"What is Relevant in a Text Document?": An Interpretable Machine Learning Approach

Leila Arras[1], Franziska Horn[2], Grégoire Montavon[2], Klaus-Robert Müller[2,3,4*], 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
4 Max Planck Institute for Informatics, Saarbrücken, Germany

* klaus-robert.mueller@tu-berlin.de, wojciech.samek@hhi.fraunhofer.de

Abstract

Text documents can be described by a number of abstract concepts such as semantic category, writing style, or sentiment. Machine learning (ML) models have been trained to automatically map documents to these abstract concepts, allowing to annotate very large text collections, more than could be processed by a human in a lifetime. Besides predicting the text's category very accurately, it is also highly desirable to understand *how* and *why* the categorization process takes place. In this paper, we demonstrate that such understanding can be achieved by tracing the classification decision back to individual words using layer-wise relevance propagation (LRP), a recently developed technique for explaining predictions of complex non-linear classifiers. We train two word-based ML models, a convolutional neural network (CNN) and a bag-of-words SVM classifier, on a topic categorization task and adapt the LRP method to decompose the predictions of these models onto words. Resulting scores indicate how much individual words contribute to the overall classification decision. This enables one to distill relevant information from text documents without an explicit semantic information extraction step. We further use the word-wise relevance scores for generating novel vector-based document representations which capture semantic information. Based on these document vectors, we introduce a measure of *model explanatory power* and show that, although the SVM and CNN models perform similarly in terms of classification accuracy, the latter exhibits a higher level of explainability which makes it more comprehensible for humans and potentially more useful for other applications.

1 Introduction

A number of real-world problems related to text data have been studied under the framework of natural language processing (NLP). Examples of such problems include topic categorization, sentiment analysis, machine translation, structured information extraction, and automatic summarization. Due to the overwhelming amount of text data available on the Internet from various sources such as user-generated content or digitized books, methods to automatically and intelligently process large collections of text documents are in high demand. For several text applications, machine learning (ML) models based on global word statistics like TFIDF [1, 2] or linear classifiers are known to perform remarkably well, e.g. for unsupervised keyword extraction [3] or document classification [4]. However more recently, neural network models based on vector space representations of words (like [5]) have shown to be of great benefit to a large number of tasks. The trend was initiated by the seminal work of [6] and [7], who introduced word-based neural networks to perform various NLP tasks such as language modeling, chunking, named entity recognition, and semantic role labeling. A number of recent works (e.g. [7, 8]) also refined the basic neural network architecture by incorporating useful structures such as convolution, pooling, and parse tree hierarchies, leading to further improvements in model predictions. Overall, these ML models have permitted to assign automatically and accurately concepts to entire documents or to sub-document levels like phrases; the assigned information can then be mined on a large scale.

In parallel, a set of techniques were developed in the context of image categorization to explain the predictions of convolutional neural networks (a state-of-the-art ML model in this field) or related models. These techniques were able to associate to each prediction of the model a meaningful pattern in the space of input features [9–11] or to perform a decomposition onto the input pixels of the model output [12–14]. In this paper, we will make use of the layer-wise relevance propagation (LRP) technique [13], which has already been substantially tested on various datasets and ML models [15–18].

In the present work, we propose a method to identify which words in a text document are important to explain the category associated to it. The approach consists in using a ML classifier to predict the categories as accurately as possible, and in a second step, decompose the ML prediction onto the input domain, thus assigning to each word in the document a relevance score. The ML model of study will be a word-embedding based convolutional neural network that we train on a text classification task, namely topic categorization of newsgroup documents. As a second ML model we consider a classical bag-of-words support vector machine (BoW/SVM) classifier.

We contribute the following:

(i) The LRP technique [13] is brought to the NLP domain and its suitability for identifying relevant words in text documents is demonstrated.

(ii) LRP relevances are validated, at the document level, by building document heatmap visualizations, and at the dataset level, by compiling representative words for a text category. It is also shown quantitatively that LRP better identifies relevant words than sensitivity analysis.

(iii) A novel way of generating vector-based document representations is introduced and it is verified that these document vectors present semantic regularities within their original feature space akin to word vector representations.

(iv) A measure for model explanatory power is proposed and it is shown that two ML models, a neural network and a BoW/SVM classifier, although presenting similar classification performance, may substantially differ in terms of explainability.

The work is organized as follows. In Section 2 we describe the related work for explaining classifier decisions with respect to input space variables. In Section 3 we introduce our neural network ML model for document classification, as well as the LRP decomposition procedure associated to its predictions. We describe how LRP relevance scores can be used to identify important words in documents and introduce a novel way of condensing the semantic information of a text document into a single document vector. Likewise in section 3 we introduce a baseline ML model for document classification, as well as a gradient-based alternative for assigning relevance scores to words. In Section 4 we define objective criteria for evaluating word relevance scores, as well as for assessing model explanatory power. In Section 5 we introduce the dataset and experimental setup, and in Section 6 we present the results. Finally, Section 7 concludes our work.

2 Related Work

Explanation of individual classification decisions in terms of input variables has been studied for a variety of machine learning classifiers such as additive classifiers [19], kernel-based classifiers [20] or hierarchical networks [12]. Model-agnostic methods for explanations relying on random sampling have also been proposed [21–23]. Despite their generality, the latter however incur an additional computational cost due to the need to process the whole sample to provide a single explanation. Other methods are more specific to deep convolutional neural networks used in computer vision: the authors of [9] proposed a network propagation technique based on deconvolutions to reconstruct input image patterns that are linked to a particular feature map activation or prediction. The work of [10] is aimed at revealing salient structures within images related to a specific class by computing the corresponding prediction score derivative with respect to the input image. The latter method is based on gradient magnitude, and thus reveals the *sensitivity* of the classifier decision to some *local variation* of the input image; this technique is related to sensitivity analysis [24, 25].

In contrast, the LRP method of [13] corresponds to a *full decomposition* of the classifier's actual prediction score value for the *current* input image. One can show that sensitivity analysis decomposes the gradient square norm of the function f, i.e., $\sum_i R_i = \|\nabla_x f(x)\|^2$, whereas LRP decomposes the function value itself $\sum_i R_i = f(x)$. Intuitively, when the classifier e.g. detects cars in images, then sensitivity analysis answers the question "what makes this car image more or less a car?", whereas LRP answers the more fundamental question "what makes this image a car at all?". Note that the LRP framework can be applied to various models such as kernel support vector machines and deep neural networks [13, 18]. We refer the reader to [15] for a comparison of the three explanation methods, and to [14] for a view of particular instances of LRP as a "deep Taylor decomposition" of the decision function. A tutorial on methods for interpreting and understanding deep neural networks can be found in [26].

In the context of neural networks for text classification [27] proposed to extract salient sentences from text documents using loss gradient magnitudes. In order to validate the pertinence of the sentences extracted via the neural network classifier, the latter work proposed to subsequently use these sentences as an input to an external classifier and compare the resulting classification performance to random and heuristic sentence selection. The work by [28] also employs gradient magnitudes to identify salient words within sentences, analogously to the method proposed in computer vision by [10]. However their analysis is based on qualitative interpretation of saliency heatmaps for exemplary sentences. In addition to the heatmap visualizations, we provide a classifier-intrinsic quantitative validation of the word-level relevances. We furthermore extend previous work from [29] by adding a BoW/SVM baseline to the experiments and proposing a new criterion for assessing model explanatory power. Recent work from [30, 31] uses LRP to explain recurrent neural network predictions in sentiment analysis and machine translation.

3 Interpretable Text Classification

In this Section we describe our method for identifying words in a text document, that are relevant with respect to a given category of a classification problem. For this, we assume that we are given a vector-based word representation and a convolutional neural network that has already been trained to map accurately documents to their actual category. Our method can be divided into four steps: (1) Compute an input representation of a text document based on word vectors. (2) Forward-propagate the input representation through the convolutional neural network until the output is reached. (3) Backward-propagate the output through the network using the layer-wise relevance propagation (LRP) method, until the input is reached. (4) Pool the relevance scores associated to each input variable of the network onto the words to which they belong. As a result of this four-step procedure, a decomposition of the prediction score for a category onto the words of the documents is obtained. Decomposed terms are called relevance scores. These relevance scores can be viewed as highlighted text or can be used to form a list of top-words in the document. The whole procedure is also described visually in Fig 1. While we detail in this Section the LRP method for a specific network architecture and with predefined choices of layers, the method can in principle be extended to any architecture composed of a similar or larger number of layers.

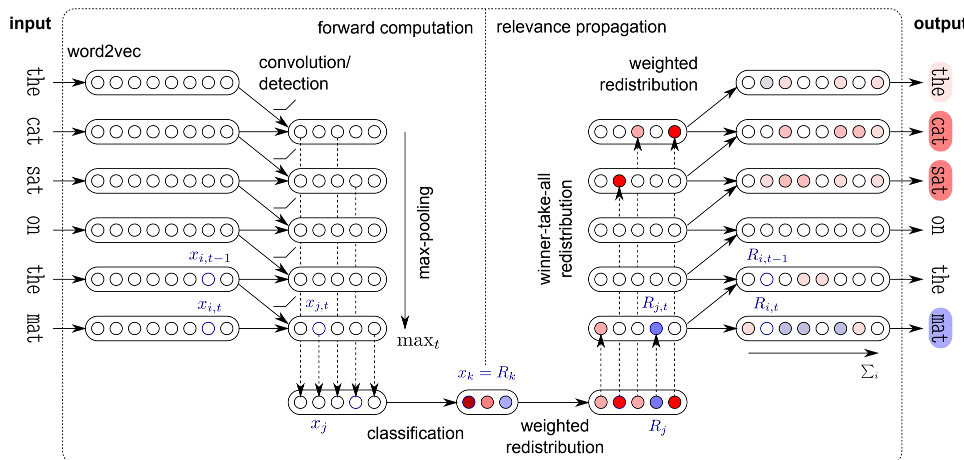

Fig 1. Diagram of a CNN-based interpretable machine learning system. It consists of a forward processing that computes for each input document a high-level concept (e.g. semantic category or sentiment), and a redistribution procedure that explains the prediction in terms of words.

At the end of this Section we introduce different methods which will serve as baselines for comparison. A baseline for the convolutional neural network model is the BoW/SVM classifier, with the LRP procedure adapted accordingly [13]. A baseline for the LRP relevance decomposition procedure is gradient-based sensitivity analysis (SA), a technique which assigns sensitivity scores to individual words. In the vector-based document representation experiments, we will also compare LRP to uniform and TFIDF baselines.

3.1 Representing Words and Documents

Prior to training the neural network and using it for prediction and explanation, we first derive a numerical representation of the text documents that will serve as an input to the neural classifier. To this end, we map each individual word in the document to a vector embedding, and concatenate these embeddings to form a matrix of size the number of words in the document times the dimension of the word embeddings. A distributed representation of words can be learned from scratch, or fine-tuned simultaneously with the classification task of interest. In the present work, we use only pre-training as it was shown that, even without fine-tuning, this leads to good neural network classification performance for a variety of tasks like e.g. part-of-speech tagging or sentiment analysis [7, 32].

One shallow neural network model for learning word embeddings from unlabeled text sources, is the continuous bag-of-words (CBOW) model of [33], which is similar to the log-bilinear language model from [34, 35] but ignores the order of context words. In the CBOW model, the objective is to predict a target middle word from the average of the embeddings of the context words that are surrounding the middle word, by means of direct dot products between word embeddings. During training, a set of word embeddings for context words v and for target words v' are learned separately. After training is completed, only the context word embeddings v will be retained for further applications. The CBOW objective has a simple maximum likelihood formulation, where one maximizes over the training data the sum of the logarithm of probabilities of the form:

$$P(w_t|w_{t-n:t+n}) = \frac{\exp\left((\frac{1}{2n} \cdot \sum_{-n \leq j \leq n, j \neq 0} \ v_{w_{t+j}})^\top v'_{w_t} \right)}{\sum_{w \in V} \ \exp\left((\frac{1}{2n} \cdot \sum_{-n \leq j \leq n, j \neq 0} \ v_{w_{t+j}})^\top v'_{w} \right)}$$

where the softmax normalization runs over all words w in the vocabulary V, $2n$ is the number of context words per training text window, w_t represents the target word at the t^{th} position in the training data and $w_{t-n:t+n}$ represent the corresponding context words.

In the present work, we utilize pre-trained word embeddings obtained with the CBOW architecture and the negative sampling training procedure [5]. We will refer to these embeddings as word2vec embeddings.

3.2 Predicting Category with a Convolutional Neural Network

Our ML model for classifying text documents, is a word-embedding based convolutional neural network (CNN) model similar to the one proposed in [32] for sentence classification, which itself is a slight variant of the model introduced in [7] for semantic role labeling. This architecture is depicted in Fig 1 (left) and is composed of several layers.

As previously described, in a first step we map each word in the document to its word2vec vector. Denoting by D the word embedding dimension and by L the document length, our input is a matrix of shape $D \times L$ (e.g., for the purpose of illustration, in Fig 1 we have $D = 8$ and $L = 6$). We denote by $x_{i,t}$ the value of the i^{th} component of the word2vec vector representing the t^{th} word in the document. The convolution/detection layer produces a new representation composed of F sequences indexed by j, where each element of the sequence is computed as:

$$\forall j, t: \ x_{j,t} = \max\left(0, \ \sum_{i,\tau} x_{i,t-\tau} \ w^{(1)}_{i,j,\tau} + b^{(1)}_j\right) = \max\left(0, \ \sum_i \ \left(x_i * w^{(1)}_{i,j}\right)_t + b^{(1)}_j\right)$$

where t indicates a position within the text sequence, j designates a feature map, and $\tau \in \{0, 1, \ldots, H-1\}$ is a delay with range H, the filter size of the one-dimensional convolutional operation $*$. After the convolutional operation, which yields F features maps of length $L - H + 1$, we apply the ReLU non-linearity element-wise (e.g., in Fig 1, we have $F = 5$ features maps and a filter size $H = 2$, hence we use $\tau \in \{0, 1\}$ and the resulting feature maps have a length of 5). Note that the trainable parameters $w^{(1)}$ and $b^{(1)}$ do not depend on the position t in the text document, hence the convolutional processing is equivariant with this physical dimension. The next layer computes, for each dimension j of the previous representation, the maximum over the entire text sequence of the document:

$$\forall j: \ x_j = \max_t \left\{x_{j,t}\right\}$$

This layer creates invariance to the position of the features in the document. Finally, the F pooled features are fed into a logistic classifier where the unnormalized log-probability of each of the C classes, indexed by the variable k are given by:

$$\forall k: \ x_k = \textstyle\sum_j x_j \, w_{jk}^{(2)} + b_k^{(2)}$$

where $w^{(2)}$, $b^{(2)}$ are trainable parameters of size $F \times C$ resp. size C defining a fully-connected linear layer (e.g., in Fig 1, $C = 3$). The outputs x_k can be converted to probabilities through the softmax function $p_k = \exp(x_k)/\sum_{k'} \exp(x_{k'})$. For the LRP decomposition we take the unnormalized classification scores x_k as a starting point.

3.3 Explaining Predictions with Layer-wise Relevance Propagation

Layer-wise relevance propagation (LRP) [13, 36] is a recently introduced technique for estimating which elements of a classifier input are important to achieve a certain classification decision. It can be applied to bag-of-words SVM classifiers as well as to layer-wise structured neural networks. For every input data point and possible target class, LRP delivers one scalar relevance value per input variable, hereby indicating whether the corresponding part of the input is contributing *for* or *against* a specific classifier decision, or if this input variable is rather uninvolved and irrelevant to the classification task.

The main idea behind LRP is to redistribute, for each possible target class separately, the output prediction score (i.e. a scalar value) that causes the classification, back to the input space via a backward propagation procedure that satisfies a layer-wise conservation principle. Thereby each intermediate classifier layer up to the input layer gets allocated relevance values, and the sum of the relevances per layer is equal to the classifier prediction score for the class being considered. Denoting by $x_{i,t}\,, x_{j,t}\,, x_j\,, x_k$ the neurons of the CNN layers presented in the previous Section, we associate to each of them respectively a relevance score $R_{i,t}\,, R_{j,t}\,, R_j\,, R_k$. Accordingly the layer-wise conservation principle can be written as:

$$\textstyle\sum_{i,t} R_{i,t} = \sum_{j,t} R_{j,t} = \sum_j R_j = \sum_k R_k \tag{1}$$

where each sum runs over all neurons of a given layer of the network. To formalize the redistribution process from one layer to another, we introduce the concept of messages $R_{a \leftarrow b}$ indicating how much relevance circulates from a given neuron b to a neuron a in the next lower-layer. We can then express the relevance of neuron a as a sum of incoming messages using: $R_a = \sum_{b \in \text{upper}(a)} R_{a \leftarrow b}$ where upper(a) denotes the upper-layer neurons connected to a. To bootstrap the propagation algorithm, we set the top-layer relevance vector to $\forall_k : R_k = x_k \cdot \delta_{kc}$ where δ is the Kronecker delta function, and c is the *target* class of interest for which we would like to explain the model prediction in isolation from other classes.

In the top fully-connected layer, messages are computed following a weighted redistribution formula:

$$R_{j \leftarrow k} = \frac{z_{jk}}{\sum_j z_{jk}} R_k \tag{2}$$

where we define $z_{jk} = x_j w_{jk}^{(2)} + F^{-1}(b_k^{(2)} + \epsilon \cdot (1_{x_k \geq 0} - 1_{x_k < 0}))$. This formula redistributes relevance onto lower-layer neurons in proportion to z_{jk} representing the contribution of each neuron to the upper-layer neuron value in the forward propagation, incremented by a small stabilizing term ϵ that prevents the denominator from nearing zero, hence avoiding positive and negative relevance messages that are too large. In the limit case where $\epsilon \to \infty$, the relevance is redistributed uniformly along the network connections. As a stabilizer value we use $\epsilon = 0.01$ as introduced in [13]. After computation of the messages according to Eq 2, the latter can be pooled onto the corresponding neuron by the formula $R_j = \sum_k R_{j \leftarrow k}$.

The relevance scores R_j are then propagated through the max-pooling layer using the formula:

$$R_{j,t} = \begin{cases} R_j & \text{if } \ t = \arg\max_{t'} \ x_{j,t'} \\ 0 & \text{else} \end{cases} \tag{3}$$

which is a "winner-take-all" redistribution analogous to the rule used during training for backpropagating gradients, i.e. the neuron that had the maximum value in the pool is granted all the relevance from the upper-layer neuron. Finally, for the convolutional layer we use the weighted redistribution formula:

$$R_{(i,t-\tau) \leftarrow (j,t)} = \frac{z_{i,j,\tau}}{\sum_{i,\tau} z_{i,j,\tau}} R_{j,t} \tag{4}$$

where $z_{i,j,\tau} = x_{i,t-\tau} w^{(1)}_{i,j,\tau} + (HD)^{-1}(b^{(1)}_j + \epsilon \cdot (1_{x_{j,t}>0} - 1_{x_{j,t} \leq 0}))$, which is similar to Eq 2 except for the increased notational complexity incurred by the convolutional structure of the layer. Messages can finally be pooled onto the input neurons by computing $R_{i,t} = \sum_{j,\tau} R_{(i,t) \leftarrow (j,t+\tau)}$.

3.4 Word Relevance and Vector-Based Document Representation

So far, the relevance has been redistributed only onto individual components of the word2vec vector associated to each word, in the form of single input neuron relevances $R_{i,t}$. To obtain a word-level relevance value, one can pool the relevances over all dimensions of the word2vec vector, that is computed as:

$$R_t = \textstyle\sum_i R_{i,t} \tag{5}$$

and use this value to highlight words in a text document, as shown in Fig 1 (right). These word-level relevance scores can further be used to condense the semantic information of text documents, by building vectors $\boldsymbol{d} \in \mathbb{R}^D$ representing full documents through linearly combining word2vec vectors:

$$\forall_i : \ d_i = \textstyle\sum_t R_t \cdot x_{i,t} \tag{6}$$

The vector $\boldsymbol{d}$ is a summary that consists of an additive composition of the semantic representation of all relevant words in the document. Note that the resulting document vector lies in the same semantic space as word2vec vectors. A more fined-grained extraction technique does not apply word-level pooling as an intermediate step and extracts only the relevant subspace of each word:

$$\forall_i : \ d_i = \textstyle\sum_t R_{i,t} \cdot x_{i,t} \tag{7}$$

This last approach is particularly useful to address the problem of word homonymy, and will thus result in even finer semantic extraction from the document. In the remaining we will refer to the semantic extraction defined by Eq 6 as word-level extraction, and to the one from Eq 7 as element-wise (ew) extraction. In both cases we call vector $\boldsymbol{d}$ a *document summary vector*.

3.5 Baseline Methods

In the following we briefly mention methods which will serve as baselines for comparison.

Sensitivity Analysis. Sensitivity analysis (SA) [20, 24, 25] assigns scores $R_{i,t} = (\partial x_k / \partial x_{i,t})^2$ to input variables representing the steepness of the decision function in the input space. These partial derivatives are straightforward to compute using standard gradient propagation [37] and are readily available in most neural network implementations. We would like to point out that, per definition, sensitivity analysis redistributes the quantity $\|\nabla x_k\|_2^2$, while LRP redistributes x_k. However, the local steepness information is a relatively weak proxy of the actual function value, which is the real quantity of interest when estimating the contribution of input variables with respect to a current classifier's decision. We further note that relevance scores obtained with LRP are signed, while those obtained with SA are positive.

BoW/SVM. As a baseline to the CNN model, a bag-of-words linear SVM classifier will be used to predict the document categories. In this model each text document is first mapped to a vector x with dimensionality V the size of the training data vocabulary, where each entry is computed as a term frequency - inverse document frequency (TFIDF) score of the corresponding word. Subsequently these vectors x are normalized to unit Euclidean norm. In a second step, using the vector representations x of all documents, C maximum margin separating hyperplanes are learned to separate each of the classes of the classification problem from the other ones. As a result we obtain for each class $c \in C$ a linear prediction score of the form $s_c = w_c^\top x + b_c$, where $w_c \in \mathbb{R}^V$ and $b_c \in \mathbb{R}$ are class specific weights and biases. In order to obtain a LRP decomposition of the prediction score s_c for class c onto the input variables, we simply compute $R_i = (w_c)_i \cdot x_i + b_c/D$, where D is the number of non-zero entries of x. Respectively, the sensitivity analysis redistribution of the prediction score squared gradient reduces to $R_i = (w_c)_i^2$.

Note that the BoW/SVM model, being a linear predictor relying directly on word frequency statistics, lacks expressive power in comparison to the CNN model which additionally learns intermediate hidden layer representations and convolutional filters. Moreover the CNN model can take advantage of the semantic similarity encoded in the distributed word2vec representations, while for the BoW/SVM model all words are "equidistant" in the bag-of-words semantic space. As our experiments will show, these limitations lead the BoW/SVM model to sometimes identify spurious words as relevant for the classification task.

In analogy to the semantic extraction proposed in Section 3.4 for the CNN model, we can build vectors $\boldsymbol{d}$ representing documents by leveraging the word relevances obtained with the BoW/SVM model. To this end, we introduce a binary vector $\tilde{x} \in \mathbb{R}^V$ whose entries are equal to one when the corresponding word from the vocabulary is present in the document and zero otherwise (i.e. $\tilde{x}$ is a binary bag-of-words representation of the document). Thereafter, we build the document summary vector $\boldsymbol{d}$ component-wise, so that $\boldsymbol{d}$ is just a vector of word relevances:

$$\forall_i : \; d_i = R_i \cdot \tilde{x}_i \tag{8}$$

Uniform/TFIDF based Document Summary Vector. Instead of the word-level relevance R_t resp. R_i used in Eq 6 and Eq 8, we can apply a uniform weighting. This corresponds to building the document vector $\boldsymbol{d}$ as an average of word2vec word embeddings in the first case, and to taking a binary bag-of-words vector as the document representation $\boldsymbol{d}$ in the second case. Moreover, we can replace R_t in Eq 6 by an inverse document frequency (IDF) score, and R_i in Eq 8 by a TFIDF score. Both correspond to TFIDF weighting of either word2vec vectors, or of one-hot vectors representing words.

4 Quality of Word Relevances and Model Explanatory Power

In this Section we describe how to evaluate and compare the outcomes of algorithms which assign relevance scores to words (such as LRP or SA) through intrinsic validation. Furthermore, we propose a measure of model explanatory power based on an extrinsic validation procedure. The latter will be used to analyze and compare the relevance decompositions or *explanations* obtained with the neural network and the BoW/SVM classifier. Both types of evaluations will be carried out in Section 6.

4.1 Measuring the Quality of Word Relevances through Intrinsic Validation

An evaluation of how well a method identifies relevant words in text documents can be performed qualitatively, e.g. at the document level, by inspecting the heatmap visualization of a document, or by reviewing the list of the most (or the least) relevant words per document. A similar analysis can also be conducted at the dataset level, e.g. by compiling the list of the most relevant words for one category across all documents. The latter allows one to identify words that are representatives for a document category, and eventually to detect potential dataset biases or classifier specific drawbacks. However, in order to quantitatively compare algorithms such as LRP and SA regarding the identification of relevant words, we need an objective measure of the quality of the explanations delivered by relevance decomposition methods. To this end we adopt an idea from [15]: A word w is considered highly relevant for the classification $f(x)$ of the document x if removing it and classifying the modified document $\tilde{x}$ results in a strong decrease of the classification score $f(\tilde{x})$. This idea can be extended by sequentially deleting words from the most relevant to the least relevant or the other way round. The result is a graph of the prediction scores $f(\tilde{x})$ as a function of the number of deleted words. In our experiments, we employ this approach to track the changes in classification performance when successively deleting words according to their relevance value. By comparing the relative impact on the classification performance induced by different relevance decomposition methods, we can estimate how appropriate these methods are at identifying words that are really important for the classification task at hand. The above procedure constitutes an intrinsic validation, as it does not rely on an external classifier.

4.2 Measuring Model Explanatory Power through Extrinsic Validation

Although intrinsic validation can be used to compare relevance decomposition methods for a given ML model, this approach is not suited to compare the explanatory power of different ML models, since the latter requires a common evaluation basis. Furthermore, even if we would track the classification performance changes induced by different ML models using an external classifier, it would not necessarily increase comparability, because removing words from a document may affect different classifiers very differently, so that their graphs $f(\tilde{x})$ are not comparable. Therefore, we propose a novel measure of model explanatory power which does not depend on a classification performance change, but only on the word relevances. Hereby we consider ML model A as being more explainable than ML model B if its word relevances are more "semantic extractive", i.e. more helpful for solving a semantic related task such as the classification of document summary vectors.

More precisely, in order to quantify the ML model explanatory power, we undertake the following steps:

(1) Compute document summary vectors for all test set documents using Eq 6 or 7 for the CNN and Eq 8 for the BoW/SVM model. Hereby use the ML model's predicted class as target class for the relevance decomposition (i.e. the summary vector generation is unsupervised).

(2) Normalize the document summary vectors to unit Euclidean norm, and perform a K-nearest-neighbors (KNN) classification of half of these vectors, using the other half of summary vectors as neighbors (using standard KNN classification, i.e. nearest neighbors are identified by Euclidean distance and neighbor votes are weighted uniformly). Use different hyperparameters K.

(3) Repeat step (2) over 10 random data splits, and average the KNN classification accuracies for each K. Finally, report the maximum (over different K) KNN accuracy as explanatory power index (EPI). The higher this value, the more *explanatory power* the ML model and the corresponding document summary vectors, will have.

In a nutshell, our EPI metric of explanatory power of a given ML model "f", combined with a relevance map "R", can informally be summarized as:

$$\boldsymbol{d}(x) = \textstyle\sum_t \, [R(f(x)) \odot x]_t$$

$$\mathrm{EPI}(f, R) \;=\; \max_K \;\; \texttt{KNN\_accuracy}\Big(\{\boldsymbol{d}(x^{(1)}), \ldots, \boldsymbol{d}(x^{(N)})\}, K\Big) \tag{9}$$

where $\boldsymbol{d}(x)$ is the document summary vector for input document x, and subscript t denotes the words in the document. The sum $\sum_t$ and element-wise multiplication $\odot$ operations stand for the weighted combination specified explicitly in Eq 6 - 8. The KNN accuracy is estimated over all test set document summary vectors indexed from 1 to N, and K is the number of neighbors.

In the proposed evaluation procedure, the use of KNN as a common external classifier enables us to compare different ML models in an unbiased manner, in terms of the density and local neighborhood structure of the semantic information extracted via the summary vectors in input feature space. Indeed we recall that summary vectors constructed via Eq 6 and 7 lie in the same semantic space as word2vec embeddings, and that summary vectors obtained via Eq 8 lie in the bag-of-words space.

5 Experimental Setup

This Section describes the dataset, preprocessing and training procedure used in our experiments.

5.1 Dataset

We consider a topic categorization task, and employ the freely available 20Newsgroups dataset consisting of newsgroup posts evenly distributed among twenty fine-grained categories. More precisely we use the 20news-bydate version, which is already partitioned into 11314 training and 7532 test documents corresponding to different periods in time.

5.2 Preprocessing and Training

As a first preprocessing step, we remove the headers from the documents (by splitting at the first blank line) and tokenize the text with NLTK. Then, we filter the tokenized data by retaining only tokens composed of the following four types of characters:

alphabetic, hyphen, dot and apostrophe, and containing at least one alphabetic character. Hereby we aim to remove punctuation, numbers or dates, while keeping abbreviations and compound words. We do not apply any further preprocessing, as for instance stop-word removal or stemming, except for the SVM classifier where we additionally perform lowercasing, as this is a common setup for bag-of-words models. We truncate the resulting sequence of tokens to a chosen fixed length of 400 in order to simplify neural network training (in practice our CNN can process any arbitrary sized document). Lastly, we build the neural network input by horizontally concatenating pre-trained word embeddings, according to the sequence of tokens appearing in the preprocessed document. In particular, we take the 300-dimensional freely available word2vec embeddings [5]. Out-of-vocabulary words are simply initialized to zero vectors. As input normalization, we subtract the mean and divide by the standard deviation obtained over the flattened training data. We train the neural network by minimizing the cross-entropy loss via mini-batch stochastic gradient descent using l_2-norm and dropout as regularization. We tune the ML model hyperparameters by 10-fold cross-validation in case of the SVM, and by employing 1000 random documents as fixed validation set for the CNN model. However, for the CNN hyperparameters, we did not perform an extensive grid search and stopped the tuning once we obtained models with reasonable classification performance for the purpose of our experiments.

6 Results

This Section summarizes our experimental results. We first describe the classification accuracy of the four ML models: three CNNs with different filter sizes and a BoW/SVM classifier. Remaining results are divided into two parts, first a *qualitative* one and then a *quantitative* one. In the qualitative part, we demonstrate that LRP can be used to identify relevant words in text documents. We also compare heatmaps for the best performing CNN model and the BoW/SVM classifier, and report the most representative words for three exemplary document categories. These results demonstrate qualitatively that the CNN model produces better explanations than the BoW/SVM classifier. After that we move to the evaluation of the document summary vectors, where we show that a 2D PCA projection of the document vectors computed from the LRP scores groups documents according to their topics (without requiring the true labels). Since worse results are obtained when using the SA scores or the uniform or TFIDF weighting, this indicates that the explanations produced by LRP are semantically more meaningful than the former. In the quantitative part, we confirm the observations made before, namely that (1) the LRP decomposition method provides better explanations than SA and that (2) the CNN model outperforms the BoW/SVM classifier in terms of explanatory power.

6.1 Performance Comparison

Table 1 summarizes the performance of our trained models. Herein CNN1, CNN2, CNN3 respectively denote neural networks with convolutional filter size H equal to 1, 2 and 3 (i.e. covering 1, 2 or 3 consecutive words in the document). One can see that the linear SVM performs on par with the neural networks, i.e. the non-linear structure of the CNN models does not yield a considerable advantage toward classification accuracy. Similar results have also been reported in previous studies [38], where it was observed that for document classification a convolutional neural network model starts to outperform a TFIDF-based linear classifier only on datasets in the order of millions of documents. This can be explained by the fact that for most topic categorization tasks, the different categories can be separated linearly in the very high-dimensional bag-of-words or bag-of-N-grams space thanks to sufficiently disjoint sets of features. However, despite similar performance, the CNN models present some advantages over the SVM in that their computational costs scale linearly with the training data size, and that each training iteration involves a mini-batch of fixed size; besides, they can take advantage of the word similarity information encoded in the distributed word embeddings, whereas for a BoW/SVM model, any training algorithm that solves the dual optimization problem for an arbitrary kernel has a computational cost that scales at least quadratically in the number of training samples (moreover when the kernel matrix does not fit in memory, this constitutes a major factor in the computation time) [39], additionally the latter model does not consider any word similarity.

Table 1. Test set performance of the ML models for 20-class document classification.

ML Model	**Test Accuracy (%)**
BoW/SVM ($V = 70631$ words)	80.10
CNN1 ($H = 1$, $F = 600$)	79.79
CNN2 ($H = 2$, $F = 800$)	**80.19**
CNN3 ($H = 3$, $F = 600$)	79.75

6.2 Identifying Relevant Words

Fig 2 compiles the resulting LRP heatmaps we obtain on an exemplary `sci.space` test document that is correctly classified by the SVM and the best performing neural network model CNN2. Note that for the SVM model the relevance values are computed per bag-of-words feature, i.e., the same words will have the same relevance irrespectively of their context in the document, whereas for the CNN classifier we visualize one relevance value per word position. Here we consider as target class for the LRP decomposition the classes `sci.space` and `sci.med`. We can observe that the SVM model considers insignificant words like *the*, *is*, *of* as very relevant (either negatively or positively) for the target class `sci.med`, and at the same time mistakenly estimates words like *sickness*, *mental* or *distress* as negatively contributing to this class (indicated by blue coloring). Besides, in the present work, we compute the TFIDF score of a word w as the raw word count multiplied by $(1 + \log \frac{1+N}{1+n_w})$, where N is the total number of training documents, and n_w is the number of training documents in which w occurs, hence the inverse document frequency has a minimum value of one; this further explains, in part, why frequent words like *the* are not entirely ignored by the SVM model. On the other hand, we notice that the CNN2 heatmap is consistently more sparse and concentrated on semantically meaningful words. This sparsity property can be attributed to the max-pooling non-linearity which for each feature map in the neural network selects the most relevant feature that occurs in the document. As can be seen, it significantly simplifies the interpretability of the results by a human. Another disadvantage of the SVM model is that it relies entirely on local and global word statistics, thus can only assign relevances proportionally to the TFIDF BoW features (plus a class-dependent bias term), while the neural network model benefits from the knowledge encoded in the word2vec embeddings. For instance, the word *weightlessness* is not highlighted by the SVM model for the target class `sci.space`, because this word does not occur in the training data and thus is simply ignored by the SVM classifier. The neural network however is able to detect and attribute relevance to unseen words thanks to the semantic information encoded in the pre-trained word2vec embeddings.

As a dataset-wide analysis, we determine the words identified through LRP and SA as class representatives. For that purpose we set one class as target class for the relevance decomposition, and conduct LRP, resp. SA, over all test set documents (i.e. irrespectively of the true or ML model's predicted class). Subsequently, we sort all the

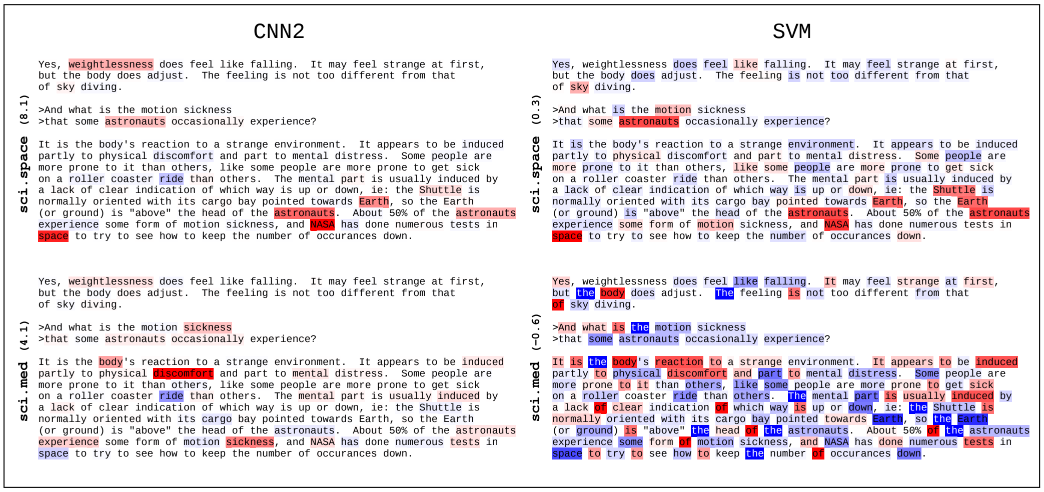

Fig 2. LRP heatmaps of the document `sci.space 61393` for the CNN2 and SVM model. Positive relevance is mapped to red, negative to blue. The color opacity is normalized to the maximum absolute relevance per document. The LRP target class and corresponding classification prediction score is indicated on the left.

words appearing in the test data in decreasing order of the obtained word-level relevance values, and retrieve the twenty most relevant ones. The result is a list of words identified via LRP or SA as being highly supportive for a classifier decision toward the considered class. Fig 3 and 4 list the most relevant words for different target classes, as well as the corresponding word-level relevance values for the CNN2 and the SVM model. Through underlining we indicate words that do not occur in the training data. Interestingly, we observe that some of the most "class-characteristic" words identified via the neural network model correspond to words that do not even appear in the training data. In contrast, such words are simply ignored by the SVM model as they do not occur in the bag-of-words vocabulary. Similarly to the previous heatmap visualizations, the class-specific analysis reveals that the SVM classifier occasionally assigns high relevances to semantically insignificant words like for example the pronoun *she* for the target class `sci.med` (20^{th} position in the first row left column of Fig 4), or to the names *pat*, *henry*, *nicho* for the target the class `sci.space` (resp. 7, 13, 20^{th} position in the first row middle column of Fig 4). In the former case the high relevance is due to a high term frequency of the word (indeed the word *she* achieves its highest term frequency in one `sci.med` test document where it occurs 18 times), whereas in the latter case this can be explained by a high inverse document frequency or by a class-biased occurrence of the corresponding word in the training data (*pat* appears within 16 different training document categories but 54.1% of its occurrences are within the category `sci.space` alone, 79.1% of the 201 occurrences of *henry* appear among `sci.space` training documents, and *nicho* appears exclusively in nine `sci.space` training documents). On the contrary, the neural network model seems less affected by word count regularities and systematically attributes the highest relevances to words semantically related to the target class. These results demonstrate that, subjectively, the neural network is better suited to identify relevant words in text documents than the BoW/SVM model. For a given classifier, when comparing the lists of the most relevant words obtained with LRP or SA in Fig 3 and 4, we do not discern any qualitative difference between LRP and SA class representatives. Nevertheless, we recall that these "keywords" correspond to the greatest observed relevance scores *across* the test set documents, and do not reflect the differences between LRP and SA at the document level. Indeed in practice we noticed

	sci.med	sci.space	comp.graphics
LRP	symptoms (7.3), treatments (6.6), medication (6.4), osteopathy (6.3), ulcers (6.2), sciatica (6.0), hypertension (6.0), herb (5.6), doctor (5.4), physician (5.1), Therapy (5.1), antibiotics (5.1), Asthma (5.0), renal (5.0), medicines (4.9), caffeine (4.9), infection (4.9), gastrointestinal (4.8), therapy (4.8), homeopathic (4.7).	spacecraft (11.0), orbit (10.8), NASA (8.6), Mars (7.8), moon (7.1), orbiting (7.1), Martian (6.8), orbital (6.8), shuttle (6.7), SMOS (6.6), telescope (6.5), Space (6.5), rocket (6.3), GRBs (6.0), Earth (6.0), astronaut (5.9), Moon (5.7), Shuttle (5.7), lander (5.6), Flyby (5.3).	Graphics (6.9), raytracing (6.8), graphics (6.8), polygon (6.5), animation (6.3), Image (6.2), shaders (6.2), pixel (5.7), fractal (5.5), viewports (5.5), Autodesk (5.4), visualization (5.2), RGB (5.1), images (5.0), TIFF (5.0), Corel (4.9), Studio (4.9), algorithm (4.8), Bezier (4.8), polygons (4.7).
SA	sciatica (0.4), symptoms (0.4), osteopathy (0.4), Therapy (0.3), treatments (0.3), herb (0.3), cancer (0.3), allergic (0.3), cravings (0.3), ulcers (0.3), Asthma (0.3), IVF (0.3), medication (0.2), Addiction (0.2), caffeine (0.2), drug (0.2), sulfation (0.2), Dr. (0.2), penicillin (0.2), gel (0.2).	orbit (0.4), Martian (0.4), spacecraft (0.4), rocket (0.4), NASA (0.4), GRBs (0.4), telescope (0.3), Mars (0.3), docking (0.3), SMOS (0.3), moon (0.3), Earth (0.3), hyperspace (0.3), Space (0.3), Galileo (0.3), space (0.3), planetary (0.3), satellite (0.3), Shuttle (0.3), Astronomy (0.3).	raytracing (0.4), animation (0.4), Autodesk (0.4), RGB (0.3), graphics (0.3), Graphics (0.3), Image (0.3), pixel (0.3), visualization (0.3), fractal (0.3), Pixel (0.3), GIF (0.3), CG (0.3), viewports (0.3), Corel (0.3), shaders (0.3), polygon (0.3), TIFF (0.3), JPEG (0.3), Rodchenko (0.3).

Fig 3. The 20 most relevant words per class for the CNN2 model. The words are listed in decreasing order of their LRP(first row)/SA(second row) relevance (value indicated in parentheses). Underlined words do not occur in the training data.

that *within* a document, the first two most relevant words identified with LRP or SA are often identical, but the remaining words will be ordered differently in terms of their LRP or SA relevance. As an example, if we consider the test documents of the classes `sci.space`, `sci.med`, and `comp.graphics`, retaining only documents with a length greater or equal to 10 tokens (this amounts to 1165 documents), and perform a relevance decomposition for the true document class using the CNN2 model, then in 81,9% of the cases LRP and SA will identify the same word as the most relevant per document, and if the two most relevant words are considered, 42.6% of the cases will be equal, 15,8% for three words and only 4.0% for four words. Similar results are obtained using the SVM model: in 50.9% of the cases LRP and SA identify the same word as the most relevant, if we consider the two most relevant, 23.1% of the cases are identical, 7.3% for three words and 1.6% for four words. In addition, the differences between LRP and SA are confirmed by the quantitative evaluation in Section 6.4.

6.3 Document Summary Vectors

The word2vec embeddings are known to exhibit linear regularities representing semantic relationships between words [5, 33]. We explore whether these regularities can be

	sci.med	sci.space	comp.graphics
LRP	cancer (1.4), photography (1.0), doctor (1.0), msg (0.9), disease (0.9), medical (0.8), sleep (0.8), radiologist (0.7), eye (0.7), treatment (0.7), prozac (0.7), vitamin (0.7), epilepsy (0.7), health (0.6), yeast (0.6), skin (0.6), pain (0.5), liver (0.5), physician (0.5), she (0.5).	space (1.6), launch (1.4), ics.uci.edu (1.2), moon (1.1), orbit (1.0), mars (1.0), pat (1.0), nasa (0.9), dietz (0.9), shuttle (0.8), solar (0.7), command (0.7), henry (0.6), fred (0.6), gamma (0.6), sci.space (0.6), pluto (0.6), satellite (0.6), dc-x (0.6), nicho (0.6).	graphics (2.0), phigs (1.4), image (1.4), images (1.4), xv (1.3), tiff (1.2), polygons (1.1), comp.graphics (1.0), mpeg (1.0), format (1.0), siggraph (1.0), povray (0.9), quicktime (0.8), bockamp (0.8), surface (0.8), animation (0.8), iges (0.8), studio (0.8), jpeg (0.8), pov (0.7).
SA	disease (6.2), msg (5.9), doctor (5.3), treatment (5.3), counselor (4.5), medical (4.2), cancer (4.1), geb (3.5), photography (3.3), gordon (3.3), health (3.1), banks (3.0), symptoms (2.8), dyer (2.7), needles (2.7), epilepsy (2.6), pain (2.4), prozac (2.3), patients (2.2), physician (2.2).	space (19.6), nasa (8.0), orbit (6.8), launch (5.9), moon (5.7), sci.space (5.4), pat (4.8), dietz (4.2), flight (3.7), solar (3.5), fred (3.5), rockets (3.4), spacecraft (3.3), lunar (3.1), nick (2.9), satellite (2.9), shuttle (2.9), mars (2.8), funding (2.8), henry (2.7).	graphics (14.3), image (8.8), images (6.7), animation (5.8), pov (5.4), polygon (4.6), tiff (4.4), impulse (4.3), studio (4.0), comp.graphics (4.0), vesa (3.7), viewer (3.6), surface (3.5), mpeg (3.5), routine (3.4), format (3.3), algorithm (3.2), daemon (3.1), fractal (3.1), polygons (3.0).

Fig 4. The 20 most relevant words per class for the BoW/SVM model. The words are listed in decreasing order of their LRP(first row)/SA(second row) relevance (value indicated in parentheses). Underlined words do not occur in the training data.

transferred to a new document representation, which we denote as document summary vector, when building this vector as a weighted combination of word2vec embeddings (see Eq 6 and Eq 7) or as a combination of one-hot word vectors (see Eq 8). We compare the weighting scheme based on the LRP relevances to the following baselines: SA relevance, TFIDF and uniform weighting (see Section 3.5).

The two-dimensional PCA projection of the summary vectors obtained via the CNN2 resp. the SVM model, as well as the corresponding TFIDF/uniform weighting baselines are shown in Fig 5. In these visualizations we group the 20Newsgroups test documents into six top-level categories (the grouping is performed according to the dataset website), and we color each document according to its *true* category (note however that, as mentioned earlier, the relevance decomposition is always performed in an unsupervised way, i.e., with the ML model's *predicted* class). For the CNN2 model, we observe that the two-dimensional PCA projection reveals a clear-cut clustered structure when using the element-wise LRP weighting for semantic extraction, while no such regularity is observed with uniform or TFIDF weighting. The word-level LRP or SA weightings, as well as the element-wise SA weighting present also a form of bundled layout, but not as dense and well-separated as in the case of element-wise LRP. For the SVM model, the two-dimensional visualization of the summary vectors exhibits partly a cross-shaped layout for LRP and SA weighting, while again no particular structure is observed for TFIDF or uniform semantic extraction. This analysis confirms the observations made in the last Section, namely that the neural network outperforms the BoW/SVM classifier in terms of subjective human interpretability. Fig 5 furthermore suggests that LRP provides semantically more meaningful semantic extraction than the baseline methods. In the next Section we will confirm these observations quantitatively.

6.4 Quantitative Evaluation

6.4.1 How well does LRP identify relevant words ?

In order to quantitatively validate the hypothesis that LRP is able to identify words that either *support* or *inhibit* a specific classifier decision, we conduct several word-deleting experiments on the CNN models using LRP scores as relevance indicator. More specifically, in accordance with the word-level relevances we delete a sequence of words from each document, re-classify the documents with "missing words", and report the classification accuracy as a function of the number of deleted words. The word-level relevances are computed on the original documents (with no words deleted). For the deleting experiments, we consider only 20Newsgroups test documents that have a length greater or equal to 100 tokens (after prepocessing), this amounts to 4963 test documents, from which we delete up to 50 words. For deleting a word we simply set the corresponding word embedding to zero in the CNN input. Moreover, in order to assess the pertinence of the LRP decomposition method as opposed to alternative relevance models, we additionally perform word deletions according to SA word relevances, as well as random deletion. In the latter case we sample a random sequence of 50 words per document, and delete the corresponding words successively from each document. We repeat the random sampling 10 times, and report the average results (the standard deviation of the accuracy is less than 0.0141 in all our experiments). We additionally perform a biased random deletion, where we sample only among words contained in the word2vec vocabulary (this way we avoid deleting words we have already initialized as zero-vectors as they are outside the word2vec vocabulary, however as our results show this biased deletion is almost equivalent to strict random selection).

As a first deletion experiment, we start with the subset of test documents that are initially correctly classified by the CNN models, and successively delete words in decreasing order of their LRP/SA word-level relevance. In this first deletion experiment,

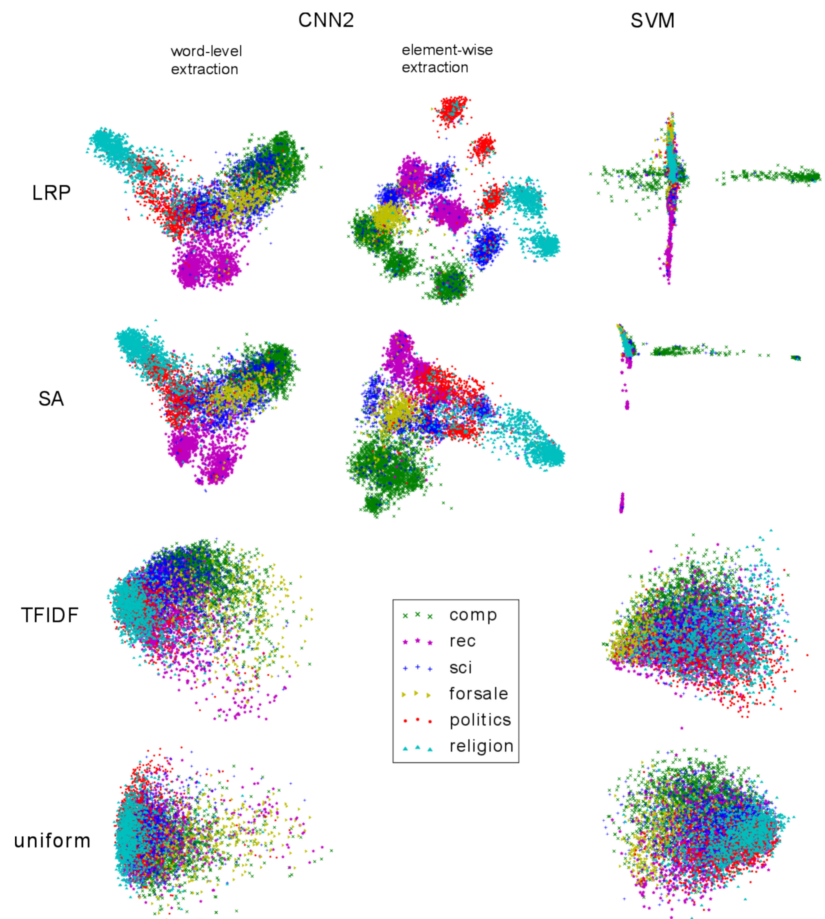

Fig 5. PCA projection of the summary vectors of the 20Newsgroups test documents. The LRP/SA based weightings were computed using the ML model's predicted class, the colors denote the true labels.

the LRP/SA relevances are computed with the true document class as target class for the relevance decomposition. In a second experiment, we perform the opposite evaluation. Here we start with the subset of initially falsely classified documents, and delete successively words in increasing order of their relevance, while considering likewise the true document class as target class for the relevance computation. In the third experiment, we start again with the set of initially falsely classified documents, but now delete words in decreasing order of their relevance, considering the classifier's initially predicted class as target class for the relevance decomposition.

Fig 6 summarizes the resulting accuracies when deleting words from the CNN1, CNN2 and CNN3 input documents respectively (each row in the figure corresponds to one of the three deletion experiments). Note that we do not report results for the BoW/SVM model, as our focus here is the comparison between LRP and SA and not between different ML models. Besides we note that intrinsic validation is also not the right tool for comparing the BoW/SVM and the CNN models, as the resulting accuracies are not directly comparable (deleting a word from the bag-of-words document representation has a different effect than setting a word to zero in the CNN input). Through successive deletion of either "positive-relevant" words in decreasing order of their LRP relevance, or of "negative-relevant" words in increasing order of their LRP relevance, we confirm that both extremal LRP relevance values capture pertinent information with respect to the classification problem. Indeed in all deletion experiments, we observe the most pronounced decrease resp. increase of the classification accuracy when using LRP as relevance model. We additionally note that SA, in contrast to LRP, is largely unable to provide suitable information to pinpoint words that speak *against* a specific classification decision. Instead it appears that the lowest SA relevances (which mainly correspond to zero-valued relevances) are more likely to identify words that have no impact on the classifier decision at all, as this deletion scheme has even less impact on the classification performance than random deletion when deleting words in increasing order of their relevance, as shown by the second deletion experiment.

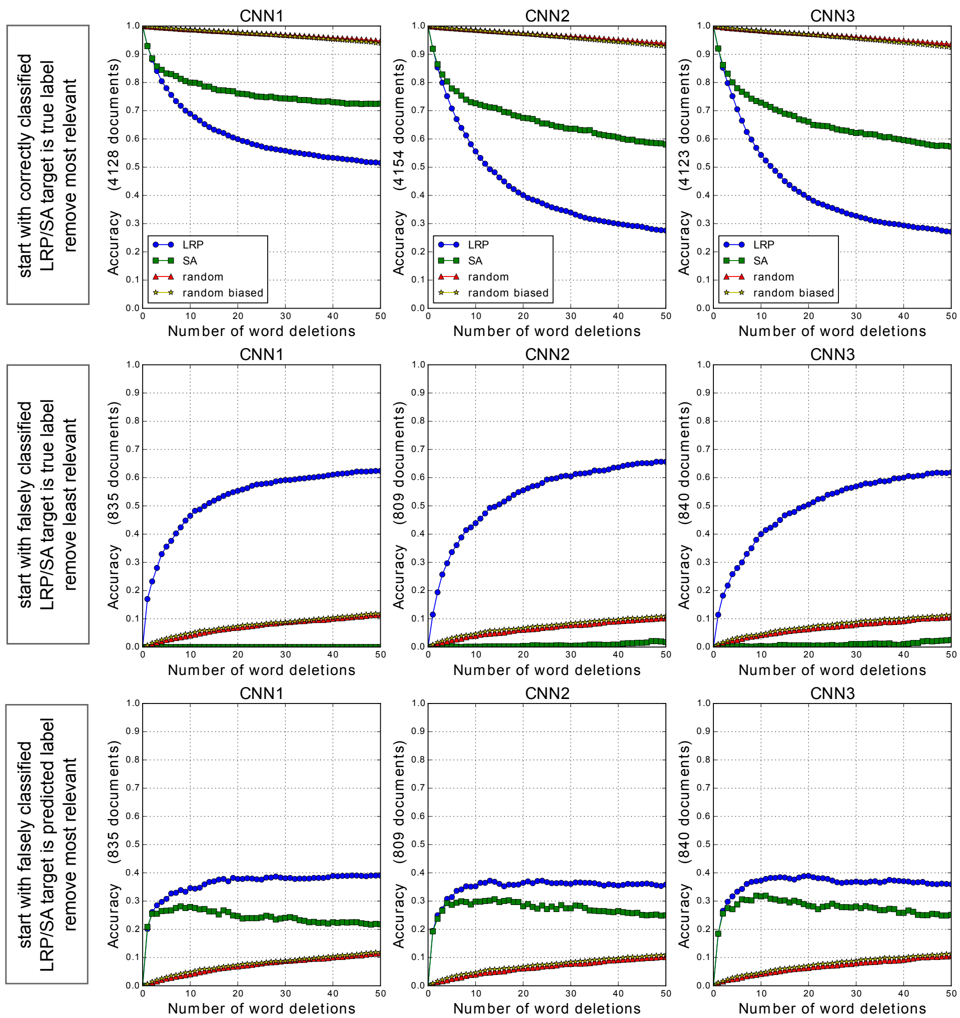

Fig 6. Word deletion experiments for the CNN1, CNN2 and CNN3 model. The LRP/SA target class is either the true document class, and words are deleted in decreasing (first row, lower curve is better) resp. increasing (second row, higher curve is better) order of their LRP/SA relevance, or else the target class is the predicted class (third row, higher curve is better) in which case words are deleted in decreasing order of their relevance. Random (biased) deletion is reported as average over 10 runs.

When comparing the different CNN models, we observe that the CNN2 and CNN3 models, as opposed to CNN1, produce a steeper decrease of the classification performance when deleting the most relevant words from the initially correctly classified documents, both when considering LRP as well as SA as relevance model, as shown by the first deletion experiment. This indicates that the networks with greater filter sizes are more sensitive to single word deletions, most likely because during these deletions the meaning of the surrounding words becomes less obvious to the classifier. This also provides some weak evidence that, while CNN2 and CNN3 behave similarly (which suggests that a convolutional filter size of two is already enough for the considered classification problem), the learned filters in CNN2 and CNN3 do not only focus on isolated words but additionally consider bigrams or trigrams of words, as their results differ a lot from the CNN1 model in the first deletion experiment.

6.4.2 Quantifying the Explanatory Power

In order to quantitatively evaluate and compare the ML models in combination with a relevance decomposition or *explanation* technique, we apply the evaluation method described in Section 4.2. That is, we compute the accuracy of an external classifier (here KNN) on the classification of document summary vectors (obtained with the ML model's predicted class). For these experiments we remove test documents which are empty or contain only one word after preprocessing (this amounts to remove 25 documents from the 20Newsgroups test set). The maximum KNN mean accuracy obtained when varying the number of neighbors K (corresponding to our EPI metric of explanatory power) is reported for several models and explanation techniques in Table 2.

Table 2. Results averaged over 10 random data splits. For each semantic extraction method, we report the dimensionality of the document summary vectors, the explanatory power index (EPI) corresponding to the *maximum* mean KNN accuracy obtained when varying the number of neighbors K, the corresponding standard deviation over the multiple data splits, and the hyperparameter K that led to the maximum accuracy.

Dim	Semantic Extraction		Explanatory Power Index	KNN Parameter
300	word2vec/CNN1	LRP (ew)	**0.8045** (± 0.0044)	$K = 10$
		SA (ew)	0.7924 (± 0.0052)	$K = 9$
		LRP	0.7792 (± 0.0047)	$K = 8$
		SA	0.7773 (± 0.0041)	$K = 6$
	word2vec/CNN2	LRP (ew)	**0.8076** (± 0.0041)	$K = 10$
		SA (ew)	0.7993 (± 0.0045)	$K = 9$
		LRP	0.7847 (± 0.0043)	$K = 8$
		SA	0.7767 (± 0.0053)	$K = 8$
	word2vec/CNN3	LRP (ew)	**0.8034** (± 0.0039)	$K = 13$
		SA (ew)	0.7931 (± 0.0048)	$K = 10$
		LRP	0.7793 (± 0.0037)	$K = 7$
		SA	0.7739 (± 0.0054)	$K = 6$
	word2vec	TFIDF	**0.6816** (± 0.0044)	$K = 1$
		uniform	0.6208 (± 0.0052)	$K = 1$
70631	BoW/SVM	LRP	**0.7978** (± 0.0048)	$K = 14$
		SA	0.7837 (± 0.0047)	$K = 17$
	BoW	TFIDF	**0.7592** (± 0.0039)	$K = 1$
		uniform	0.6669 (± 0.0061)	$K = 1$

When comparing the best CNN based weighting schemes with the corresponding TFIDF baseline result from Table 2, we find that all LRP element-wise weighted combinations of word2vec vectors are statistical significantly better than the TFIDF weighting of word embeddings at a significance level of 0.05 (using a corrected resampled t-test [40]). Similarly, in the bag-of-words space, the LRP combination of one-hot word vectors is significantly better than the corresponding TFIDF document representation with a significance level of 0.05. Lastly, the best CNN2 explanatory power index is significantly higher than the best SVM based explanation at a significance level of 0.10. Although the CNN2 model has only a slightly superior result over the SVM model, the document vectors obtained through the former model have a much lower dimensionality than those extracted via the SVM.

In Fig 7 we plot the mean accuracy of KNN (averaged over ten random test data splits) as a function of the number of neighbors K, for the CNN2 and the SVM model, as well as the corresponding TFIDF/uniform weighting baselines (for CNN1 and CNN3 we obtained similar plot as for CNN2). One can further see from Fig 7 that (1) (element-wise) LRP provides consistently better semantic extraction than all baseline methods and that (2) the CNN2 model has a greater explanatory power than the BoW/SVM classifier since it produces semantically more meaningful summary vectors for KNN classification.

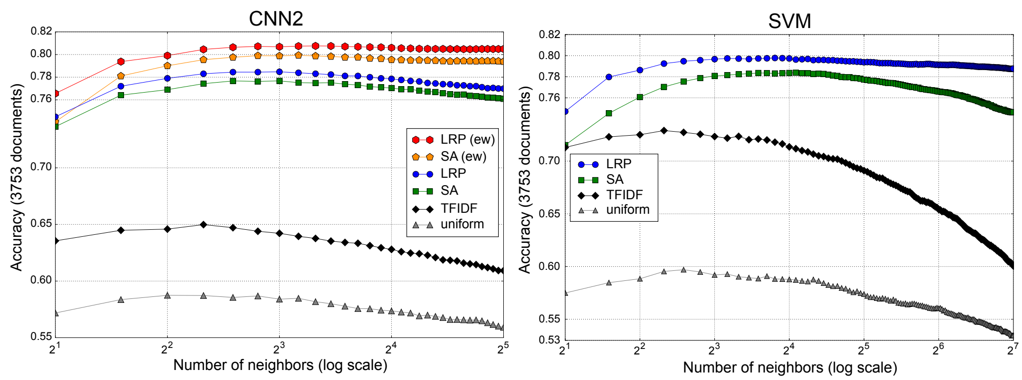

Fig 7. KNN accuracy when classifying the document summary vectors. The accuracy is computed on one half of the 20Newsgroups test documents (other half is used as neighbors). Results are averaged over 10 random data splits.

Overall the good performance, both qualitatively as well as quantitatively, of the element-wise combination of word2vec embeddings according to the LRP relevance illustrates the usefulness of LRP for extracting a new vector-based document representation preserving semantic neighborhood regularities in the input feature space.

7 Conclusion

We have demonstrated qualitatively and quantitatively that LRP constitutes a useful tool for identifying, both for fine-grained analysis at the document level and as a dataset-wide introspection across documents, words that are important to a classifier's decision. This knowledge enables us to broaden the scope of applications of standard machine learning classifiers like support vector machines or neural networks, by extending the primary classification result with additional information linking the classifier's decision back to components of the input, in our case words in a document. Furthermore, based on LRP relevance, we have introduced a new way of condensing the semantic information contained in word embeddings (such as word2vec) into a document vector representation that can be used for nearest neighbors classification, and that leads to better performance than standard TFIDF weighting of word embeddings. The resulting document vector is the basis of a new measure of model explanatory power which was proposed in this work, and its semantic properties could find applications in various visualization and search tasks, where the document similarity is expressed as a dot product between vectors. Another future application of

LRP-based semantic extraction could be the aggregation of word representations into sub-document representations like phrases, sentences or paragraphs.

Our work is a first step toward applying the LRP decomposition to the NLP domain, and we expect this technique to be also suitable for types of applications that are based on other neural network architectures such as character-based or recurrent network classifiers, or on other types of classification problems (e.g. sentiment analysis). More generally, LRP could contribute to the design of more accurate and efficient classifiers, not only by inspecting and leveraging the input space relevances, but also through the analysis of intermediate relevance values "at classifier hidden layers".

Acknowledgments

This work was supported by the German Ministry for Education and Research as Berlin Big Data Center BBDC, funding mark 01IS14013A, by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (No. 2017-0-00451) and by DFG. KRM thanks for partial funding by the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology in the BK21 program.

References

1. Jones KS. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation. 1972;28:11–21.

2. Salton G, Wong A, Yang CS. A Vector Space Model for Automatic Indexing. Communications of the ACM. 1975;18(11):613–620.

3. Hasan KS, Ng V. Conundrums in Unsupervised Keyphrase Extraction : Making Sense of the State-of-the-Art. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters (COLING); 2010. p. 365–373.

4. Aggarwal CC, Zhai C. A Survey of Text Classification Algorithms. In: Aggarwal CC, Zhai C, editors. Mining Text Data. Springer; 2012. p. 163–222.

5. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed Representations of Words and Phrases and their Compositionality. In: Advances in Neural Information Processing Systems 26 (NIPS); 2013. p. 3111–3119.

6. Bengio Y, Ducharme R, Vincent P, Jauvin C. A Neural Probabilistic Language Model. Journal of Machine Learning Research (JMLR). 2003;3:1137–1155.

7. Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research (JMLR). 2011;12:2493–2537.

8. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng A, et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics; 2013. p. 1631–1642.

9. Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. In: Computer Vision - ECCV 2014: 13th European Conference; 2014. p. 818–833.

10. Simonyan K, Vedaldi A, Zisserman A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In: International Conference on Learning Representations Workshop (ICLR); 2014.

11. Schütt KT, Arbabzadah F, Chmiela S, Müller KR, Tkatchenko A. Quantum-Chemical Insights from Deep Tensor Neural Networks. Nature Communications. 2017;8:13890.

12. Landecker W, Thomure MD, Bettencourt LMA, Mitchell M, Kenyon GT, Brumby SP. Interpreting Individual Classifications of Hierarchical Networks. In: IEEE Symposium on Computational Intelligence and Data Mining (CIDM); 2013. p. 32–38.

13. Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. PLOS ONE. 2015;10(7):e0130140.

14. Montavon G, Lapuschkin S, Binder A, Samek W, Müller KR. Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognition. 2017;65:211–222.

15. Samek W, Binder A, Montavon G, Lapuschkin S, Müller KR. Evaluating the visualization of what a Deep Neural Network has learned. IEEE Transactions on Neural Networks and Learning Systems. 2017;PP(99):1–14.

16. Arbabzadah F, Montavon G, Müller KR, Samek W. Identifying Individual Facial Expressions by Deconstructing a Neural Network. In: Pattern Recognition - 38th German Conference, GCPR 2016. vol. 9796 of LNCS. Springer; 2016. p. 344–354.

17. Sturm I, Lapuschkin S, Samek W, Müller KR. Interpretable Deep Neural Networks for Single-Trial EEG Classification. Journal of Neuroscience Methods. 2016;274:141–145.

18. Lapuschkin S, Binder A, Montavon G, Müller KR, Samek W. Analyzing Classifiers: Fisher Vectors and Deep Neural Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 2912–2920.

19. Poulin B, Eisner R, Szafron D, Lu P, Greiner R, Wishart DS, et al. Visual Explanation of Evidence in Additive Classifiers. In: Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence (IAAI). AAAI Press; 2006. p. 1822–1829.

20. Baehrens D, Schroeter T, Harmeling S, Kawanabe M, Hansen K, Müller KR. How to Explain Individual Classification Decisions. Journal of Machine Learning Research (JMLR). 2010;11:1803–1831.

21. Strumbelj E, Kononenko I. An Efficient Explanation of Individual Classifications using Game Theory. Journal of Machine Learning Research (JMLR). 2010;11:1–18.

22. Ribeiro MT, 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; 2016. p. 1135–1144.

23. Turner R. A Model Explanation System: Latest Updates and Extensions. arXiv. 2016;1606.09517.

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

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

26. Montavon G, Samek W, Müller KR. Methods for Interpreting and Understanding Deep Neural Networks. arXiv. 2017;1706.07979.

27. Denil M, Demiraj A, de Freitas N. Extraction of Salient Sentences from Labelled Documents. arXiv. 2014;1412.6815.

28. Li J, Chen X, Hovy E, Jurafsky D. Visualizing and Understanding Neural Models in NLP. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT); 2016. p. 681–691.

29. Arras L, Horn F, Montavon G, Müller KR, Samek W. Explaining Predictions of Non-Linear Classifiers in NLP. In: Proceedings of the 1st Workshop on Representation Learning for NLP. Association for Computational Linguistics; 2016. p. 1–7.

30. Arras L, Montavon G, Müller KR, Samek W. Explaining Recurrent Neural Network Predictions in Sentiment Analysis. arXiv. 2017;1706.07206.

31. Ding Y, Liu Y, Luan H, Sun, M. Visualizing and Understanding Neural Machine Translation. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics; 2017.

32. Kim Y. Convolutional Neural Networks for Sentence Classification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014. p. 1746–1751.

33. Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. In: International Conference on Learning Representations Workshop (ICLR); 2013.

34. Mnih A, Hinton G. Three New Graphical Models for Statistical Language Modelling. In: Proceedings of the International Conference on Machine Learning (ICML); 2007. p. 641–648.

35. Mnih A, Teh YW. A fast and simple algorithm for training neural probabilistic language models. In: Proceedings of the International Conference on Machine Learning (ICML); 2012. p. 1751–1758.

36. Lapuschkin S, Binder A, Montavon G, Müller KR, Samek W. The Layer-wise Relevance Propagation Toolbox for Artificial Neural Networks. Journal of Machine Learning Research. 2016;17(114):1–5.

37. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323:533–536.

38. Zhang X, Zhao J, LeCun Y. Character-level Convolutional Networks for Text Classification. In: Advances in Neural Information Processing Systems 28 (NIPS); 2015. p. 649–657.

39. Bottou L, Lin CJ. Support Vector Machine Solvers. In: Bottou L, Chapelle O, DeCoste D, Weston J, editors. Large Scale Kernel Machines. Cambridge, MA.: MIT Press; 2007. p. 1–27.

40. Nadeau C, Bengio Y. Inference for the Generalization Error. Machine Learning. 2003;52(3):239–281.