Weakly Supervised Short Text Categorization Using World Knowledge

Rima Türker[1,2][0000−0001−8332−7241], Lei Zhang[1][0000−0001−8184−9353], Mehwish Alam[1,2][0000−0002−7867−6612], and Harald Sack[1,2][0000−0001−7069−9804]

[1] FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany
[2] Karlsruhe Institute of Technology, Institute AIFB, Germany
{firstname.lastname}@fiz-karlsruhe.de
{firstname.lastname}@kit.edu

Abstract. Short text categorization is an important task in many NLP applications, such as sentiment analysis, news feed categorization, etc. Due to the sparsity and shortness of the text, many traditional classification models perform poorly if they are directly applied to short text. Moreover, supervised approaches require large amounts of manually labeled data, which is a costly, labor intensive, and time-consuming task. This paper proposes a weakly supervised short text categorization approach, which does not require any manually labeled data. The proposed model consists of two main modules: (1) a data labeling module, which leverages an external Knowledge Base (KB) to compute probabilistic labels for a given unlabeled training data set, and (2) a classification model based on a Wide & Deep learning approach. The effectiveness of the proposed method is validated via evaluation on multiple datasets. The experimental results show that the proposed approach outperforms unsupervised state-of-the-art classification approaches and achieves comparable performance to supervised approaches.

Keywords: Short Text Categorization · Weakly Supervised Short Text Categorization · Wide & Deep Model.

1 Introduction

Due to rapid growth of the Web content, short text data such as search snippets, news feeds, short messages, etc. is drastically multiplying online [3]. Hence, short text categorization has become a crucial task for a wide range of applications including sentiment analysis and news feed categorization [14]. While conventional text classification methods such as Support Vector Machines (SVMs) have demonstrated their success in classifying long and well structured text, as e.g., news articles, in case of short text they seem to have a substandard performance [33]. Moreover, due to the main characteristics of short text, i.e., limited context, sparsity and ambiguity, the traditional classification methods based on Bag of Words (BOW) [31] or approaches that utilize word embeddings perform poorly if directly applied to short text. Besides, such approaches often lead to

inaccurate results on new and rare words. Thus, to overcome these challenges, it is indispensable to use external sources such as Knowledge Bases (KBs) to enrich and obtain more advanced text representations [30].

Recently, several deep learning approaches have been proposed for short text classification, which demonstrated remarkable performance in this task [15,4]. The two main advantages of these models for the classification task are that minimum effort is required for feature engineering and their classification performance is better in comparison to traditional text classification approaches [16]. However, the requirement of large amounts of labeled data remains the main bottleneck for neural network based approaches [16]. Acquiring labeled data for the classification task is costly and time-consuming. Especially, if the data to be labeled is of a specific domain then only a limited number of domain experts are able to label them correctly, which makes it a labor intensive task.

To overcome this bottleneck several *dataless* [12,9], *semi supervised* [19,32], and *weakly supervised* [16,17] classification algorithms have been proposed. The *dataless classification* algorithms do not require any labeled data to perform text categorization. Instead, they project each predefined label and document into a common vector space by exploiting the words present in the labels and the documents. As a second step, based on the vector similarity a label is assigned to each document. However, the most prominent dataless classification methods are designed for long text, e.g., news article classification [12]. In addition, for addressing the labeled data scarcity problem, *semi supervised* text classification algorithms have been proposed. However, they also require some set of labeled data. Yet, generating small training sets for semi supervised methods still remains an expensive process due to the diversity of the documents in many applications. Furthermore, there has been a considerable amount of studies in *weakly supervised* text classification approaches. Most of these methods require user-given weak supervision sources such as some labeled documents, class related keywords, etc. for the classification task. Besides, existing weakly supervised text classification solutions mostly rely on hard-coded heuristics, such as looking for specific keywords or syntactical patterns in text, which still requires domain expertise and is especially prone to noise. Moreover, the most well-known weakly supervised methods are designed for long text classification.

Motivated by the aforementioned challenges, this paper proposes a novel model for **We**akly **S**upervised **S**hort **Te**xt **C**ategorization using World Knowledge[3] (WESSTEC). The proposed approach does not require any labeled data for short text categorization. It exploits Knowledge Bases and embedding models such as Doc2Vec [10], LINE [26], Word2Vec [18] etc. as weak supervision sources without requiring any manual effort. Instead, given a list of labels and unlabeled short text documents, the proposed method first associates each text with its relevant concepts in a KB to enhance the semantic representation of short texts and then generates labels for each document by utilizing the aforementioned embedding models. In the second step, words and concepts from the labeled documents are exploited for training a Wide & Deep learning based clas-

[3] https://github.com/ISE-FIZKarlsruhe/WESSTEC

sification model [5]. Finally, the trained model is used to categorize new short text documents. Overall, the main contributions of the paper are as follows:

- a new paradigm for short text categorization, based on a knowledge based weak supervision;
- a method to combine weak supervision sources to generate labeled data which can be used for any arbitrary classification model;
- adaptation of a Wide & Deep model for weakly supervised short text categorization;
- utilizing multiple features, i.e., both words and entities present in a given short text and their combination for the Wide & Deep model;
- an experimental evaluation using four different standard datasets for short text categorization.

The rest of this paper is structured as follows: Section 2 provides a review of the related work. In Section 3, the proposed approach for short text categorization is explained. Section 4 presents the experimental setup for the evaluation of the proposed approach and the discussion of the achieved results. Finally, Section 5 concludes the paper with open issues and future work.

2 Related Work

The aim of this study is to categorize short text documents under a weak supervision setting without requiring any manually labeled data. Hence, this section presents prior related studies on *Short Text Classification*, *Weakly Supervised Text Classification* as well as *Dataless Text Classification*.

Short Text Classification. Recent works [2,30,31] have proposed deep neural network based models to overcome the problem of data sparsity that arises when dealing with short text classification. The main characteristic of short text is the insufficient text length, which is no longer than 200 characters [24]. While [2,30,13] utilize an external knowledge to enrich the representation of short text, [31] exploits word embedding models and Convolutional Neural Network (CNN) to expand the information contained in short text. On the other hand, instead of focusing on expanding the representation of short text, [33] proposes topic memory networks which aim to encode latent topic representation of class labels for short text classification. In addition, recently, more sophisticated deep neural network based short text classification methods [4,15] have been proposed for sentiment analysis. Although the aforementioned approaches have demonstrated superior performance in text classification, they require huge amounts of labeled data. Conversely, the proposed method in this study does not require any manually labeled data for short text categorization.

Weakly Supervised Text Classification. There has been a considerable amount of studies related to weakly supervised text classification to address the problem of missing labeled data [16,17,21]. Most of these methods require user-given weak supervision sources such as class related key words, small amount of labeled data, etc. Hence, the requirement of domain expertise is still inevitable.

On the contrary, the proposed approach does not require such manually designed weak supervision sources for the categorization task. Instead, it utilizes unsupervised embedding models such as Word2Vec, Doc2Vec and LINE as weak supervision sources.

Dataless Text Classification. To address the problem of missing labeled data, [1] introduced a dataless text classification method by representing documents and labels into a common semantic space. Then, the classification is performed by considering the vector similarity between the documents and the labels. The most prominent dataless classification methods [1,12,9] utilize only words present in documents for the classification task and they ignore the entities. However, entities carry much more information than the words. Moreover, aforementioned studies are designed for the classification of long and well structured documents such as news articles. Such methods use traditional supervised approaches i.e., Naive Bayes (NB), Support Vector Machine (SVM), etc. with the features calculated based on the term frequency and the inverse document frequency to perform classification. In contrast to these studies, the proposed approach aims to categorize short text documents without requiring any labeled data and it utilizes entities as well as words present in documents for the classification task. Further, our approach exploits the Wide & Deep model for short text classification in the dataless scenario. The Wide & Deep model has been proposed by [5] for Recommendation Systems with the goal of jointly training a wide linear model (for memorization) alongside a deep neural network (for generalization).

The most recent work related with ours is Knowledge-Based Short Text Categorization (KBSTC) [29], which is a probabilistic model and does not require any labeled training data to perform short text categorization. Instead, the category of the given text is derived based on the semantic similarity between the entities present in the text and the set of predefined categories. KBSTC utilizes only entities and ignores the words. However, WESSTEC exploits words as well as entities. In addition, the proposed model leverages both textual information (in Doc2Vec model) and structural information (in LINE model) from KBs to better capture the semantic representation of entities, however, KBSTC uses only structural information of entities. Further, while KBSTC labels the input text only based on the heuristics of semantic similarity, WESSTEC adapts an additional classification model using Wide & Deep learning.

Last but not least, all previous approaches rely on a single model (e.g., [1] utilizes only word2vec, [29] utilizes only entity-and-category embedding) to categorize text data, while WESSTEC combines different embedding models to increase the accuracy and coverage.

3 Weakly Supervised Short Text Categorization

This section provides a formal definition of the short text classification task, followed by the description of the proposed approach.

Problem Formulation. Given an input short text t and n predefined labels $L = \{l_1, l_2, .., l_n\}$, the output is the most relevant label $l_i \in L$ for the given short

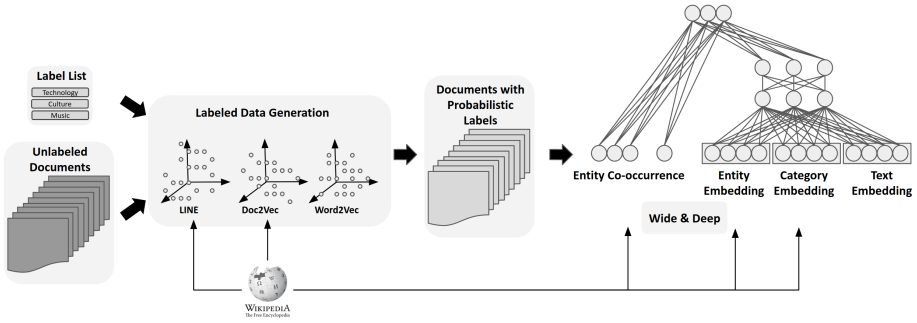

Fig. 1. The workflow of WESSTEC

text t, i.e., we compute the label function $f_{lab}(t) = l_i$, where $l_i \in L$.

Method Overview. The general workflow of WESSTEC is shown in Fig. 1. Given a list of labels and a set of unlabeled short text documents, the *Labeled Data Generation* module is responsible for generating *probabilistic training labels* for each document. In other words, it utilizes three different embedding models, i.e., LINE [26], Doc2Vec [10] and Word2Vec [18] to estimate the probability of each predefined label for a given document. This module generates documents with probabilistic labels as training data.

The second main module of the workflow is a Wide & Deep learning based classification model [5], which utilizes the documents with the probabilistic labels for training. Several different feature sets are extracted from the documents to train the Wide & Deep model. Note that in this work we have utilized Wikipedia as a KB.

Section 3.1 and Section 3.2 provide a detailed description of each module and the feature sets that have been utilized by each module.

3.1 Labeled Data Generation

The aim of this module is to generate labeled documents from a given label list and unlabeled set of documents (see Fig. 1). In other words, given a short text t and n labels $L = \{l_1, l_2, .., l_n\}$, the goal of this module is to produce a probabilistic label for t as $\mathbf{y_t} = [p_1, p_2, ..., p_n]$ where $p_i \in [0, 1]$, and p_i is the corresponding probability of l_i for t. To this end, this module utilizes three different embedding models, namely, LINE, Doc2Vec and pre-trained Word2Vec to capture the semantic correlations between the predefined labels and the words as well as entities present in a short text. First, each document and label is projected into common vector spaces, then the probabilistic labels of given texts are calculated based on the cosine similarity between documents and the set of predefined labels.

LINE is a network embedding model, which is designed to learn embedding of arbitrary types of large-scale networks (weighted, directed, undirected, etc.). The model has been trained by utilizing Wikipedia hyperlink structure to obtain a vector representation of each entity from Wikipedia. In other words, from

Wikipedia hyperlink structure, an *entity-network* has been constructed to be utilized by this model. More technical details about the construction of the *entity-network* can be found in [29]. To obtain a document vector with the help of LINE, we simply take the average of entity vectors present in that document. To extract entities from a document an *anchor text dictionary* [29,27,28] is used. The anchor text dictionary is constructed by leveraging the anchor texts of hyperlinks of Wikipedia, which are pointing to any article in Wikipedia. The anchor texts are considered as entity mentions and the links refer to the corresponding entities.

Doc2Vec creates the distributed representation of documents by utilizing the context words present in the corresponding documents. This model has been trained on Wikipedia articles and contains a vector representation of each entity of Wikipedia. Note that we consider each Wikipedia article page as an entity. To form a document vector for a given text, the average of entity vectors present in that text is considered.

Word2Vec learns the low dimensional distributed representation of words. We use the pre-trained Word2Vec model[4] for our approach. To create document vectors with Word2Vec, the average of the word vectors in that document is considered.

Moreover, each given label is also mapped to its corresponding vector in the respective vector space, e.g., the label *Music* is mapped to the word vector of *Music* from Word2Vec and it is also mapped to the entity vector of *Music* from Doc2Vec and LINE.

After embedding each text and label into common vector spaces, each embedding model assigns the most similar label to each text based on the vector similarity between the text and the labels. As there are three embedding models, for each given text three labels are generated. These labels can overlap or conflict. Then, the goal of the remaining process of this module is to convert the outputs of the embedding measures into *probabilistic training labels*. In order to achieve that a heuristic approach has been employed.

Based on outputs of each embedding measure for all texts, the heuristic approach estimates the *confidence* of each embedding model by considering the output label agreement and disagreement rates. The *confidence* of an embedding model EM_i is defined as follows:

$$C_{EM_i} = \frac{Agg_{EM_i} + noneAgg}{TotalAgg + noneAgg}\,, \tag{1}$$

where Agg_{EM_i} is the number of documents, on which the model EM_i agreed for a label with at least one of the other embeddings, $noneAgg$ is the number of documents, on which none of the embedding models agreed for the assigned label and $TotalAgg$ is the number of documents which at least two embedding models agreed on their labels (i.e., $TotalAgg = \#TotalDocuments - noneAgg$).

The confidence values are exploited to convert the generated labels into probabilistic training labels $\mathbf{y_t}$. For each text, the preferred label from each embed-

[4] https://code.google.com/archive/p/word2vec/

ding measure will be weighted using its confidence and then all three weighted labels are combined together, which could result in three probabilistic labels when three measures disagree with each other or two probabilistic labels when two measures agree or one label when all agree on it. Finally, these values are normalized to produce the probabilistic training labels $\mathbf{y_t}$.

Given a short text t and n labels $L = \{l_1, l_2, .., l_n\}$, let $\mathbf{y_t} = [p_1, p_2, ..., p_n]$ denote t's probabilistic training labels, where $p_i \in [0, 1]$. To calculate the probability p_i of the label l_i for t, we define the following formula:

$$p_i(t) = \frac{\sum_{j=1}^{e} C_{EM_j} I^i_{EM_j}(t)}{\sum_{k=1}^{n} \sum_{j=1}^{e} C_{EM_j} I^k_{EM_j}(t)}, \tag{2}$$

where e is the total number of embedding models that are utilized in *labeled data generation* module, C_{EM_j} is the confidence of embedding model EM_j, n is the total number of predefined labels and $I^i_{EM_j}$ is defined as

$$I^i_{EM_j}(t) = \begin{cases} 1 & \text{if } EM_j \text{ assigns } l_i \text{ to } t \\ 0 & \text{otherwise}. \end{cases}, \tag{3}$$

3.2 Wide and Deep Model for Short Text Categorization

The second main module of our workflow is a Wide & Deep learning based classification model which was proposed in [5] for Recommender Systems. We adapt this approach for the short text categorization task. To the best of our knowledge, this is the first attempt of utilizing the Wide & Deep model for short text categorization.

The model consists of two main components i.e., Wide Component and Deep Component. Moreover, the model has the ability of memorizing feature interactions and generalizing feature combinations by jointly training the wide and deep components as shown in Fig. 1 (right).

In the following, we first introduce the Wide model and Deep model separately and then present the joint Wide & Deep model.

Wide Model: The wide part has the ability of memorizing feature interactions. In other words, it is able to learn the frequent co-occurrence of features. Hence, we design this model to be able to capture the correlation between the co-occurrence of features and the target labels. In our approach, ***Entity co-occurrence*** information of each document is used as a feature for the wide part (see Fig. 1). Given a short text t let $\mathbf{x_t} = [x_1, x_2, x_3, ..., x_m]$ denote the m entities present in t. To construct the d dimensional *Entity co-occurrence* feature vector we apply cross-product transformation [5] as:

$$\phi_k(\mathbf{x_t}) = \prod_{i=1}^{m} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\}, \tag{4}$$

where c_{ki} is a boolean variable that is 1 if the i-th feature is part of the k-th transformation ϕ_k, and 0 otherwise. The wide part is a model of the form as:

$$P(Y = l_i|t) = softmax(\mathbf{w}_i^T \phi(\mathbf{x_t}) + b_i), \tag{5}$$

where t is a given short text, $\phi(\mathbf{x_t}) = [\phi_1(\mathbf{x_t}), \phi_2(\mathbf{x_t}), ..., \phi_d(\mathbf{x_t})]$ is the cross product transformations of $\mathbf{x_t}$, $\mathbf{w}_i = [w_1, w_2, ..., w_d]$ and b_i are the model parameters corresponding to the i-th label l_i. The *softmax* function is defined as:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{z_j \in \mathbf{z}} e^{z_j}}, \tag{6}$$

for $i = 1, 2, .., n$ and $\mathbf{z} = (z_1, z_2, ..., z_n) \in R^n$.

We give the following example to illustrate how an *Entity co-occurrence* feature vector can be formed. Given a short text *"Motorola and HP in Linux tie-up"*, the extracted entities are $E' = \{Motorola, HP, Linux\}$ and the possible entity pairs are $E'_p = \{(Motorola, HP), (Motorola, Linux), (HP, Linux)\}$. The dimension of the vector is the number of all the possible entity pairs of the dataset and each dimension corresponds to an entity pair. For each entity pair $e_{p_i} \in E'_p$, the value of the corresponding dimension of the vector would be 1 and the rest would be 0.

Deep Model: The deep part is a neural network, which is capable of generalization of feature combinations through low-dimensional dense embeddings. In our approach, three different embedding vectors, i.e., ***Entity Embedding***, ***Category Embedding*** and ***Text Embedding*** are utilized as an input to the deep part (see Fig. 1).

To construct each feature vector, different embedding models are utilized, i.e., for *Entity Embedding* LINE, for *Category Embedding* the joint entity and category embedding model [29] and for *Text Embedding* Word2Vec. The joint entity and category embedding model has been proposed by [29] to capture the semantic relations between entities and categories from a KB. This model first constructs a weighted network of entities and categories, and then jointly learns their embeddings from the network.

In order to form an *Entity Embedding* vector for a given text, entities present in the document are extracted with the help of a prefabricated Anchor-Text dictionary [29] and then the average of the vector representations of these entities is taken. For the *Category Embedding* feature vector, all the categories that are directly associated with the entities appearing in the text are collected from Wikipedia, then the average of the category vector representations is taken. Finally, for a given text a *Text Embedding* feature vector is constructed by taking the average of the word vector representations in that document.

The deep part is a feed forward neural network, which takes low-dimensional embedding vectors as an input i.e., $[\mathbf{e}^e_t, \mathbf{e}^c_t, \mathbf{e}^t_t]$, where $\mathbf{e}^e_t$ is the entity embedding, $\mathbf{e}^c_t$ is the category embedding and $\mathbf{e}^t_t$ is the text embedding.
The deep part is the model of the form as:

$$P(Y = l_i|t) = softmax(\mathbf{w}_i^T a^{(lf)} + b), \tag{7}$$

where $\mathbf{w}_i$ are the weights that are applied on the final activation $a^{(lf)}$ for the i-th label l_i, l is the layer number and f is the activation function which is ReLU.

We have built 3-layer feed forward neural network for the deep part and each hidden layer of this model performs the following computation [5]:

$$a^{(l+1)} = f(W^{(l)}a^{(l)} + b^{(l)}), \tag{8}$$

$a^{(l)}$ is activations, $b^{(l)}$ is bias and $W^{(l)}$ is model weights at l-th layer.

Wide & Deep Model: The wide and the deep components are combined for joint training by back propagating the gradients from the output of both wide and deep parts simultaneously. The combined model is illustrated in Fig. 1 (right). For a given short text t the prediction of Wide & Deep model is:

$$P(Y = l_i|t) = softmax({\mathbf{w}_{wide_i}}^T \phi(\mathbf{x}_t) + {\mathbf{w}_{deep_i}}^T a^{(lf)} + b_i). \tag{9}$$

In order to deal with the probabilistic training labels, we configure our model to train with a noise-aware loss function, i.e., cross-entropy between the probability of each training label and the output of the softmax function, which is defined as:

$$H(p, q) = -\sum_n p_i(t) * log(P(Y = l_i|t)) \tag{10}$$

Note that the reason of exploiting different feature sets in Labeled Data Generation and Wide & Deep modules is mainly two-fold: (1) Combining different features into Labeled Data Generation module requires much more feature engineering efforts. In other words, the Wide & Deep model can automatically learn the weights of the feature sets, however, it is not the same case with the proposed heuristic model designed for labeled data generation. (2) There are some features (e.g., entity co-occurrence) that cannot be straightforwardly integrated into heuristic algorithms to help calculate semantic similarity between input text and labels and do the labeling. However, such "non-heuristic" features can be transferred into the final classification model trained on labeled data generated by the heuristic algorithms using other features. Overall, we expect the trained model to provide performance gains over the heuristics that it is trained on both by applying to "non-heuristic" features (e.g., entity co-occurrence), and by learning to generalize beyond heuristics, i.e., putting weights on more subtle features that each individual heuristic algorithm cannot cover.

4 Experimental Results

This section provides a description of the datasets and the baselines, followed by the experimental results and a comparison to the state-of-the-art text categorization approaches.

4.1 Datasets

Four different real-world datasets have been used to evaluate the performance of the proposed approach: **AG News** [34], which contains the title and a short description of the news articles, **Snippets** [20], which contains short snippets from Google search results, **DBpedia Ontology** classification dataset [11], which is constructed by selecting 14 non-overlapping classes from DBpedia 2014 and

Twitter[5] topic categorization dataset contains tweets belong to 6 different categories. The Twitter dataset is preprocessed, in other words, the dataset does not contain hash symbols, emoticons, user mentions, etc. Besides, the special characters and numbers present in each dataset have been removed, further, each sample has been converted to lower case. Table 1 shows the distribution of the datasets, the average number of entities and words as well as the standard deviation of entities and words per text in each dataset.

Furthermore, as WESSTEC does not require any labeled training data, the training datasets of AG News, Snippets, DBpedia and Twitter have been used without their labels. In other words, the training set of each dataset without their labels have been utilized as an input to *Labeled Data Generation* module of the WESSTEC framework (see Fig. 1) to generate the training labels.

Table 1. Statistics for the short text datasets

Dataset	#Category	#Train	#Test	Avg. #Ent	Avg. #Word	SD Ent	SD Word
AG News	4	120,000	7,600	11.83	38.65	3.80	9.8
Snippets	8	10,060	2,280	8.90	17.97	3.56	4.84
DBpedia	14	560,000	70,000	15.30	46.49	6.9	21.57
Twitter	6	9,879	3,697	4.31	12.36	2.29	5.13

4.2 Baseline Approaches

To demonstrate the performance of the proposed approach, the following models have been selected as baselines:

- **Dataless ESA and Dataless Word2Vec:** Two variants of the state-of-the-art dataless approach [25] are considered as baselines, which are based on different methods to compute word similarity, i.e., ESA [8] and Word2Vec [18].
- **KBSTC [29]:** Knowledge-based short text categorization, which does not require any labeled data for short text categorization. Instead it relies on the semantic similarity between the given short text and predefined labels to categorize a given short text.
- **SVM+tf-idf:** In this model, the term frequency-inverse document frequency (tf-idf) is calculated as features for a subsequent Support Vector Machine (SVM) classifier.
- **CNN [35]+Word2Vec, CNN+Ent and CNN+Category:** A Convolutional Neural Network (CNN) is applied on text, entity and category matrices separately. These matrices are constructed by using Word2Vec, LINE, joint entity and category embedding model [29] respectively.
- **LSTM:** The standard LSTM model is composed of a single LSTM layer followed by a dense output layer.
- **charCNN [34]:** This model learns character embeddings using "one-hot" encoding. Subsequently, CNN is applied for the classification process.

[5] https://github.com/madhasri/Twitter-Trending-Topic-Classification/tree/master/data

Table 2. The classification accuracy of different models with different features

Model	Feature	AG News	Snippets	DBpedia	Twitter
Wide	Entity Co-occurance (Ent Co)	0.561	0.447	0.499	0.278
Deep	Text	0.802	0.795	0.786	0.555
	Entity	0.790	0.764	0.775	0.521
	Category	0.773	0.698	0.754	0.444
	Text+Entity	0.793	0.785	0.779	0.524
	Text+Category	0.801	0.794	0.786	0.554
	Entity+Category	0.792	0.771	0.771	0.534
	Text+Entity+Category	0.792	0.786	0.785	0.529
Wide & Deep	Ent Co+Text	0.807	0.792	0.786	0.556
	Ent Co+Entity	0.791	0.774	0.768	0.520
	Ent Co+Category	0.792	0.693	0.774	0.446
	Ent Co+Text+Entity	0.787	0.802	0.776	0.53
	Ent Co+Text+Category	**0.814**	0.803	**0.792**	**0.581**
	Ent Co+Entity+Category	0.791	0.770	0.766	0.544
	Ent Co+Text+Entity+Category	0.790	**0.805**	0.778	0.572

– **BERT [6]:** The state-of-the-art language representation model[6] have been commonly leveraged to derive sentence embeddings. To produce BERT embeddings, first, each sentence has been passed through pre-trained BERT, then the outputs of the model have been averaged, which is the most common way of obtaining sentence embeddings from BERT [23]. In the experiments, the BERT embeddings have been generated as features for the subsequent 3-layer feed forward neural network.

4.3 Feature Sets

This section describes the feature sets that have been extracted from the *Documents with Probabilistic Labels* (see Fig. 1) and utilized to train the Wide & Deep model. To construct feature sets, words and entities present in texts as well as parent categories of entities from Wikipedia have been leveraged.

As shown in Fig. 1, the wide part exploits the **Entity Co-occurrence (Ent Co)** information as a feature and the deep part utilizes three different feature sets, namely, **Text Embedding (Text)**, **Entity Embedding (Entity)** and **Category Embedding (Category)** vectors as well as their combinations, such as Text+Entity (see Table 2) refers to the concatenation of text embeddings and entity embeddings. The detailed construction of the feature sets is explained in Section 3.2.

4.4 Evaluation of WESSTEC

Table 2 depicts the classification accuracy of the Wide & Deep model of WESSTEC, in comparison to individual Wide-only and Deep-only models with different features on AG News, Snippets, DBpedia and Twitter datasets.

[6] https://github.com/google-research/bert

It has been observed that the jointly trained Wide & Deep model outperforms the individual Wide-only and Deep-only models on each dataset. The reason here can be attributed to the benefit of utilizing the Wide & Deep model to achieve both memorization and generalization of features for short text classification. In addition, we have observed that some of the wrongly classified samples with the Deep part, have been correctly classified after combining the Wide part and jointly training the model.

Wide model performs best on the AG News dataset. This dataset has the least number of categories and the length of the samples are not as limited as Twitter dataset, therefore, it is easier for the Wide model to handle this dataset in comparison to other datasets. The reason of the general low accuracy of the Wide model (in comparison to the Deep model and Wide & Deep model) is that a very sparse set of features have been used to train the model. It is a well known fact that the Deep Neural Networks (DNNs) can be much more powerful than the linear models. Therefore, the Deep model always outperforms the Wide model on each dataset. Similar to the Wide model, with the Deep model the best classification accuracy has been obtained on the AG news.

On the other hand, despite the specific properties of Tweets (e.g., out-of-vocabulary words) WESSTEC can still obtain reasonable accuracy on the Twitter dataset. To illustrate the difficulty of categorizing tweets, we give the following tweet from the Twitter dataset as an example: *"BSE NSE Stock Tip HINDUSZI"*, which is labeled as *"Business"*. The categorization of such tweets is rather difficult for many standard categorization models, which rely on only words. However, WESSTEC enriches text representations by leveraging entities present in texts and their associated categories with the help of a KB. For the given example the detected entities are `Bombay_Stock_Exchange`, `National_Stock_Exchange_of_India` and `Stock`, which capture very useful information for categorization of the tweet. Further, even for out-of-vocabulary words such as *"BSE"*, WESSTEC can still detect entities, which are crucial for the categorization task.

This study has also investigated the impact of each feature combination on the classification performance. The Deep model performs the best when utilizing only words. Whereas, the Wide & Deep model enjoys the combination of the feature sets. However, it has been observed that using entity features in both wide and deep parts could result in a bias of the whole model towards entity information, which might not reflect the entire semantics of text, especially when the text is longer such that there could be some more words that cannot be detected as entities (e.g., in AG News and DBpedia). This suggests that our Wide & Deep model (Ent-Co+Text+Category) using Entity Co-occurrence (Ent-Co) as a feature in the wide part as well as Text Embedding (Text) and Category Embedding (Category) as features in the deep part could be the most promising combination. The results in Table 2 also shows that (Ent-Co+Text+Category) clearly yields best results on AG News, DBpedia and Twitter datasets and performs only slightly worse than (Ent-Co+Text+Entity+Category) on Snippets dataset (with the difference of 0.002 for accuracy).

Overall, the experiments show that, firstly, it is possible to perform short text categorization with a high accuracy in the complete absence of labeled data with our proposed approach and secondly, the Wide & Deep model can be successfully applied for the short text categorization problem.

Since WESSTEC achieves almost the best performance with the combination of Ent-Co+Text+Category features, we use the results of this model for the comparison between WESSTEC and other approaches in the rest of the experiments.

4.5 Comparison of WESSTEC with the Unsupervised Approaches

Table 3 presents the classification accuracy of WESSTEC in comparison to the text classification approaches that do not require any labeled data.

It is observed that the proposed approach based on the Wide & Deep model considerably outperforms the dataless approaches as well as KBSTC. Although the dataless approaches achieved promising results in case of longer news articles in [25], they cannot perform well on short text due to the data sparsity problem.

KBSTC is a probabilistic model which does not require any training phase and it utilizes entities and categories from a KB for the categorization process. Whereas, WESSTEC first generates documents with probabilistic labels from a given unlabeled document set, then it utilizes those documents to train a Wide & Deep model to classify new documents. Moreover, WESSTEC exploits words present in text as well as entities and their directly associated categories from a KB for categorization. Hence, the proposed model is much more sophisticated and utilizes more features than the KBSTC model. Therefore, as expected the classification performance has been improved with the proposed approach.

Table 3. The classification accuracy against the unsupervised baselines

Model	AG News	Snippets	DBpedia	Twitter
Dataless ESA [25]	0.641	0.485	0.551	0.317
Dataless Word2Vec [25]	0.527	0.524	0.679	0.5
KBSTC [29]	0.805	0.720	0.460	0.359
WESSTEC	**0.814**	**0.803**	**0.792**	**0.581**

4.6 Comparison of WESSTEC with the Supervised Approaches

In order to show the effectiveness of the Wide & Deep Module (see Sec 3.2), its performance has been compared with the supervised baselines. The generated training sets of respective datasets (see Sec 3.1) have been utilized to train Wide & Deep as well as the baseline models. The respective original test datasets have been used for evaluating the trained models. Table 4 reports the classification performance.

The results show that the proposed Wide & Deep model can yield better accuracy in comparison to the baselines. This is due to the fact that in contrast to

other approaches, the Wide & Deep model is capable of both memorization and generalization of features and thus it performs the best among all the approaches. Moreover, especially on the Snippets dataset, Wide & Deep model significantly outperforms all the baselines. The reason here can be attributed to the different characteristics of this dataset. The Snippets dataset has less average number of entities, words per text and the size of the training set is much smaller in comparison to AG News and DBpedia (see Table 1). In contrast to baselines, the proposed Wide & Deep model utilizes different resources from a KB to enrich the semantic representations of texts. Thus, it is capable of categorizing of such a dataset with a high accuracy.

Another advantage of the Wide & Deep model over the baselines is different feature combinations (e.g., entity co-occurrence, text embedding, entity embedding, etc.) can be easily exploited by the model for the categorization task.

Furthermore, a statistical significance test, namely, the 5x2cv paired t-test [7] has been also performed to compare the results of Wide & Deep and BERT. This test has been proposed to overcome the drawbacks of other significance tests (e.g., resampled paired t-test) and it is based on five iterations of two-fold cross validation. According to 5x2cv paired t-test, the experimental results are significantly different at 95% level of significance with 5 degrees of freedom.

Overall, the obtained results in Table 4 suggest that in comparison to the baselines the Wide & Deep model is better suited for the short text categorization task by utilizing the generated labeled data for training.

Table 4. The classification accuracy against the supervised baselines. The baselines have been trained with the generated training sets (see Sec 3.1) of respective datasets.

Model	AG	Snippets	DBpedia	Twitter
SVM+tf-idf	0.808	0.696	0.784	0.513
CNN+W2V	0.796	0.787	0.784	0.542
CNN+Ent	0.794	0.703	78.24	0.456
CNN+Category	0.779	0.656	0.762	0.449
LSTM	0.786	0.693	0.796	0.473
charCNN	0.773	0.497	0.760	0.472
BERT	0.806	0.801	**0.804**	0.560
Wide & Deep	**0.814**	**0.803**	0.792	**0.581**

4.7 Evaluation of the Generated Labeled Data

To evaluate the performance of each embedding model, i.e., Word2Vec, Doc2Vec and LINE in the context of labeling the training data, we have conducted a set of experiments. First, each of the unlabeled documents and predefined labels has been projected into common vector spaces. Then each embedding model has assigned the most similar label to the documents based on the vector similarity. Additionally, by considering a simple majority vote of all the embedding models each document has also been labeled. The accuracy of labeled datasets has been calculated by comparing them with the original hand-labeled data. Table 5

presents the accuracy of the labeled training data based on the individual embedding models and the majority vote. The results suggest that considering all the embedding models for the labeling task can help in assigning more accurate labels. Therefore, to estimate the probabilistic labels for each training sample, all the embedding models have been used in the *Labeled Data Generation* module (see Sec. 3.1).

Further experiments have been conducted to asses the performance of the Wide & Deep model when it is trained on the training samples that are labeled based on majority vote. Table 6 presents the classification accuracy. The results show that using probabilistic labels in WESSTEC leads to higher-quality supervision for training the end classification model.

Table 5. The accuracy of generated training data based on the embedding models

Model	AG News	Snippets	DBpedia	Twitter
Vector Similarity LINE	0.776	0.657	0.708	0.536
Vector Similarity Doc2Vec	0.651	0.644	0.672	0.479
Vector Similarity Word2Vec	0.612	0.692	0.702	0.527
Vector Similarity (Majority)	**0.778**	**0.709**	**0.757**	**0.555**

Table 6. The classification accuracy of WESSTEC against the Wide & Deep model trained on majority vote based training set

Model	AG News	Snippets	DBpedia	Twitter
Wide & Deep (Majority)	0.812	0.799	0.772	0.559
WESSTEC	**0.814**	**0.803**	**0.792**	**0.581**

5 Conclusion and Future Work

In this study we have proposed WESSTEC, a new paradigm for weakly supervised short text categorization using world knowledge. The proposed model does not require any labeled data for categorizing documents. Instead, it first generates labeled training data from unlabeled documents by utilizing three different embedding models, i.e., Word2Vec, LINE, Doc2Vec. Several features are extracted from the labeled documents to train the Wide & Deep classification model. Finally, the new documents are classified with the help of this model. The experimental results have proven that WESSTEC is capable of categorizing short text documents with a high accuracy without requiring any labeled data and it significantly outperforms the classification approaches which do not require any labeled data. As for future work, we aim to (1) improve the labeled data generation process by exploiting advanced weak supervision approaches such as Snorkel [22]; (2) adopt WESSTEC with different KBs; (3) evaluate the performance of WESSTEC on more text classification benchmarks.

References

1. Chang, M.W., Ratinov, L.A., Roth, D., Srikumar, V.: Importance of semantic representation: Dataless classification. In: AAAI (2008)
2. Chen, J., Hu, Y., Liu, J., Xiao, Y., Jiang, H.: Deep short text classification with knowledge powered attention. In: AAAI (2019)
3. Chen, M., Jin, X., Shen, D.: Short text classification improved by learning multi-granularity topics. In: IJCAI (2011)
4. Chen, P., Sun, Z., Bing, L., Yang, W.: Recurrent attention network on memory for aspect sentiment analysis. In: EMNLP (2017)
5. Cheng, H., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H.: Wide & deep learning for recommender systems. In: DLRS@RecSys (2016)
6. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)
7. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation (1998)
8. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJCAI (2007)
9. Hingmire, S., Chougule, S., Palshikar, G.K., Chakraborti, S.: Document classification by topic labeling. In: SIGIR (2013)
10. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML (2014)
11. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web (2015)
12. Li, C., Xing, J., Sun, A., Ma, Z.: Effective document labeling with very few seed words: A topic model approach. In: CIKM (2016)
13. Li, M., Clinton, G., Miao, Y., Gao, F.: Short text classification via knowledge powered attention with similarity matrix based CNN. CoRR (2020)
14. Linmei, H., Yang, T., Shi, C., Ji, H., Li, X.: Heterogeneous graph attention networks for semi-supervised short text classification. In: EMNLP-IJCNLP (2019)
15. Ma, Y., Peng, H., Cambria, E.: Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM. In: AAAI (2018)
16. Meng, Y., Shen, J., Zhang, C., Han, J.: Weakly-supervised neural text classification. In: CIKM (2018)
17. Meng, Y., Shen, J., Zhang, C., Han, J.: Weakly-supervised hierarchical text classification. In: AAAI (2019)
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013)
19. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.M.: Text classification from labeled and unlabeled documents using EM. Machine Learning (2000)
20. Phan, X.H., Nguyen, L.M., Horiguchi, S.: Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In: WWW (2008)
21. Rabinovich, E., Sznajder, B., Spector, A., Shnayderman, I., Aharonov, R., Konopnicki, D., Slonim, N.: Learning concept abstractness using weak supervision. In: EMNLP (2018)
22. Ratner, A., Bach, S.H., Ehrenberg, H.R., Fries, J.A., Wu, S., Ré, C.: Snorkel: Rapid training data creation with weak supervision. PVLDB (2017)

23. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: EMNLP-IJCNLP (2019)
24. Song, G., Ye, Y., Du, X., Huang, X., Bie, S.: Short text classification: A survey. Journal of multimedia (2014)
25. Song, Y., Roth, D.: On dataless hierarchical text classification. In: AAAI (2014)
26. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: WWW (2015)
27. Türker, R., Zhang, L., Koutraki, M., Sack, H.: TECNE: knowledge based text classification using network embeddings. In: EKAW (2018)
28. Türker, R., Zhang, L., Koutraki, M., Sack, H.: "the less is more" for text classification,. In: SEMANTiCS (2018)
29. Türker, R., Zhang, L., Koutraki, M., Sack, H.: Knowledge-based short text categorization using entity and category embedding. In: ESWC (2019)
30. Wang, J., Wang, Z., Zhang, D., Yan, J.: Combining knowledge with deep convolutional neural networks for short text classification. In: IJCAI (2017)
31. Wang, P., Xu, B., Xu, J., Tian, G., Liu, C.L., Hao, H.: Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. Neurocomputing (2016)
32. Xuan, J., Jiang, H., Ren, Z., Yan, J., Luo, Z.: Automatic bug triage using semi-supervised text classification. In: SEKE (2010)
33. Zeng, J., Li, J., Song, Y., Gao, C., Lyu, M.R., King, I.: Topic memory networks for short text classification. In: EMNLP (2018)
34. Zhang, X., LeCun, Y.: Text understanding from scratch. CoRR (2015)
35. Zhang, Y., Wallace, B.C.: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. CoRR (2015)