CNN-based Text Classification with Pre-trained Word2Vec Vectors

Tyler Wengert
UCCS
twengert@uccs.edu

Abstract

One of the large challenges faced by researchers is text classification. In the early and mid 2010s, with the introduction of word vectors and Convolutional Neural Networks (CNNs), neural networks started achieving results on classification problems that met and soon far exceeded the state of the art. This paper describes the setup of a text classification CNN based on the architecture of (Kim, 2014). As previously shown by Kim's model, a convolutional network setup like this achieves excellent results, my model achieving average results across all data sets of about 84%.

1 Introduction

Text classification has a wide variety of applications across natural language processing, including sentiment analysis and document identification. For many years, image classification was achieved through complex mathematical calculation and hand picked features and filters. It was tedious work. For classification, we want the high level of captured feature information that convolutions provide, without the necessity of picking desired features yourself, and potentially missing important information. CNN architectures allow researchers to avoid hand picking features, since the neural network will learn the features and filters itself. It still maintains the high level of scrutiny afforded by traditional convolutions, and will likely learn things that researchers would miss.

The advantage of using CNNs over other types of neural networks for classification is in the way that the CNN looks at its input data. Nodes focus only on specific parts of the data, instead of looking at the whole thing, and determine features locally that are then integrated into the global set. The convolutional filters that the system uses are applied to small frames of data instead of the whole body of data, which allows the system to learn incredibly well.

2 Related Works

The first CNN architecture was described by (LeCun et al., 1998). They used two convolutional layers, each followed by a pooling layer, and finally a fully connected layer as the output layer. The researchers had successfully developed a model that would learn features without the need for hand-crafted feature extractors. Trained on over 500,000 images, they achieved an accuracy of 82% classifying ASCII characters on checks. This was an almost 20% improvement against older models that they compared against.

CNN architectures did not see widespread use until 2012, when the AlexNet architecture, as described by (Krizhevsky et al., 2012), competed in an won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The AlexNet model consisted of five convolutional layers, some followed by max-pooling layers, three fully connected layers, the final one being a soft-max layer. The results achieved by AlexNet were state of the art at the time. Kim (2014) described an architecture for text classification using multiple convolutional channels. In The results of all of the channels were merged using a max-pooling label to come to the best classification prediction. In addition to using a multi-channel approach, the system was also initialized with pre-trained Word2Vec vectors, as described by (Mikolov et al., 2013). In the paper, Kim noted that the Word2Vec vectors seem to be good feature extractors, and show

promise in use with many different data sets.

3 Implementation

This program was built using Python 3.7.6 64-bit using the IDLE IDE. The program uses the GenSim package (Řehůřek and Sojka, 2010) to load in the word vectors from a format file provided by Google, as well as NumPy (Oliphant, 2006–) for mathematics involving the vectors, and NLTK (Loper and Bird, 2002) for preprocessing of the corpora.

3.1 Word2Vec

The Word2Vec representation was described by (Mikolov et al., 2013), and has since become a standard for many NLP tasks looking to assess the semantic and syntactic relationships among words. Word2Vec vectors are highly dense vectors, especially compared to one-hot encodings, and can improve training results as well as shorten training time. These vectors are especially useful when there isn't a readily available corpus of sufficient size for a certain task.

In this application, the Word2Vec vectors give the model all the information it needs to know about word relationships in the English language, and from their all the model needs to learn is which ones are typically associated with either objective or subjective content.

3.2 CNN Architecture

This architecture is largely based on (Kim, 2014), whose model set the standard for classification when it was published. In his paper, Kim compared the results from both single and multi-channel CNNs; here, a version of the single channel model is reproduced.

The first layer in the model is the Embedding layer. It is here that the model is fed in the pre-trained Word2Vec vectors associated with the corpus words. The Embedding layer's 'trainable' parameter is set to false, since the model will use the pre-trained word vectors and there is no need to train its own. In addition to providing transfer learning to the model, this reduces the trainable parameters for the system to run, and greatly reduces training time. The Embedding layer feeds into a 1D Convolutional layer. This Convolutional layer uses 32 filters, a kernel size of 4, and a ReLU activation function. From there, the data is passed into a Dropout layer, which dropped 50% of the nodes. After applying dropout, data was fed into a Max-Pooling layer and a RelU activated, dense hidden layer. The output layer was where this model differed from Kim's; his output was a dense soft-max layer, whereas this model uses a dense sigmoid layer. In his research, Kim worked with classification among several different categories, and so soft-max was appropriate. However, this data set only tests binary categorization. Since there were only two classifications (0 = objective and 1 = subjective; 0 = male and 1 = female), the soft-max layer was pushing all the results to 1. The model achieved nearly perfectly 50% accuracy at every epoch of training as well as during evaluation. This makes sense, since it was just guessing 1 every time. The sigmoid function is much better suited to binary classification, and the final model shows this by performing far above 50% accuracy.

4 Results

4.1 Subjectivity Data Set

The model performed quite well on both models, but produced better results for the Subjectivity data set.

Training Accuracy	Test Accuracy
91.6%	87.9%

Training accuracy was evaluated by running the completed model on the same data set that it was trained on. Test accuracy was evaluated by running the model on a set of examples that it had never seen before. The training/testing split used in the data set was .2, meaning that 8,000 examples comprised the training set, and the remaining 2,000 were the testing set, out of 10,000 total examples.

It seems like the subjectivity data set had greater vocabulary difference between the objective sentences and the subjective sentences than did the male and female written blog posts. Certain concrete descriptor words could be flags to the system that a sentence is more or less subjective. The subjective sentences tended to use more colorful language in describing a film, and so many words appeared in these examples that did not in the objective ones. This could be one reason that the model achieved higher accuracy on these examples.

4.2 Author Gender Data Set

While it achieved fairly high results, the model did not do quite as well in labeling a blog author's gender as it did in subjectivity classification. The training/testing split used in the data set was .2, meaning that 2,585 examples comprised the training set, and the remaining 647 were the testing set, out of 3,232 total examples.

Training Accuracy	Test Accuracy
80.4%	76.0%

As noted in the previous section, the blog gender data set did not seem to have quite as large of a difference in vocabulary between the two classes as did the subjectivity data set. There are simply less words that are specifically for usage by males or by females than words that are specifically for usage in objective statements or in subjective statements. In general, the differences in the gender data set seemed to be more structural and tonal. So while the two genders may have used largely similar vocabularies, the way they present those words was certainly distinct, and the model did a good job picking up on those relationships.

As a note, in the previous section and in this one, the gender data set is described as having less lexical diversity and more structural relations, while the subjectivity data set is described as having the opposite characteristics. This is not to say that the gender data does not have any lexical diversity, nor that the subjectivity has no structural relations, just that in relating the data sets to each other, some characteristics were more prominent than others.

5 Further Work and Improvements

One immediately clear way to improve on this model would be to upgrade it from a single channel model to a multi-channel setup, as also described in (Kim, 2014). The multi-channel model feeds the same input to many Convolutional layers, each with its own unique hyperparameters to evaluate the data, and then pools the results together to get the best resulting prediction.

While it is not implemented here, a multi-channel approach using Tensorflow was researched, and its architecture described. Tensorflow provides two simple ways to vary channels in a model. Each Convolutional layer is initialized with a number of filters. The network learns and refines these filters itself during training. By varying the number of filters that the convolution applies, you can get a variety of results from of a Convolutional model, and so by having, suppose, three channels, applying 16, 32, and 64 filters respectively, you can merge all those results and get the benefits from each convolution. In addition, Tensorflow also allows the user to specify kernel size. This is the window over which the filter is applied. One can also improve or detriment a model by changing the kernel size, and so this is also a good variable to change in each channel. Given three channels, one could apply kernel sizes of 2, 4, and 8 to get different results to merge. You could vary one or both of these variables in your channels, conceivably having up to 9 convolutional channels using the 6 previously mentioned kernel and filter sizes, or more if you want to use more sizes. Outputs from the Convolutional layer undergo Max-Pooling, then the results are flattened, and concatenated with the flattened results from each of the other channels. This concatenated result undergoes Global Max-Pooling, and is finally output with a soft-max layer, as in Kim's multi-channel model, or for binary classification, a sigmoid layer.

6 Conclusion

Since CNNs came into wide use for classification problems, they have continued to improve and provide state of the art results. My simple, single channel CNN shows the excellent results these methods can achieve, as well as the meaningful, dense features that Convolutional layers are able to capture from data. While the model already has quite high results on subjectivity classification, it has a little more room for improvement in gender classification. It is likely that better results could be achieved by extending the model to a multi-channel approach.

7 Acknowledgements

The following are papers that were referenced, but not cited specifically in this paper: (Wang et al., 2018), (Joulin et al., 2017), (Liu et al., 2017)

References

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. pages 427–431.

Yoon Kim. 2014. 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. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 115–124.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Travis Oliphant. 2006–. NumPy: A guide to NumPy. USA: Trelgol Publishing.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Shiyao Wang, Minlie Huang, and Zhidong Deng. 2018. Densely connected cnn with multi-scale feature attention for text classification. In IJCAI, pages 4468–4474.