Mixture of Expert/Imitator Networks: Scalable Semi-Supervised Learning Framework

Shun Kiyono,[1] Jun Suzuki,[1,2] Kentaro Inui[1,2]
[1]Tohoku University, [2]RIKEN Center for Advanced Intelligence Project
{kiyono, jun.suzuki, inui}@ecei.tohoku.ac.jp

Abstract

The current success of deep neural networks (DNNs) in an increasingly broad range of tasks involving artificial intelligence strongly depends on the quality and quantity of labeled training data. In general, the scarcity of labeled data, which is often observed in many natural language processing tasks, is one of the most important issues to be addressed. Semi-supervised learning (SSL) is a promising approach to overcoming this issue by incorporating a large amount of unlabeled data. In this paper, we propose a novel scalable method of SSL for text classification tasks. The unique property of our method, Mixture of Expert/Imitator Networks, is that imitator networks learn to "imitate" the estimated label distribution of the expert network over the unlabeled data, which potentially contributes a set of features for the classification. Our experiments demonstrate that the proposed method consistently improves the performance of several types of baseline DNNs. We also demonstrate that our method has the *more data, better performance* property with promising scalability to the amount of unlabeled data.

1 Introduction

It is commonly acknowledged that deep neural networks (DNNs) can achieve excellent performance in many tasks across numerous research fields, such as image classification (He et al. 2016), speech recognition (Amodei et al. 2016), and machine translation (Wu et al. 2016). Recent progress in these tasks has been primarily driven by the following two factors: (1) A large amount of labeled training data exists. For example, ImageNet (Deng et al. 2009), one of the major datasets for image classification, consists of approximately 14 million labeled images. (2) DNNs have the property of achieving better performance when trained on a larger amount of labeled training data, namely, the *more data, better performance* property.

However, collecting a sufficient amount of labeled training data is not always easy for many actual applications. We refer to this issue as the *labeled data scarcity* issue. This issue is particularly crucial in the field of natural language processing (NLP), where only a few thousand or even a few hundred labeled data are available for most tasks. This is because, in typical NLP tasks, creating the labeled data often requires the professional supervision of several highly skilled annotators. As a result, the cost of data creation is high relative to the amount of data.

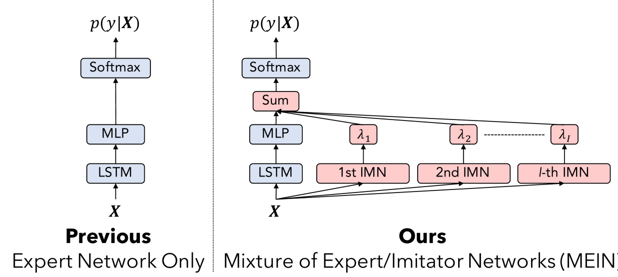


Figure 1: Overview of our framework: the Mixture of Expert/Imitator Networks (MEIN)

Unlike labeled data, unlabeled data for NLP tasks is essentially a collection of raw texts; thus, an enormous amount of unlabeled data can be obtained from the Internet, such as through the Common Crawl website[1], at a relatively low cost. With this background, semi-supervised learning (SSL), which leverages unlabeled data in addition to labeled training data for training the parameters of DNNs, is one of the promising approaches to practically addressing the labeled data scarcity issue in NLP. In fact, some intensive studies have recently been undertaken with the aim of developing SSL methods for DNNs and have shown promising results (Mikolov et al. 2013; Dai and Le 2015; Miyato, Dai, and Goodfellow 2017; Clark, Luong, and Le 2018; Peters et al. 2018).

In this paper, we also follow this line of research topic, i.e., discussing SSL suitable for NLP. Our interest lies in the *more data, better performance* property of the SSL approach over the unlabeled data, which has been implicitly demonstrated in several previous studies (Pennington, Socher, and Manning 2014; Peters et al. 2018). In order to take advantage of the huge amount of unlabeled data and improve performance, we need an SSL approach that scales with the amount of unlabeled data. However, the scalability of an SSL approach has not yet been widely dis-

[1]http://commoncrawl.org

cussed, since the primary focus of many of the recent studies on SSL in DNNs has been on improving the performance. For example, several studies have utilized unlabeled data as additional training data, which essentially increases the computational cost of (often complex) DNNs (Miyato, Dai, and Goodfellow 2017; Clark, Luong, and Le 2018; Sato et al. 2018). Another SSL approach is to (pre-)train a gigantic bidirectional language model (Peters et al. 2018). Nevertheless, it has been reported that the training of such a network requires 3 weeks using 32 GPUs (Jozefowicz et al. 2016). By developing a scalable SSL method, we hope to broaden the usefulness and applicability of DNNs since, as mentioned above, the amount of unlabeled data can be easily increased.

In this paper, we propose a novel scalable method of SSL, which we refer to as the Mixture of Expert/Imitator Networks (MEIN). Figure 1 gives an overview of the MEIN framework, which consists of an expert network (EXN) and at least one imitator network (IMN). To ensure scalability, we design each IMN to be computationally simpler than the EXN. Moreover, we use unlabeled data exclusively for training each IMN; we train the IMN so that it *imitates* the label estimation of the EXN over the unlabeled data. The basic idea underlying the IMN is that we force it to perform the imitation with only a limited view of the given input. In this way, the IMN effectively learns a set of features, which potentially contributes to the EXN. Intuitively, our method can be interpreted as a variant of several training techniques of DNNs, such as the mixture-of-experts (Jacobs et al. 1991; Shazeer et al. 2017), knowledge distillation (Ba and Caruana 2014; Hinton, Vinyals, and Dean 2015), and ensemble techniques.

We conduct experiments on well-studied text classification datasets to evaluate the effectiveness of the proposed method. We demonstrate that the MEIN framework consistently improves the performance for three distinct settings of the EXN. We also demonstrate that our method has the *more data, better performance* property with promising scalability to the amount of unlabeled data. In addition, a current popular SSL approach in NLP is to pre-train the language model and then apply it to downstream tasks (Mikolov et al. 2013; Dai and Le 2015; McCann et al. 2017; Peters et al. 2017; 2018). We empirically prove in our experiments that MEIN can be easily combined with this approach to further improve the performance of DNNs.

2 Related Work

There have been several previous studies in which SSL has been applied to text classification tasks. A common approach is to utilize unlabeled data as additional training data of the DNN. Studies employing this approach mainly focused on developing a means of effectively acquiring a teaching signal from the unlabeled data. For example, in virtual adversarial training (VAT) (Miyato, Dai, and Goodfellow 2017) the perturbation is computed from unlabeled data to make the baseline DNN more robust against noise. Sato et al. (2018) proposed an extension of VAT that generates a more interpretable perturbation. In addition, cross-view training (CVT) (Clark, Luong, and Le 2018) considers the auxiliary loss by making a prediction from an unlabeled input with a restricted view. On the other hand, in our MEIN framework, we do not use unlabeled data as additional training data for the baseline DNN. Instead, we use the unlabeled data to train the IMNs to imitate the baseline DNN. The advantage of such usage is that one can choose an arbitrary architecture for the IMNs. In this study, we design the IMN to be computationally simpler than the baseline DNN to ensure better scalability with the amount of unlabeled data (Table 4).

The idea of our *expert-imitator* approach originated from the SSL framework proposed by Suzuki and Isozaki (2008). They incorporated several simple generative models as a set of additional features for a supervised linear conditional random field classifier. Our EXN and IMN can be regarded as their linear classifier and the generative models, respectively. In addition, they empirically demonstrated that the performance has a linear relationship with the logarithm of the unlabeled data size. We empirically demonstrate that the proposed method also exhibits similar behavior (Figure 3), namely, increasing the amount of unlabeled data reduces the error rate of the EXN.

One of the major SSL approaches in NLP is to pre-train a language model over unlabeled data. The pre-trained weights have many uses, such as parameter initialization (Dai and Le 2015) and as a source of additional features (McCann et al. 2017; Peters et al. 2017; 2018), in downstream tasks. For example, Peters et al. (2018) have recently trained a bi-directional LSTM language model using the One Billion Word Benchmark dataset (Chelba et al. 2014). They utilized the hidden state of the LSTM as contextualized embedding, called *ELMo* embedding, and achieved state-of-the-art results in many downstream tasks. In our experiment, we empirically demonstrate that the proposed MEIN is complementary to the pre-trained language model approach. Specifically, we show that by combining the two approaches, we can further improve the performance of the baseline DNN.

3 Task Description and Notation Rules

This section gives a formal definition of the text classification task discussed in this paper. Let $\mathcal{V}$ represent the vocabulary of the input sentences. $\boldsymbol{x}_t \in \{0,1\}^{|\mathcal{V}|}$ denotes the one-hot vector of the t-th token (word) in the input sentence, where $|\mathcal{V}|$ represents the number of tokens in $\mathcal{V}$. Here, we introduce the short notation form $(\boldsymbol{x}_t)_{t=1}^{T}$ to represent a sequence of vectors for simplicity, that is, $(\boldsymbol{x}_t)_{t=1}^{T} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$. Suppose we have an input sentence that consists of T tokens. For a succinct notation, we introduce $\boldsymbol{X}$ to represent a sequence of one-hot vectors that corresponds to the tokens in the input sentence, namely, $\boldsymbol{X} = (\boldsymbol{x}_t)_{t=1}^{T}$. $\mathcal{Y}$ denotes a set of output classes. Let $y \in \{1, \dots, |\mathcal{Y}|\}$ be an integer that represents the output class ID. In addition, we define $\boldsymbol{X}_{a:b}$ as the subsequence of $\boldsymbol{X}$ from index a to index b, namely, $\boldsymbol{X}_{a:b} = (\boldsymbol{x}_a, \boldsymbol{x}_{a+1} \dots, \boldsymbol{x}_b)$ and $1 \leq a \leq b \leq T$. We also define $\boldsymbol{x}[i]$ as the i-th element of vector $\boldsymbol{x}$. For example, if $\boldsymbol{x} = (5, 2, 1, -1)^\top$, then $\boldsymbol{x}[2] = 2$ and $\boldsymbol{x}[4] = -1$.

In the supervised training framework for text classification tasks modeled by DNNs, we aim to maximize the (conditional) probability $p(y|\boldsymbol{X})$ over a given set of labeled training data $(\boldsymbol{X}, y) \in \mathcal{D}_s$ by using DNNs. In the semi-supervised training, the objective of maximizing the probability is identical but we also use a set of unlabeled training data $\boldsymbol{X} \in \mathcal{D}_u$.

4 Baseline Network: LSTM with MLP

In this section, we briefly describe a baseline DNN for text classification. Among the many choices, we select the *LSTM-based text classification model* described by Miyato, Dai, and Goodfellow (2017) as our baseline DNN architecture since they achieved the current best results on several well-studied text classification benchmark datasets. The network consists of the LSTM (Hochreiter and Schmidhuber 1997) cell and a multi layer perceptron (MLP).

First, the LSTM cell calculates a hidden state sequence $(\boldsymbol{h}_t)_{t=1}^{T}$, where $\boldsymbol{h}_t \in \mathbb{R}^H$ for all t and H is the size of the hidden state, as $\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{E}\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$. Here, $\boldsymbol{E} \in \mathbb{R}^{D\times|\mathcal{V}|}$ is the word embedding matrix, D denotes the size of the word embedding, and $\boldsymbol{h}_0$ is a zero vector.

Then the T-th hidden state $\boldsymbol{h}_T$ is passed through the MLP, which consists of a single fully connected layer with ReLU nonlinearity (Glorot, Bordes, and Bengio 2011), to compute the final hidden state $\boldsymbol{s} \in \mathbb{R}^M$. Specifically, $\boldsymbol{s}$ is computed as $\boldsymbol{s} = \mathrm{ReLU}(\boldsymbol{W}_h\boldsymbol{h}_T + \boldsymbol{b}_h)$, where $\boldsymbol{W}_h \in \mathbb{R}^{M\times H}$ is a trainable parameter matrix and $\boldsymbol{b}_h \in \mathbb{R}^M$ is a bias term. Here, M denotes the size of the final hidden state of the MLP.

Finally, the baseline DNN estimates the conditional probability from the final hidden state $\boldsymbol{s}$ as follows:

$$z_y = \boldsymbol{w}_y^\top \boldsymbol{s} + b_y, \tag{1}$$

$$p(y|\boldsymbol{X}, \boldsymbol{\Theta}) = \frac{\exp(z_y)}{\sum_{y'\in\mathcal{Y}} \exp(z_{y'})}, \tag{2}$$

where $\boldsymbol{w}_y \in \mathbb{R}^M$ is the weight vector of class y and b_y is the scalar bias term of class y. Also, $\boldsymbol{\Theta}$ denotes all the trainable parameters of the baseline DNN.

For the training process of the parameters in the baseline DNN $\boldsymbol{\Theta}$, we seek the (sub-)optimal parameters that minimize the (empirical) negative log-likelihood for the given labeled training data $\mathcal{D}_s$, which can be written as the following optimization problem:

$$\boldsymbol{\Theta}' = \underset{\boldsymbol{\Theta}}{\arg\min} \big\{ L_s(\boldsymbol{\Theta}|\mathcal{D}_s) \big\}, \tag{3}$$

$$L_s(\boldsymbol{\Theta}|\mathcal{D}_s) = -\frac{1}{|\mathcal{D}_s|} \sum_{(\boldsymbol{X},y)\in\mathcal{D}_s} \log\big(p(y|\boldsymbol{X}, \boldsymbol{\Theta})\big), \tag{4}$$

where $\boldsymbol{\Theta}'$ represents the set of obtained parameters in the baseline DNN, by solving the above minimization problem. Practically, we apply a variant of a stochastic gradient descent algorithm such as Adam (Kingma and Ba 2015).

5 Proposed Model: Mixture of Expert/Imitator Networks (MEIN)

Figure 1 gives an overview of the proposed method, which we refer to as MEIN. MEIN consists of an expert network (EXN) and a set of imitator networks (IMNs). Once trained, the EXN and the set of IMNs jointly predict the label of a given input $\boldsymbol{X}$. Figure 1 shows the baseline DNN (LSTM with MLP) as an example of the EXN. Note that MEIN can adopt an arbitrary classification network as the EXN.

5.1 Basic Idea

A brief description of MEIN is as follows: (1) The EXN is trained using labeled training data. Thus, the EXN is expected to be very accurate over inputs that are similar to the labeled training data. (2) IMNs (we basically assume that we have more than one IMN) are trained to imitate the EXN. To accomplish this, we train each IMN to minimize the Kullback–Leibler (KL) divergence between estimations of label distributions of the EXN and the IMNs over the unlabeled data. (3) Our final classification network is a mixture of the EXN and IMN(s). Here, we fine-tune the EXN using the labeled training data jointly with the estimations of all the IMNs.

The basic idea underlying MEIN is that we force each IMN to imitate estimated label distributions with only a limited view of the given input. Specifically, we adopt a sliding window to divide the input into several fragments of n-grams. Given a large amount of unlabeled data and the estimation by the EXN, the IMN learns to represent the label "tendency" of each fragment in the form of a label distribution (i.e., certain n-grams are more likely to have positive/negative labels than others). Our assumption here is that this tendency can potentially contribute a set of features for the classification. Thus, after training the IMNs, we jointly optimize the EXN and the weight of each feature. Here, MEIN may control the contribution of each feature by updating the corresponding weight.

Intuitively, our MEIN approach can be interpreted as a variant of several successful machine learning techniques for DNNs. For example, MEIN shares the core concept with the mixture-of-experts technique (MoE) (Jacobs et al. 1991; Shazeer et al. 2017). The difference is that MoE considers a mixture of several EXNs, whereas MEIN generates a mixture from a single EXN and a set of IMNs. In addition, one can interpret MEIN as a variant of the ensemble, bagging, voting, or boosting technique since the EXN and the IMNs jointly make a prediction. Moreover, we train each IMN by minimizing the KL-divergence between the EXN and the IMN through unlabeled data. This process can be seen as a form of "knowledge distillation" (Ba and Caruana 2014; Hinton, Vinyals, and Dean 2015). We utilize these methodologies and formulate the framework as described below.

5.2 Network Architecture

Let $\sigma(\cdot)$ be the sigmoid function defined as $\sigma(\lambda) = (1 + \exp(-\lambda))^{-1}$. $\boldsymbol{\Phi}$ denotes a set of trainable parameters of the IMNs and I denotes the number of IMNs. Then, the EXN combined with a set of IMNs models the following (condi-

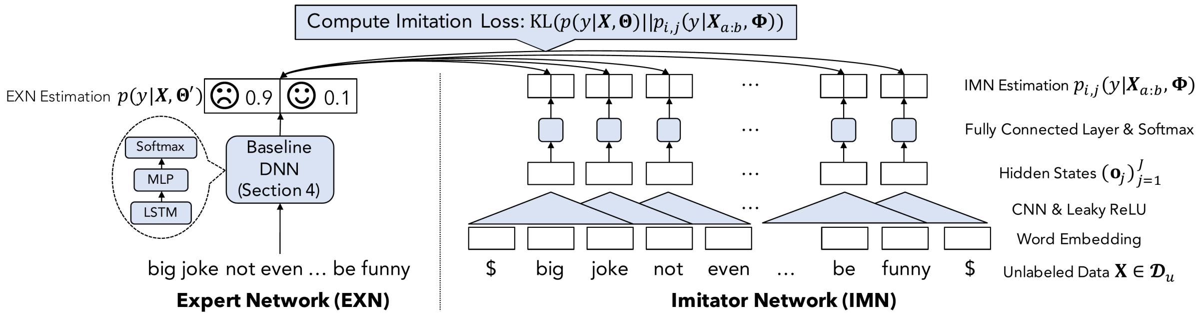


Figure 2: Overview of the 1st IMN ($c_1 = 1$). The IMN must predict the label estimation of the EXN from a limited amount of information. $ denotes a special token used to pad the input (a zero vector).

tional) probability:

$$p(y|\boldsymbol{X}, \boldsymbol{\Theta}, \boldsymbol{\Phi}, \boldsymbol{\Lambda}) = \frac{\exp(z'_y)}{\sum_{y' \in \mathcal{Y}} \exp(z'_{y'})}, \tag{5}$$

$$\text{where} \quad z'_y = z_y + \sum_{i=1}^{I} \sigma(\lambda_i)\boldsymbol{\alpha}_i[y]. \tag{6}$$

λ_i is a scalar parameter that controls the contribution of logit $\boldsymbol{\alpha}_i$ of the i-th IMN and $\boldsymbol{\Lambda}$ is defined as $\boldsymbol{\Lambda} = \{\lambda_1, \dots, \lambda_I\}$. Here, logit $\boldsymbol{\alpha}_i$ represents an estimated label distribution, which we assume to be a feature. Note that the first term of Equation 6 is the baseline DNN logit $z_y = \boldsymbol{w}_y^\top \boldsymbol{s} + b_y$ (Equation 1). In addition, if we set $\sigma(\lambda_i) = 0$ for all i, then Equation 5 becomes identical to Equation 2 regardless of the value of $\boldsymbol{\Phi}$.

c_i denotes the window size of the i-th IMN. Given an input $\boldsymbol{X}$ and the i-th IMN, we create J inputs with a sliding window of size c_i. Then the IMN predicts the EXN for each input and generates J predictions as a result. We compute the i-th imitator logit $\boldsymbol{\alpha}_i$ by taking the average of these predictions. Specifically, $\boldsymbol{\alpha}_i$ is defined as

$$\boldsymbol{\alpha}_i = \log\left(\frac{1}{J}\sum_{j=1}^{J} p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi})\right), \tag{7}$$
$$\text{where} \quad a = j - c_i \quad \text{and} \quad b = j + c_i.$$

Here, a is a scalar index that represents the beginning of the window. Similarly, b represents the last index of the window.

5.3 Definition of IMNs

Note that the architecture of the IMN used to model Equation 7 is essentially arbitrary. In this research, we adopt a single-layer CNN for modeling $p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi})$. This is because a CNN has high computational efficiency (Gehring et al. 2017), which is essential for our primary focus: scalability with the amount of unlabeled data.

Figure 2 gives an overview of the architecture of the IMN. First, the IMN takes a sequence of word embeddings of input $\boldsymbol{X}$ and computes a sequence of hidden states $(\boldsymbol{o}_j)_{j=1}^J$ by applying a *one-dimensional convolution* (Kalchbrenner, Grefenstette, and Blunsom 2014) and leaky ReLU nonlinearity (Maas, Hannun, and Ng 2013). We ensure that J is always equal to T. To achieve this, we pad the beginning and the end of the input $\boldsymbol{X}$ with zero vectors $\mathbf{0} \in \mathbb{R}^{|\mathcal{V}'| \times c_i}$, where $|\mathcal{V}'|$ denotes the vocabulary size of the IMN.

As explained in Section 5.2, each IMN has a predetermined and fixed window size c_i. One can choose an arbitrary window size for the i-th IMN. Here, we define c_i as $c_i = i$ for simplicity. For example, as shown in Figure 2, the 1st IMN ($i = 1$) has a window size of $c_1 = 1$. Such a network imitates the estimation of the EXN from three consecutive tokens.

Then the i-th IMN estimates the probability $p_{i,j}(y|\boldsymbol{X}, \boldsymbol{\Phi})$ from each hidden state $\boldsymbol{o}_j$ as

$$p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi}) = \frac{\exp(\boldsymbol{w}'^\top_{i,y}\boldsymbol{o}_j + b'_{i,y})}{\sum_{y' \in \mathcal{Y}} \exp(\boldsymbol{w}'^\top_{i,y'}\boldsymbol{o}_j + b'_{i,y'})}, \tag{8}$$

where $\boldsymbol{w}'_{i,y} \in \mathbb{R}^N$ is the weight vector of the i-th IMN and $b'_{i,y}$ is the scalar bias term of class y. N denotes the CNN kernel size.

Algorithm 1: Training framework of MEIN

Data: Labeled data $\mathcal{D}_s$ and unlabeled data $\mathcal{D}_u$
Result: Trained set of parameters $\widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{\Phi}}, \widehat{\boldsymbol{\Lambda}}$

$\boldsymbol{\Theta}' \leftarrow \underset{\boldsymbol{\Theta}}{\arg\min}\{L_s(\boldsymbol{\Theta}|\mathcal{D}_s)\}$ ▷ Train EXN (Equation 3)

$\widehat{\boldsymbol{\Phi}} \leftarrow \underset{\boldsymbol{\Phi}}{\arg\min}\{L_u(\boldsymbol{\Phi}|\boldsymbol{\Theta}', \mathcal{D}_u)\}$ ▷ Train IMN(s) (Equation 11)

$\widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{\Lambda}} \leftarrow \underset{\boldsymbol{\Theta},\boldsymbol{\Lambda}}{\arg\min}\{L'_s(\boldsymbol{\Theta}, \boldsymbol{\Lambda}|\widehat{\boldsymbol{\Phi}}, \mathcal{D}_s)\}$ ▷ Train EXN (Equation 13)

5.4 Training Framework

First, we define the *imitation loss* of each IMN as the KL-divergence between the estimations of the label distributions of the EXN and the IMN given (unlabeled) data $\boldsymbol{X}$, namely, $\mathrm{KL}(p(y|\boldsymbol{X}, \boldsymbol{\Theta})||p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi}))$. Note that this imitation loss is defined for an input with the sliding window $\boldsymbol{X}_{a:b}$. Thus, this definition effectively accomplishes the concept, i.e., the IMN making a prediction $p_{i,j}(y|\boldsymbol{X}_{a:b}, \boldsymbol{\Phi})$ from only a limited view of the given input $\boldsymbol{X}_{a:b}$.

Next, our objective is to estimate the set of optimal parameters by minimizing the negative log-likelihood of Equation 5 while also minimizing the total imitation losses for all

IMNs as biases of the network. Therefore, we jointly solve the following two minimization problems for the parameter estimation of MEIN:

$$\widehat{\boldsymbol{\Theta}}, \widehat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Theta},\boldsymbol{\Lambda}}{\arg\min}\{L'_s(\boldsymbol{\Theta},\boldsymbol{\Lambda}|\widehat{\boldsymbol{\Phi}},\mathcal{D}_s)\} \tag{9}$$

$$\widehat{\boldsymbol{\Phi}} = \underset{\boldsymbol{\Phi}}{\arg\min}\{L_u(\boldsymbol{\Phi}|\boldsymbol{\Theta}',\mathcal{D}_u)\}. \tag{10}$$

As described in Equations 9 and 10, we update the different sets of parameters depending on the labeled/unlabeled training data. Specifically, we use the labeled training data $(\boldsymbol{X}, y) \in \mathcal{D}_s$ to update the set of parameters in the EXN, $\boldsymbol{\Theta}$, and the set of mixture parameters of the IMNs, $\boldsymbol{\Lambda}$. In addition, we use the unlabeled training data $\boldsymbol{X} \in \mathcal{D}_u$ to update the parameters of the IMNs, $\boldsymbol{\Phi}$.

To ensure an efficient training procedure, the training framework of MEIN consists of three consecutive steps (Algorithm 1). First, we perform standard supervised learning to obtain $\boldsymbol{\Theta}'$ using labeled training data while keeping $\lambda_i = -\infty$ unchanged for all i during the training process to ensure that $\sigma(\lambda_i) = 0$ in Equation 6. Note that this optimization step is essentially equivalent to that of the baseline DNN (Equation 4).

Second, we estimate the set of IMN parameters $\boldsymbol{\Phi}$ by solving the minimization problem in Equation 10 with the following loss function:

$$L_u(\boldsymbol{\Phi}|\boldsymbol{\Theta}',\mathcal{D}_u) = \frac{1}{|\mathcal{D}_u|}\sum_{\boldsymbol{X}\in\mathcal{D}_u}\sum_{i=1}^{I}\sum_{j=1}^{J}\mathrm{KL}(p||p_{i,j}), \tag{11}$$

$$\mathrm{KL}(p||p_{i,j}) = -\sum_{y\in\mathcal{Y}} p(y|\boldsymbol{X},\boldsymbol{\Theta}')\log\left(p_{i,j}(y|\boldsymbol{X}_{a:b},\boldsymbol{\Phi})\right) + const, \tag{12}$$

where $\mathrm{KL}(p||p_{i,j})$ is a shorthand notation of the imitation loss $\mathrm{KL}(p(y|\boldsymbol{X},\boldsymbol{\Theta})||p_{i,j}(y|\boldsymbol{X}_{a:b},\boldsymbol{\Phi}))$ and $const$ is a constant term that is independent of $\boldsymbol{\Phi}$.

Finally, we estimate $\boldsymbol{\Theta}$ and $\boldsymbol{\Lambda}$ by solving the minimization problem in Equation 9 with the following loss function:

$$L'_s(\boldsymbol{\Theta},\boldsymbol{\Lambda}|\widehat{\boldsymbol{\Phi}},\mathcal{D}_s) = -\frac{1}{|\mathcal{D}_s|}\sum_{(\boldsymbol{X},y)\in\mathcal{D}_s}\log\Big(p(y|\boldsymbol{X},\boldsymbol{\Theta},\widehat{\boldsymbol{\Phi}},\boldsymbol{\Lambda})\Big). \tag{13}$$

6 Experiments

To investigate the effectiveness of MEIN, we conducted experiments on two text classification tasks: (1) a sentiment classification (SEC) task and (2) a category classification (CAC) task.

6.1 Datasets

For SEC, we selected the following widely used benchmark datasets: IMDB (Maas et al. 2011), Elec (Johnson and Zhang 2015), and Rotten Tomatoes (Rotten) (Pang and Lee 2005). For the Rotten dataset, we used the Amazon Reviews dataset (McAuley and Leskovec 2013) as unlabeled data, following previous studies (Dai and Le 2015; Miyato, Dai, and Goodfellow 2017; Sato et al. 2018). For CAC, we used the RCV1 dataset (Lewis et al. 2004). Table 1 summarizes the characteristics of each dataset[2].

Task	Dataset	Classes	Train	Dev	Test	Unlabeled
SEC	Elec	2	22,500	2,500	25,000	200,000
	IMDB	2	21,246	3,754	25,000	50,000
	Rotten	2	8,636	960	1,066	7,911,684
CAC	RCV1	55	14,007	1,557	49,838	668,640

Table 1: Summary of datasets. Each value represents the number of instances contained in each dataset.

6.2 Baseline DNNs

In order to investigate the effectiveness of the MEIN framework, we combined the IMN with following three distinct EXNs and evaluated their performance:

- LSTM: This is the baseline DNN (LSTM with MLP) described in Section 4.
- LM-LSTM: Following Dai and Le (2015), we initialized the embedding layer and the LSTM with a pre-trained RNN-based language model (LM) (Bengio et al. 2003). We trained the language model using the labeled training data and unlabeled data of each dataset. Several previous studies have adopted this network as a baseline (Miyato, Dai, and Goodfellow 2017; Sato et al. 2018).
- ADV-LM-LSTM: Adversarial training (ADV) (Goodfellow, Shlens, and Szegedy 2015) adds small perturbations to the input and makes the network robust against noise. Miyato, Dai, and Goodfellow (2017) applied ADV to LM-LSTM for a text classification. We used the reimplementation of their network.

Note that these three EXNs have an identical network architecture, as described in Section 4. The only difference is in the initialization or optimization strategy of the network parameters.

To the best of our knowledge, ADV-LM-LSTM provides a performance competitive with the current best result for the configuration of supervised learning (using labeled training data only).

Thus, if the IMN can improve the performance of a strong baseline, the results will strongly indicate the effectiveness of our method.

6.3 Network Configurations

Table 2 summarizes the hyperparameters and network configurations of our experiments. We carefully selected the settings commonly used in the previous studies (Dai and Le 2015; Miyato, Dai, and Goodfellow 2017; Sato et al. 2018).

We used a different set of vocabulary for the EXN and the IMNs. We created the EXN vocabulary $\mathcal{V}$ by following the previous studies (Dai and Le 2015; Miyato, Dai, and Goodfellow 2017; Sato et al. 2018), i.e., we removed the tokens

[2]DBpedia (Lehmann et al. 2015) is another widely adopted CAC dataset. We did not use this dataset in our experiment because it does not contain unlabeled data.

	Hyperparameter	Value
EXN (baseline DNN)	Word Embedding Dim. (D)	256
	Embedding Dropout Rate	0.5
	LSTM Hidden State Dim. (H)	1024
	MLP Dim. (M) for SEC Task	30
	MLP Dim. (M) for CAC Task	128
	Activation Function	ReLU
IMN	CNN Kernel Dim. (N)	512
	Word Embedding Dim.	512
	Activation Function	Leaky ReLU
	Number of IMNs (I)	4
Optimization	Algorithm	Adam
	Mini-Batch Size	32
	Initial Learning Rate	0.001
	Fine-tune Learning Rate	0.0001
	Decay Rate	0.9998
	Baseline Max Epoch	30
	Fine-tune Max Epoch	30

Table 2: Summary of hyperparameters

that appear only once in the whole dataset. We created the IMN vocabulary $\mathcal{V}'$ by byte pair encoding (BPE) (Sennrich, Haddow, and Birch 2016)[3]. The BPE merge operations are jointly learned from the labeled training data and unlabeled data of each dataset. We set the number of BPE merge operations to 20,000.

6.4 Results

Table 3 summarizes the results on all benchmark datasets, where the evaluation metric is the error rate. Therefore, a lower value indicates better performance. Here, all the reported results are **the average of five distinct trials** using five different random seeds. Moreover, for each trial, we automatically selected the best network in terms of the performance on the validation set among the networks obtained at every epoch. For comparison, we also performed experiments on training baseline DNNs (LSTM, LM-LSTM, and ADV-LM-LSTM) with incorporating random vectors as the replacement of IMNs, which is denoted as "+IMN (Random)". Moreover, we present the published results of VAT-LM-LSTM (Miyato, Dai, and Goodfellow 2017) and iVAT-LSTM (Sato et al. 2018) in the bottom three rows of Table 3, which are the current state-of-the-art networks that adopt unlabeled data. For VAT-LM-LSTM, we also report the result of the reimplemented network, denoted as "VAT-LM-LSTM (rerun)". As shown in Table 3, incorporating the IMNs consistently improved the performance of all baseline DNNs across all benchmark datasets. Note that the source of these improvements is not the extra set of parameters $\boldsymbol{\Lambda}$ but the outputs of the IMNs. We can confirm this fact by comparing the results of IMNs, "+IMN", with those of random vectors, "+IMN (Random)", since the difference between these two settings is the incorporation of IMNs or random vectors.

[3]We used sentencepiece (Kudo and Richardson 2018) (https://github.com/google/sentencepiece) for the BPE operations.

Method	Elec	IMDB	Rotten	RCV1
LSTM	10.09	10.98	26.47	14.14
LSTM+IMN (Random)†	9.87	10.75	27.27	14.04
LSTM+IMN†	**8.83**	**10.04**	**24.93**	**12.31**
LM-LSTM†	5.72	7.25	16.80	8.37
LM-LSTM+IMN (Random)†	5.71	7.01	16.78	7.83
LM-LSTM+IMN†	**5.48**	**6.51**	**15.91**	**7.53**
ADV-LM-LSTM†	5.38	6.58	15.73	7.89
ADV-LM-LSTM+IMN (Random)†	5.34	6.27	15.11	7.78
ADV-LM-LSTM+IMN†	**5.14***	**6.07***	**13.98**	**7.51***
VAT-LM-LSTM (rerun) †	5.47	6.20	18.50	8.44
VAT-LM-LSTM (`Miyato 2017`)†	5.54	5.91	19.1	7.05
VAT-LM-LSTM (`Sato 2018`)†	5.66	5.69	14.26	11.80
iVAT-LSTM (`Sato 2018`)†	5.18	5.66	14.12	11.68

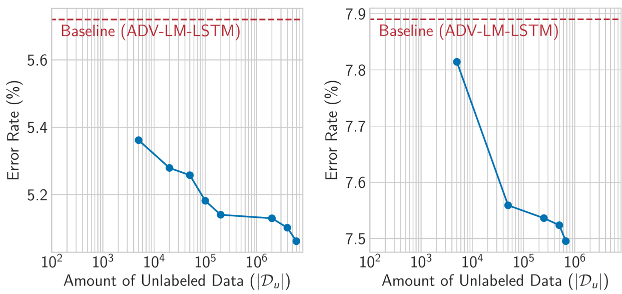
Table 3: Test performance (error rate (%)) on each dataset. **A lower error rate indicates better performance.** Models using the unlabeled data are marked with †. Results marked with * are statistically significant compared with ADV-LM-LSTM. `Miyato 2017`: the result reported by Miyato, Dai, and Goodfellow (2017). `Sato 2018`: the result reported by Sato et al. (2018).

The most noteworthy observation about MEIN is that the amount of the improvement upon incorporating the IMN is nearly consistent, regardless of the performance of the base EXN. For example, Table 3 shows that the IMN reduced the error rates of LSTM, LM-LSTM, and ADV-LM-LSTM by 1.54%, 0.89%, and 1.22%, respectively, for the Rotten dataset. From these observations, the IMN has the potential to further improve the performance of much stronger EXNs developed in the future.

We also remark that our best configuration, ADV-LM-LSTM+IMN, outperformed VAT-LM-LSTM (rerun) on all datasets[4]. In addition, the best configuration outperformed the current best published results on the Elec and Rotten datasets, establishing new state-of-the-art results.

As a comparison with the current strongest SSL method, we combined the IMN with the current state-of-the-art VAT method, namely, VAT-LM-LSTM+IMN. In the Elec dataset, the IMN improved the error rate from 5.47% to 5.16%. This result indicates that the IMN and VAT have a complementary relationship. Note that utilizing VAT is challenging in terms of the scalability with the amount of unlabeled data. However, if sufficient computing resources exist, then VAT and the IMN can be used together to achieve even

[4]The performance of our VAT-LM-LSTM (rerun) is lower than the performances reported by Miyato, Dai, and Goodfellow (2017) except for the Elec and Rotten datasets. Through extensive trials to reproduce their results, we found that the hyperparameter of the RNN language model is extremely important in determining the final performance; therefore, the strict reproduction of the published results is significantly difficult. In fact, a similar difficulty can be observed in Table 3, where VAT-LM-LSTM (`Sato 2018`) has lower performance than VAT-LM-LSTM (`Miyato 2017`) on the Elec and RCV1 datasets. Thus, we believe that VAT-LM-LSTM (rerun) is the most reliable result for the comparison.



(a) Elec (b) RCV1

Figure 3: Error rate (%) at different amounts of unlabeled data. The x-axis is in log-scale. **A lower error rate indicates better performance.** The dashed horizontal line represents the performance of the base EXN (ADV-LM-LSTM).

higher performance.

7 Analysis

7.1 More Data, Better Performance Property

We investigated whether the MEIN framework has the *more data, better performance* property for unlabeled data. Ideally, MEIN should achieve better performance by increasing the amount of unlabeled data. Thus, we evaluated the performance while changing the amount of unlabeled data used to train the IMN.

We selected the Elec and RCV1 datasets as the focus of this analysis. We created the following subsamples of the unlabeled data for each dataset: {5K, 20K, 50K, 100K, Full Data} for Elec and {5K, 50K, 250K, 500K, Full Data} for RCV1. In addition, for the Elec dataset, we sampled extra unlabeled data from the electronics section of the Amazon Reviews dataset (McAuley and Leskovec 2013) and constructed {2M, 4M, 6M} unlabeled data[5]. For each (sub)sample, we trained ADV-LM-LSTM+IMN as explained in Section 6.

Figures 3a and 3b demonstrate that increasing the amount of unlabeled data improved the performance of the EXN. It is noteworthy that in Figure 3a, ADV-LM-LSTM+IMN trained with 6M data achieved an error rate of 5.06%, outperforming the best result in Table 3 (5.14%). These results explicitly demonstrate the *more data, better performance* property of the MEIN framework. We also report that the training process on the largest amount of unlabeled data (6M) only took approximately a day.

7.2 Scalability with Amount of Unlabeled Data

The primary focus of the MEIN framework is its scalability with the amount of unlabeled data. Thus, in this section, we compare the computational speed of the IMNs with that of the base EXN. We also compare the IMNs with the state-of-the-art SSL method, VAT-LM-LSTM, and discuss their scalability. Here, we focus on the computation in the training phase of the network, where the network processes both forward and backward computations.

Method	Tokens/sec	Relative Speed
LM-LSTM	41,914	-
ADV-LM-LSTM	13,791	0.33x
VAT-LM-LSTM	9,602	0.23x
IMN ($c_i = 1$)	555,613	13.26x
IMN ($c_i = 1, 2$)	236,065	5.63x
IMN ($c_i = 1, 2, 3$)	122,076	2.91x
IMN ($c_i = 1, 2, 3, 4$)	75,393	1.80x

Table 4: Number of tokens processed per second during the training

We measured the number of tokens that each network processes per second. We used identical hardware for each measurement, namely, a single NVIDIA Tesla V100 GPU. We used the cuDNN implementation for the LSTM cell since it is highly optimized and substantially faster than the naive implementation (Bradbury et al. 2017).

Table 4 summarizes the results. The table shows that even the slowest IMN ($c_i = 1, 2, 3, 4$) was 1.8 times faster than the optimized cuDNN LSTM network and eight times faster than VAT-LM-LSTM. This indicates that it is possible to use an even larger amount of unlabeled data in a practical time to further improve the performance of the EXN. In addition, note that each IMN can be trained in *parallel*. Thus, if multiple GPUs are available, the training can be carried out much faster than reported in Table 4.

7.3 Effect of Window Size of the IMN

In this section, we investigate the effectiveness of combining IMNs with different window sizes c_i on the final performance of the EXN. Figure 4 summarizes the results across all datasets. The figure shows that integrating an IMN with a greater window size consistently reduced the error rate, and the IMN with the greatest window size (**D**: $c_i = 1, 2, 3, 4$) achieved the best performance. This observation implies that the context, which is captured by a greater window size, contributes to the performance.

8 Discussion

8.1 Variations of the IMN

In this section, we discuss two possible variations of the IMN to better understand its effectiveness in the MEIN framework.

Incorporating IMN with Greater Window Size As discussed in Section 7.3, Figure 4 demonstrates that increasing the window size of the IMN consistently improves the performance. From this observation, one may hypothesize that integrating an IMN with an even greater window size will be beneficial. Thus, we carried out an experiment with such

[5]We discarded instances from the unlabeled data when the non stop-words overlap with instances in the Elec test set. Thus, the unlabeled data and the Elec test set had no instances in common.

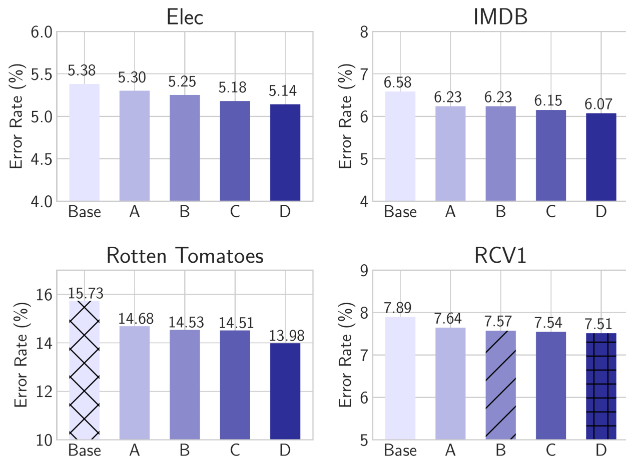


Figure 4: Effect of the IMN with different window sizes c_i on the final error rate (%) of ADV-LM-LSTM. **A lower error rate indicates better performance. Base**: EXN (ADV-LM-LSTM) without the IMN, **A**: $c_i = 1$, **B**: $c_i = 1, 2$, **C**: $c_i = 1, 2, 3$, **D**: $c_i = 1, 2, 3, 4$.

Window Size	Error Rate (%)
$c_i = 1, 2, 3, 4$	5.14
$c_i = 2, 3, 4$	5.18
$c_i = 3, 4$	5.26
$c_i = 4$	5.23

Table 5: Effect of removing IMNs with smaller window sizes on the error rate (%) of ADV-LM-LSTM on the Elec dataset. **A lower error rate indicates better performance.**

a configuration, i.e., $c_i = 1, 2, 3, 4, 5$, and found that the hypothesis is valid. For example, the error rates of ADV-LM-LSTM+IMN ($c_i = 1, 2, 3, 4, 5$) were 5.12% and 6.00% for Elec and IMDB, respectively, which are better than the values reported in Table 3.

However, we found that a large window size has a major drawback; the training of IMNs becomes significantly slower. This undesirable property must be avoided as our primary focus is the scalability with the amount of unlabeled data. Thus, we do not report these values as the main results of the experiment in Table 3.

Removing IMNs with Smaller Window Sizes We also investigated the effectiveness of utilizing IMNs with smaller window size in addition to the larger window sizes. Table 5 gives the results of this investigation, and we can see that combining IMNs with smaller window sizes works better than incorporating a single IMN with the greatest window size.

8.2 Stronger Baseline DNN

In this section, we discuss the results of two attempts to improve the performance of baseline DNNs.

Increasing Number of Parameters The most straightforward means of improving the performance of baseline DNNs is to increase the number of parameters. Thus, we doubled the word embedding dimension and trained ADV-LM-LSTM, namely, the ADV-LM-LSTM-Large model. This model has approximately the same number of parameters as the ADV-LM-LSTM+IMN. However, the performance did not improve from that of the original ADV-LM-LSTM. Specifically, the error rate degraded by 0.08 points for the IMDB dataset and was unchanged for the Elec dataset.

Combining ELMo ELMo (Peters et al. 2018) is one of the strongest SSL approaches in the research field. Thus, we conducted an experiment with a baseline that utilizes ELMo. Specifically, we combined LSTM with the ELMo embeddings, namely, ELMo-LSTM[6]. The error rate of this network on the IMDB test set was 8.67%, which is worse than that of LM-LSTM reported in Table 3. This result suggests that, at least in this task setting, pre-training the RNN language model for initialization is more effective than using the ELMo embeddings.

9 Conclusion

In this paper, we proposed a novel method for SSL, which we named Mixture of Expert/Imitator Networks (MEIN). The MEIN framework consists of a baseline DNN, i.e., an EXN, and several auxiliary networks, IMNs. The unique property of our method is that the IMNs learn to "imitate" the estimated label distribution of the EXN over the unlabeled data with only a limited view of the given input. In this way, the IMNs effectively learn a set of features that potentially contributes to improving the classification performance of the EXN.

Experiments on text classification datasets demonstrated that the MEIN framework consistently improved the performance of three distinct settings of the EXN. We also trained the IMNs with extra large-scale unlabeled data and achieved a new state-of-the-art result. This result indicates that our method has the *more data, better performance* property. Furthermore, our method operates eight times faster than the current strongest SSL method (VAT), and thus, it has promising scalability to the amount of unlabeled data.

References

Amodei, D.; Ananthanarayanan, S.; Anubhai, R.; Bai, J.; Battenberg, E.; Case, C.; Casper, J.; Catanzaro, B.; Cheng, Q.; Chen, G.; et al. 2016. Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. In *ICML*, 173–182.

Ba, J., and Caruana, R. 2014. Do Deep Nets Really Need to be Deep? In *NIPS*, 2654–2662.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A Neural Probabilistic Language Model. *JMLR* 3(Feb):1137–1155.

[6]We used the implementation available in AllenNLP (Gardner et al. 2017).

Bradbury, J.; Merity, S.; Xiong, C.; and Socher, R. 2017. Quasi-Recurrent Neural Networks. In *ICLR*.

Chelba, C.; Mikolov, T.; Schuster, M.; Ge, Q.; Brants, T.; Koehn, P.; and Robinson, T. 2014. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *INTERSPEECH*, 2635–2639.

Clark, K.; Luong, T.; and Le, Q. V. 2018. Cross-View Training for Semi-Supervised Learning. In *ICLR*.

Dai, A. M., and Le, Q. V. 2015. Semi-supervised Sequence Learning. In *NIPS*, 3079–3087.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 248–255.

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N. F.; Peters, M.; Schmitz, M.; and Zettlemoyer, L. S. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional Sequence to Sequence Learning. In *ICML*, 1243–1252.

Glorot, X.; Bordes, A.; and Bengio, Y. 2011. Deep Sparse Rectifier Neural Networks. In *AISTATS*, 315–323.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3(1):79–87.

Johnson, R., and Zhang, T. 2015. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In *NIPS*, 919–927.

Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; and Wu, Y. 2016. Exploring the Limits of Language Modeling. *arXiv preprint arXiv:1602.02410*.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A Convolutional Neural Network for Modelling Sentences. In *ACL*, 655–665.

Kingma, D., and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kudo, T., and Richardson, J. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *EMNLP*, 66–71.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P. N.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; et al. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6(2):167–195.

Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *JMLR* 5:361–397.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In *ACL*, 142–150.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.

McAuley, J., and Leskovec, J. 2013. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *RecSys*, 165–172.

McCann, B.; Bradbury, J.; Xiong, C.; and Socher, R. 2017. Learned in Translation: Contextualized Word Vectors. In *NIPS*, 6294–6305.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*, 3111–3119.

Miyato, T.; Dai, A. M.; and Goodfellow, I. 2017. Adversarial Training Methods For Semi-Supervised Text Classification. In *ICLR*.

Pang, B., and Lee, L. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL*, 115–124.

Pennington, J.; Socher, R.; and Manning, C. 2014. GloVe: Global Vectors for Word Representation. In *EMNLP*, 1532–1543.

Peters, M.; Ammar, W.; Bhagavatula, C.; and Power, R. 2017. Semi-supervised Sequence Tagging with Bidirectional Language Models. In *ACL*, 1756–1765.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *NAACL*, 2227–2237.

Sato, M.; Suzuki, J.; Shindo, H.; and Matsumoto, Y. 2018. Interpretable Adversarial Perturbation in Input Embedding Space for Text. In *IJCAI*, 4323–4330.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL*, 1715–1725.

Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*.

Suzuki, J., and Isozaki, H. 2008. Semi-Supervised Sequential Labeling and Segmentation Using Giga-Word Scale Unlabeled Data. In *ACL*, 665–673.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144*.