5MinuteCNN: A surprisingly simple, fast, accurate architecture for cross domain sentiment classification

Prajwal K R, Shruti S, Sivasankar E

Department of Computer Science and Engineering,
National Institute of Technology, Tiruchirappalli.
prajwalrenukanand@gmail.com
nks.shruti@gmail.com
sivasankar@nitt.edu

Abstract. In this paper, we propose a deep learning solution for cross domain sentiment classification. Our solution is a small (memory-footprint), simple convolutional neural network [10] that trains in about five minutes to produce results comparable to the state of the art in cross domain sentiment analysis [13], and in several cases, exceeds it. Our solution is also unique when compared to the other state of the art solutions, that it does not require any data from the target domain, labelled or unlabelled. That is, this solution adapts to different domains without requiring explicit domain adaptation. Our tests show that our solution trained exclusively on a single source domain can generalise across 6 different domains with zero prior knowledge about the target domain. We think that this serves as a strong state-of-the-art baseline for cross domain sentiment analysis, as these results are entirely achieved by no explicit domain adaptation.

1 Introduction

Over the recent years, there has been a growing need for identifying sentiments in online content like tweets and product reviews. Sentiment analysis is a widely-explored problem in NLP, and several approaches [16] [9] have achieved excellent results, typically using a deep learning approach. These approaches however require huge number of training examples to train. Most of the domains, however, have very little or no labelled data to train upon. In this scenario, domain adaptation is particularly significant given the large number of domains, for example various categories in product reviews. In cross-domain sentiment classification, we train our model on a source category and test its performance on a target category. For example, Home & Kitchen could be a source category, Books could be a target category.

Cross Domain Sentiment Classification, however, is a challenging problem to solve. The word "sharp" would carry a positive sentiment for a kitchen product like a knife, but would not mean anything in another domain like "Books" or "Movies". Similarly, "thrilling" would carry a positive sentiment in a domain

like “Books” or “Movies” and would carry no significance in “Home & Kitchen”. A good domain adaptation algorithm enables one to train on a single (source) domain and classify on various other (target) domains. In an ideal case, no labelled data from the target domain is used. In our approach, we take it even further, where we require no data from the target domain to get near-state-of-the-art results.

2 Literature review

The importance of domain adaptation has motivated several prominent work in this area ([2] [14] [4] [3]). In recent years, with the advances of deep learning in NLP, multi-layer neural network models such as RNNs and CNNs have been widely used in sentiment classification and have achieved good performance ([16] [9] [8]). While [16] employs Convolutional Neural Networks for sentiment classification, [9] takes a different approach, a character-level model using both CNN and LSTM architectures. We use Convolutional Neural Networks and pre-trained word vectors similar to the one specified in Kim (2014) [8].
There have also been several deep learning approaches for the problem of domain adaptation by generating an intermediate shared representation that tries to capture shared features of the source and target domains, like [20] [13], and these papers have achieved state of the art results. They try to reduce the training-testing distribution gap using different methodologies. [20] uses domain independent sentiment word lists, which are obtained by removing pivot words from the review texts. A joint learning mechanism is used to achieve domain adaptation. [13] takes the advantage of marginalised Stacked Denoising Autoencoders (mSDA) to learn an intermediate shared representation by regenerating a perturbation of the input. The SVM classifier classifies the intermediate layers for sentiment. To our knowledge, we found the results of this publication [13] to be the state of the art in cross domain sentiment classification.
In this paper, we propose a simple plain CNN architecture that takes about 5 minutes on a modest GPU to train on a hundred thousand training examples and gives comparable results to the state of the art, in some cases, even exceeds them.

3 Methodology

In this paper, we use Amazon product reviews dataset, published by Julian McAuley, UCSD, and was also used in the paper [12]. The dataset contains several categories of products with several hundred thousand reviews in each category. We choose 6 categories : “Books”, “Movies”, “Clothing”, “DVD”, “Home & Kitchen”, “Electronics”. We chose only a subset of the reviews in each category; all the categories except “Clothing” have exactly hundred thousand reviews, while “Clothing” category has around 52000 reviews. This subset of data is balanced between positive and negative reviews. The reviews in the dataset are rated out of five. We consider the positive reviews as reviews with rating strictly

greater than three out of five, the negative reviews as those with rating strictly less than three. Each review consists of several sentences.

3.1 Preprocessing

We consider each review as a sequence of words. Each word is a token. Therefore, each review is a sequence of tokens. The length of each review is clipped to a maximum of 200 words. Reviews with lesser number of words are padded. We clip the vocabulary size to most common 40,000 words in each category based on word frequency.

3.2 Embedding Matrix generation and use of Transfer Learning

In Convolutional Neural Networks for Sentence Classification, Kim (2014)[8] has obtained excellent results by using pre-trained word vectors for sentence classification using Convolutional Neural Networks. Using pre-trained word vectors provides an increase of around 1-2% in cross domain classification accuracy compared to learning word vectors from scratch. We initialize our input vectors with GloVe [15] vectors of 50 dimensions. The GloVe embeddings we used were trained on Wikipedia 2014 + Gigaword 5 dumps with 6 billion tokens. The sequence of 200 words for each review now become a matrix of 200 x 50 dimensions. The words not present in the GloVe corpus are represented as a vector of zeros. In Convolutional Neural Networks for Sentence Classification, Kim (2014)[8], it has been shown that pretrained word vectors increase model performance because they contain several intrinsic features of the language. They obtain higher accuracy in many cases by keeping the word vectors static. However, we deviate in this aspect. In our implementation, the word vectors are fine-tuned, i.e., the word vectors are not kept frozen during training. We found that keeping the word vectors frozen gives much sub-optimal results.

3.3 Convolutional Neural Network

The embedding matrix obtained in the previous stage is fed as input the Convolutional Neural Network. The CNN contains 3 Convolutional layers and 3 Pooling layers. We use Exponential Linear Units [6] as the activation layer. ELUs attempt to make the mean activations closer to zero and this speeds up the learning process. It has been shown that ELUs obtain higher classification accuracy than RELUs. [6]

$$f(x) = \begin{cases} x, & \text{if } x >= 0 \\ a(e^x - 1), & \text{otherwise} \end{cases} \tag{1}$$

We also experiment with Dropout [18] and we found it to improve accuracy when used with a low dropout probability at the input layer. Throughout this model, we try to enhance the ability of the model to generalise. We strive to keep the

model as small as possible while also capturing the maximum number of features.

Architecture and Implementation Details

1. The first layer in an input layer (200 word vectors of dimension 50 each).
2. We also test the model with Dropout after the Input layer, this shall be described later.
3. We use a total of 3 convolutional layers.
4. We start with 16 filters in the first layer and double the number of filters in the subsequent layers.
5. Each filter spans across a window of 3 words. We keep the filter size small, following the pattern in the VGGNet [17]. Zero padding is done appropriately. We found that accuracy drops significantly due to loss of information when padding is not done.
6. The Convolutional Layer is activated by the ELU activation layer.
7. It is followed by a max-pooling layer with stride and width 5. We want to minimize overfitting and a smaller architecture does exactly that. In the beginning layers, we can use a larger pool width and stride to reduce the input sizes to the next layer. The next max-pooling layer has a small stride and width of just 2, again following the pattern in VGGNet [17].
8. We do not use a fully connected layer. Our network is a Fully convolutional Network (FCN) [11]. Using this instead of a fully connected layer increased accuracy by reducing overfitting. Instead of a fully connected layer, we do a Global Max Pooling to get a single vector which is used to predict a single output.
9. The output is activated by the classic sigmoid activation function, which gives the probability of the review being positive. We use cross entropy loss and RMS Propagation [19] as the optimizer.
10. The model is trained on about hundred thousand training examples (except Clothing category which had around 52000 examples) with an Nvidia GeForce GTX 860m for about 20 epochs (14s / epoch) to reach maximum validation accuracy in the source domain. We evaluate the final model on the target domain.
11. The entire architecture is implemented in Keras library [5] with Tensorflow [1] backend.

Our final architecture is depicted in Fig. 1, along with the dropout layer.

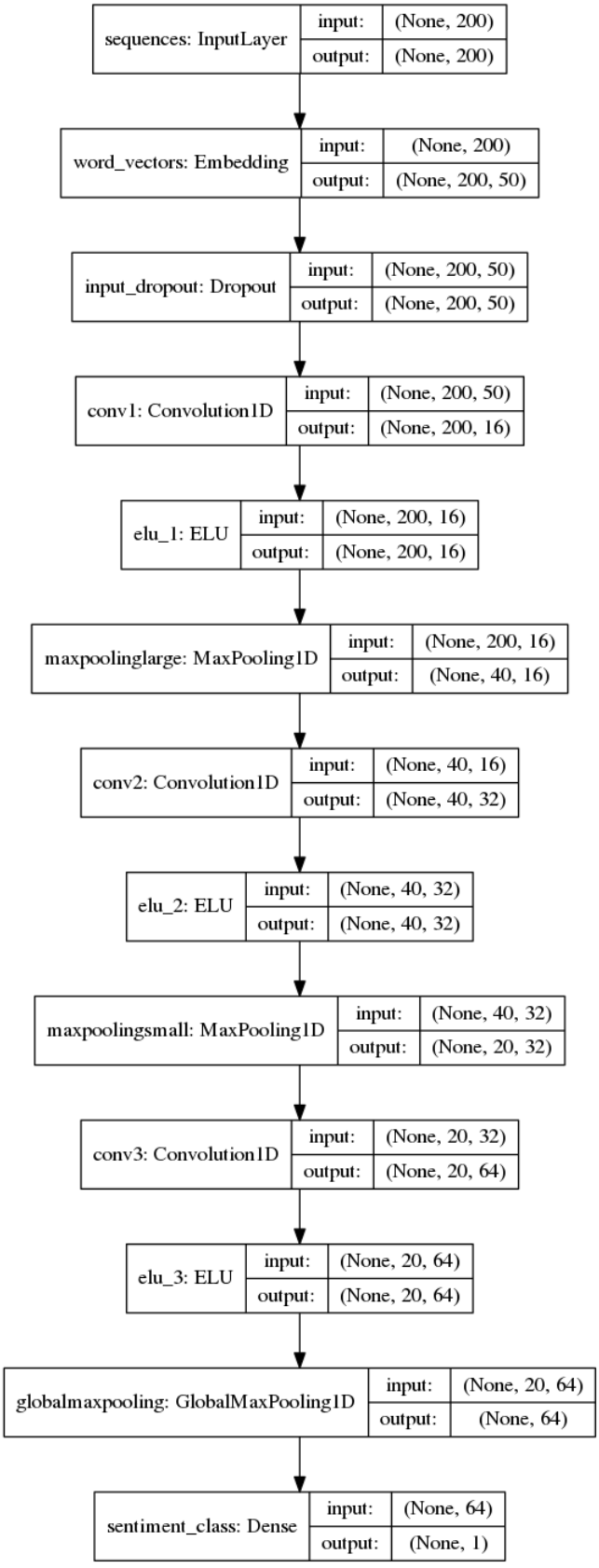

Fig. 1. CNN Architecture

4 Experimental results

We found our model to generalise across domains, but we wanted to see how much generalization ability we can squeeze out of as memory efficient model as possible. Apart from trying different methods to improve generalisation and reduce memory usage, we also tested with different optimizer functions and dataset sizes.

1. Making the model / vocabulary smaller
   (a) We tried halving the number of filters in the model, but that led to significant drop in accuracy, due to lesser number of features captured.
   (b) We tried reducing the maximum number of words in each review but that led to a slight drop in accuracy.
   (c) We could not also trim the vocabulary size by more than one-thirds the maximum vocabulary size while considering all the reviews in a category.
   (d) We increased the review length to 200 words and it could generalise better across domains as different domains had reviews of varying length distribution.
2. Using Dropout across the layers
   (a) We experimented with dropout of 10% - 20% drop ratio across the convolutional layers but it led to drop in accuracy. Convolutional layers have an inherent tendency to combat overfitting and using dropout did not benefit.
   (b) However, adding a dropout with probability 0.1 after the input layer led to an average of 1% to 1.5% increase in accuracy with no modification to the model. It increased the training time by a few epochs.
   (c) Using greater dropout ratios at the input layer performed equally well as the case without dropout or even worse. In many cases of using high dropout ratio (30%), it started leading to erratic behavior due to significant portions of input being dropped out.
3. Dataset size
   (a) Halving the dataset size reduces cross domain accuracy by a marginal 1%
   (b) However we observed that it also reduces the models confidence in its predictions (higher cross entropy loss).

Table 1. Accuracy across various source-target pairs without using dropout

Source/Target	Electronics	Home & Kitchen	Clothing	Movies	Books	DVD
Electronics	89.7	89.1	88.8	80.6	78.1	78.8
Home & Kitchen	86.2	89.3	88.7	76	76.3	77
Clothing	82.1	84.5	89.5	74.4	74.4	74
Movies	81.3	80.7	83.6	88.7	86.5	86.6
Books	79.4	80.8	82.4	86.3	89.3	84
DVD	79.1	79	81.6	85.7	83.9	88.7

Table 2. Accuracy across various source-target pairs after using dropout

Source/Target	Electronics	Home & Kitchen	Clothing	Movies	Books	DVD
Electronics	90.2	89.5	89.4	81.3	78.7	81
Home & Kitchen	87	90.2	89.2	78.2	77.3	77.1
Clothing	82.8	85.1	90	76	74.8	75.6
Movies	81.4	81.9	84.5	88.7	87.1	87.2
Books	80	81.3	82.8	86.9	90.1	85.3
DVD	79.6	79.6	81.8	86.3	84.7	89.2

5 Conclusion

We propose a competitive state-of-the-art baseline that exceeds results of models that perform explicit domain adaptation. Our CNN is memory-efficient, simple, fast and has the ability to generalise across domains. Since our model can get the experimental results shown above with its inherent domain generalisation capabilities, we think that further improvements to the near-state-of-the-art results can be made with explicit domain adaptation methods.

References

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
2. John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.
3. Danushka Bollegala, Tingting Mu, and John Yannis Goulermas. Cross-domain sentiment classification using sentiment sensitive embeddings. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):398–410, 2016.

4. Danushka Bollegala, David Weir, and John Carroll. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *Knowledge and Data Engineering, IEEE Transactions on*, 25(8):1719–1731, 2013.
5. François Chollet. Keras, 2015.
6. Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.
7. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.
8. Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.
9. Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.
10. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
11. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
12. Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2015.
13. Debora Nozza, Elisabetta Fersini, and Enza Messina. Deep learning and ensemble methods for domain adaptation. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 184–189. IEEE, 2016.
14. Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.
15. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.
16. Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962. ACM, 2015.
17. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
18. Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
19. Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.
20. Jianfei Yu and Jing Jiang. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification.