Weakly-supervised Text Categorization from Different Sources

Yu Meng
University of Illinois at Urbana-Champaign
yumeng5@illinois.edu

ABSTRACT
Supervised and semi-supervised text categorization have been well studied in data mining and NLP applications. Many conventional approaches rely on large amount of labeled documents to train a good classifier that is used later to categorize unlabeled data. In real-world scenarios, however, labeling a large number of documents is time consuming and sometimes even impossible. Therefore, it is often desirable to use methods that leverage weak supervision sources instead of massive labeled documents. In this paper, we propose a framework for text categorization that can leverage different kinds of weak supervision sources from users, including class surface names, class keywords or few labeled documents. Our experiments on several real-world datasets show that our framework outperforms several state-of-the-art weakly-supervised and semi-supervised algorithms, and that although weak supervision sources can be of different forms, if leveraged properly, they can all contribute to accurate text categorization.

KEYWORDS
Text Classification; Weakly-supervised Learning; Convolutional Neural Network

1 INTRODUCTION
Text categorization is a fundamental task in many data mining and NLP applications, such as web searching, information retrieval, and sentiment analysis. In recent years, the popularity of neural network models has resulted in wide application of many supervised and semi-supervised machine learning models in text categorization. [8, 9, 28, 29] show that convolutional neural network (CNN) can not only make effective use of word order for accurate text categorization but also be applied to understand text based on characters. Recurrent neural network (RNN)[20–22] can memorize a longer history and effectively construct sentence representations. While these neural network models achieve state-of-the-art performances in different domains of text categorization, they all rely on massive amount of labeled documents as training data.

However, in real-world applications, obtaining large amounts of training data is expensive because labeling documents requires human efforts and even domain specific knowledge. Even if human labeling is not a problem, sometimes obtaining enough raw data to be labeled can even be difficult. For example, onion sites on the darkweb use the Tor Hidden Service protocol[3] to hide their locations on the Internet. Since they are hosted in an anonymous manner, it is difficult to detect and shut them down using legal action. Many malicious and illegal business and trade can therefore take place in the marketplaces on the darkweb. In order to detect and monitor potential criminal activities carried out on the darkweb, it is vital to categorize and label product descriptions (short documents) on onion sites to identify malicious actions, such as illegal trade in weapons or drugs. Nevertheless, applying conventional text categorization approaches to label potentially illegal transactions is infeasible because collecting large amounts of training data requires crawling contents from onion sites, which is very complicated and potentially dangerous.

To deal with cases where training data is lacking, one may consider applying unsupervised algorithms that can automatically categorize documents into different clusters without the need of supervision sources. However, an obvious drawback of unsupervised learning is that user supervision cannot be effectively leveraged to affect categorization. This could be especially bad for corpora with multi-dimensional information because it is difficult to use the desired criterion to guide categorization results. For example, in the darkweb markets,
a product description usually contains information including its vendor, category and origin. If an unsupervised algorithm is directly applied, we have no control over which types of features (vendor, category or origin) will be mainly used for categorization.

Therefore, a desirable way of performing text categorization is to leverage weak supervision sources which are easier to obtain and can be used to guide categorization. Contrary to conventional supervision sources, i.e., massive labeled documents, weak supervision sources require minimum efforts from users and can be of different types. In this work, we aim to leverage three types of weak supervision sources: class label surface names, class keywords and few labeled documents. We first make an assumption that words, documents and class share a joint semantic space where we model the semantic of each class as a probability distribution from different types of weak supervision sources. Since no or few labeled documents are provided for training a classifier, we generate pseudo-documents to construct training data based on the distribution of each class. Finally, we use the generated pseudo-documents to pre-train a CNN and then apply a self-training procedure to iteratively refine CNN based on its high-confidence predictions.

The main contributions of this paper include:

- We propose a novel framework that models the semantic of each class as a probability distribution from different types of weak supervision sources.
- We propose a method for training CNN for text categorization under a semi-supervised or weakly-supervised setting.
- We conduct extensive experiments on several real-world datasets for weakly-supervised text categorization. Our method outperforms state-of-the-art algorithms on each type of the three kinds of weak supervision sources.

2 RELATED WORK

In this section, we introduce several weakly-supervised and semi-supervised approaches for text categorization.

Seed-guided topic model[13] takes a small set of seed words that are relevant to the semantic meaning of the category, and then predicts the category labels of the documents through two kinds of topic influence: category-topics and general-topics. Category-topics are associated with specific categories; the general-topics cover the general semantics of the whole corpus. Documents are classified based on posterior category-topic assignment.

Unsupervised neural categorization[14] takes category names as input and applies a cascade embedding approach: First the seeded category names and other significant phrases (concepts) are embedded into vectors to capture concept semantics. Then the concepts are embedded into a hidden category space to make the category information explicit. Finally documents are categorized by jointly considering the category attribution of their concepts.

Dataless classification[5] takes category names and project each word and document into the same semantic space of Wikipedia concepts. Each category is represented with words in the category label. The document classification is performed based on the vector similarity between a document and a category using explicit semantic analysis. [23] applies the same dataless approach for hierarchical text classification.

Predictive text embedding[26] is a semi-supervised algorithm that utilizes both labeled and unlabeled documents to learn text embedding specifically for a task. Labeled data and different levels of word co-occurrence information are first represented as a large-scale heterogeneous text network and then embedded into a low dimensional space that preserves the semantic similarity of words and documents. Classification is performed by using one-vs-rest logistic regression model as classifier and the learned embedding as input.

3 PRELIMINARIES

3.1 CNN for Text Classification

For text classification with CNN[9], the input to CNN is a document of length n represented by a concatenation of word vectors, i.e.,

\[ d = x_1 \oplus x_2 \oplus \cdots \oplus x_n \]

where \( \oplus \) is the concatenation operator; \( x_i \in \mathbb{R}^p \) is the p dimensional word vector of the ith word in the document. We use \( x_{i:j} \) to represent the concatenation of word vectors \( x_i, x_{i+1}, \ldots, x_j \). For window size of \( h \), a feature \( c_i \) is generated from a window of words \( x_{i:i+h-1} \) by the following convolution operation

\[ c_i = f(w \cdot x_{i:i+h-1} + b) \]
where $b \in \mathbb{R}$ is a bias term, $w \in \mathbb{R}^{hp}$ is the filter operating on $h$ words.

For each possible size-$h$ window of words, a feature map is generated as

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

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

### 3.2 Von Mises-Fisher Distribution

The von Mises-Fisher distribution (vMF)[2, 7] is a probability distribution on the $(p - 1)$ dimensional sphere in $\mathbb{R}^p$ whose probability density function is given by

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

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

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

where $I_r(\cdot)$ represents the modified Bessel function of the first kind at order $r$.

To fit a vMF distribution $f(x; \mu, \kappa)$ to a set of $n$ unit vectors, i.e.,

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

we use the maximum likelihood estimates[2, 24] for finding the parameters $\hat{\mu}$ and $\hat{\kappa}$ of the vMF distribution:

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i}{\| \sum_{i=1}^{n} x_i \|}$$

and

$$\frac{I_{p/2}(\hat{\kappa})}{I_{p/2-1}(\hat{\kappa})} = \frac{\| \sum_{i=1}^{n} x_i \|}{n}$$

It is not possible to obtain an analytic solution for $\hat{\kappa}$ because the formula involves an implicit equation which is a ratio of Bessel functions. An approximation of $\hat{\kappa}$ can be obtained by numerical or asymptotic methods[2].

### 4 METHODOLOGY

In semi-supervised learning, generative models and self-training techniques are widely used to leverage unlabeled data for accurate classification. However, previous semi-supervised methods still require considerable labeled data for pre-training models or model selection, and therefore fail to generalize to real-world applications where no or few labeled data are available. In this section, we describe a framework for weakly-supervised learning that involves a generative model and a self-training strategy: the generative model generates pseudo-documents which are used to pre-train CNN, then the self-training strategy is applied to iteratively refine the current predictions by learning from the CNN’s high confidence classifications.

#### 4.1 Pseudo-documents Generation

The power of generative models in traditional semi-supervised classification tasks have been largely exploited. In generative adversarial networks (GANs)[6], the generative model produces samples from a data distribution and the discriminative model distinguishes the generated samples from the real ones. [19] extends GANs for semi-supervised classification by adding a "generated" class and maximizing the probability that real samples belong to real classes and generated samples belong to the generated class. However, in order for the model to perform well, massive labeled data are still required. Another attempt[10] leveraging generative models constructs a stacked generative semi-supervised model and perform variational inference to predict missing labels, which are treated as latent variables of the model. In a weakly-supervised classification setting, however, variational inference can be problematic unless an efficient and appropriate generative model is carefully selected, which is unrealistic due to the lack of training and validation sets. In this work, the proposed generative model only leverages weak user supervision sources and does not require labeled data.

##### 4.1.1 Word-Document-Class Joint Semantic Space Assumption

Similar to the assumptions in [1, 15] that topic/sentence embeddings reside in the same $p$-dimensional space as word embeddings, we assume that each document $D_i$ is associated with a document vector $d_i \in \mathbb{R}^p$ in the word embedding space. Since in semantic analysis for both words and documents, the vector direction is usually relevant instead of its magnitude[2, 12, 25], we further assume that document vectors and word
embeddings are all unit vectors generated by class related distributions. Based on the above assumptions, we model the semantic of each class as a vMF distribution \( f(x; \mu, \kappa) = c_p(\kappa)e^{\kappa\mu^T x} \), where by analogy with mean and variance in Gaussian distribution, we can interpret \( \mu \) as the semantic centroid and \( \kappa \) as the semantic concentration of the class. Therefore, we have assumed that words, documents and classes share a joint semantic space \( \mathbb{R}^p \) where words and documents are represented by \( p \)-dimensional unit vectors and classes are modeled by vMF distributions on the \( (p-1) \) dimensional sphere in \( \mathbb{R}^p \). This assumption provides flexibility for handling different kinds of weak supervision sources.

### 4.1.2 Handling Different Supervision Sources

We can model each class by a vMF distribution from different kinds of weak supervision sources: class label surface names, keywords, or few labeled documents. Below we explain the details of finding the vMF distribution based on each kind of supervision source (we assume that both document vectors and word embeddings are already normalized unit vectors):

- **Labels**: Class label surface names
  For each class \( j \), given the class label surface name \( w_j \), we first form a \( t \) keyword cluster by finding the top-\( t \) words with the most similar word embedding (measured by dot product) with \( w_j \), and then fit a vMF distribution to the embeddings of the \( t \) words. \( t \) can be set as the largest number that does not result in overlapping of keyword clusters across different classes.

- **Keywords**: Class keyword lists
  For each class \( j \), given the class keyword list \( L_j = \{w_{j1}, w_{j2}, \ldots, w_{jn}\} \), we first form a \( t(t \geq n) \) keyword cluster by finding the top-\( t \) words with the most similar word embedding (measured by dot product) with the average embedding of all keywords in \( L_j \), and then fit a vMF distribution to the embeddings of the \( t \) words.

- **Documents**: Few labeled documents
  For each class \( j \), given few labeled documents \( D_{1j}, D_{2j}, \ldots, D_{cj} \) of class \( j \), we first extract \( n \) keywords based on averaged tf-idf weighting over the \( c \) documents. The remaining procedure becomes the same with keyword case.

### 4.1.3 Generating Pseudo-documents

To generate a pseudo-document \( D_i^j \) (we use \( D_i^j \) instead of \( D_i \) to denote it is a pseudo-document) of class \( j \) given class \( j \)'s distribution \( f(x; \mu_j, \kappa_j) \), we first sample a document vector \( d_i \) from \( f(x; \mu_j, \kappa_j) \), then build a keyword vocabulary \( V_{di} \) for \( d_i \) that contains the top-\( m \) words with most similar word embedding (measured by dot product) with \( d_i \). The \( m \) words in \( V_{di} \) are highly semantically relevant with the topic of pseudo-document \( D_i^j \) and will appear frequently in \( D_i^j \). Finally, the pseudo-document is generated according to the following word probability distribution

\[
p(w | d_i) = \begin{cases} \alpha p_B(w) & w \notin V_{di} \\ \alpha p_B(w) + (1-\alpha) \frac{\exp(d_i^T w)}{\sum_{w' \in V_{di}} \exp(d_i^T w')} & w \in V_{di} \end{cases}
\]

where \( p_B(w) \) is the word embedding for \( w \); \( p_B(w) \) is the background word distribution in the entire corpus. This generative model design is similar to [1]; the model generates a word from the background distribution with probability \( \alpha \) and from the document keyword vocabulary with probability \( 1-\alpha \). The length of each pseudo-document is equal to the average document length among the entire corpus.

### 4.2 CNN Training

Self-training[17, 18] is a common strategy used in semi-supervised learning. A simple way of performing self-training is to first train the model with labeled data, and then bootstrap the learning model with its current highly-confident predictions. When no or few labeled data are available, however, it becomes difficult to pre-train a good model for the later bootstrapping step. In this part, we describe the approach to pre-train CNN with the generated pseudo-documents and self-train CNN to improve its classification performance.

#### 4.2.1 CNN Pre-training

If supervision source is **Labels** or **Keywords**, only pseudo-documents generated as described in Section 4.1 will be used to pre-train the CNN model; otherwise (supervision source is **Documents**), we pre-train the CNN model using both pseudo-documents and the provided labeled documents simultaneously. A naive way of creating the label for a pseudo-document \( D_i^j \) is to treat \( D_i^j \) as if it is a given labeled document and use one hot encoding where the class that \( D_i^j \) is generated from takes value 1 and all other classes are set to 0. However, this approach will result in CNN over-fitting to pseudo-documents and behaving poorly when
classifying real documents. To avoid this problem, we create pseudo-labels for pseudo-documents. In [11], the pseudo-labels are target classes (still one hot encodings) for unlabeled data and are used the same way as true labels. However, the pseudo-labels we describe here are designed for pseudo-documents instead of unlabeled documents, and they represent probability distributions among all classes instead of one hot encodings. Based on equation 1, a pseudo-document is generated from a mixture of background word distributions weighted by $\alpha$ and the keyword vocabulary weighted by $1 - \alpha$. We evenly split the fraction of the background distribution into all $N$ classes, and the pseudo-label $l_i$ for pseudo-document $D_i'$ is designed to be

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

Pre-training is performed by generating $k$ pseudo-documents for each class, and minimizing the KL divergence loss from CNN outputs $Y$ to the pseudo-labels $L$, i.e.

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

### 4.2.2 CNN Self-training.

After the pre-training step, we use the pre-trained CNN to classify all unlabeled documents in the corpus and then apply a self-training strategy to improve the current predictions. During the self-training process, we iteratively compute pseudo-labels based on CNN’s current predictions and refine CNN’s parameters by training CNN with pseudo-labels. Given the current CNN outputs $Y$, the pseudo-labels are computed using the same self-training formula as in [27]:

$$l_{ij} = \frac{y_{ij}^2/f_j}{\sum_f y_{ij}^2/f_f}$$

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

Self-training is performed by iteratively computing pseudo-labels and minimizing the KL divergence loss from the current predictions $Y$ to the pseudo-labels $L$. This process stops when less than $\delta%$ of the documents in the corpus change class assignments between two iterations. Although self-training and pre-training both create pseudo-labels and use them to train CNN, the reasonings of using pseudo-labels are different: in pre-training, pseudo-labels are paired with generated pseudo-documents to distinguish them from given labeled documents (if provided) and prevent CNN from overfitting to pseudo-documents; in self-training, pseudo-labels are paired with every unlabeled real documents from corpus and reflect CNN’s high confidence predictions.

## 5 EXPERIMENTS

### 5.1 Datasets

Many traditional benchmark text classification datasets have equal number of documents per class, but real-world corpus usually have unbalanced number of documents. Therefore, in addition to using balanced benchmark datasets, we also create unbalanced real-world datasets to thoroughly test the performances of different methods. Table 1 provides the statistics of all the document corpora, including the dataset name, classification type, class names with corresponding number of documents, average document length in the corpus and document length used in classification (shorter documents and pseudo-documents will be padded and longer ones will be truncated to classification length).

**The New York Times**: We crawl 13081 news articles using the New York Times API\(^1\). The corpus covers 5 major news topics.

**AG’s News**: We employ the same AG’s News dataset from [29] and take the training set portion (120000 documents evenly distributed into 4 classes) as the corpus for evaluation.

**Yelp Review**: We employ the Yelp reviews polarity dataset from [29] and take the testing set portion (38000 documents evenly distributed into 2 classes) as the corpus for evaluation.

### 5.2 Baselines

We compare the performance of our method with multiple baselines that perform semi-supervised or weakly-supervised document classification. We mark the types of weak supervision sources that can be leveraged by each baseline.

**IR with TF-IDF**: If supervision source is *Labels*, we treat each class label name as a keyword query and compute the TF-IDF weighting of each class name for each document. The class label with the highest TF-IDF value is assigned to the corresponding document. If supervision source is *Keywords*, we compute the average TF-IDF weightings of keywords in each class

\(^1\)http://developer.nytimes.com/
Table 1: Dataset Statistics

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Type</th>
<th>Classes (# Documents)</th>
<th>Avg. Len.</th>
<th>Classification Len.</th>
</tr>
</thead>
<tbody>
<tr>
<td>The New York Times</td>
<td>Topic</td>
<td>Politics (1451), Arts (1043), Business (1429), Science (519), Sports (8639)</td>
<td>778</td>
<td>1500</td>
</tr>
<tr>
<td>AG’s News</td>
<td>Topic</td>
<td>Politics (30000), Sports (30000), Business (30000), Technology (30000)</td>
<td>45</td>
<td>100</td>
</tr>
<tr>
<td>Yelp Review</td>
<td>Sentiment</td>
<td>Good (19000), Bad (19000)</td>
<td>155</td>
<td>500</td>
</tr>
</tbody>
</table>

for each document and assign the class label according to the highest average TF-IDF value of keywords.

**Topic Modeling:** We train LDA model[4] on the entire corpus. If supervision source is **Labels**, we compute the likelihood of observing a class label name given a document and assign the most likely label to it. If supervision source is **Keywords**, we compute the likelihood of observing each keyword given a document. The class with the maximum average likelihood of keywords will be assigned as the label of the document.

**Dataless:** Dataless classification[5, 23] takes **Labels** as weak supervision source. It leverages Wikipedia and use explicit semantic analysis to derive vector representations of labels and documents. Labels are assigned based on vector similarity between labels and documents.

**UNEC:** Unsupervised neural categorization[14] takes **Labels** as weak supervision source. It categorizes documents by learning the semantics and category attribution of concepts inside the corpus.

**CNN:** We use the same CNN model that directly trains on **Documents** as weak supervision source. Then the CNN is used as the classifier to label all documents in the corpus.

**PTE:** Predictive text embedding[26] is a semi-supervised algorithm that utilizes both labeled and unlabeled data to learn text embedding and use logistic regression model as classifier for text categorization. **Documents** will be used in both the representation learning phase and the classifier learning phase.

**Pre-training:** We use our proposed method but only with pre-training procedure and without self-training procedure, i.e., after pre-training CNN with pseudo-documents, CNN will be directly used to classify documents.

5.3 Experiment Settings

In weakly-supervised learning, it is not possible to tune hyperparameters by cross-validation due to the lack of training and validation set. Therefore, although dataset specific settings of hyperparameters might result in better performance on each dataset, we avoid this kind of unrealistic parameter tuning as much as possible. For all datasets, we use a simple CNN model as the classifier with one convolutional layer, one max pooling layer and one fully connected layer with softmax output. The CNN filter window sizes are 2, 3, 4, 5 with 20 feature maps each. Pre-training and self-training are performed using Stochastic Gradient Descent with mini-batch size of 256. For all datasets, the word embeddings have dimensionality of 100 and are trained on the corresponding corpus using the continuous bag-of-words architecture of word2vec[16]. For hyperparameters used in our proposed framework, we set the background word distribution weight $\alpha = 0.2$, number of pseudo-documents for pre-training $k = 500$ and self-training stopping criterion $\delta = 0.5$.

When supervision source is **Labels**, we directly use the label surface names of all classes; when supervision source is **Keywords**, we manually select 3 keywords which do not include the class label name for each class as shown in Tables 2, 3 and 4; when supervision source is **Documents**, we randomly sample $c$ documents from the corpus and use them as the given labeled documents. Since **Documents** as supervision source involves randomness, we set 10 random seeds and show the performances with average and standard deviation values. We use macro-F1 scores (Macro) and micro-F1 scores (Micro) as the evaluation metrics.

Table 2: Keyword Lists for The New York Times Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th>Keyword List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Politics</td>
<td>democracy, religion, liberal</td>
</tr>
<tr>
<td>Arts</td>
<td>music, movie, dance</td>
</tr>
<tr>
<td>Business</td>
<td>investment, economy, industry</td>
</tr>
<tr>
<td>Science</td>
<td>scientists, biological, computing</td>
</tr>
<tr>
<td>Sports</td>
<td>hockey, tennis, basketball</td>
</tr>
</tbody>
</table>
Table 3: Keyword Lists for AG’s News Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th>Keyword List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Politics</td>
<td>government, military, war</td>
</tr>
<tr>
<td>Sports</td>
<td>basketball, football, athletes</td>
</tr>
<tr>
<td>Business</td>
<td>stocks, markets, industries</td>
</tr>
<tr>
<td>Technology</td>
<td>computer, telescope, software</td>
</tr>
</tbody>
</table>

Table 4: Keyword Lists for Yelp Review Dataset

<table>
<thead>
<tr>
<th>Class</th>
<th>Keyword List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good</td>
<td>terrific, great, awesome</td>
</tr>
<tr>
<td>Bad</td>
<td>horrible, disappointing, lousy</td>
</tr>
</tbody>
</table>

Figure 1: Dataset: The New York Times

5.4 Experiment Results

For all datasets under three different kinds of weak supervision sources, we show the self-training procedure in Figures 1, 2 and 3 and the categorization results in Tables 5, 6 and 7.

Our proposed method achieves the best performances among all the baselines on all datasets under three types of weak supervision sources. Interestingly, for different datasets, although the three types of supervision sources result in comparable performances, the best supervision source for each dataset is different: for the New York Times dataset, Labels gives the best performance; for AG’s News dataset, Documents results in the best performance; for Yelp Review dataset, Documents achieves the best.

6 DISCUSSIONS AND CONCLUSIONS

In this work, we propose a framework that can leverage three kinds of weak supervision sources for text categorization. We experiment with several real-world datasets and show that (1) our method outperforms all other state-of-the-art baselines under three different kinds of weak supervision sources; our method can be practically applied to real-world scenarios without requiring massive amount of labeled documents; (2) although Labels, Keywords and Documents are different types of supervision sources, they can produce comparably accurate categorization results if leveraged properly; (3) different real-world dataset “favors” different weak supervision sources; none of Labels, Keywords and Documents as weak supervision sources can always result in better categorization results than the other two sources for all datasets.

For an unseen dataset, it is usually not easy to predict which type of weak supervision sources will work best. Therefore, it is highly desirable to apply a heterogeneous weak supervision approach – combining
Table 5: Performances of all methods on The New York Times Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Labels</th>
<th>Keywords</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Macro</td>
<td>Micro</td>
<td>Macro</td>
</tr>
<tr>
<td>IR with TF-IDF</td>
<td>0.319</td>
<td>0.240</td>
<td>0.509</td>
</tr>
<tr>
<td>Topic Modeling</td>
<td>0.301</td>
<td>0.666</td>
<td>0.253</td>
</tr>
<tr>
<td>Dataless</td>
<td>0.484</td>
<td>0.710</td>
<td>-</td>
</tr>
<tr>
<td>UNEC</td>
<td>0.690</td>
<td>0.810</td>
<td>-</td>
</tr>
<tr>
<td>CNN</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pre-training</td>
<td>0.781</td>
<td>0.881</td>
<td>0.676</td>
</tr>
<tr>
<td>This Method</td>
<td>0.844</td>
<td>0.917</td>
<td>0.744</td>
</tr>
</tbody>
</table>

Table 6: Performances of all methods on AG's News Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Labels</th>
<th>Keywords</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Macro</td>
<td>Micro</td>
<td>Macro</td>
</tr>
<tr>
<td>IR with TF-IDF</td>
<td>0.187</td>
<td>0.292</td>
<td>0.258</td>
</tr>
<tr>
<td>Topic Modeling</td>
<td>0.496</td>
<td>0.584</td>
<td>0.723</td>
</tr>
<tr>
<td>Dataless</td>
<td>0.688</td>
<td>0.699</td>
<td>-</td>
</tr>
<tr>
<td>UNEC</td>
<td>0.659</td>
<td>0.668</td>
<td>-</td>
</tr>
<tr>
<td>CNN</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pre-training</td>
<td>0.679</td>
<td>0.704</td>
<td>0.751</td>
</tr>
<tr>
<td>This Method</td>
<td>0.829</td>
<td>0.830</td>
<td>0.821</td>
</tr>
</tbody>
</table>

Table 7: Performances of all methods on Yelp Review Dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Labels</th>
<th>Keywords</th>
<th>Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Macro</td>
<td>Micro</td>
<td>Macro</td>
</tr>
<tr>
<td>IR with TF-IDF</td>
<td>0.533</td>
<td>0.548</td>
<td>0.638</td>
</tr>
<tr>
<td>Topic Modeling</td>
<td>0.333</td>
<td>0.500</td>
<td>0.333</td>
</tr>
<tr>
<td>Dataless</td>
<td>0.337</td>
<td>0.500</td>
<td>-</td>
</tr>
<tr>
<td>UNEC</td>
<td>0.602</td>
<td>0.603</td>
<td>-</td>
</tr>
<tr>
<td>CNN</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PTE</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pre-training</td>
<td>0.506</td>
<td>0.586</td>
<td>0.796</td>
</tr>
<tr>
<td>This Method</td>
<td>0.730</td>
<td>0.731</td>
<td>0.828</td>
</tr>
</tbody>
</table>
multiple kinds of weak supervision sources for better categorization results, which could be an interesting topic for future work.

REFERENCES


