master/dataless-classifier

[5] https://github.com/alexander-rakhlin/CNN-for-Sentence-Classification-in-Keras

$\alpha = 0.2$, 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 (Hochreiter and Schmidhuber 1997) and Hierarchical Attention Networks (Yang et al. 2016). 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).

## Quantitative Comparision

We show the overall text classification results in Table 4. 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.

## 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 (Meng et al. 2018): 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 5. 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 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 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.

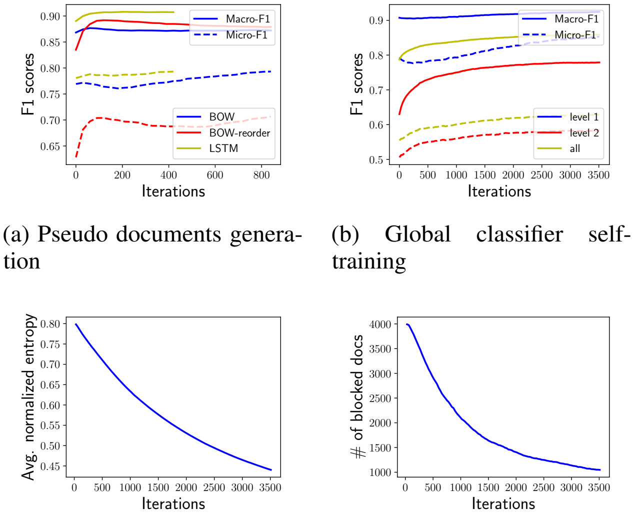
**Blocking During Self-training** We demonstrate the dynamics of the blocking mechanism during self-training. Figure 2(c) shows the average normalized entropy of the corresponding local classifier output for each document in **NYT** dataset, and Figure 2(d) shows the total number of blocked documents 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.

Table 4: 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<br>Avg. (Std.) | Micro<br>Avg. (Std.) | Macro | Micro | Macro<br>Avg. (Std.) | Micro<br>Avg. (Std.) | Macro | Micro | Macro<br>Avg. (Std.) | Micro<br>Avg. (Std.) |
| 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 5: 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 . . . |



(a) Pseudo documents generation

(b) Global classifier self-training

(c) Average normalized entropy

(d) Number of blocked documents

Figure 2: Component-wise evaluation on **NYT** dataset.

# Related Work

## 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 (Meng et al. 2018) 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 (Chen et al. 2015) 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. (Ganchev et al. 2010) propose to encode prior knowledge and indirect supervision in constraints on posteriors of latent variable probabilistic models. Predictive text embedding (Tang, Qu, and Mei 2015) 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.

## Hierarchical Text Classification

There have been efforts on using SVM for hierarchical classification. (Dumais and Chen 2000; Liu et al. 2005) 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. (Cai and Hofmann 2004) 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 (Peng et al. 2018) to convert text to graph-of-words, on which the graph convolution operations are applied for feature extraction. FastXML (Prabhu and Varma 2014) 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 WeSH-

Class does not require much training data but only weak supervision from users.

Hierarchical dataless classification (Song and Roth 2014) 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.

## Conclusions

We proposed 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.

## Acknowledgements

This research 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). We thank anonymous reviewers for valuable and insightful feedback.

## References

Banerjee, A.; Dhillon, I. S.; Ghosh, J.; and Sra, S. 2005. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research* 6:1345–1382.

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

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

Chen, X.; Xia, Y.; Jin, P.; and Carroll, J. A. 2015. Dataless text classification with descriptive lda. In *AAAI*.

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

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

Ganchev, K.; Graça, J.; Gillenwater, J.; and Taskar, B. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research* 11:2001–2049.

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

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

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

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL* 3:211–225.

Li, X., and Roth, D. 2002. Learning question classifiers. In *COLING*.

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

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

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

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

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

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

Song, Y., and Roth, D. 2014. On dataless hierarchical text classification. In *AAAI*.

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

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

Tang, J.; Qu, M.; and Mei, Q. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*.

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

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A. J.; and Hovy, E. H. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*.

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

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