Entity-based Short Text Classification using Convolutional Neural Networks

Mehwish Alam[1,2], Qingyuan Bie[2], Rima Türker[1,2], and Harald Sack[1,2]

[1] FIZ Karlsruhe, Leibniz Institute for Information Infrastructure, Germany
[2] Karlsruhe Institute of Technology (KIT), Germany
{firstname.lastname}@fiz-karlsruhe.de

Abstract. It is beyond human capabilities to analyze a huge amount of short text produced on the World Wide Web in the form of search queries, social media platforms, etc. Due to many difficulties underlying short text for automated processing, i.e, sparsity and insufficient context, the traditional text classification approaches cannot easily be applied to short text. This study discusses a Convolutional Neural Network (CNN) based approach for short text classification. Given a short text, the model generates the text representation by leveraging words together with the entities. To validate the effectiveness of the model, several experiments have been conducted on different datasets. The results suggest that the proposed model is capable of performing short text classification with a high accuracy and outperforms the baseline.

Keywords: Short text classification, Convolutional Neural Networks, Text Classification

1 Introduction

Recently, with the advent of the World Wide Web (WWW) and the digitization of all areas an explosive growth of globally available textual data has been observed. Additionally, short text has become one of the fundamental ways to express and share opinions over different online platforms. A huge amount of such a content is generated each second such as tweets, search queries, snippets, blogs, news feeds, product reviews, etc. Performing analytics over these data is beyond human capability. Consequently, there is a special need to automatically process and analyze these data for many downstream Natural Language Processing (NLP) tasks such as text classification, text summarization and recommendation [19], fake news detection [10,12], etc.

To-date many text classification algorithms have been proposed [5]. However, different kinds of challenges are encountered while handling short text, i.e., **(i)** short texts such as search queries do not follow proper syntax rules of a written natural language. Slang language is very frequent in tweets along with many typing errors. In such cases, the algorithms based on syntactic parsing fail. **(ii)** Short text has limited context, e.g., search queries, limited characters in a tweet, etc. which makes the text more ambiguous.

Search engines are usually able to handle the search queries, however, this is mainly due to the fact that a user clicks through the relevant links which helps in resolving the ambiguity that the selected links might be the preferred meaning of what the user is searching for. Now, the question arises, how does the human mind understand the information contained in this short text? The human mind makes use of external information in order to make sense of this limited and ambiguous content. One of the ways to make this short text machine understandable is conceptualization [13], where the short text is mapped to the concepts defined in a taxonomy or a knowledge base (external). Such approaches can be classified under explicit representation models. On the other hand, many methods use implicit representation models, i.e., deep learning techniques to generate latent semantic representations of short-text [15]. Following these lines, this study combines implicit as well as explicit representation models by making use of the information provided by entities in the external knowledge base as well as the latent representations of the words and the entities. The study proposes a Convolutional Neural Network (CNN) based model due to its outstanding empirical performance on short text classification. In order to enrich the text representation, the model utilizes both words and entities together from the content of the documents for their representation. After entity linking is performed over the short text, the proposed model exploits several language models such as Word2Vec, Doc2Vec, BERT, Wikipedia2Vec, etc. An experimental evaluation was performed over several datasets for short text classification belonging to different genres such as twitter, news articles, etc. The proposed model outperforms the baseline with high accuracy.

This paper is organized as follows: Section 2 gives an insight into the existing methods for short text classification while Section 3 details the proposed approach. In order to prove the feasibility of our approach, several experiments were conducted which are given in Section 4. Finally, Section 5 concludes this study and discusses the future work.

2 Related Work

Text classification is one of the fundamental tasks in NLP. A huge amount of literature on text classification has been summarized in [5]. This section further dives into the existing approaches for short text classification. In [14], the authors combine the implicit and explicit representations, i.e., they introduce a method which in the first step conceptualizes, i.e., annotates the short text with concepts from a knowledge base (explicit representation) and then uses implicit representation, i.e, their corresponding embeddings along with the character level features for short text classification. In [2], the authors use Convolutional Neural Networks along with attention mechanism for sentiment-aware short text classification, however, the approach proposed in this study is applied for short text classification in general without targeting a specific task. The authors in [19] exploit topic memory networks for short text classification. [17] propose a model, which consists of two modules. The first module extracts the concepts and context features then applies an attention mechanism to find the context relevant features. Then the second module leverages a convolutional neural network with the high-level features along with the context relevant features. To alleviate the data sparsity problem [1] leverages knowledge from an external source. In other words, the model utilizes conceptual information from a KB to enhance the text representation. Moreover, the approach utilizes an attention mechanism to measure the importance of the information for classifying the input text.

3 Short Text Classification using Words and Entities

This section dives into the proposed framework for short text classification by combining both the explicit and implicit representation model.

3.1 Entity Linking and Vector Representations

Entity Linking and Entity Embeddings. Given a sentence, TagMe[3] [3] is used for entity identification. It is a tool which augments the plain text with the hyperlinks from Wikipedia, i.e., connecting it to Wikipedia entities. TagMe was chosen because it proves to have good performance over short text. For vector representations of entities Wikipedia2Vec [18] was used. It is a python based tool which jointly learns word and entity embeddings where similar words and entities are close to one another in the vector space.

Word and Contextual Embeddings. The vector representations of words is obtained with the help of two embedding methods, i.e., (i) Word2Vec [8], (ii) Doc2Vec [6]. In order to incorporate contextual information in the embedding spaces the contextual embedding approach, BERT [11] was used.

3.2 Entity Convolutional Neural Networks

The overall workflow of the proposed approach, i.e., Entity Convolutional Neural Networks (EntCNN) is shown in Figure 1. The input layer takes a word and an entity matrix. The second layer uses 3 sets of convolution kernels with width of 2, 3 and 4, each size has 2 kernels. Each of the convolution kernels slides over the whole sentence to generate a feature map.

Input (Embedding) Layer. As a first step, an embedding matrix is created from the preprocessed sentence. This word embedding matrix maps vocabulary word indices into low-dimensional vector representations. The vectors for out of vocabulary words were randomly initialized. Similarly, an entity embedding matrix was created by using Wikipedia2Vec. This entity embedding matrix should have the same length as the word embedding matrix hence the matrix is zero padded. Zero-padding strategy was adopted for the entities which don't have entity embeddings in the pre-trained model.

[3] https://tagme.d4science.org/tagme/

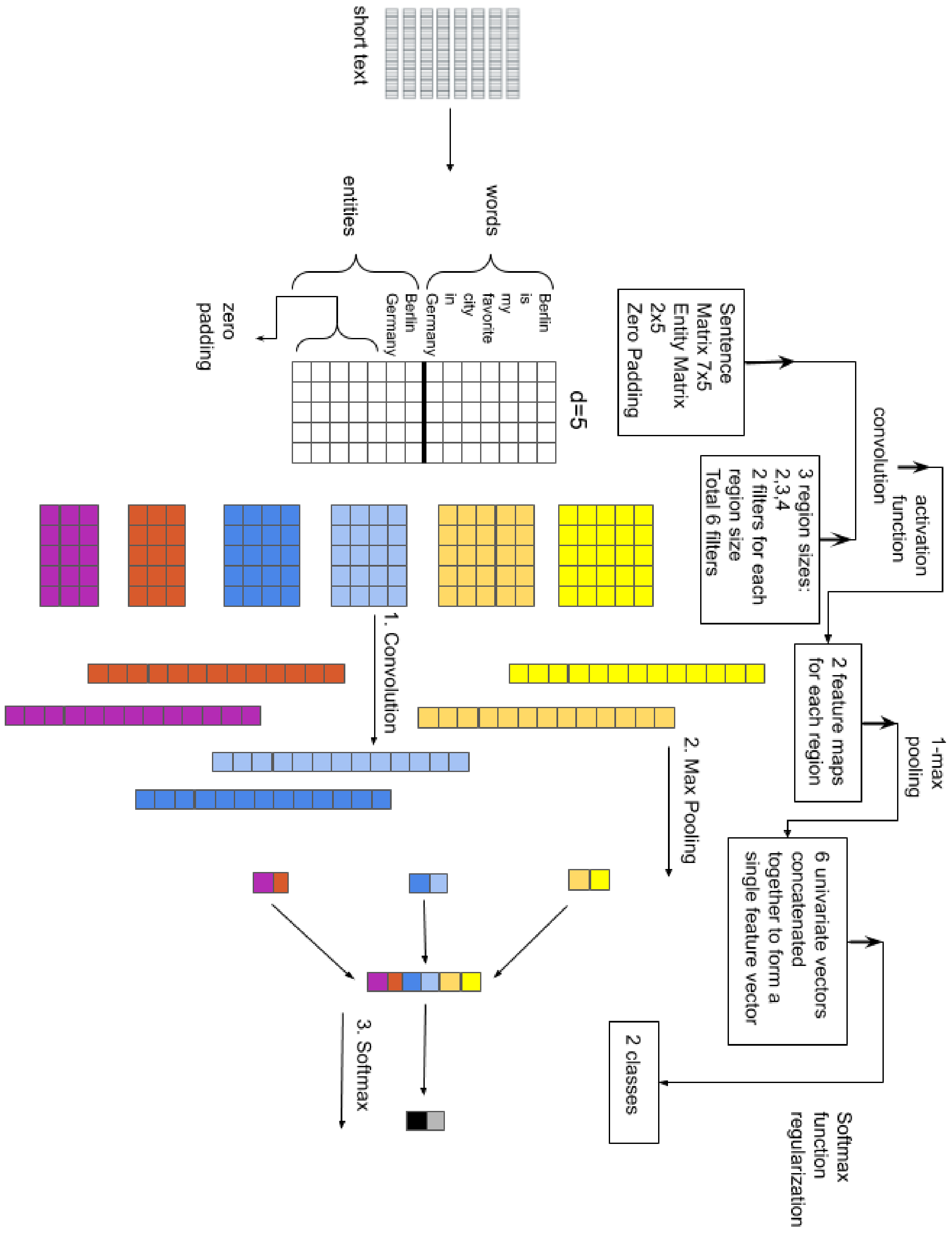


Fig. 1: Overall Architecture of EntCNN

Assume the maximum sentence length of a given document is l. The dimensionality of word and entity vectors are denoted as d. The embedding matrix W by concatenating the word and entity embedding matrix is given as: $W = W_w \oplus W_e$, where W_w and W_e denote the embedding matrix of words and entities respectively. $\oplus$ stands for concatenation operator. Therefore, given a sentence, its embedding matrix can be represented as:

$$W = \boldsymbol{v}_1^w \oplus \boldsymbol{v}_2^w \oplus \cdots \oplus \boldsymbol{v}_l^w \oplus \boldsymbol{v}_1^e \oplus \boldsymbol{v}_2^e \oplus \cdots \oplus \boldsymbol{v}_l^e$$

Convolution and Max-Pooling Layer. Unlike convolution operations in computer vision where filter width can be one or any other integer, filter width should be equal to the dimensionality of word and entity vectors (i.e., d) in text classification. For example, for word "example", its hard to recognize if only *xa* or *ampl* are seen. Thus, the height f of the filter will be varied. Given a filter (convolution kernel) which has size $f \times d$, suppose the words and entities are concatenated $\boldsymbol{v}_i, \boldsymbol{v}_{i+1}, \ldots, \boldsymbol{v}_{i+j}$, denoted as $[\boldsymbol{v}_i : \boldsymbol{v}_{i+f-1}]$. A feature map is then generated from this window of words and entities by:

$$c_i = g(\boldsymbol{w} \cdot [\boldsymbol{v}_i : \boldsymbol{v}_{i+f-1}] + b)$$

where g is a non-linear activation function and b is a bias term. Various non-linear activation functions are applied to introduce non-linearity into deep neural networks. Commonly used functions include sigmoid, hyperbolic tangent (tanh) and rectified linear unit (ReLU). EntCNN uses ReLU because it has characteristics such as simplicity, non-saturating, non-linearity which helps to avoid the vanishing gradient problem, and it has been observed to be able to accelerate the convergence of stochastic gradient descent (SGD). For every possible window of words and entities in a sentence $\{\boldsymbol{x}_{1:f}, \boldsymbol{x}_{2:f+1}, \ldots, \boldsymbol{x}_{n:f+1:n}\}$ EntCNN uses this filter to produce feature maps:

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

A max-over-time pooling function is then applied to this feature map to create a fixed length vector. Max-pooling is a strategy where the maximum value is taken, reducing its dimensionality while capturing the most important features. Note that EntCNN uses filters of different region sizes (heights). Multiple filters were used to learn different features for the same region size (height). Because each filter produces vectors of different shapes, there is a need to iterate over them, create a layer for each of them, adopt the same pooling scheme, and then merge the results into one feature vector.

Dropout Layer. This feature vector is then passed to a dropout layer. A dropout layer stochastically disables a fraction of its neurons during training by setting them to zero, which prevents neurons from co-adapting and forces them to learn individually useful features, thus significantly reduces over-fitting. The output of the dropout layer is denoted as h.

Softmax Layer. The feature vector from dropout layer is finally passed to a fully connected layer. Label predictions are generated by performing matrix multiplication and picking the class with the highest score. Softmax function was applied to convert raw scores into a probability distribution over labels.

$$p = softmax(w \cdot h + b)$$
$$\hat{p} = argmax(p)$$

Training. The goal of training is to minimize the loss function. The cross-entropy error is the standard loss function for categorization problems. Compared with cross-entropy error, the mean squared error (MSE) and classification error are not suitable loss functions for classification. For binary classification, cross-entropy error can be calculated as:

$$L = -(ylog(p) + (1 - y)log(1 - p))$$

In the case of multi-class classification, a separate loss for each class per observation is calculated and the sum of the result is taken. L2-regularization is used to prevent over-fitting.

4 Experimental Results

This section discusses details about the datasets used for evaluation along with their statistics. In order to show that the applicability of the proposed framework, an empirical analysis over the results was performed.

4.1 Datasets.

The experiments are conducted on four widely used datasets for different text classification task, i.e., (i) Twitter, (ii) Movie Review, (iii) TREC, and (iv) AG-News. Table 1 shows the statistics of the datasets used for evaluation.

- **Twitter.** The current study specifically uses the *Sentiment Analysis in Twitter* (task [9]) from SemEval 2013 and more specifically the dataset related to polarity classification (subtask B)[4]. The noisy nature of the natural language in tweets makes it a good candidate for the evaluation.
- **Movie Review** is a widely used dataset for evaluating the algorithms for polarity detection[5]. The dataset does not have a standard test set, therefore, 10% of the training data have been selected as the development set.
- **TREC** was first introduced for evaluating question classification method [7]. The dataset is freely available online[6]. It includes the following six different question labels: *Abbreviation*, *Entity*, *Description*, *Human*, *Location*, and *Numeric count*.

[4] https://www.cs.york.ac.uk/semeval-2013/task2/
[5] http://www.cs.cornell.edu/people/pabo/movie-review-data/
[6] http://cogcomp.cs.illinois.edu/Data/QA/QC/

Datasets	#classes	#labels	$\|S\|$	#train set	#test set	$\|V\|$	$\|V_{pre}\|$	Avg. Len
TREC	6	10	33	5452	500	8602	7223	10
Twitter	3	19	35	9684	3547	20755	12140	19
AG News	4	7	23	120000	7600	38916	25310	7
Movie Review	2	21	59	10662	CV	19897	16394	20

Table 1: $|S|$ represents the max. sentence length, $|V|$ represents the vocabulary size, $|V_{pre}|$ represents the number of words present in pre-trained model. CV stands for cross-validation, i.e., there is no standard train/test split, thus 10-fold cross validation is used.

– **AG News** was first introduced in [20]. The original dataset consists of news title and their description[7]. From this dataset, four largest categories were chosen, i.e., world, sports, business, sci/tech. The current experiment only uses news titles.

During the preprocessing step the sentences were converted to lower case and then tokenization, stop-words removal, stemming and lemmatization were performed. In case of Twitter dataset all the URLs as well as user mentions were removed and retweets were retained by removing *"RT:"*. The hashtags were replaced with the corresponding words in the hashtag.

4.2 Experimental Setup

All the experiements were conducted on a workstation having 3.60GHz Intel Xeon 4 Cores CPUs, 1 single NVIDIA GeForce GTX 1060 GPU with CUDA 10.0.130, cuDNN 7.3.0 and TensorFlow 1.3. The pre-trained word2vec model trained on Google News was used. The model contains 300 dimensional vectors for 3 million words and phrases. For the out of vocabulary words the vectors were randomly initialized from a uniform distribution [-0.25,0.25]. The pre-trained Wikipedia2Vec[8] model trained on English Wikipedia dump was used. Doc2Vec model was trained on the latest English Wikipedia dump with 300 dimensions and 5 epochs. The model exploits different feature combinations as shown in Table 2. In the first set of experiments, only word vectors (Word(Word2Vec), Word(Doc2Vec), Word(BERT)) from the respective embedding model and entity vectors (Entity(Wikipedia2vec)) were utilized separately for the classification task. In the second set of experiments, combination of word and entity vectors (Word(Word2Vec) + Entity(Wikipedia2Vec), Word(Wikipedia2Vec) + Entity(Wikipedia2Vec, etc.) were leveraged as a feature set.

The following hyper-parameters were used for the experiments (here we state the optimal parameters):

– filter sizes; lower:[3, 4, 5], upper:[3, 4, 5, 6],

[7] http://groups.di.unipi.it/ gulli/AG_corpus_of_news_articles.html
[8] https://wikipedia2vec.github.io/wikipedia2vec/pretrained/

– number of filters; lower:16, upper:256,
– dropout rate; 0.5,
– batch size; 64,
– l2 regularization; 3.0,
– embedding dimension; lower:100, upper:300,
– learning rate; 0.005.

4.3 Evaluation Results

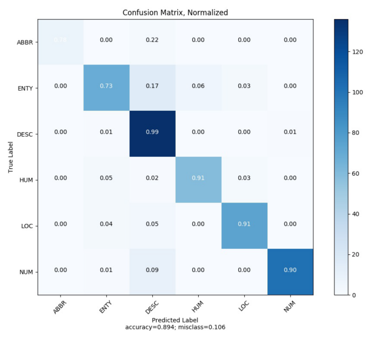
The baseline used for the experimentation was Knowledge Powered Convolutional Neural Network (KPCNN) [14]. It uses external knowledge from Probase [16] along with the character, word and concept embeddings. It finally uses a CNN based model for short-text classification. KPCNN was chosen because the method is very close to our proposed approach. The method is also compared against TextCNN [4] which uses only word vectors with CNN.

Methods	**Twitter**	**TREC**	**MR**	**AG News**
Baseline: KPCNN	57.24	89.3	81.5	86.1
Word(Word2Vec) (TextCNN)	68.1	89.4	**83.0**	87.4
Word(Doc2Vec)	**69.3**	91.6	81.5	87.6
Word(BERT)	**69.3**	90.2	79.3	87.4
Entity(Wikipedia2vec)	46.4	61.4	51.8	79.3
Word(Word2Vec) + Entity(Wikipedia2Vec)	68.2	90.4	**83.0**	87.9
Word(Wikipedia2Vec) + Entity(Wikipedia2Vec)	68.8	89.8	82.8	87.9
Word(Word2Vec) + Entity(Doc2Vec)	**69.3**	89.0	82.0	**88.1**
Word(Doc2Vec) + Entity(Doc2Vec)	**69.3**	**93.0**	81.3	87.4

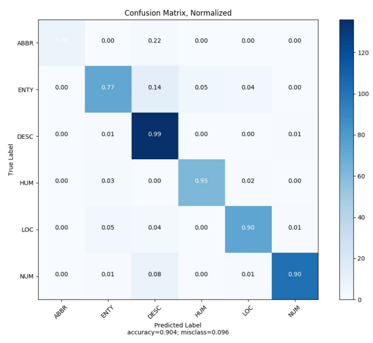
Table 2: Classification accuracy of proposed model against the baseline

The results of the experiments are illustrated in Table 2. The results show that the entities play an important role in classifying short text. It can be seen that the best classification accuracy, i.e., TREC 93.0%, Movie Review 83.0% and AG News 88.1% has been achieved by including words as well as entities in the classification process. In one of the experiments with the Movie Review dataset, where only words have been utilized, the achieved accuracy is 83.0% which is equivalent to the results based on entities leading to the fact that entities have not played a role in the classification performance. The reason here can be attributed to the characteristics of the dataset. The proportion of entities in the Movie Review dataset (83.5%) is less in comparison to other two datasets (92%). Additionally, it can be seen that for all the datasets our method outperforms the baseline. Interestingly, for each dataset a different embedding model helps to obtain the best performance but all of them use entities. Overall, it can be concluded that no single word embedding or entity embedding works best for all datasets and combination of word and entity features can help to improve the overall performance of short text classification.

The accuracy on Twitter 68.1% is much lower than three other datasets. It can be attributed to the fact that tweets contain more complicated language such as slang language and short-hand language as well as noisy or faulty sentences. When compared with TREC and AG News, it is obviously more difficult to analyze and label the sentences. In the original SemEval task [9], the best result of Twitter Task 2 subtask B is 69% which is closer to our results.

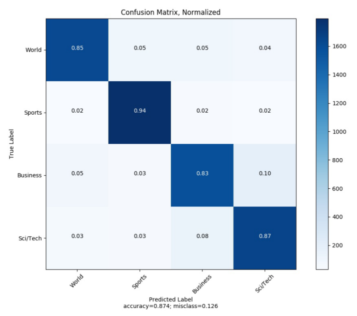


TextCNN

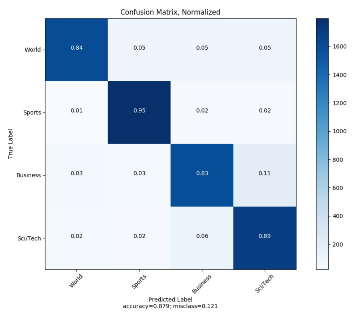


EntCNN

Fig. 3: Normalized confusion matrix on TREC



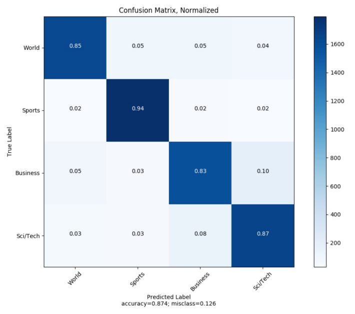
TextCNN



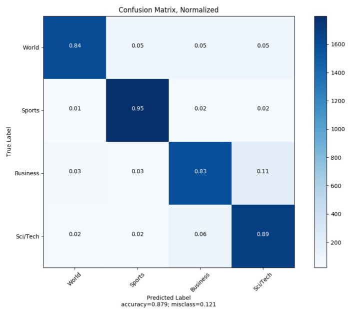
EntCNN

Fig. 5: Normalized confusion matrix on AGNews

From the confusion matrices shown in Figure 7(a) and Figure 7(b), it can be seen that our model EntCNN (i.e., text classification using words and entities) improves the classification accuracy on label `HUM` from 91% to 95%, `ENTY` from 73% to 77% respectively, and has overall higher accuracy than TextCNN. Another observation about the classification results is that labels `ABBR` and `ENTY` have much lower accuracy than other labels. One possible explanation is that label `ABBR` and `ENTY` lack training data. The confusion matrices presented for other datasets can also be interpreted in the same way.



TextCNN



EntCNN

Fig. 7: Normalized confusion matrix on Twitter

Finally, in order to take advantage of the sequential information contained in a sentence, methods such as Bi-LSTM and Recurrent Convolutional Neural Networks (RCNN) were used. The results are shown in Table 3. It can be observed that TextCNN outperforms RCNN and BiLSTM. In Table 2, it can be seen that the entity based methods outperform TextCNN.

Model	Twitter	TREC	Movie Review	AG News
TextCNN	68.1	89.4	83.0	87.4
RCNN	67.6	90.4	82.3	88.3
Bi-LSTM	63.6	88.4	83.7	88.2

Table 3: Comparison to RCNN and Bi-LSTM

5 Conclusion and Future Work

In this study, a CNN based model has been utilized to perform short text classification. In contrast to traditional classification models, our approach utilizes both words and entities together from the documents to represent documents as well as for dealing with the problem of ambiguity accompanied by short text. The experimental results illustrate that entities play an important role for short text classification, especially when the available context is rather limited. As future perspective, short text understanding would be exploited by targeting the similar problems posed by short text (as discussed in the current study) for generating a machine readable representation from such text.

References

1. Jindong Chen, Yizhou Hu, Jingping Liu, Yanghua Xiao, and Haiyun Jiang. Deep short text classification with knowledge powered attention. In *AAAI*, 2019.

2. Zeyu Chen, Yan Tang, Zuowei Zhang, Chengyang Zhang, and Luwei Wang. Sentiment-aware short text classification based on convolutional neural network and attention. In *In: IEEE - ICTAI*, 2019.
3. Paolo Ferragina and Ugo Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*, 2010.
4. Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.
5. Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.
6. Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, 2014.
7. Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Nat. Lang. Eng.*, 12(3):229–249, 2006.
8. Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
9. Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Theresa Wilson. Semeval-2013 task 2: Sentiment analysis in twitter. *CoRR*, abs/1912.06806, 2019.
10. Ray Oshikawa, Jing Qian, and William Yang Wang. A survey on natural language processing for fake news detection. *CoRR*, abs/1811.00770, 2018.
11. Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *In: NAACL-HLT*, 2018.
12. Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *In: EMNLP*, 2017.
13. Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. Short text conceptualization using a probabilistic knowledgebase. In *In: IJCAI*. IJCAI/AAAI, 2011.
14. Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In Carles Sierra, editor, *In: IJCAI*, 2017.
15. Zhongyuan Wang and Haixun Wang. Understanding short texts. In *the Association for Computational Linguistics (ACL) (Tutorial)*, August 2016.
16. Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. Probase: a probabilistic taxonomy for text understanding. In *In: ACM SIGMOD*, 2012.
17. Jingyun Xu and Yi Cai. Incorporating context-relevant knowledge into convolutional neural networks for short text classification. In *AAAI*, 2019.
18. Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280v3*, 2020.
19. Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu, and Irwin King. Topic memory networks for short text classification. In *In: EMNLP*, 2018.
20. Xiang Zhang and Yann LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015.