Deep NLP Explainer: Using Prediction Slope to Explain NLP Models

Reza Marzban(✉) and Christopher Crick

Computer Science Department, Oklahoma State University, Stillwater, OK, USA
reza.marzban@okstate.edu

Abstract. Natural Language Processing models have been increasingly used for many tasks, from sentiment analysis to text summarization. Most of these models are reaching the performance of human experts. Unfortunately, not only are these models not intuitive to the end-user, but they are also not even interpretable to highly-skilled Machine Learning scientists. We need explainable artificial intelligence to be able to trust models in high-stakes scenarios, and also to develop insights to optimize them by removing existing limitations and biases. In this paper, we devise a new tool called "Prediction Slope" that can be applied to any NLP model, extracting the importance rate of the component words and thereby helping to explain the model. It uses the average effect each word has on the final prediction slope as the word importance rate. We compared our technique with preceding approaches and observed that although they perform similarly, the earlier approaches do not generalize as well. Our method is independent of the model's architecture and details.

Keywords: Natural language processing · Deep learning · Artificial neural networks · Explainable artificial intelligence · Transformers

1 Introduction

The rapid growth in the amount of data available has provided both opportunities and challenges. It has helped us to build and optimize new models like different architectures of deep learning models. On the other hand, much of the available data cannot be processed by traditional statistical models and machine learning algorithms. These algorithms are desirable when we have small to medium-sized formatted data in tables inserted by a domain expert, but they are not able to handle modern tasks that require analyzing unstructured data (e.g. texts, movies, pictures). Neural networks are not a new technology – perceptrons were invented in 1958 – but they were not terribly successful until recently, due to the small amount of available data and weak computing power. However, both of these conditions have changed recently, and deep artificial neural networks have become the superhero of each and every Artificial Intelligence (AI) task from Natural Language Processing (NLP) to voice recognition and machine vision.

I. Farkaš et al. (Eds.): ICANN 2021, LNCS 12892, pp. 447–458, 2021.
https://doi.org/10.1007/978-3-030-86340-1_36

Although we have seen a huge jump in deep learning models' performance, they are not without drawbacks. The most important is that they are not as intuitive as basic machine learning models like decision trees. These models are more like black boxes, in that we throw data at them, use the output, and hope for the best, but we do not understand how or why. If the user does not understand the logic behind a model's decision, it will cause distrust, especially in high-stakes situations like autonomous vehicles. As a result, recently researchers have made large efforts toward explainable artificial intelligence. Not only should this boost the users' trust, but interpretability also helps developers, ML experts, and data scientists learn the defects of their models, detect bias, and tune them for further improvement.

In the relevant literature, many papers have contributed toward making deep learning models interpretable and intuitive, although they have mostly concentrated on image processing problems, as 2D pictures are much easier to visualize. They use a broad range of techniques like segmentation or creating heatmaps and saliency maps to highlight the pixels that are critical to a specific final model decision. Such approaches help users to understand the logic behind each decision, as humans can digest 2-dimensional images and find patterns in them. On the other hand, there is a huge gap in the literature on explainable AI in other contexts like NLP. Natural Language Processing is the science of enabling machines to communicate (understand and generate) in human languages. Textual data that is consisted of sentences, words, and letters are very hard to visualize, especially in a 2d space where humans can find patterns, even though accessible visualization is a key component of explainable AI.

Deep learning models come in various flavors with different architectures. Convolutional neural networks (CNNs) were originally designed for image classification but can be applied to other types of data like texts. Recurrent neural networks (RNNs) are assumed to be a natural choice for time-series data; Long Short Term Memory (LSTMs) and Gated Recurrent Units (GRUs) are common types of RNN. LSTMs are believed to be one of the most effective options in NLP tasks, as they are constructed with time series in mind. Each word or token can be looked at as a time step in a sentence. CNNs (1-dimensional versions) can also be used on textual data. Their performance is comparable with LSTM on well-known textual benchmarks for various tasks like sentiment analysis. They are also much faster than LSTMs.

Our contribution in this paper creates a brand new explainable AI technique that can be applied to any type of NLP model. Our technique uses a model's inner logic to come up with an importance rate for each and every unique word in the corpus. This has many benefits: we can take a look at the model's most important words to understand its overall general logic. It also can be used to inject insights into future models for further performance improvements. We observed that using our technique, models that were trained on just the 5% most important words perform equally as well as baseline models that have access to 100% of data. However, because only a small fraction of words are used, the model's speed is much greater. In order to create an importance rate for all words, we use and compare the mean significance of the effect of each unique word to the overall sentence prediction. In other words, we compare the change in the prediction of all sentences that contain a specific word with and without that word and use the average prediction change throughout all sentences in the corpus to create an importance rate.

Previously there has been some related work on finding and targeting the most important words to a model, but they mainly suffer from a couple of disadvantages. They provide the most important words locally, in a single output to a specific decision, which is useful but does not help in understanding the logic of the model in general. In addition, the techniques that are used for extracting the most important words are highly dependent on the architecture of the model and are thus limited to specific types of NLP models (e.g. CNNs). However, our technique can be applied to all types of NLP models and provides general explainability for the overall model.

2 Related Work

If we want users to understand and trust deep learning models we should provide justifications along with predictions. Explainable artificial intelligence (XAI) [10] attempts to address this problem, as well as helping data scientists to find the models' weaknesses, biases, and blind spots and thereby improve them.

XAI enables models to explain themselves to satisfy non-technical users [9], and helps developers to justify and improve them. XAI approaches can have various flavors [1]; they can provide local explanations of each and every prediction or globally explain the logic of the model as a whole. Layer-wise relevance propagation (LRP) [3,21] matches each prediction in the model to the input features that have a significant effect on the prediction. LIME [26] is a technique for providing local interpretable model-agnostic explanations. These tools and techniques help us to trust deep learning models.

Almost all deep learning researchers working toward XAI have concentrated on image processing and machine vision, as humans find it easy to understand and find patterns in visual data. This research has created heat maps, saliency maps [27] and attention networks [29]. However, other artificial intelligence fields, such as NLP, have seen far fewer research efforts. NLP has made many significant improvements in model performance on various types of tasks and data in recent years [7], but very few of them concentrate on creating self-explanatory models.

Arras [2] identified the words that support or contradict a specific classification using LRP, highlighting them to create a visual aid for the user to understand the reasoning behind each model's decisions and predictions. This can help identify when a model arrives at a correct prediction through incorrect logic or bias, and provide clues toward fixing such errors. This technique is local, which helps to confirm single model predictions, but in order to improve and optimize models, we need tools for understanding their global logic.

RNNs and LSTMs [11] are efficient architectures for NLP tasks and textual data; however, 1-dimensional CNNs are also used for common NLP tasks like sentence classification [13] and modeling [12]. Le [14] demonstrates how CNN depth affects performance in sentiment analysis. Yin [32] compares RNN and CNN performance on various NLP tasks. Wood [30] proves that CNNs might outperform RNNs on textual data, in addition to being faster.

In 2017, a new generation of NLP models appeared, starting with Vaswani's first Transformer attention-based architecture [28]. Instead of remembering an entire text, it assigns an attention weight to each token, which allows it to process much longer texts. The attention technique enabled the creation of much more advanced transformer-based models like BERT [8], RoBERTa [16], and GPT-3 [5]. All of these models have tried to overcome their predecessor models' limitations. Some researchers have changed the inner architecture of these models and others have created auto-encoders to overcome the sequence length limitation of these models [19], adding a custom encoder layer to compress the input so that models like LSTM and BERT can accept and process longer texts.

Many researchers have tried to interpret and visualize CNN models, often on famous visual object recognition databases and benchmarks like ImageNet [34]. There are four basic techniques to visualize models in image processing tasks: activation maximization, network inversion, deconvolutional neural networks, and network dissection [23]. Yosinski [33] has devised tools to visualize features of a CNN model at each layer in image space. Model explanation, visualization, and interpretation for other types of data, such as text, are nowhere near as well-developed, but there have been a few attempts. Choi [6] attempted to explain a CNN model that classifies genres of music, and showed that deeper layers capture textures. Xu [31] used attention-based models to describe the contents of images in natural language, showing saliency relationships between image contents and word generation.

One of the hardest challenges in NLP is visualizing data after tokenizing textual data with available tools like NLTK [4]. Each token or word is represented by an embedding [17,20,25]. An embedding is a vector of numbers that represent a word's semantic relationship to other words. Pre-trained embeddings like GloVe [22] are available that are trained on a huge corpus. However, they are not understandable by humans, and it is very challenging to explain models that use them. Li [15] created methods to illustrate the saliency of word embeddings and their contribution to the overall model's comprehension. Rajwadi [24] trained a 1-dimensional CNN for a sentiment analysis task and used a deconvolution method to explain text classification. They estimate the importance of each word to the overall decision by masking it and checking its effect on the final prediction score.

Activation Maximization (AM) is a technique that can be applied on CNN models trained on textual data; some research has focused on creating an importance rate for each unique word in a corpus using AM on CNNs by analyzing the convolution filter weights [18]. However, instead of creating a local explanation for each prediction and decision, they used this technique to describe the whole model's logic and tried to explain it in a layer-wise manner by studying the filters of the trained model. They used the IMDb dataset [17] as their benchmark. It is very useful as their result is not dependent on every prediction, but provides a general justification for overall model logic. However, it is limited to CNN models, while we need a technique that is independent of the model architecture.

In this paper, we created a brand new tool to generate a word importance rate for an entire model, for all unique words in a corpus. This is similar to previous tools, except that this new technique is independent of the models' inner details and architecture. In other words, it can be applied to any type of NLP model.

3 Technical Description

3.1 Dataset Introduction and Preprocessing

In this research, we used two benchmark datasets with different tasks. The first is the IMDb review dataset [17],[1] which contains movie reviews and a binary target value (no neutral reviews are included). The task of this benchmark is sentiment analysis, one of the basic but crucial NLP tasks. The second dataset is the Stack Overflow dataset[2], in which each question is tagged with one of 20 possible tags. In other words, it is a multinomial classification. Obviously, the first task is easier for models as it contains only two classes.

Both of these benchmark datasets were preprocessed by removing all stop-words, special characters, numbers, HTML tags, and hapax legomena (words that appear only once in an entire corpus). All characters were converted to lower case. We used NLTK [4] to tokenize the reviews. Word2Vec [25] was used to generate 100-dimensional embeddings for each word. In the final results, our IMDb dataset had around 43,000 documents and 23,000 unique words while the Stack Overflow dataset had around 40,000 documents and 28,000 unique words. Our final step was splitting them into training and test sets.

3.2 Overview of the Latest Importance Rate (Activation Maximization)

Most of the work on XAI in NLP fields concentrates on providing local interpretability or justifying each and every prediction for all inputs. In contrast, we need a global explainability technique or tool to understand the overall logic of a model. Recently, some research has tried to handle that issue by identifying the most important words to the whole model [18].

They used a 1-d CNN model and activation maximization to create an importance rate with Eq. 1. They used this technique to inject insights into newer models. They proved that new models that use only a tiny fraction of the most

[1] https://ai.stanford.edu/~amaas/data/sentiment/.
[2] https://console.cloud.google.com/marketplace/product/stack-exchange/stack-overflow.

important words (extracted with the help of their equation) result in no significant accuracy change and dramatic increase in speed.

$$importance = \left\{ \sum_{f=1}^{F} \sum_{s=1}^{S} \sum_{i=1}^{I} |w_i * Filter_{f*s*i}| \, | w \in Corpus, Filter \right\} \tag{1}$$

In Eq. 1, F is the number of filters in the CNN layer, S is the size of the filters, and I is the embedding length. w is a word embedding vector with a length of I. Corpus is a matrix of the entire word embedding of size $m * I$, in which m is the count of unique words in our corpus dictionary. Filter is a 3-D tensor of size $F * S * I$. This equation calculates the sum of activations of all filters caused by a single word from the Corpus.

While this technique (which will be referred to as **"activation maximization"** in this paper) is innovative, providing as it does a global interpretability rather than a local justification, it has one main limitation. The equation is highly dependent on CNN filters, and it can only be used on CNN models trained on textual data. We need similar tools that can be applied to any type of model inner architecture. In this paper, our contribution is to create a brand new technique for choosing the most important words, that is independent of the NLP model architecture.

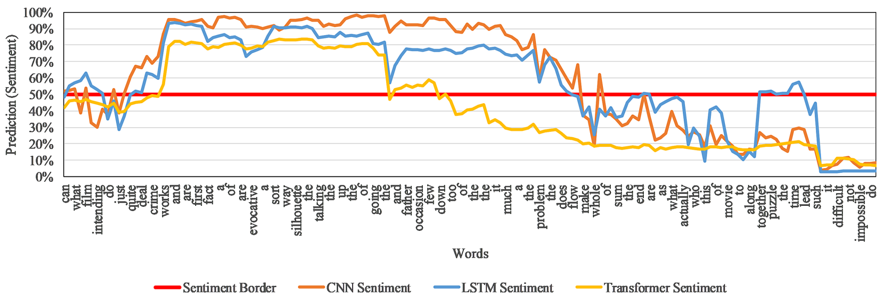


Fig. 1. Effect of each word in an IMDb document on the binary prediction of 3 different models (CNN, LSTM, and Transformer). Predictions above 50% represent positive sentiment and below 50% represent negative sentiment.

3.3 Introduction of Prediction Slope

In order to create an importance rate that is independent of the model's inner architecture and details, we created the concept of the prediction slope, which will be defined and clarified in this section. In all machine learning models, and specifically in Artificial Neural Networks (ANNs), there is a final prediction (a.k.a. output layer), where the model's decisions are found. In NLP our inputs consist of words or tokens. To understand the significance of each of these words on the final prediction, we can feed them one by one into our model and observe the effect of each word on the final prediction.

In Fig. 1, an IMDb document with negative sentiment is randomly chosen to visualize the prediction slope in a binary classification problem. In a multiclass task like the Stack Overflow dataset, we would have 20 predictions due to the fact that there are 20 classes, and the highest probability output is taken as the model's decision. We observe this maximum probability class and the effect of adding new words.

$$S_i = F(x_0, x_1, x_2, ..., x_{i-1}, x_i) - F(x_0, x_1, x_2, ..., x_{i-1}) \tag{2}$$

In Eq. 2, S_i is the prediction slope of the i^{th} word in a document, and F is simply the function defined by the model, which receives a sentence as input and produces a prediction. x_n is the n^{th}word in a document.

3.4 Extracting Word Importance Rate from the Prediction Slope

The prediction slope technique is not entirely new; it has been used in the literature to map a prediction to the most important words in a single input, and is sometimes also known as the temporal score. However, until now it has not been considered as a tool to perform a similar technique to an entire model, and this is where our contribution comes into play.

In each document of our corpus, we can monitor the local significance and effect of each word on the final prediction slope, but we needed a way to find the global importance rate of each unique word to the whole model. In order to do so, we use Eq. 3, in which S_i is extracted from Eq. 2, D_j is all documents in our corpus that contain the j^{th} unique word, and $|D_j|$ is the count of those documents. Notice that the j^{th} unique word in our corpus is the i^{th} word that appears in a document. After applying this equation, we will a global importance rate for all unique words applicable to the whole model rather than just a single prediction. This importance rate is the mean value of the prediction slope of a particular word in all sentences containing it.

$$importance_j = \frac{\sum_{D_j} S_i}{|D_j|} \tag{3}$$

3.5 Comparing Importance Rates

The prediction slope importance rate technique can be applied to any type of model, but we chose to use it on a basic Transformer model, as it is one of the hardest models to interpret and understand. Now that we have two importance rates at hand, one generated by activation maximization and the other created by prediction slope, we performed experiments to compare them. As a result, we created several brand new models that were trained on a subset of the unique words which were selected by one of our two importance rates, and we examined their respective performances.

4 Experimental Results

In order to test our hypothesis on both of our datasets, and compare the performance of each of the two importance rates extracted, we designed new models that were just trained on the most important words based on three different algorithms: **Activation Maximization, Prediction Slope, Random**. In the random technique, words are chosen randomly as a naive baseline for comparison with our two other models. We also compare them against the **Base Model** that uses all 100% of the words. The final model is called **Hybrid**, and it averages the importance rates for each word generated by each of the two techniques.

We created a threshold that identified the percentage of the most important words that our models would train on. We tested different threshold values: 10%, 5%, 2%, 1%, 0.5%.

4.1 Comparing Importance Rates on the IMDb Dataset

In our IMDb dataset, as it is a sentiment analysis problem with a binary target value, the prediction accuracy starts from 50% and the baseline accuracy with access to all words was 84%. Results are shown in Fig. 2.

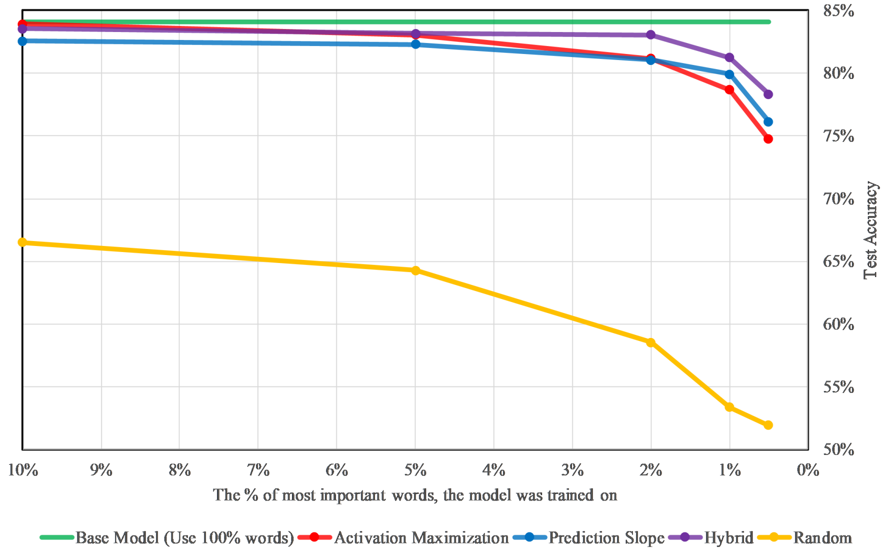


Fig. 2. Comparison of different importance rate techniques on IMDb dataset.

Both models are interchangeable, with no significant difference in their performance. In addition, both perform superbly while using just 2% of the data. Accuracies are very similar to the base model while they are much faster. The random model performs poorly as expected.

4.2 Comparing Importance Rates on the Stack Overflow Dataset

The Stack Overflow dataset presents a multinomial task with 20 possible classes or tags. Accuracy, therefore, starts at 5%, and the accuracy of the base model with access to all words is 74%.

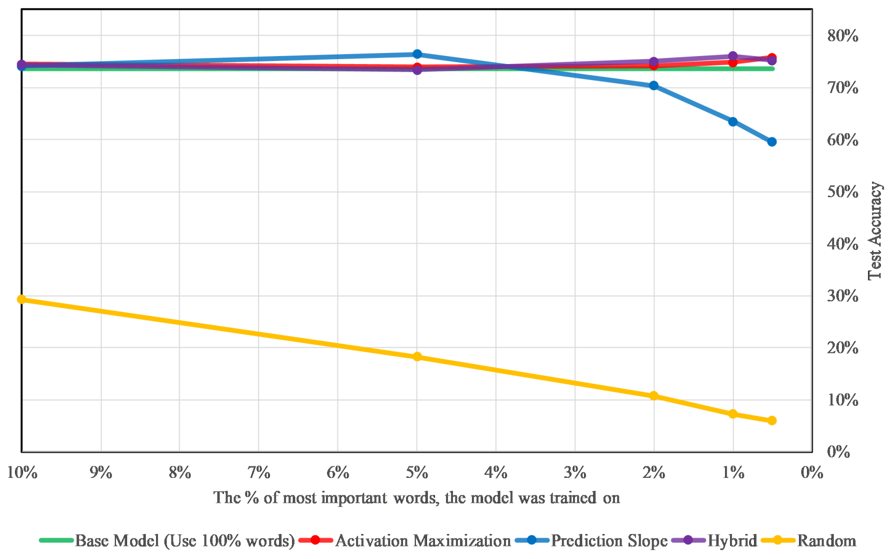
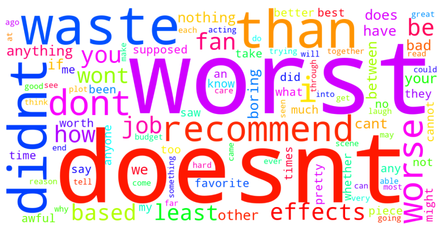


Fig. 3. Comparison of different importance rate techniques on Stack Overflow dataset.

Figure 3 shows that both models are still very similar, although the prediction slope technique has lower performance at thresholds smaller than 2%. The activation maximization model, however, has very good performance – even slightly better than the baseline model – even when training on only 0.5% of words. The model focuses on critical keywords, and it turns out that in this task, they are extremely predictive. Both models still perform well overall, even when they have access to a small subset of the data. Again, they are much faster than the baseline model. Also as in the previous experiment, the random model performs weakly as expected.

4.3 Analysis of the Result

It was observed that both techniques have quite similar performance (They performed equally in the IMDb dataset, Activation Maximization was slightly better in the Stack Overflow dataset), and both are much faster than the base model as they are just using a small subset of the input. However, the Activation Maximization can just be applied on CNN models and is very dependent on the model architecture, while the Prediction Slope can be applied on any type of NLP model architecture (In our case it was applied on a transformer model). This is very beneficial to have a tool that can analyze the model independent of the inner architecture or details and gives us insights from the model's global logic.


(a) Activation Maximization

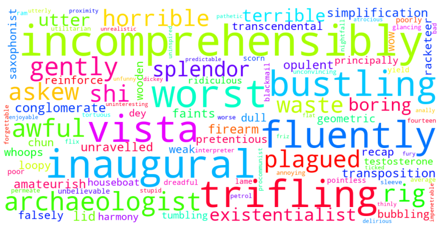

(b) Prediction Slope

Fig. 4. Wordcloud of the top 100 most important words in IMDb dataset

These techniques can be used to generate insights to improve future models, and we can also use them to visualize the most important words to a model to make it more explainable and understandable. Figure 4 shows the top 100 most important words extracted from both techniques in the IMDb dataset. The size of the word represents its importance rate. The most important words according to the activation maximization technique exhibited higher document frequencies.

5 Conclusion

Now that many AI models, for many tasks, have reached acceptable performance levels, and often even surpass human experts, it is time to focus on other aspects of machine learning models beyond their raw accuracies. Machine learning models raise many challenging questions about AI fairness, ethical issues, and biases. In order to answer all these questions, we need to develop infrastructure and tools that make our models explainable in order to justify their decisions. Our contribution in this paper was to generate a new method for explaining NLP models' logic. Our experiments show that our method is as accurate as previous ones, while it is much more generalized. Our technique is not dependent on the NLP architecture and type and can be applied to any NLP model and task.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
2. Arras, L., Horn, F., Montavon, G., Müller, K.R., Samek, W.: "What is relevant in a text document?": an interpretable machine learning approach. PLoS ONE **12**(8), e0181142 (2017)

3. Bach, S., et al.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**(7), e0130140 (2015)
4. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media Inc., Sebastopol (2009)
5. Brown, T.B., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
6. Choi, K., Fazekas, G., Sandler, M.: Explaining deep convolutional neural networks on music classification. arXiv preprint arXiv:1607.02444 (2016)
7. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Du, M., Liu, N., Hu, X.: Techniques for interpretable machine learning. Commun. ACM **63**(1), 68–77 (2019)
10. Gunning, D.: Explainable artificial intelligence (XAI). Defense Advanced Research Projects Agency (DARPA), nd Web 2 (2017)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
13. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
14. Le, H.T., Cerisara, C., Denis, A.: Do convolutional networks need to be deep for text classification? In: Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence (2018)
15. Li, J., Chen, X., Hovy, E., Jurafsky, D.: Visualizing and understanding neural models in NLP. arXiv preprint arXiv:1506.01066 (2015)
16. Liu, Y., et al.: Roberta: a robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
17. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 142–150. Association for Computational Linguistics (2011)
18. Marzban, R., Crick., C.: Interpreting convolutional networks trained on textual data. In: Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods, ICPRAM, vol. 1, pp. 196–203. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010205901960203
19. Marzban, R., Crick., C.: Lifting sequence length limitations of NLP models using autoencoders. In: Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods, ICPRAM, vol. 1, pp. 228–235. INSTICC, SciTePress (2021). https://doi.org/10.5220/0010239502280235
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
21. Montavon, G., Samek, W., Müller, K.R.: Methods for interpreting and understanding deep neural networks. Digit. Signal Process. **73**, 1–15 (2018)
22. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)

23. Qin, Z., Yu, F., Liu, C., Chen, X.: How convolutional neural network see the world-a survey of convolutional neural network visualization methods. arXiv preprint arXiv:1804.11191 (2018)
24. Rajwadi, M., Glackin, C., Wall, J., Chollet, G., Cannings, N.: Explaining sentiment classification. In: Interspeech 2019, pp. 56–60 (2019)
25. Rehurek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Citeseer (2010)
26. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144 (2016)
27. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
28. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
29. Wang, F., et al.: Residual attention network for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3156–3164 (2017)
30. Wood-Doughty, Z., Andrews, N., Dredze, M.: Convolutions are all you need (for classifying character sequences). In: Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text, pp. 208–213 (2018)
31. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: International Conference on Machine Learning, pp. 2048–2057 (2015)
32. Yin, W., Kann, K., Yu, M., Schütze, H.: Comparative study of CNN and RNN for natural language processing. arXiv preprint arXiv:1702.01923 (2017)
33. Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579 (2015)
34. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53