An Integrative Approach to Visualizing Recurrent Neural Networks

Gonçalo Lopes
goncalo.chincho.lopes@ist.utl.pt

Abstract

With the growth of computational power and data availability, deep neural networks have been successfully utilized for learning complex patterns in large amounts of data. There has been a corresponding need to understand these models and be able to explain their decisions, in consequence, building trust about how they will behave in a real-world scenario. Natural Language Processing is one of the fields where such models have shown success, but interpreting them is still an open problem. We explore the adaptation of a technique that has seen success in computer vision tasks, Activation Maximization, to the task of text classification, with the intent to obtain insights on the inner workings of the deep network, together with Feature Importance and exploration of Dataset Examples.

1 Introduction

Word Embeddings are the most widely used method to represent words in a Natural Language Processing tasks. Significant progress has been made in this domain, particularly when these representations are utilized in conjunction with the complex Deep Neural Networks (DNNs). These DNNs are capable of calculating inner representations of a task and from those representations obtain state-of-the-art results in a multitude of problems. However, these inner representations consist in high dimensional matrices, which, to humans, are difficult to understand. Thus, one paradigm of interpretability, Feature Visualization, focuses on the study of techniques that can transform these matrices into concepts understandable by humans.

A characteristic of most existing explanation producing methods is that they tend to work in isolation. The goal is to integrate different types of explanations to obtain different explanations for the network's output.

We aim to answer the following questions:

- **Q1 - How did certain words influence the network's decision?** The answer to this question must not only answer if a word had a negative or positive impact, but must also explain how the words were utilized throughout the network to form a decision, an explanation not provided by current attribution methods.
- **Q2 - What is the network looking for in the input?** We would like to know what do the network's neurons fire to, what kind of hidden representations were formed after training.
- **Q3 - Which training set examples are most similar with the current input?** With the aid of dataset examples, users can obtain insight on *where* or *if* the network has seen a similar instance to the current input during its training phase.

Our work is focused on text processing tasks with RNN architectures, namely LSTMs.

2 Related Work

Work in interpreting Deep Neural Networks has been rapidly expanding over the last few years. We propose the following taxonomy for the current state of the art explanation producing methods:

- **Attribution** methods assign an importance to the input's features. This importance is commnly obtained by reducing the complexity of the inner workings of a Neural Network by approximating the relations that occur between the layers with a simpler, interpretable model. Examples of such methods are (Ribeiro et al., 2016), (Ribeiro et al., 2018), (Zilke et al., 2016), (Murdoch and Szlam, 2017) and (Murdoch et al., 2018).

- **Feature Visualization** methods analyze the roles of layers and units as data flows through the network. Examples of these methods are (Erhan et al., 2009) and (Strobelt et al., 2018).

- **Dataset Examples** methods utilize examples of the training set to explain the model's decisions. An example of the application of this approach is (Papernot and McDaniel, 2018).

3 Activation Maximization

In order to answer question **Q2 - What is the network looking for in the input?**, we try to synthesize an optimal input, which when fed as input to the model, results in a high activation value, and consequently,as very positive sentiment.

3.1 Gradient Ascent

Activation Maximization (Erhan et al., 2009), aims to obtain an input x^* from the d-Dimensional space X that maximizes the value of a chosen network parameter, in the case of RNNs we choose the activation value $h_t(\theta, x)$ at a certain time step t.

$$x^* = \underset{X}{\operatorname{argmax}}\, h_t(\theta, x) \tag{1}$$

The result is often seen as a representation of a concept that the model activates to the most, which aids users in model interpretation.

However, the usage of this technique as been mostly limited to image processing tasks, where the optimized input is an image that, although it doesn't represent something real, represents visual concepts such as edges, or shapes. In NLP the challenge is due to the discrete nature of words, a word embedding that optimizes a model parameter doesn't correspond to a word in the vocabulary, and as such the high dimensional vectors do not directly translate to an understandable concept.

3.2 Corpus Search

Due to the discrete property of textual data, another approach is to iterate the vocabulary and directly find words that maximize the network's output. Compared to Gradient Descent, this approach is computationally more complex, as it requires performing a forward pass on each word of the vocabulary. We aim to compare the output to these two approaches to input maximization.

4 Training Set Search

To complement the optimized output obtained from activation maximization with examples from the training set, we test two ways of comparing inputs:

- **Activation Similarity** compares an arbitrary input's activation values with the activations obtained from the examples of the training set, and selecting examples with most similar activation values. This approach is described in 4.1.

- **Token Search** looks for training set examples that contain a given token.This approach is described in 4.2.

4.1 Activation Similarity

To test this approach, we run each of the examples of the training set through the model, and store the respective activation values for each example. Given an input sentence, we can then choose which part of the sentence should be compared to the training set, and iterate the activation value collection to find the most similar sentence segments.

To compare activation vectors, we calculate the norm of the obtained difference between the two vectors.

4.2 Token Search

For Token Search, we run through the dataset's sentence inputs and select sentences that contain the same tokens as the arbitrary input tokens. For each selected training set sentences, we compile the association model predictions in order to compare how these words got classified in the training set with how the word was classified in the given example.

5 Experiments

5.1 Data and setup

For this work, we the sentiment analysis IMDB reviews dataset. This dataset contains 50,000 labeled highly polar movie reviews. The classification tasks consists in predicting a polarity label - whether a review holds a negative or a positive sentiment. For training and validation purposes, the dataset was split into 17,500 examples for training, 7,500 examples for validation, and 25,000 examples for testing.

The data were imported utilizing the *torchtext* package, which already contains a predefined train and test split for this dataset, along with an *IMDB* class that encapsulates different methods and attributes that facilitated working with the dataset, such as example iteration, and data structures to store the examples.

For this work, we the sentiment analysis IMDB reviews dataset. This dataset contains 50,000 labeled highly polar movie reviews. The classification tasks consists in predicting a polarity label - whether a review holds a negative or a positive sentiment. For training and validation purposes, the dataset was split into 17,500 examples for training, 7,500 examples for validation, and 25,000 examples for testing.

The data were imported utilizing the *torchtext* package, which already contains a predefined train and test split for this dataset, along with an *IMDB* class that encapsulates different methods and attributes that facilitated working with the dataset, such as example iteration, and data structures to store the examples.

Table 5.1 contains the accuracy on the training, validation and test sets for the LSTM model.

Model	*TrainAcc*	*ValAcc*	*TestAcc*
LSTM	93.21%	84.98%	84.64%

Table 1: Accuracy on the training, validation and test sets for the LSTM model.

5.2 Feature Importance

To study which words that had the most impact in the network's decision and answer question **Q1 - How did certain words influence thenetwork's decision?**, we start by obtaining word importance through the usage of LIME (Ribeiro et al., 2016). To study the application of the studied explanation methods, two different input reviews were selected from the movie review website *www.rottentomatoes.com.* The first review consists of a positive review, that the model correctly classifies as being positive with a confidence of 98%, and the second review is also positive, however, the LSTM model classifies this review as negative with a confidence of 84%. Figures 1 and 2 contain the reviews and the respective explanation obtained by applying the LIME Python package.[1].

5.3 Activation Maximization

The results for applying activation maximization with no previous hidden vectors as input is shown in figure 3. This optimized input vector has an associated confidence for the "positive" class of 98%, which shows that the optimization was successful. The 10 most similar tokens correspond to word embeddings with a similar cosine similarity to the optimized word vector. We can verify that 4 of these tokens correspond to scores explicitly stated in the review, which, in sentiment analysis are an objective indicator of the review's sentiment. An important observation is that not only good scores are present in the similar words, but also bad scores, such as "3/10". This can be explained by similarity function used, cosine similarity. This similarity compares word embedding angles, and ignores vector norms, which conceptually translates into similar words with shorter cosine distances are those that are conceptually similar, which is the case with different scores, be them positive or negative. This thus shows that the model in question responds with high activation values when scores are in the input sentence.

Comparing the Activation Maximization results with those from Corpus Search in figure 4, we confirm that the words in the vocabulary that maximize the activation values are the tokens associated with scores. This means we obtain the same results with both methods, but with significantly less computational complexity with AM.

[1]https://github.com/marcotcr/lime

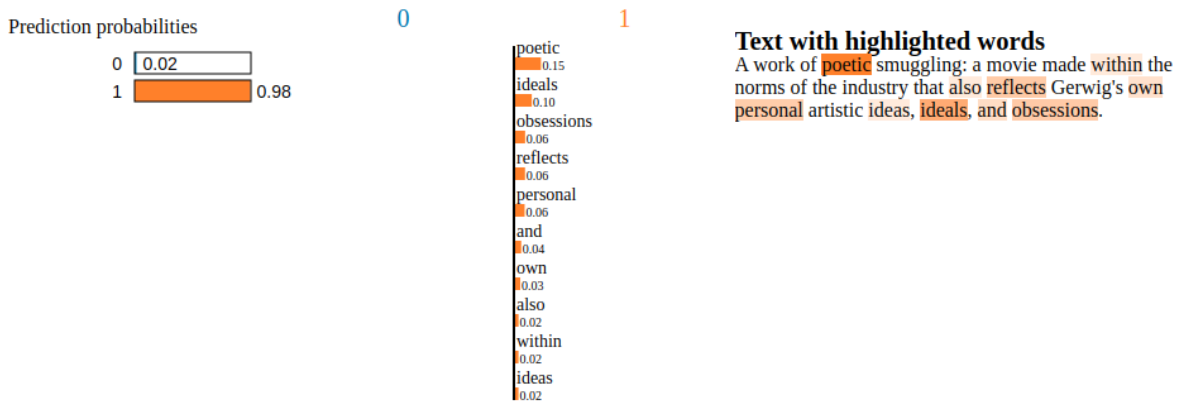

Figure 1: LIME explanation for the first review. Class probability distribution is shown on the left, features importance in the middle, and highlighted words and the review on the right.

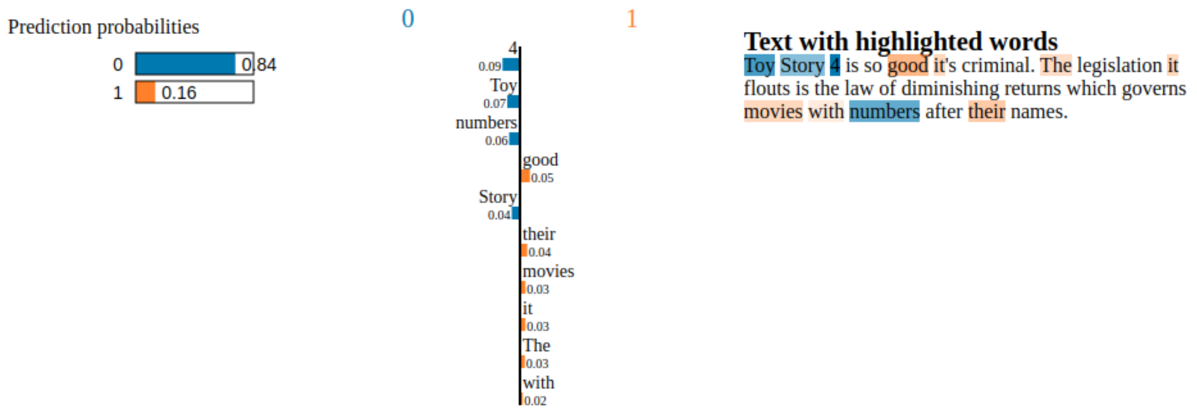

Figure 2: LIME explanation for the second review. Class probability distribution is shown on the left, features importance in the middle, and highlighted words and the review on the right.

```
['premise.<br',
 '6/10',
 '3/10',
 'friggin',
 'EVER',
 '7/10',
 'Ever',
 'scariness',
 'Surreal',
 '5/10'],
```

Figure 3: Activation Maximization results with no previous hidden vector.

```
['Treat',
 'McBain',
 '10/10',
 '9/10',
 '/>9/10',
 'being.<br',
 '/>10/10',
 '/>13',
 '7/10',
 '8/10',
 'DW'],
```

Figure 4: Corpus results with no previous hidden vector.

5.4 Dataset Examples

To find similar examples in the training set, we applied the two approaches describes in section 4.

For Activation Similarity, we obtained the activations for the tokens "poetic" for the first review, and "Toy Story" for the second review, and searched the training examples for similar activation values. We split the training set examples into bigrams and calculated the activation for each. The most similar bigrams in the training set for the first and second reviews are shown in tables 2 and 3 respectively.

Token	Sentence Score
Dog Days	98%
true hero	97%
Tigerland follows	97%

Table 2: Top 3 bigrams and the model's confidence for the positive class for the bigram's respective sentences for the first example sentence.

Token	Sentence Score
In 1988	3%
In an	9%
Awful Movie	2%

Table 3: Top 3 bigrams and the model's confidence for the positive class for the bigram's respective sentences for the second example sentence.

The obtained bigrams consist of the first two tokens of training set sentences with a very positive prediction for the first review a for very negative training set sentences for the second review. The results show that if a similarity is performed by activations, word similarity is ignored, and the returned bigrams only show examples of positive or negative examples.

For token search, the results are shown in figures 5 and 6. For the first review, the distribution of the resulting predictions of the training sets show a greater number of positive training set examples with the word "$poetic$", while for the second review, "Toy" and "$Story$" have a more balanced number of negative and positive examples. Comparing the scores for both reviews' tokens, "$ToyStory$" contain a higher percentage of negative examples, however it still contains more positive examples, and doesn't truly justify the very negative scores for these two tokens.

6 Conclusions

We now address the proposed goals, if these were met, and how:

- **Q1 - How did certain words influence the network's decision?** - The answer to this question was obtained through the usage of LIME, and as such the importance of the input words was obtained, but with no information on how this importance oscillated throughout the forward pass calculations. For the correctly classified example, expected word importances were obtained, where positive adjectives held the highest importance values. For the misclassified example, a movie title itself showed to be the main reason for the misclassification, which is a situation that illustrates how the utilization of attribution methods is not always enough to the understanding of a model's decisions, and as such can benefit from a deeper analysis.

- **Q2 - What is the network looking for in the input?** - For the problem of sentiment analysis in the IMDB dataset, we found through Activation Maximization that tokens that represent movie ratings induce the highest possible activations on the last layer of utilized LSTM. These results were further sustained by the application of brute force approach, Corpus Search, which resulted in conceptually the same kind of tokens (movie raings), but with an much higher computational complexity. The results of both these methods showed that the model is capable of correctly utilizing objective tokens with no ambiguity when it comes to sentimental polarity to sustain it's decisions, a complementary analysis when understanding the model as a whole.

- **Q3 - Which training set examples are most similar with the current input?** - The obtained dataset examples did not explain the anomalous behavior of the misclassified example, as the distribution of the reviews containing the most important tokens had the opposite behavior than what was expected, i.e., the bigram *Toy Story* was present in a significantly greater number of positiv ereviews in the training set than negative ones. The comparison by activation vectors did not result in useful outputs aswell, possibly due to the comparison method used.

7 Future Work

Each of the three main approaches can be augmented by exploration of different methods in the area, such as the utilization of anchors (Ribeiro et al., 2018) for feature importance and (Papernot and McDaniel, 2018) for the search of dataset examples, together with the work on different neural network architectures and different word embeddings, such as BERT (Devlin et al., 2019). An interactive visualization that integrative these components may also prove to be a step towards creation of commonly used tools for the generation of model explanations. This tool could include the work described in this document, together as the improvements stated in this section, while providing visual aids of the propagation of the inputs through the network nodes and layers, having as a base-line the work of LSTMVis [25], mainly augmenting it with the network-specific artifacts, attribution and optimization techniques.

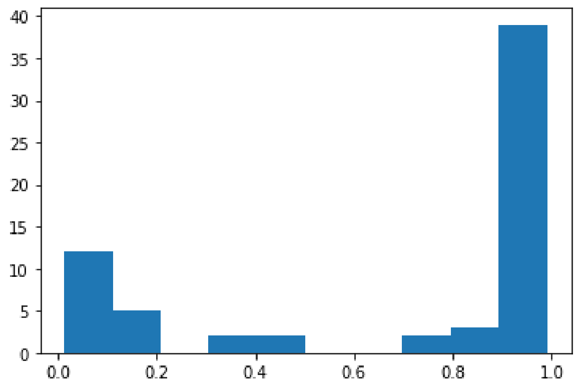
Figure 5: Distribution of the activation values for the training set examples that contain the token *poetic*.

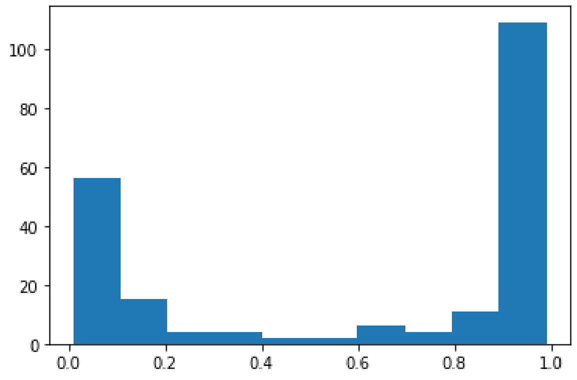
Figure 6: Distribution of the activation values for the training set examples that contain the bigram *Toy Story*.

References

M. T. Ribeiro, S. Singh, and C. Guestrin. "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, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence*, 2018.

J. R. Zilke, E. L. Mencía, and F. Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pages 457–473. Springer, 2016.

W. J. Murdoch and A. Szlam. Automatic rule extraction from long short term memory networks. *arXiv preprint arXiv:1702.02540*, 2017.

W. J. Murdoch, P. J. Liu, and B. Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *arXiv preprint arXiv:1801.05453*, 2018.

D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE* *transactions on visualization and computer graphics*, 24(1):667–676, 2018.

N. Papernot and P. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.