Transfer Learning for Text Classification

Hui Wei
hw1666@nyu.edu

Xin Chen
xc1113@nyu.edu

December 21, 2018

Abstract

Transfer learning is considered as the next frontier of machine learning field. It is one of the key points for building the general artificial intelligence. Recently, transfer learning has become a hot topic and has excellent performance in computer vision field. However, for natural language processing, it has not achieved such a huge success. In this paper, we try to modify a transfer learning model used for healthcare time series classification problem on NLP, test it for text classification task, and then prove the effectiveness and robustness of this transfer learning method.

1 Introduction and Related Works

It was said by Andrew Ng on NIPS 2016 that transfer learning will be the next driver of machine learning. Nevertheless, most current state-of-the-art machine learning models focus only on solving one specific task. For example, AlphaGo performs better than any human experts, but, unlike human beings, which can transfer knowledge from other tasks, AlphaGo cannot tackle the real-world problems. In some fields, such as healthcare and medicine, due to the issue of privacy, at the same time, expensive and time-consuming labeling methods, data are very scarce, thus it is quite hard to train a machine learning model on those tasks from scratch. Also, with the development of deep learning, the model needs more and more computing resources to gain a high performance, so it seems that except some technique companies, we cannot train a better model, which will push the whole field forward. Fortunately, transfer learning can help for those problems.

Intuitively, transfer learning is to transfer some "knowledge" learned from the source domain and task to tackling tasks in the target domain. Mathematically, Domain $D = \{X, P(X)\}$, where, X is the dataset, and P(X) is the statistical distribution of it. Target $T = \{y, P(y|X)\}$, includes the label y, and the conditional probability $P(y|X)$ stands for the task to predict the label given a data point. To use the transfer learning, the summary paper [1] introduces several conditions the dataset should satisfy: 1) $D_S \neq D_T$ and/or $T_S \neq T_D$, where S and T stands for source and target, respectively. 2)The number of the training samples for source should be much larger than that of the target. 3)the source and the target should be different, but at least they are related; otherwise, it will lead to negative transfer, which, instead of helping deal with the target problem, does harm to tackling it.

For deep neural network based transfer learning models, [2] categorizes them into four classes: 1) instance-based, which used ways such like Adaboost to re-weight the source domain, giving those points which are more similar to the target domain with higher weights and combining them with the target training set to train the model. 2) Mapping-based methods, trying to find a new space, on which the target and source domain are more similar, then train a model on this new space combining the training samples from both domains. 3) Network-based, whose another name is "fine-tuning". Basically, it pre-trains a network using source samples, freezes some layers, then continues to train the remaining layers using the target training samples. 4) Adversarial-based methods, inspired by Generative Adversarial Network (GAN), utilize the same network as the feature extractor for both domains, then feed those features into a discriminator to enforce the features are common.

Text classification is one of the basic problems in natural language process. Some other problems can be essentially categorized as it, such like sentimental analysis, word tagging and spam email detection. As other classification problems, text classification associates a label with each word or the entire sentence or paragraph. In our problem, given the training dataset which is a binary classification problem, we try to transfer to other datasets, one of which is also binary classification problem, while the other is the multi-classification problem. Also, our task is to assign a lable to a whole paragraph, which is much harder than tasks assigning for each word, such like word tagging, because of more noise.

As this paper, [3] also tries to use transfer learning model to solve multi-class text classification problem. They use TF-IDF based vector as the word representation, and try to learn a transfer learning model based on Naive Bayes, softmax learning and non-parametric learning. Compared to our work using GloVe, TF-IDF word vector grabs the statistical information for the corpus, but it fails to incorporate the semantic and syntactic properties of words, thus losing some information. Also, their word is to specifically solve the text classification problem and cannot easily modified to other problems. Unlike that, our work is easy to transfer to other time series classification problems, such like in healthcare.

Natural language is one of the specific forms of times series data. For tackling the problem in this field, people always get inspired by other field. [4] uses GRU and Logistic Regression model which is transferable for healthcare times series classification problem. In the work, they use MIMIC III dataset, pre-train GRU on the source dataset as a feature extractor, then fix parameter of GRU and train the LR model on the target data.

From 2012, machine learning field has been pushed forward fast by the application of deep neural network. CNN [5, 6], inspired by the visual system of animals, has dominated the area and outperforms traditional computer vision methods in different problems. Using CNN instead of RNN which is designed for time series data, [7] not only improves the accuracy of text classification problem, also accelerates the entire training and inference speed.

Inspired by the last two work [4, 7], we designed a model using CNN as the pre-trained feature extractor and SVM as the classifier to substitute the final fully connected layer of CNN. As our final results show, our model is effective and robust enough that it can get a high performance on both binary and multi-labeled classification task.

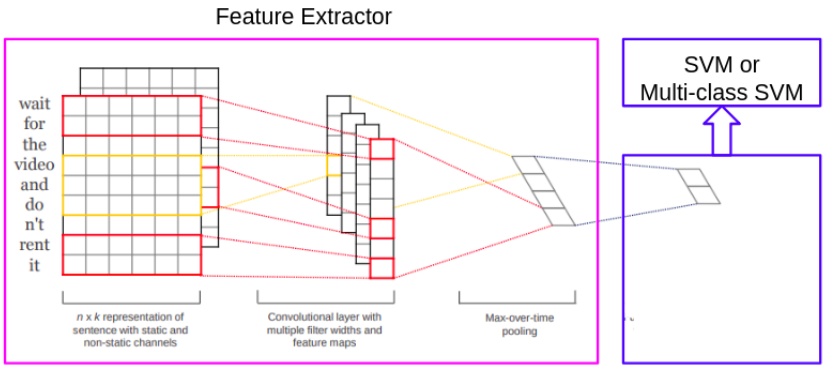

Figure 1: Transfer learning model for text classification. The CNN part pre-trained on Amazon-binary dataset is used as feature extractor for sentences and rather than using the fully connected layer, SVM is used for the classifier.

2 Model

As shown in Figure 1, our transfer learning model consists of two parts: CNN, which is used as the feature extractor, and SVM part, used for classification.

We use the pre-trained GloVe [8] model to map each word into a fixed-size vector, thus the input of the model is a 2-d $n \times k$ matrix, where n is the number of words in one review, and k is the length of word embedding, which in our experiments, is 50-d.

Let $x_i \in R^k$ denotes a k-dimensional word vector of the i-th word in the sentence. There are filters of different sizes and each kind of filter has multiple channels. Given a filter of size h, the feature c_i in the feature map could be generated by

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b)$$

where $\mathbf{x}_{i:i+h-1}$ denotes the concatenation from x_i to x_{i+h}. After the convolutional layer, we can get a feature map

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

for each filter, where n is the length of a sentence. Note that sometimes $n+1 \leq h$, when the maximum length of sentences in the same batch is less than the size of filters, we should pad all of them to the size of $h-1$, otherwise we only need to pad all sentences to the maximum sentence length in the same batch.

Then 1D max pooling layer is applied to each feature map and the results of pooling are concatenated. The idea is to capture the most important feature for each feature map. For avoiding the overfitting, we use dropout after the concatenation layer with a constraint on L2-norms of the weight vectors. Dropout prevents co-adaptation of hidden units by randomly dropping out a proportion p of the hidden units during forward back-propagation.

To improve the robustness and transferability, the last FC layer is substituted by SVM. Unlike FC layers which has huge number of parameters and needs large number of training samples to tune, SVM only relies on supports vectors, the number of which is much smaller than the training samples. That's quite

suitable for transfer learning since the size of target dataset is far less than the size of source dataset and can't be guaranteed to be sufficient to train a model well. Several SVM models are tested in our experiments. It is shown that linear kernel with L1-regularization is the best for our problem.

3 Experiments

3.1 Datasets

To train the transfer learning model and test the performance of it, we used the Amazon-binary classification dataset for source, Yelp-binary and Yelp-full dataset for target. All three datasets are also used in [8]. Amazon-binary and Yelp-binary dataset contains reviews crawled from Amazon.com and Yelp.com, respectively. The task is to determine which review is positive. Yelp-full is the multi-label dataset, which aims to assign the score from 1 to 5 according to the customers' review. The original datasets are divided into training and test set. To facilitate our experiments, we randomly sampled 25% of each training set as the validation set. For the details of the number of samples in each sets, please see Table 1.

Table 1: Number of samples for each dataset

Dataset	**Training**	**Validation**	**Test**
Amazon-binary	2,700,000	900,000	400,000
Yelp-binary	420,000	140,000	38,000
Yelp-full	487,498	162,500	50,000

3.2 Preprocessing

Like other natural language datasets, three datasets used in our experiments also exist noises. Most of the noises are special characters used in the reviews by people. Thus, like [7], we first whiten the dataset to remove those special characters, then according to words in GloVe [8] dictionary, make all the letters lowercase and add the space between word and the successor like "'m", "'re", "'d", etc. Then tokenize the cleaned dataset, and convert them using GloVe vectors [8]. For those words which do not appear in GloVe, we generate a random vector with same dimension for each such word using the uniform distribution with lowest bound of -0.25 and highest bound of 0.25, which is the same as GloVe.

3.3 Support Vector Machines

SVM model is one of the baseline models used in our experiments. We trained it on both Yelp-binary and Yelp-full training set, and then valued its accuracy on the test set of these two datasets. In the experiments, we tested two kernels for SVM: linear kernel and RBF kernel. For text classification problem with GloVe word representation, our experiments show that SVM with linear kernel performs better than using RBF kernel as shown in Table 2, thus we only used

the linear SVM for the following experiments. For the loss function, we used hinge loss for SVM, and also used L1 regularization to enforce the sparsity of the entire model. To select the best model, cross-validation was used to choose hyper-parameter $C \in \{0.01, 0.1, 1, 10, 100\}$ in optimization problem.

Table 2: Best Accuracy for Linear and RBF Kernel

Dataset	**Linear Kernel**	**RBF Kernel**
Yelp-binary	81.087%	50.166%
Yelp-full	44.100%	20.205%

3.4 Convolutional Neural Networks

Apart from SVM, CNN is another baseline model to compare with our transfer learning model. Like SVM, we train the CNN model on the target training datasets, and evaluate the performance of it on target test datasets to compare with transfer model.

We follow the hyperparameters of CNN part in the implementation of [7], but adjust the learning rate scheduler to narrower the loss function. We also use momentum optimizer of 0.9 to gain faster convergence and reduced oscillation. There are filters of size 3, 4 and 5, and filter of each size has 100 channels, so there will be 300 feature maps after the convolutional layer, which is also the size of output vector after our feature extraction. The model is trained for 10 epochs. When it comes to the middle stage, training accuracy becomes stable and we pick out the checkout point who has the highest validation accuracy. The loss on training dataset will continue to decrease but it might be resulted from over-fitting.

Table 3: Best Accuracy for Transfer Model on Target Datasets

Dataset	**Binary Classification**	**Multiclass Classification**
Yelp-100%	93.135%	51.260%
Yelp-60%	93.131%	51.790%
Yelp-20%	93.070%	51.489%
Yelp-5%	93.133	50.586%
Yelp-0.5%	91.715%	47.622%
Yelp-0.05%	89.147%	39.071%
Yelp-0.005%	86.419%	35.700%

3.5 Training and Testing of Transfer Learning Model

To test the performance of the transfer learning model and baseline models, we did two series of experiments. In the first one, we used Amazon binary dataset which is larger for the source and trained the CNN part of the transfer learning on it. Then froze the parameter of this part, and trained the SVM part of it on the Yelp-binary dataset. For those datasets, the domain of the source and target is different, but at least the task of them is the same, both of which are binary

classification problem. To added the difficulty for the transfer learning model, we did the same training process except training SVM on Yelp-full dataset as the second series. For this experiment, both the domain and task for source and target are different. Furthermore, both baseline models, SVM and CNN, were only trained and tested on the target datasets.

To prove the robustness of our models, when training them, we reduced the training sets for target step by step. For experiments, we chose 0.005%, 0.05%, 0.5%, 5%, 20%, 60%, 100% of the original training size, and trained same models on them.

Like in section 3.4, we fine-tune the hyperparameters on the different datasets. For the last SVM part, we use L1 regularization and linear kernel as stated before, and train it for max iteration of 1000. For the final test results for the transfer learning model, please see Table 3.

4 Results and Observations

Figure 2 shows the classification results on target test sets. Since our ratio of the whole training set is varied in 0.005%, 0.05%, 0.5%, 5%, 20%, 60%, 100%, to keep the plot balanced, we use the logarithm of those ratios as x-axis, and classification accuracy for y-axis.

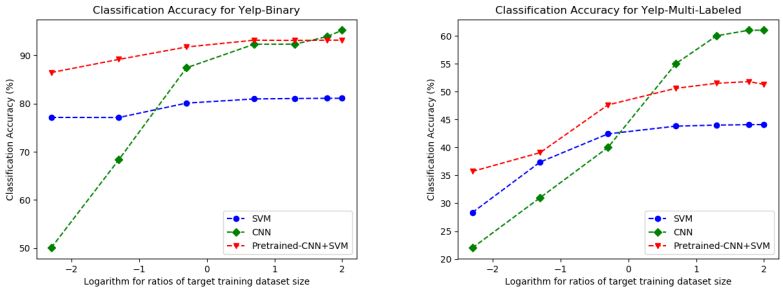

Figure 2: The test accuracy for target sets: Yelp-binary(left), Yelp-full(right). The x-axis is the logarithm based on 10 for the ratios from 0.005% to 100% of the target training set sizes used in the experiments.

Effectiveness of transfer learning model. For both the Yelp-binary and Yelp-full test datasets, on average, the performance of the transfer learning model is the best. This is surprising, since our transfer learning model was mainly trained on the source Amazon-binary datasets and all the baseline models were specifically trained on the target training datasets. Furthermore, since we pre-trained CNN on Amazon-binary once, and then froze it and used for different tasks. For those different test sets, the transfer learning model performs both better than SVM, which demonstrates that CNN is good at extracting features automatically and the features from our pre-trained CNN are general enough and well transferable.

Robustness of transfer learning model. As mentioned above, to test for the robustness of the transfer learning model, we consider the subsets of the entire target training sets. From Figure 3, we can see that when the size of training samples decreases, the speed of decreasing of the transfer learning

model is the lowest. At the same time, when the ratios of the training set is lower than 60%, the transfer learning model outperforms the other two baseline models by a large margin. Therefore, the transfer learning is insensitive to the decreasing number of training samples, which is more robust than other models.

Apart from the above observations, we can also see from those two figures that although CNN has good ability to extract features, it is hungry for data. Training CNN needs more data than transfer learning model and SVM, which illustrates why as the number of the target training samples decreases, the performance of CNN degrades so fast.

5 Conclusion

In this work, we proposed a new transfer learning model which is proved to be effective and robust. Inspired by the prior papers, we overcome their shortcomings and get a better model which is easier to transfer. For the future work, the reasons of the error will be analyzed by printing out the erroneously classified examples by the model, and according to it, we will improve our model to achieve a higher accuracy.

6 Acknowledgements

The authors of this paper thanks Professor Mehryar Mohri, who gave an entirely amazing and excellent machine learning course in this semester, through which the authors gained a solid theoretical background for analyzing and applying machine learning algorithms. The authors also want to thank for all the work done by graders. Without your help, we could not have gotten such a rapid feedback for our questions and assignments.

References

[1] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[2] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A Survey on Deep Transfer Learning. In *International Conference on Artificial Neural Networks*, pages 270–279. Springer, 2018.

[3] JC. Do and A. Ng. Transfer learning for text classification. In *NIPS*, 2005.

[4] P. Gupta, P. Malhotra, L. Vig,and G. Shroff. Transfer Learning for Clinical Time Series Analysis using Recurrent Neural Networks. In MLMH Workshop, *KDD*, August 19-23, 2018, London.

[5] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, et al. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, 1990.

[6] A. Krizhevsky, I.Sutskever, and G.E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS, pp. 1106–1114*, 2012.

[7] Y. Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2014.

[8] X. Zhang, J. Zhao, and Y. LeCun. Character-level Convolutional Networks for Text Classification. In *Proceedings of NIPS*, 2015.