Research Journal of Computer and Information Technology Sciences ISSN 2320 – 6527
Vol. **7(2),** 1-5, December **(2019)** *Res. J. Computer and IT Sci.*

Hyper-parameter optimization: towards practical sentiment analysis using a Convolutional Neural Network (CNN)

Zachary Kirori
Kirinyaga University, Kerugoya, Kenya
zkirori@kyu.ac.ke

Available online at: www.isca.in
Received 3rd August 2019, revised 15th November 2019, accepted 10th December 2019

Abstract

Current advancements of in Deep Neural Networks (DNNs) has made it possible to assess the latent and sentimental polarity in text. This is helpful in many applications such as hate speech recognition and mood determination. Sentiment analysis, as it is popularly known in machine learning, is a typical task in text classification – an active research area of Natural Language Processing (NLP). One of the main challenges in machine learning is the ability to create accurate models applicable to real-life problems. In this paper, we report on the experimental results of hyper-parameter optimization of a Convolution Neural Network (CNN) – one of the most promising deep machine learning architecture for text processing as an attempt to move towards practical sentiment analysis models. The results indicate remarkable improvements over ordinary CNNs on typical machine learning datasets.

Keywords: Sentiment analysis, convolution neural networks, hyper-parameter optimization, machine learning, deep neural networks.

Introduction

Text classification is a key task in Natural Language Processing (NLP) with varied applications in sentiment processing, spam detection, topic labeling among others. It is the task of assigning a set of pre-defined corpus tags to unstructured text that might include web pages, social media, chats, emails, support tickets and survey response according to their characteristics. In as much as text is a rich source of information, extracting useful information therein is usually hard and consumes a lot of time because of its unstructuredness. Many entrepreneurs are progressively turning to this approach as a way of enhancing decision-making and automating business processes[1].

Manual and automatic text classification methods are common[2]. In the manual approach, a human annotator infers the textual content and assigns it a category. Although time-consuming and effort hungry, this method typically provides good results. Automatic techniques rely on machine learning approaches, to automatically classify text - which is faster, convenient and more efficient. Machine learning approaches typically appear as rule-based, machine learning or hybrid systems, Machine Learning approaches are trained to learn to make latent-based categorizations using past data. Through supervised learning of data instances, these algorithms learn by associating instances of input to the corresponding output thus tagging the instances accordingly.

The learning process typically begins with feature extraction in order to transform each textual input into feature vectors and then a bag of words (BOW) model is used to map the vectors using the word frequencies in a predefined dictionary of words[3]. For instance, we could define a dictionary with the words: {The, appealing, bad, is, not, football}, the BOW model would vectorize (or tokenize depending on the approach) the text *"The appealing football"* as: (1, 1, 0, 0, 0, 1). The resulting classification model would appear as shown in Figure-1.

Upon training, the deep learning model is used to make predictions on unseen data. In this step, the feature extractor transforms new input text to an array of feature sets, which are then directed to the classification system to make predictions on the new tags according to the task category as illustrated in Figure-2.

The most common text categorization algorithms include support vector machines (SVMs), deep machine learning and naive bayes. The Deep machine learning category comprises a set of techniques and methods inspired by the natural brain. Two of the viable candidates in the arena of deep machine learning architectures for text categorization are Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). The key advantage of the deep machine learning approaches is that they get better with additional training though this may require more time to reach convergence[4].

Related research: Several researchers have attempted to solve the sentiment analysis problem to varied degrees of success. A state of the art system was programmed known as NOVEL2GRAPH to give visual summarization of text corpora enhanced by artificial intelligence[5]. Another deep machine learning approach for latent modeling for the amharic language was described and experimented[6]. Jaspreet Singh et al[7], compared optimization procedures for a set of tradition recommendation algorithms and reported One R to outperform the rest. The works of Samuel Pilcer[8], reports on the application of artificial intelligence techniques to sentiment modeling while Support Vector Machine (SVM) optimization was experimented[9].

Hybrid approaches have also been experimented in the past. An experiment on a mixed method of rule-based techniques and deep machine learning to enhance aspect-level latent modeling has been reported[10]. The research output of Md Shad Akhtar et al[11] reported a hybrid approach: the multi-objective optimization (MOO) framework for SVM classification. Yet another hybrid technique was also experimented using Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to characterize a deep learning model on the Stanford Sentiment Treebank andSAR14 data sets[12].

Convolution Neural Network optimization on big social data using text polarity feature has been reported[13]. Re-inforcement learning has been applied to sentiment analysis to comparable performance among many other optimization experiments[14]. An experiment with BowTie - a deep machine model based on a feed forward network for sentiment modeling was reported in Apostol Vassilev[15].

Methodology

Deep Learning Platform: The selected development platform consisted of a set of Python DML libraries and frameworks available for the experiment under the MIT permissive license namely: Keras and Tensorflow. Tensorflow is one of the numerical backend platforms in Python. They provide an experimental platform for empirical Deep Machine Learning development and research. Keras runs on advanced versions of Python with options to execute on CPUs and GPUs based on available hardware the underlying frameworks[16].

Dataset: The dataset for this experiment was the Sentences Data Set of Labelled Sentiment data provided by the publicly available Machine Learning database at UCI. The dataset is a collection of reviews (labeled) from Amazon, IMD and Yelp. The reviews are marked with scores of 0 or 1 for a negative and positive sentiments respectively.

Data Pre-processing: After extracting the download, we then go ahead to train a guided neural model for predicting the latent of a given statement or sentence. The tokenizer feature provided by the scikit-learn library within Keraswas used to tokenize sentences by breaking them into constituent fragments known as tokens. By taking the content of each statement, it creates a dictionary of words based on their respective levels of uniqueness. The resultant vocabulary was then used to generate an array of feature vectors based on word count. More specifically, the tokenizer decomposes the statements into a set of "tokens" and further removes special characters and punctuation attributes as well as any further preprocessing needed to every input word.

For instance, after tokenizing the sentences {'Mary enjoys chocolate', 'Mary hates ice cream'}, the resultant vocabulary is {'Mary': 0, 'cream': 1, 'chocolate': 2, 'ice': 3, 'enjoys': 4, 'hates': 5} and the resulting vector is a two dimensional feature vector of the original sentences, thus: array ([[1, 0, 1, 0, 1, 0], [1, 1, 0, 1, 0, 1]])

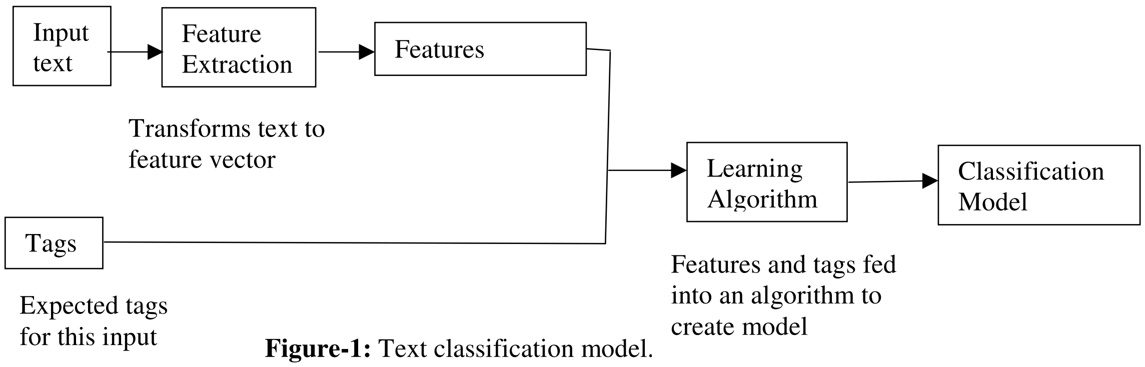
Input text
Feature Extraction
Features
Transforms text to feature vector
Learning Algorithm
Classification Model
Tags
Features and tags fed into an algorithm to create model
Expected tags for this input

Figure-1: Text classification model.

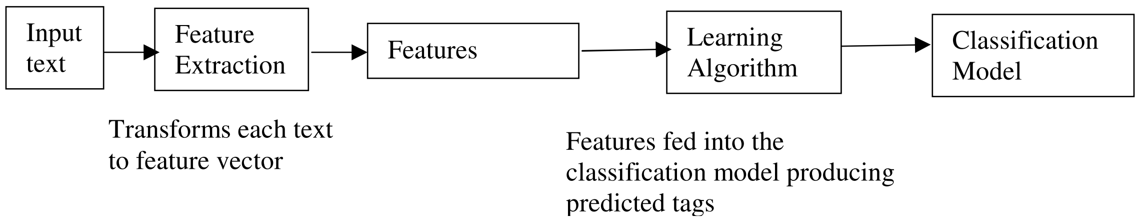
Input text
Feature Extraction
Features
Learning Algorithm
Classification Model
Transforms each text to feature vector
Features fed into the classification model producing predicted tags

Figure-2: Tags extraction pipeline.

Baseline Model

We built our baseline model using a typical deep learning neural model – the Convolutional Neural Network (CNN) also invariably referred to as a 'convnet'. CNNs have the ability to features from datasets such as images and text. A CNN hidden layers (convolutional layers) which facilitate dealing with two-dimensional (2D) matrix of numbers representing the features vectors as 2D matrices. The convolutional layers' ability to detect corners, edges and any additional kind of textual features make CNNs to be good candidates for sentiment analysis.

Figure-3 illustrates the typical working of a CNN for text processing. It begins by processing a section of input attributes using the filter kernelsize. With this section, the computed product of weights of the filter is generated and stored in memory. One dimensional (1D) CNN remains unaffected by translations, implying that certain sequences are seen at different positions – an ability useful for certain patterns in the text.

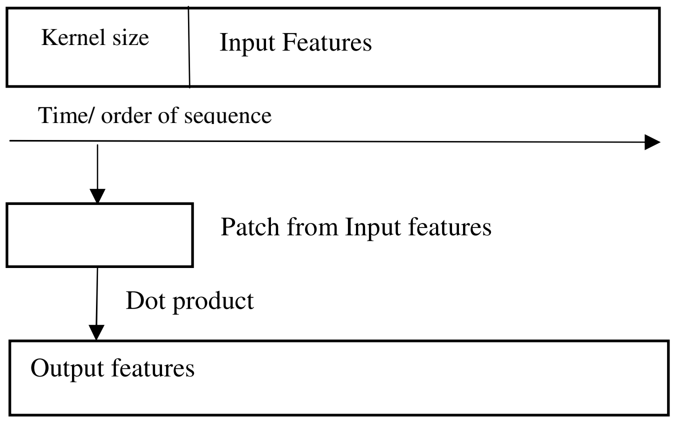

Figure-3: Feature Processing in a 1-D CNN.

With Keras' Convolutional layers, we need the Conv1D layer – the 1D layer appropriate for text processing. Selecting the required parameters such as the size, filter number, the input/output dimensions, the optimizer, the activation function among others, the layer is added between the GlobalMaxPool1D and the Embedding layers. After compiling the model, we examine its architecture using the Keras' supplied summary() function. This architecture is depicted in Table-1.

Table-1: The model architecture.

Type of Layer	Shape of the Output	No. of Parameters
Embedding	0, 0, 100	174700
Conv1D	0, 0, 128	64128
Global_max_pooling1d	Glob 0, 128	None
Dense	0, 10	1290
Dense	0, 1	11

After training the model, for 10 epochs and a batch size of 10, the baseline exhibited a training accuracy of 100% while the testing accuracy was 80.4%. The graph in Figure-4 below is the training and testing accuracies.

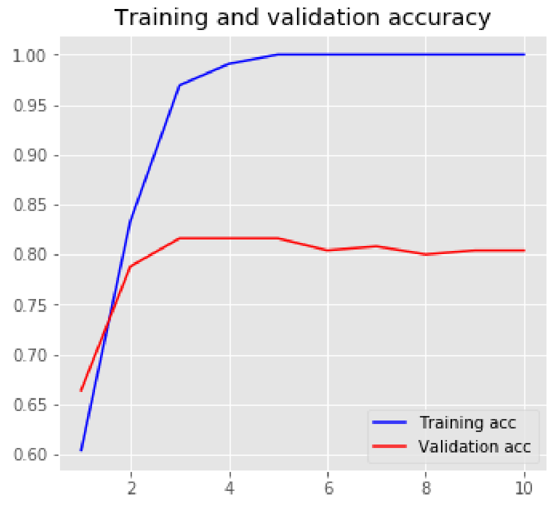

Figure-4: Training and testing accuracy.

Our Approach: Hyper-Parameter Optimization

Of the many tricks in empirical deep machine learning research, hyperparameter optimization is key. The most common technique for hyper-parameter optimization is grid search. It proceeds by taking an array of parameters and by executing the model with every possible parameter combination, it outputs the best. This procedure is very thorough but is also very computationally demanding. An alternative to grid search random search that takes random combinations of parameters and attempts to match them out.

The Keras Classifier serving as a high-level application programming interface (API) wrapper for the scikit-learn in Keras, offers a number of useful tools present in the scikit-learn library. The support class required is Randomized Search CV for cross-validation when performing random search. This has the effect of validating the model by separating the dataset it into several training and testing data sets.

K-fold cross-validation partitions the data set into *k* equal-sized portions with one set being used for testing and the rest for training. This facilitates the execution of *k* different epochs, with each partition being used for testing. In the resulting output, the higher *k* value is taken to be the more accurate. In this experiment, iterations over each data set was performed as a preprocessing step. After running the model with selected parameters for Randomized Search CV, it yielded a Best Accuracy of 0.8127 or 81.27% accuracy for hyper-parameter values of: {'num_filters': 64,'vocab_size': 4603,'kernel_size': 7, 'maxlen': 100, 'embedding_dim': 50}. The resultant test accuracy was noted to be 0.8384 or 83.8%, demonstrating an improved and reasonable level of accuracy.

Findings: The baseline model seems to suffer from over-fitting, as its training accuracy was 100% while the testing accuracy was 80.4%. However, with hyper-parameter optimization exhibited lower training accuracy noted as 81.27% with a higher testing accuracy noted as 83.8%. Perhaps due to the limited data size in this dataset and given that CNNs typically exhibit exemplary performance with massive datasets, the test accuracy remained in the 83rd percentile. Nevertheless, it is worth noting that with hyper-parameter optimization, the model exhibits better capacity given its ability to classify unseen instances batter and to a high degree of accuracy.

Conclusion

Resulting from the reported experiment, it is evident that with proper fine-tuning of hyper-parameters, we can enhance the performance of deep machine learning architectures – especially in this reported case of sentiment analysis.

Future Work: As future improvements to this work, we suggest further experimentation with additional hyper-parameters as well as trying out other methods for Cross-Validation such as the nested cross-validation.

References

1. Tang D., Qin B. and Liu T. (2015). Deep learning for sentiment analysis: successful approaches and future challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(6), 292-303. doi: 10.1002/widm.1171. Retrieved from https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1171

2. Ain Q.T., Ali M., Riaz A., Noureen A., Kamran M., Hayat B. and Rehman A. (2017). Sentiment analysis using deep learning techniques: a review. *Int J Adv Comput Sci Appl*, 8(6), 424. Retrieved from https://thesai.org/Downloads/Volume8No6/Paper_57-Sentiment_Analysis_using_Deep_Learning.pdf

3. Prajwal S. (2019). Sentiment analysis for text with Deep Learning. *Towards Data Science*. Retrieved from https://towardsdatascience.com/sentiment-analysis-for-text-with-deep-learning-2f0a0c6472b5

4. Zainab H., Muneera A., Nora A., Latifah A. and Sarah A. (2019). Arabic Sentiment Analysis Using Deep Learning: A Review. *IJCSNS International Journal of Computer Science and Network Security*, 19(4), 255. Retrieved from http://paper.ijcsns.org/07_book/201904/20190435.pdf

5. Vani K. and Alessandro A. (2019). NOVEL2GRAPH: Visual Summaries of Narrative Text Enhanced by Machine Learning. In Proceedings of the Text2StoryIR'19 Workshop, Cologne, Germany, published at http://ceur-ws.org

6. Yeshiwas G. and Abebe A. (2018). Deep learning approach for amharic sentiment analysis university of gondar. Retrieved from: https://www.researchgate.net/publication/331673959

7. Jaspreet S., Gurvinder S. and Rajinder S. (2017). Optimization of sentiment analysis using machine learning classifers. *Human-Centric Computing & Information Sciences*, 7(1), 32. DOI 10.1186/s13673-017-0116-3. Retrieved from: https://link.springer.com/content/pdf/10.1186%2Fs13673-017-0116-3.pdf.

8. Samuel P. (2018). Sentiment analysis: Machine-Learning approach. Data Driven Investor. Retrieved from https://medium.com/datadriveninvestor/sentiment-analysis-machine-learning-approach-83e4ba38b57

9. Munir A., Shabib A., Muhammad S.B., Noureen H., Iftikhar A. and Zahid N. (2018). SVM Optimization for Sentiment Analysis. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 9(4), Retrieved from https://pdfs.semanticscholar.org/3c9d/9cc4c4aad89ec00961efd76ec92c9fcbd4d2.pdf

10. Ray P. and Chakrabarti A. (2019). A Mixed approach of Deep Learning method and Rule-Based method to improve Aspect Level Sentiment Analysis. *Applied Computing and Informatics*. Retrieved from https://doi.org/10.1016/j.aci.2019.02.002

11. Md S.A., Ayush K., Asif E. and Pushpak B. (2016). A Hybrid Deep Learning Architecture for Sentiment Analysis. Proceedings of COLING 2016. In the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 482-493, 11-17. Retrieved from https://www.aclweb.org/anthology/C16-1047.

12. Chakravarthy A., Desai P., Deshmukh S., Gawande S. and Saha I. (2018). Hybrid Architecture for Sentiment Analysis Using Deep Learning. *International Journal of Advanced Research in Computer Science*, 9(1), DOI: http://dx.doi.org/10.26483/ijarcs.v9i1.5388 Volume 9, No. 1. Available Online at www.ijarcs.info

13. Komalpreet K., Chitender K. and Tarandeep K.B. (2019). An optimized CNN based robust sentiment analysis system on big social data using text polarity feature. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, ISSN: 2278-3075, 8(6). https://www.ijitee.org/wp-content/uploads/papers/v8i6/F3928048619.pdf

14. Tianyang Z., Minlie H. and Li Z. (2017). Learning Structured Representation for Text Classification via Reinforcement Learning. Association for the Advancement of Artificial Intelligence https://www.microsoft.com/en-us/research/wp-content/uploads/2017/11/zhang.pdf

15. Apostol V. (2019). BowTie – A deep learning feedforward neural network for sentiment analysis. NIST Cybersecurity

White Paper. available from: https://doi.org/10.6028/NIST.CSWP.04222019

16. Juergen S. (2015). Deep learning in neural networks: *An overview. In Neural Networks*., 61, 85-117. Retrieved from https://arxiv.org/abs/1404.7828