Explaining Predictions of Non-Linear Classifiers in NLP

Leila Arras[1], **Franziska Horn**[2], **Grégoire Montavon**[2],
Klaus-Robert Müller[2,3], **and Wojciech Samek**[1]
[1]Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany
[2]Machine Learning Group, Technische Universität Berlin, Berlin, Germany
[3]Department of Brain and Cognitive Engineering, Korea University, Seoul, Korea
{leila.arras, wojciech.samek}@hhi.fraunhofer.de
klaus-robert.mueller@tu-berlin.de

Abstract

Layer-wise relevance propagation (LRP) is a recently proposed technique for explaining predictions of complex non-linear classifiers in terms of input variables. In this paper, we apply LRP for the first time to natural language processing (NLP). More precisely, we use it to explain the predictions of a convolutional neural network (CNN) trained on a topic categorization task. Our analysis highlights which words are relevant for a specific prediction of the CNN. We compare our technique to standard sensitivity analysis, both qualitatively and quantitatively, using a "word deleting" perturbation experiment, a PCA analysis, and various visualizations. All experiments validate the suitability of LRP for explaining the CNN predictions, which is also in line with results reported in recent image classification studies.

1 Introduction

Following seminal work by Bengio et al. (2003) and Collobert et al. (2011), the use of deep learning models for natural language processing (NLP) applications received an increasing attention in recent years. In parallel, initiated by the computer vision domain, there is also a trend toward understanding deep learning models through visualization techniques (Erhan et al., 2010; Landecker et al., 2013; Zeiler and Fergus, 2014; Simonyan et al., 2014; Bach et al., 2015; Lapuschkin et al., 2016a) or through decision tree extraction (Krishnan et al., 1999). Most work dedicated to understanding neural network classifiers for NLP tasks (Denil et al., 2014; Li et al., 2015) use gradient-based approaches. Recently, a technique called layer-wise relevance propagation (LRP) (Bach et al., 2015) has been shown to produce more meaningful explanations in the context of image classifications (Samek et al., 2015). In this paper, we apply the same LRP technique to a NLP task, where a neural network maps a sequence of *word2vec* vectors representing a text document to its category, and evaluate whether similar benefits in terms of explanation quality are observed.

In the present work we contribute by (1) applying the LRP method to the NLP domain, (2) proposing a technique for quantitative evaluation of explanation methods for NLP classifiers, and (3) qualitatively and quantitatively comparing two different explanation methods, namely LRP and a gradient-based approach, on a topic categorization task using the *20Newsgroups* dataset.

2 Explaining Predictions of Classifiers

We consider the problem of explaining a prediction $f(\boldsymbol{x})$ associated to an input $\boldsymbol{x}$ by assigning to each input variable x_d a score R_d determining how relevant the input variable is for explaining the prediction. The scores can be pooled into groups of input variables (e.g. all *word2vec* dimensions of a word, or all components of a RGB pixel), such that they can be visualized as heatmaps of highlighted texts, or as images.

2.1 Layer-Wise Relevance Propagation

Layer-wise relevance propagation (Bach et al., 2015) is a newly introduced technique for obtaining these explanations. It can be applied to various machine learning classifiers such as deep convolutional neural networks. The LRP technique produces a *decomposition* of the function value $f(\boldsymbol{x})$ on its input variables, that satisfies the conservation property:

$$f(\boldsymbol{x}) = \textstyle\sum_d R_d. \qquad (1)$$

Proceedings of the 1st Workshop on Representation Learning for NLP, pages 1–7,
Berlin, Germany, August 11th, 2016.

The decomposition is obtained by performing a backward pass on the network, where for each neuron, the relevance associated with it is redistributed to its predecessors. Considering neurons mapping a set of n inputs $(x_i)_{i\in[1,n]}$ to the neuron activation x_j through the sequence of functions:

$$\begin{aligned} z_{ij} &= x_i w_{ij} + \tfrac{b_j}{n} \\ z_j &= \textstyle\sum_i z_{ij} \\ x_j &= g(z_j) \end{aligned}$$

where for convenience, the neuron bias b_j has been distributed equally to each input neuron, and where $g(\cdot)$ is a monotonously increasing activation function. Denoting by R_i and R_j the relevance associated with x_i and x_j, the relevance is redistributed from one layer to the other by defining messages $R_{i\leftarrow j}$ indicating how much relevance must be propagated from neuron x_j to its input neuron x_i in the lower layer. These messages are defined as:

$$R_{i\leftarrow j} = \frac{z_{ij} + \frac{s(z_j)}{n}}{\sum_i z_{ij} + s(z_j)} R_j$$

where $s(z_j) = \epsilon \cdot (1_{z_j \geq 0} - 1_{z_j < 0})$ is a stabilizing term that handles near-zero denominators, with ϵ set to 0.01. The intuition behind this local relevance redistribution formula is that each input x_i should be assigned relevance proportionally to its contribution in the forward pass, in a way that the relevance is preserved ($\sum_i R_{i\leftarrow j} = R_j$).

Each neuron in the lower layer receives relevance from all upper-level neurons to which it contributes

$$R_i = \textstyle\sum_j R_{i\leftarrow j}.$$

This pooling ensures layer-wise conservation: $\sum_i R_i = \sum_j R_j$. Finally, in a max-pooling layer, all relevance at the output of the layer is redistributed to the pooled neuron with maximum activation (i.e. winner-take-all). An implementation of LRP can be found in (Lapuschkin et al., 2016b) and downloaded from `www.heatmapping.org`[1].

2.2 Sensitivity Analysis

An alternative procedure called sensitivity analysis (SA) produces explanations by scoring input variables based on how they affect the decision output locally (Dimopoulos et al., 1995; Gevrey et al., 2003). The sensitivity of an input variable is given by its squared partial derivative:

$$R_d = \Big(\frac{\partial f}{\partial x_d}\Big)^2.$$

Here, we note that unlike LRP, sensitivity analysis does not preserve the function value $f(\boldsymbol{x})$, but the squared l_2-norm of the function gradient:

$$\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2^2 = \textstyle\sum_d R_d. \tag{2}$$

This quantity is however not directly related to the amount of evidence for the category to detect. Similar gradient-based analyses (Denil et al., 2014; Li et al., 2015) have been recently applied in the NLP domain, and were also used by Simonyan et al. (2014) in the context of image classification. While recent work uses different relevance definitions for a group of input variables (e.g. gradient magnitude in Denil et al. (2014) or *max*-norm of absolute value of simple derivatives in Simonyan et al. (2014)), in the present work (unless otherwise stated) we employ the squared l_2-norm of gradients allowing for decomposition of Eq. 2 as a *sum* over relevances of input variables.

3 Experiments

For the following experiments we use the *20newsbydate* version of the *20Newsgroups*[2] dataset consisting of 11314/7532 train/test documents evenly distributed among twenty fine-grained categories.

3.1 CNN Model

As a document classifier we employ a word-based CNN similar to Kim (2014) consisting of the following sequence of layers:

`Conv` → `ReLU` → `1-Max-Pool` → `FC`

By `1-Max-Pool` we denote a max-pooling layer where the pooling regions span the whole text length, as introduced in (Collobert et al., 2011). `Conv`, `ReLU` and `FC` denote the convolutional layer, rectified linear units activation and fully-connected linear layer. For building the CNN numerical input we concatenate horizontally 300-dimensional pre-trained *word2vec*[3] vectors (Mikolov et al., 2013), in the same order the corresponding words appear in the pre-processed

[1] Currently the available code is targeted on image data.

[2] `http://qwone.com/%7Ejason/20Newsgroups/`

[3] `GoogleNews-vectors-negative300, https://code.google.com/p/word2vec/`

document, and further keep this input representation fixed during training. The convolutional operation we apply in the first neural network layer is one-dimensional and along the text sequence direction (i.e. along the horizontal direction). The receptive field of the convolutional layer neurons spans the entire word embedding space in vertical direction, and covers two consecutive words in horizontal direction. The convolutional layer filter bank contains 800 filters.

3.2 Experimental Setup

As pre-processing we remove the document headers, tokenize the text with NLTK[4], filter out punctuation and numbers[5], and finally truncate each document to the first 400 tokens. We train the CNN by stochastic mini-batch gradient descent with momentum (with l_2-norm penalty and dropout). Our trained classifier achieves a classification accuracy of 80.19%[6].

Due to our input representation, applying LRP or SA to our neural classifier yields one relevance value per word-embedding dimension. From these single input variable relevances to obtain word-level relevances, we sum up the relevances over the word embedding space in case of LRP, and (unless otherwise stated) take the squared l_2-norm of the corresponding word gradient in case of SA. More precisely, given an input document d consisting of a sequence $(w_1, w_2, ..., w_N)$ of N words, each word being represented by a D-dimensional word embedding, we compute the relevance $R(w_t)$ of the tth word in the input document, through the summation:

$$R(w_t) = \sum_{i=1}^{D} R_{i,t} \tag{3}$$

where $R_{i,t}$ denotes the relevance of the input variable corresponding to the ith dimension of the tth word embedding, obtained by LRP or SA as specified in Sections 2.1 & 2.2.

In particular, in case of SA, the above word relevance can equivalently be expressed as:

$$R_{\text{SA}}(w_t) = \|\nabla_{w_t} f(d)\|_2^2 \tag{4}$$

where $f(d)$ represents the classifier's prediction for document d.

Note that the resulting LRP word relevance is signed, while the SA word relevance is positive.

In all experiments, we use the term *target* class to identify the function $f(x)$ to analyze in the relevance decomposition. This function maps the neural network input to the neural network output variable corresponding to the target class.

3.3 Evaluating Word-Level Relevances

In order to evaluate different relevance models, we perform a sequence of "word deletions" (hereby for deleting a word we simply set the word-vector to zero in the input document representation), and track the impact of these deletions on the classification performance. We carry out two deletion experiments, starting either with the set of test documents that are initially classified correctly, or with those that are initially classified wrongly[7]. We estimate the LRP/SA word relevances using as target class the true document class. Subsequently we delete words in decreasing resp. increasing order of the obtained word relevances.

Fig. 1 summarizes our results. We find that LRP yields the best results in both deletion experiments. Thereby we provide evidence that LRP positive relevance is targeted to words that *support* a classification decision, while LRP negative relevance is tuned upon words that *inhibit* this decision. In the first experiment the SA classification accuracy curve decreases significantly faster than the random curve representing the performance change when randomly deleting words, indicating that SA is able to identify relevant words. However, the SA curve is clearly above the LRP curve indicating that LRP provides better explanations for the CNN predictions. Similar results have been reported for image classification tasks (Samek et al., 2015). The second experiment indicates that the classification performance increases when deleting words with the lowest LRP relevance, while small SA values points to words that have less influence on the classification performance than random word selection. This result

[4]We employ NLTK's version 3.1 recommended tokenizers `sent_tokenize` and `word_tokenize`, module `nltk.tokenize`.

[5]We retain only tokens composed of the following characters: alphabetic-character, apostrophe, hyphen and dot, and containing at least one alphabetic-character.

[6]To the best of our knowledge, the best published *20Newsgroups* accuracy is 83.0% (Paskov et al., 2013). However we notice that for simplification we use a fixed-length document representation, and our main focus is on explaining classifier decisions, not on improving the classification state-of-the-art.

[7]For the deletion experiments we consider only the test documents whose pre-processed length is greater or equal to 100 tokens, this amounts to a total of 4963 documents.

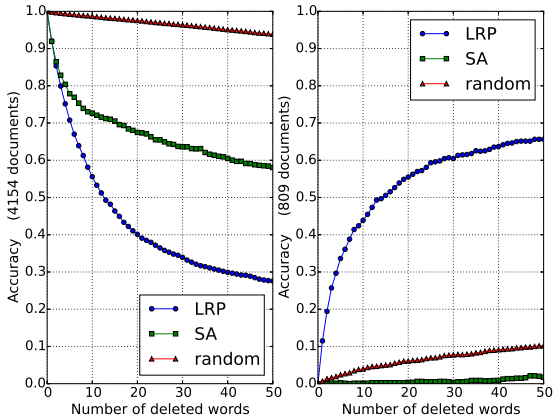

Figure 1: Word deletion on initially correct (left) and false (right) classified test documents, using either LRP or SA. The target class is the true document class, words are deleted in decreasing (left) and increasing (right) order of their relevance. Random deletion is averaged over 10 runs (std < 0.0141). A steep decline (left) and incline (right) indicate informative word relevances.

can partly be explained by the fact that in contrast to SA, LRP provides *signed* explanations. More generally the different quality of the explanations provided by SA and LRP can be attributed to their different objectives: while LRP aims at decomposing the *global* amount of evidence for a class $f(x)$, SA is build solely upon derivatives and as such describes the effect of local *variations* of the input variables on the classifier decision. For a more detailed view of SA, as well as an interpretation of the LRP propagation rules as a *deep Taylor decomposition* see Montavon et al. (2015).

3.4 Document Highlighting

Word-level relevances can be used for highlighting purposes. In Fig. 2 we provide such visualizations on one test document for different relevance target classes, using either LRP or SA relevance models. We can observe that while the word `ride` is highly negative-relevant for LRP when the target class is not `rec.motorcycles`, it is positively highlighted (even though not heavily) by SA. This suggests that SA does not clearly discriminate between words speaking *for* or *against* a specific classifier decision, while LRP is more discerning in this respect.

3.5 Document Visualization

Word2vec embeddings are known to exhibit linear regularities representing semantic relationships between words (Mikolov et al., 2013). We explore if these regularities can be transferred to a document representation, when using as a document vector a linear combination of *word2vec* embeddings. As a weighting scheme we employ LRP or SA scores, with the classifier's predicted class as the target class for the relevance estimation. For comparison we perform uniform weighting, where we simply sum up the word embeddings of the document words (SUM).

For SA we use either the l_2-norm or squared l_2-norm for pooling word gradient values along the *word2vec* dimensions, i.e. in addition to the standard SA word relevance defined in Eq. 4, we use as an alternative $R_{\mathrm{SA}(l_2)}(w_t) = \|\nabla_{w_t} f(d)\|_2$ and denote this relevance model by $\mathrm{SA}(l_2)$.

For both LRP and SA, we employ different variations of the weighting scheme. More precisely, given an input document d composed of the sequence $(w_1, w_2, ..., w_N)$ of D-dimensional *word2vec* embeddings, we build new document representations d' and $d'_{\mathrm{e.w.}}$[8] by either using word-level relevances $R(w_t)$ (as in Eq. 3), or through element-wise multiplication of word embeddings with single input variable relevances $(R_{i,t})_{i\in[1,D]}$ (we recall that $R_{i,t}$ is the relevance of the input variable corresponding to the ith dimension of the tth word in the input document d). More formally we use:

$$d' = \sum_{t=1}^{N} R(w_t) \cdot w_t$$

or

$$d'_{\mathrm{e.w.}} = \sum_{t=1}^{N} \begin{bmatrix} R_{1,t} \\ R_{2,t} \\ \vdots \\ R_{D,t} \end{bmatrix} \odot w_t$$

where $\odot$ is an element-wise multiplication. Finally we normalize the document vectors d' resp. $d'_{\mathrm{e.w.}}$ to unit l_2-norm and perform a PCA projection. In Fig. 3 we label the resulting 2D-projected test documents using five top-level document categories.

For word-based models d', we observe that while standard SA and LRP both provide similar visualization quality, the SA variant with simple l_2-norm yields partly overlapping and dense clusters, still all schemes are better than uniform[9]

[8] The subscript e.w. stands for *element-wise*.

[9] We also performed a TFIDF weighting of word embeddings, the resulting 2D-visualization was very similar to uniform weighting (SUM).

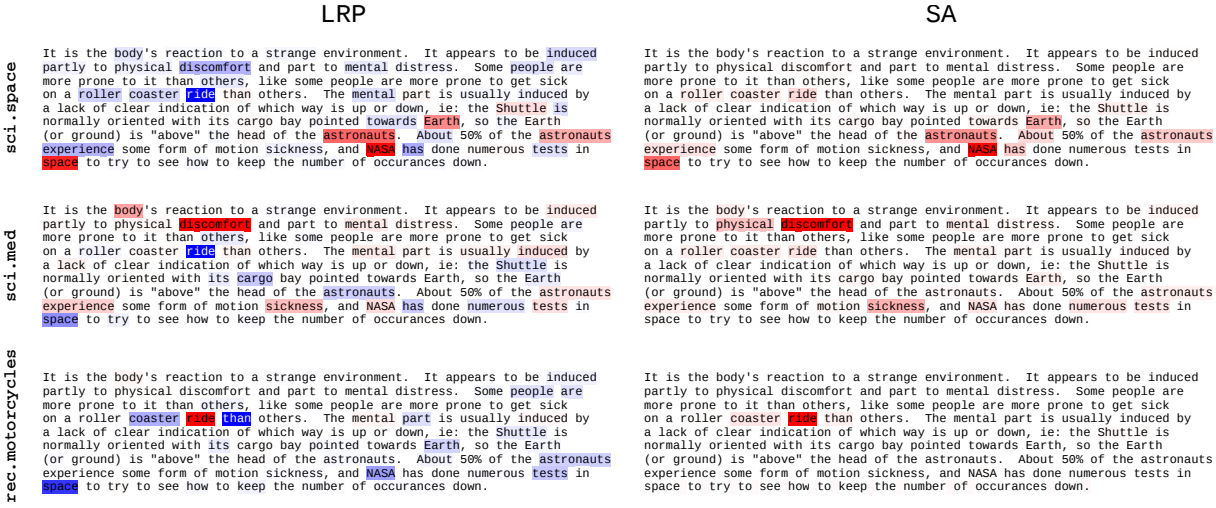

Figure 2: Heatmaps for the test document `sci.space 61393` (correctly classified), using either layer-wise relevance propagation (LRP) or sensitivity analysis (SA) for highlighting words. Positive relevance is mapped to red, negative to blue. The target class for the LRP/SA explanation is indicated on the left.

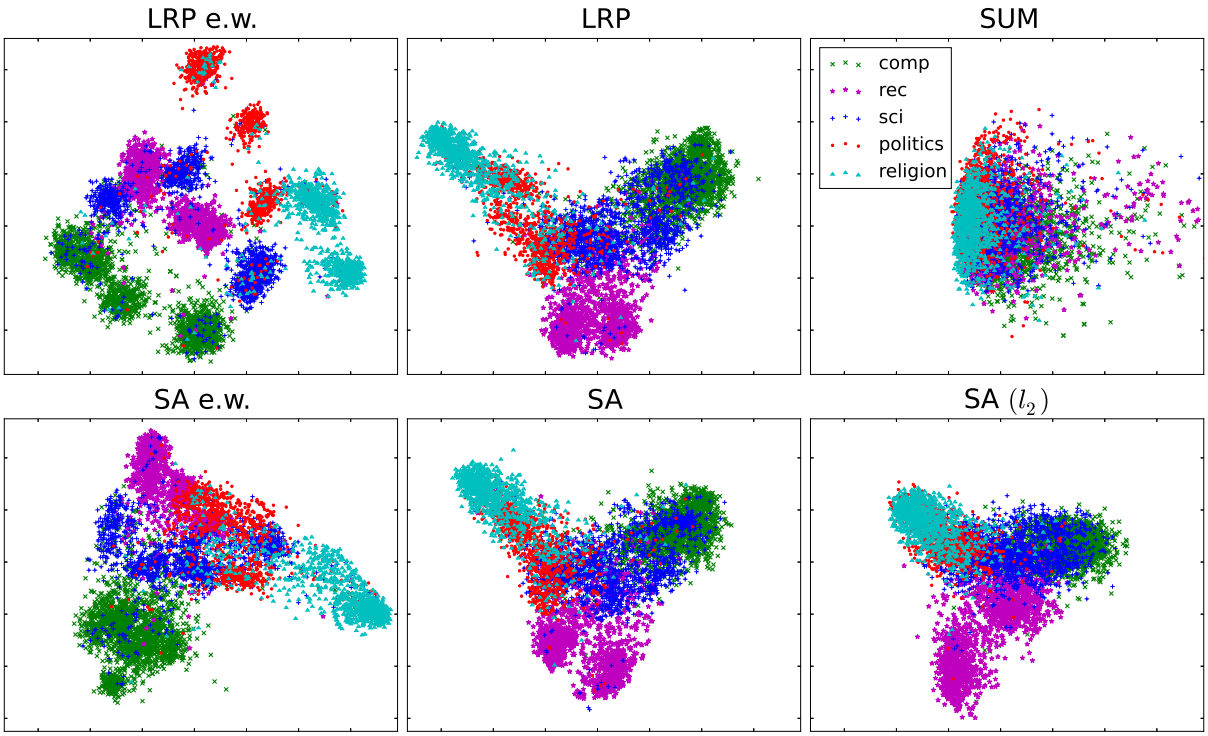

Figure 3: PCA projection of the *20Newsgroups* test documents formed by linearly combining *word2vec* embeddings. The weighting scheme is based on word-level relevances, or on single input variable relevances (e.w.), or uniform (SUM). The target class for relevance estimation is the predicted document class. SA(l_2) corresponds to a variant of SA with simple l_2-norm pooling of word gradient values. All visualizations are provided on the same equal axis scale.

weighting. In case of SA note that, even though the power to which word gradient norms are raised (l_2 or l_2^2) affects the present visualization experiment, it has no influence on the earlier described "word deletion" analysis.

For element-wise models $d'_{\text{e.w.}}$, we observe slightly better separated clusters for SA, and a clear-cut cluster structure for LRP.

4 Conclusion

Through word deleting we quantitatively evaluated and compared two classifier explanation models, and pinpointed LRP to be more effective than SA. We investigated the application of word-level relevance information for document highlighting and visualization. We derive from our empirical analysis that the superiority of LRP stems from the fact that it reliably not only links to determinant words that *support* a specific classification decision, but further distinguishes, within the preeminent words, those that are *opposed* to that decision.

Future work would include applying LRP to other neural network architectures (e.g. character-based or recurrent models) on further NLP tasks, as well as exploring how relevance information could be taken into account to improve the classifier's training procedure or prediction performance.

Acknowledgments

This work was supported by the German Ministry for Education and Research as Berlin Big Data Center BBDC (01IS14013A) and the Brain Korea 21 Plus Program through the National Research Foundation of Korea funded by the Ministry of Education.

References

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. 2015. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE*, 10(7):e0130140.

Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A Neural Probabilistic Language Model. *JMLR*, 3:1137–1155.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR*, 12:2493–2537.

M. Denil, A. Demiraj, and N. de Freitas. 2014. Extraction of Salient Sentences from Labelled Documents. Technical report, University of Oxford.

Y. Dimopoulos, P. Bourret, and S. Lek. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, 2(6):1–4.

D. Erhan, A. Courville, and Y. Bengio. 2010. Understanding Representations Learned in Deep Architectures. Technical report, University of Montreal.

M. Gevrey, I. Dimopoulos, and S. Lek. 2003. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160(3):249–264.

Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proc. of EMNLP*, pages 1746–1751.

R. Krishnan, G. Sivakumar, and P. Bhattacharya. 1999. Extracting decision trees from trained neural networks. *Pattern Recognition*, 32(12):1999–2009.

W. Landecker, M. Thomure, L. Bettencourt, M. Mitchell, G. Kenyon, and S. Brumby. 2013. Interpreting Individual Classifications of Hierarchical Networks. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 32–38.

S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, and W. Samek. 2016a. Analyzing Classifiers: Fisher Vectors and Deep Neural Networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

S. Lapuschkin, A. Binder, G. Montavon, K.-R. Müller, and W. Samek. 2016b. The Layer-wise Relevance Propagation Toolbox for Artificial Neural Networks. *JMLR*. in press.

J. Li, X. Chen, E. Hovy, and D. Jurafsky. 2015. Visualizing and Understanding Neural Models in NLP. *arXiv*, (1506.01066).

M. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Workshop Proc. ICLR*.

G. Montavon, S. Bach, A. Binder, W. Samek, and K.-R. Müller. 2015. Explaining NonLinear Classification Decisions with Deep Taylor Decomposition. *arXiv*, (1512.02479).

H.S. Paskov, R. West, J.C. Mitchell, and T. Hastie. 2013. Compressive Feature Learning. In *Adv. in NIPS*.

W. Samek, A. Binder, G. Montavon, S. Bach, and K.-R. Müller. 2015. Evaluating the visualization of what a Deep Neural Network has learned. *arXiv*, (1509.06321).

K. Simonyan, A. Vedaldi, and A. Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Workshop Proc. ICLR*.

M. D. Zeiler and R. Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *ECCV*, pages 818–833.