DOC: Deep Open Classification of Text Documents

Lei Shu, Hu Xu, Bing Liu
Department of Computer Science
University of Illinois at Chicago
{lshu3, hxu48, liub}@uic.edu

Abstract

Traditional supervised learning makes the *closed-world* assumption that the classes appeared in the test data must have appeared in training. This also applies to text learning or text classification. As learning is used increasingly in dynamic open environments where some new/test documents may not belong to any of the training classes, identifying these novel documents during classification presents an important problem. This problem is called *open-world classification* or *open classification*. This paper proposes a novel deep learning based approach. It outperforms existing state-of-the-art techniques dramatically.

1 Introduction

A key assumption made by classic supervised text classification (or learning) is that classes appeared in the test data must have appeared in training, called the *closed-world* assumption (Fei and Liu, 2016; Chen and Liu, 2016). Although this assumption holds in many applications, it is violated in many others, especially in dynamic or open environments. For example, in social media, a classifier built with past topics or classes may not be effective in classifying future data because new topics appear constantly in social media (Fei et al., 2016). This is clearly true in other domains too, e.g., self-driving cars, where new objects may appear in the scene all the time.

Ideally, in the text domain, the classifier should classify incoming documents to the right existing classes used in training and also detect those documents that don't belong to any of the existing classes. This problem is called *open world classification* or *open classification* (Fei and Liu, 2016). Such a classifier is aware *what it does and does not know*. This paper proposes a novel technique to solve this problem.

Problem Definition: Given the training data $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ is the i-th document, and $y_i \in \{l_1, l_2, \ldots, l_m\} = \mathcal{Y}$ is $\mathbf{x}_i$'s class label, we want to build a model $f(\mathbf{x})$ that can classify each test instance $\mathbf{x}$ to one of the m training or *seen* classes in $\mathcal{Y}$ or reject it to indicate that it does not belong to any of the m training or seen classes, i.e., *unseen*. In other words, we want to build a $(m+1)$-class classifier $f(\mathbf{x})$ with the classes $\mathcal{C} = \{l_1, l_2, \ldots, l_m, \textit{rejection}\}$.

There are some prior approaches for open classification. One-class SVM (Schölkopf et al., 2001; Tax and Duin, 2004) is the earliest approach. However, as no negative training data is used, one-class classifiers work poorly. Fei and Liu (2016) proposed a Center-Based Similarity (CBS) space learning method (Fei and Liu, 2015). This method first computes a center for each class and transforms each document to a vector of similarities to the center. A binary classifier is then built using the transformed data for each class. The decision surface is like a "ball" encircling each class. Everything outside the ball is considered not belonging to the class. Our proposed method outperforms this method greatly. Fei et al. (2016) further added the capability of incrementally or cumulatively learning new classes, which connects this work to *lifelong learning* (Chen and Liu, 2016) because without the ability to identify novel or new things and learn them, a system will never be able to learn by itself continually.

In computer vision, Scheirer et al. (2013) studied the problem of recognizing unseen images that the system was not trained for by reducing *open space risk*. The basic idea is that a classifier should not cover too much open space where there are few or no training data. They proposed to reduce the half-space of a binary SVM classifier

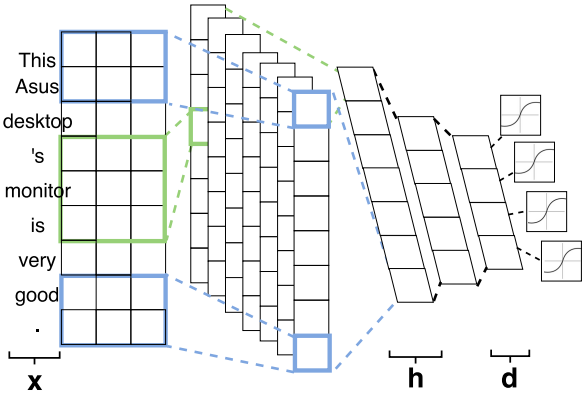

Figure 1: Overall Network of DOC

with a positive region bounded by two parallel hyperplanes. Similar works were also done in a probability setting by Scheirer et al. (2014) and Jain et al. (2014). Both approaches use probability threshold, but choosing thresholds need prior knowledge, which is a weakness of the methods. Dalvi et al. (2013) proposed a multi-class semi-supervised method based on the EM algorithm. It has been shown that these methods are poorer than the method in (Fei and Liu, 2016).

The work closest to ours is that in (Bendale and Boult, 2016), which leverages an algorithm called *OpenMax* to add the rejection capability by utilizing the logits that are trained via closed-world softmax function. One weak assumption of OpenMax is that examples with equally likely logits are more likely from the unseen or rejection class, which can be examples that are hard to classify. Another weakness is that it requires validation examples from the unseen/rejection class to tune the hyperparameters. Our method doesn't make these weak assumptions and performs markedly better.

Our proposed method, called DOC (*Deep Open Classification*), uses deep learning (Goodfellow et al., 2016; Kim, 2014). Unlike traditional classifiers, DOC builds a multi-class classifier with a 1-vs-rest final layer of sigmoids rather than softmax to reduce the open space risk. It reduces the open space risk further for rejection by tightening the decision boundaries of sigmoid functions with Gaussian fitting. Experimental results show that DOC dramatically outperforms state-of-the-art existing approaches from both text classification and image classification domains.

2 The Proposed DOC Architecture

DOC uses CNN (Collobert et al., 2011; Kim, 2014) as its base and augments it with a 1-vs-rest final sigmoid layer and Gaussian fitting for classification. Note: other existing deep models like RNN (Williams and Zipser, 1989; Schuster and Paliwal, 1997) and LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 2002) can also be adopted as the base. Similar to RNN, CNN also works on embedded sequential data (using 1D convolution on text instead of 2D convolution on images). We choose CNN because OpenMax uses CNN and CNN performs well on text (Kim, 2014), which enables a fairer comparison with OpenMax.

2.1 CNN and Feed Forward Layers of DOC

The proposed DOC system (given in Fig. 1) is a variant of the CNN architecture (Collobert et al., 2011) for text classification (Kim, 2014)[1]. The first layer embeds words in document $\mathbf{x}$ into dense vectors. The second layer performs convolution over dense vectors using different filters of varied sizes (see Sec. 3.4). Next, the max-over-time pooling layer selects the maximum values from the results of the convolution layer to form a k-dimension feature vector $\boldsymbol{h}$. Then we reduce $\boldsymbol{h}$ to a m-dimension vector $\boldsymbol{d} = d_{1:m}$ (m is the number of training/seen classes) via 2 fully connected layers and one intermediate ReLU activation layer:

$$\boldsymbol{d} = \boldsymbol{W}'(\text{ReLU}(\boldsymbol{W}\boldsymbol{h} + \boldsymbol{b})) + \boldsymbol{b}', \qquad (1)$$

where $\boldsymbol{W} \in \mathbb{R}^{r \times k}$, $\boldsymbol{b} \in \mathbb{R}^r$, $\boldsymbol{W}' \in \mathbb{R}^{m \times r}$, and $\boldsymbol{b}' \in \mathbb{R}^m$ are trainable weights; r is the output dimension of the first fully connected layer. The output layer of DOC is a 1-vs-rest layer applied to $d_{1:m}$, which allows rejection. We describe it next.

2.2 1-vs-Rest Layer of DOC

Traditional multi-class classifiers (Goodfellow et al., 2016; Bendale and Boult, 2016) typically use softmax as the final output layer, which does not have the rejection capability since the probability of prediction for each class is normalized across all training/seen classes. Instead, we build a 1-vs-rest layer containing m sigmoid functions for m seen classes. For the i-th sigmoid function corresponding to class l_i, DOC takes all examples with $y = l_i$ as positive examples and all the rest examples $y \neq l_i$ as negative examples.

The model is trained with the objective of summation of all log loss of the m sigmoid functions

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

on the training data D.

$$\text{Loss} = \sum_{i=1}^{m}\sum_{j=1}^{n} -\mathbb{I}(y_j = l_i)\log p(y_j = l_i) \\ -\mathbb{I}(y_j \neq l_i)\log(1 - p(y_j = l_i)), \quad (2)$$

where $\mathbb{I}$ is the indicator function and $p(y_j = l_i) = \text{Sigmoid}(d_{j,i})$ is the probability output from ith sigmoid function on the jth document's ith-dimension of $\boldsymbol{d}$.

During testing, we reinterpret the prediction of m sigmoid functions to allow rejection, as shown in Eq. 3. For the i-th sigmoid function, we check if the predicted probability $\text{Sigmoid}(d_i)$ is less than a threshold t_i belonging to class l_i. If all predicted probabilities are less than their corresponding thresholds for an example, the example is rejected; otherwise, its predicted class is the one with the highest probability. Formally, we have

$$\hat{y} = \begin{cases} \textit{reject}, \text{ if } \text{Sigmoid}(d_i) < t_i, \forall l_i \in \mathcal{Y}; \\ \arg\max_{l_i \in \mathcal{Y}} \text{Sigmoid}(d_i), \text{ otherwise.} \end{cases} \quad (3)$$

Note that although multi-label classification (Huang et al., 2013; Zhang and Zhou, 2006; Tsoumakas and Katakis, 2006) may also leverage multiple sigmoid functions, Eq. 3 forbids multiple predicted labels for the same example, which is allowed in multi-label classification. DOC is also related to multi-task learning (Huang et al., 2013; Caruana, 1998), where each label l_i is related to a 1-vs-rest binary classification task with shared representations from CNN and fully connected layers. However, Eq. 3 performs classification and rejection based on the outputs of these binary classification tasks.

Comparison with OpenMax: OpenMax builds on the traditional closed-world multi-class classifier (softmax layer). It reduces the open space for each seen class, which is weak for rejecting unseen classes. DOC's 1-vs-rest sigmoid layer provides a reasonable representation of all other classes (the rest of seen classes and unseen classes), and enables the 1 class forms a good boundary. Sec. 3.5 shows that this basic DOC is already much better than OpenMax. Below, we improve DOC further by tightening the decision boundaries more.

2.3 Reducing Open Space Risk Further

Sigmoid function usually uses the default probability threshold of t_i = 0.5 for classification of each class i. But this threshold does not consider potential open space risks from unseen (rejection) class data. We can improve the boundary by increasing t_i. We use Fig. 2 to illustrate. The x-axis represents d_i and y-axis is the predicted probability $p(y = l_i|d_i)$. The sigmoid function tries to push positive examples (belonging to the i-th class) and negative examples (belonging to the other seen classes) away from the y-axis via a high gain around $d_i = 0$, which serves as the default decision boundary for d_i with $t_i = 0.5$. As demonstrated by those 3 circles on the right-hand side of the y-axis, during testing, unseen class examples (circles) can easily fill in the gap between the y-axis and those dense positive (+) examples, which may reduce the recall of rejection and the precision of the i-th seen class prediction. Obviously, a better decision boundary is at $d_i = T$, where the decision boundary more closely "wrap" those dense positive examples with the probability threshold $t_i \gg 0.5$.

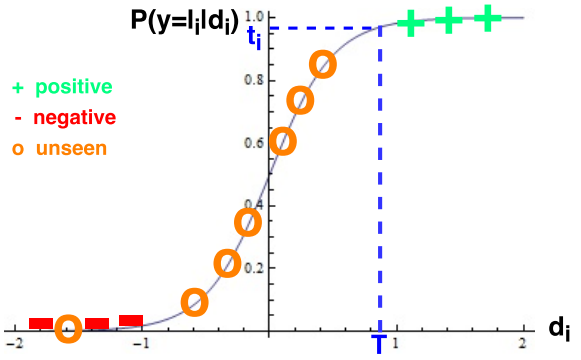

Figure 2: Open space risk of sigmoid function and desired decision boundary $d_i = T$ and probability threshold t_i.

To obtain a better t_i for each seen class i-th, we use the idea of outlier detection in statistics:

1. Assume the predicted probabilities $p(y = l_i|\mathbf{x}_j, y_j = l_i)$ of all training data of each class i follow one half of the Gaussian distribution (with mean $\mu_i = 1$), e.g., the three positive points in Fig. 2 projected to the y-axis (we don't need d_i). We then artificially create the other half of the Gaussian distributed points (≥ 1): for each existing point $p(y = l_i|\mathbf{x}_j, y_j = l_i)$, we create a mirror point $1 + (1 - p(y = l_i|\mathbf{x}_j, y_j = l_i)$ (not a probability) mirrored on the mean of 1.

2. Estimate the standard deviation σ_i using both the existing points and the created points.

3. In statistics, if a value/point is a certain number (α) of standard deviations away from the mean, it is considered an outlier. We thus set the probability threshold $t_i = \max(0.5, 1 - \alpha\sigma_i)$. The commonly used number for α is 3, which also works well in our experiments.

Note that due to Gaussian fitting, different class l_i can have a different classification threshold t_i.

3 Experimental Evaluation

3.1 Datasets

We perform evaluation using two publicly available datasets, which are exactly the same datasets used in (Fei and Liu, 2016).

(1) **20 Newsgroups**[2] (Rennie, 2008): The 20 newsgroups data set contains 20 non-overlapping classes. Each class has about 1000 documents.

(2) **50-class reviews** (Chen and Liu, 2014): The dataset has Amazon reviews of 50 classes of products. Each class has 1000 reviews. Although product reviews are used, we do not do sentiment classification. We still perform topic-based classification. That is, given a review, the system decides what class of product the review is about.

For every dataset, we keep a 20000 frequent word vocabulary. Each document is fixed to 2000-word length (cutting or padding when necessary).

3.2 Test Settings and Evaluation Metrics

For a fair comparison, we use exactly the same settings as in (Fei and Liu, 2016). For each class in each dataset, we randomly sampled 60% of documents for training, 10% for validation and 30% for testing. Fei and Liu (2016) did not use a validation set, but the test data is the same 30%. We use the validation set to avoid overfitting. For open-world evaluation, we hold out some classes (as unseen) in training and mix them back during testing. We vary the number of training classes and use 25%, 50%, 75%, or 100% classes for training and all classes for testing. Here using 100% classes for training is the same as the traditional closed-world classification. Taking 20 newsgroups as an example, for 25% classes, we use 5 classes (we randomly choose 5 classes from 20 classes for 10 times and average the results, as in (Fei and Liu, 2016)) for training and all 20 classes for testing (15 classes are unseen in training). We use macro F_1-score over $5 + 1$ classes (1 for rejection) for evaluation. Please note that examples from unseen classes are dropped in the validation set.

[2] http://qwone.com/~jason/20Newsgroups/

Table 1: Macro-F_1 scores for 20 newsgroups

% of seen classes	**25%**	**50%**	**75%**	**100%**
cbsSVM	59.3	70.1	72.0	85.2
OpenMax	35.7	59.9	76.2	91.9
DOC ($t = 0.5$)	75.9	84.0	87.4	92.6
DOC	82.3	85.2	86.2	92.6

Table 2: Macro-F_1 scores for 50-class reviews

% of seen classes	**25%**	**50%**	**75%**	**100%**
cbsSVM	55.7	61.5	58.6	63.4
OpenMax	41.6	57.0	64.2	69.2
DOC ($t = 0.5$)	51.1	63.6	66.2	69.8
DOC	61.2	64.8	66.6	69.8

3.3 Baselines

We compare DOC with two state-of-the-art methods published in 2016 and one DOC variant.

cbsSVM: This is the latest method published in NLP (Fei and Liu, 2016). It uses SVM to build 1-vs-rest CBS classifiers for multiclass text classification with rejection option. The results of this system are taken from (Fei and Liu, 2016).

OpenMax: This is the latest method from computer vision (Bendale and Boult, 2016). Since it is a CNN-based method for image classification, we adapt it for text classification by using CNN with a softmax output layer, and adopt the OpenMax layer[3] for open text classification. When all classes are seen (100%), the result from softmax is reported since OpenMax layer always performs rejection. We use default hyperparameter values of OpenMax (Weibull tail size is set to 20).

DOC($t = 0.5$): This is the basic DOC ($t = 0.5$). Gaussian fitting isn't used to choose each t_i.

Note that (Fei and Liu, 2016) compared with several other baselines. We don't compare with them as it was shown that cbsSVM was superior.

3.4 Hyperparameter Setting

We use word vectors pre-trained from Google News[4] (3 million words and 300 dimensions). For the CNN layers, 3 filter sizes are used $[3, 4, 5]$. For each filter size, 150 filters are applied. The dimension r of the first fully connected layer is 250.

[3] https://github.com/abhijitbendale/OSDN

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

3.5 Result Analysis

The results of 20 newsgroups and 50-class reviews are given in Tables 1 and 2, respectively. From the tables, we can make the following observations:

1. DOC is markedly better than OpenMax and cbsSVM in macro-F_1 scores for both datasets in the 25%, 50%, and 75% settings. For the 25% and 50% settings (most test examples are from unseen classes), DOC is dramatically better. Even for 100% of traditional closed-world classification, it is consistently better too. DOC($t = 0.5$) is better too.

2. For the 25% and 50% settings, DOC is also markedly better than DOC($t = 0.5$), which shows that Gaussian fitting finds a better probability threshold than $t = 0.5$ when many unseen classes are present. In the 75% setting (most test examples are from seen classes), DOC($t = 0.5$) is slightly better for 20 newsgroups but worse for 50-class reviews. DOC sacrifices some recall of seen class examples for better precision, while $t = 0.5$ sacrifices the precision of seen classes for better recall. DOC($t = 0.5$) is also worse than cbsSVM for 25% setting for 50-class reviews. It is thus not as robust as DOC.

3. For the 25% and 50% settings, cbsSVM is also markedly better than OpenMax.

4 Conclusion

This paper proposed a novel deep learning based method, called DOC, for open text classification. Using the same text datasets and experiment settings, we showed that DOC performs dramatically better than the state-of-the-art methods from both the text and image classification domains. We also believe that DOC is applicable to images.

In our future work, we plan to improve the cumulative or incremental learning method in (Fei et al., 2016) to learn new classes without training on all past and new classes of data from scratch. This will enable the system to learn by self to achieve continual or lifelong learning (Chen and Liu, 2016). We also plan to improve model performance during testing (Shu et al., 2017).

Acknowledgments

This work was supported in part by grants from National Science Foundation (NSF) under grant no. IIS-1407927 and IIS-1650900.

References

Abhijit Bendale and Terrance E Boult. 2016. Towards open set deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 1563–1572.

Rich Caruana. 1998. Multitask learning. In *Learning to learn*, Springer, pages 95–133.

Zhiyuan Chen and Bing Liu. 2014. Mining topics in documents: standing on the shoulders of big data. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 1116–1125.

Zhiyuan Chen and Bing Liu. 2016. *Lifelong Machine Learning*. Morgan & Claypool Publishers.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Bhavana Dalvi, William W Cohen, and Jamie Callan. 2013. Exploratory learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pages 128–143.

Geli Fei and Bing Liu. 2015. Social media text classification under negative covariate shift. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2015)*.

Geli Fei and Bing Liu. 2016. Breaking the closed world assumption in text classification. In *Proceedings of NAACL-HLT*. pages 506–514.

Geli Fei, Shuai Wang, and Bing Liu. 2016. Learning cumulatively to become more knowledgeable. In *Proceedings of SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2016)*.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. 2002. Learning precise timing with lstm recurrent networks. *Journal of machine learning research* 3(Aug):115–143.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Yan Huang, Wei Wang, Liang Wang, and Tieniu Tan. 2013. Multi-task deep neural network for multi-label learning. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*. IEEE, pages 2897–2900.

Lalit P Jain, Walter J Scheirer, and Terrance E Boult. 2014. Multi-class open set recognition using probability of inclusion. In *European Conference on Computer Vision*. Springer, pages 393–409.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Jason Rennie. 2008. 20 newsgroup dataset.

Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. 2013. Toward open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(7):1757–1772.

Walter J Scheirer, Lalit P Jain, and Terrance E Boult. 2014. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence* 36(11):2317–2324.

Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13(7):1443–1471.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Lei Shu, Hu Xu, and Bing Liu. 2017. Lifelong learning crf for supervised aspect extraction. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2017, short paper)*.

David MJ Tax and Robert PW Duin. 2004. Support vector data description. *Machine learning* 54(1):45–66.

Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3(3).

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1(2):270–280.

Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering* 18(10):1338–1351.