"What Does My Classifier Learn?" A Visual Approach to Understanding Natural Language Text Classifiers

Jonas Paul Winkler and Andreas Vogelsang

Technische Universität Berlin, Germany,
Daimler Center for Automotive IT Innovations
jonas.winkler@tu-berlin.de, andreas.vogelsang@tu-berlin.de

Abstract. Neural Networks have been utilized to solve various tasks such as image recognition, text classification, and machine translation and have achieved exceptional results in many of these tasks. However, understanding the inner workings of neural networks and explaining why a certain output is produced are no trivial tasks. Especially when dealing with text classification problems, an approach to explain network decisions may greatly increase the acceptance of neural network supported tools. In this paper, we present an approach to visualize reasons why a classification outcome is produced by convolutional neural networks by tracing back decisions made by the network. The approach is applied to various text classification problems, including our own requirements engineering related classification problem. We argue that by providing these explanations in neural network supported tools, users will use such tools with more confidence and also may allow the tool to do certain tasks automatically.

Keywords: visual feedback, neural networks, artificial intelligence, machine learning, natural language processing, explanations, requirements engineering

1 Introduction

Artificial Neural Networks have become powerful tools for performing a wide variety of tasks such as image classification, text classification, and speech recognition. Within the natural language processing community, neural networks have been used to tackle various tasks such as machine translation, sentiment analysis, authorship attribution, and also more fundamental tasks such as part-of-speech tagging, chunking, and named entity recognition [4]. More recently, convolutional neural networks that were almost exclusively used for image processing tasks were also adapted to solve natural language processing tasks [5].

However, neural networks usually do not explain why certain decisions are made. Especially when incorporating a trained neural network in a tool with heavy user interaction, users may need to understand why the network produced certain results in order to make better decisions. If such an explanation is not

provided, users may be frustrated because they do not understand the reasons behind some decisions and consequently do not profit from the tool. In order to provide such explanations, additional techniques are required.

In this paper, we propose a technique to trace back decisions made by convolutional neural networks and provide visual feedback to explain these decisions. The basic idea is to identify which parts of the network contributed the most to a decision and to further identify the corresponding words in the input sentence. We use that information to highlight certain parts within the input sentence.

The remainder of this paper is structured as follows. Our approach for computing *Document Influence Matrices* and creating visual representations is presented in Section 3. In Section 4, we evaluate our approach on three different datasets. We describe how we apply our approach on a specific use case in Section 5. Section 6 concludes.

2 Related Research

Understanding neural networks and providing explanations for network decisions is a well-established area of research. Early approaches use fuzzy logic and create rules from trained networks, ultimately explaining how input neurons relate to output neurons [2, 3]. In contrast to these works, our approach operates on individual classification results, similar to what has been presented in [1, 8]. These methods usually exploit the weights of a trained network to compute an explanation gradient. These gradients may then be used to identify individual inputs as particularly important.

Application of this type of approach to image classification tasks are presented in [11] and [10]. Here, the authors visualize intermediate networks layers to understand what the network has learned.

Within the natural language processing community, approaches to explain natural language classifiers exist as well. In [6], the authors visualize network structures to show the impact of salient words on the classification outcome.

3 Document Influence Matrices

In this section, we present our approach to compute *Document Influence Matrices*, which contain information about how strongly each word in an input document contributes to a decision made by a convolutional neural network. These matrices may then be used to create visual representations, which highlight individual words that contributed most to the classification outcome.

3.1 Classifying Text using CNNs

This section describes the operations performed by convolutional neural networks as proposed by [5]. The architecture of the network is displayed in Fig. 1.

(1) Word embedding. The first step is to transform documents into numerical representations. This is done by using the word2vec [7] word embedding

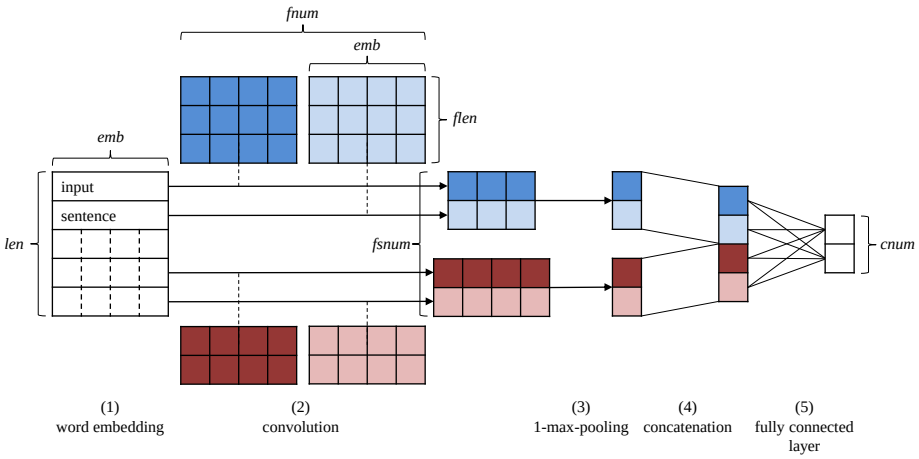

Fig. 1. Network architecture as proposed by [5]

technique. Word2vec maps individual words to vectors $v \in \mathbb{R}^{emb}$, where *emb* is the number of dimensions used for the word embedding. The word vectors are obtained by training the word2vec model on a corpus. Furthermore, each input document may be transformed into a matrix $s \in \mathbb{R}^{len,emb}$, where *len* is the length of the input document.

(2) Convolution. Next, a convolution operation applies filters to the input document matrix. A filter is a matrix $f \in \mathbb{R}^{flen,emb}$ of trainable weights, where *flen* is the length of the filter. A filter set is a rank-3 tensor $fs \in \mathbb{R}^{fnum,flen,emb}$ and contains *fnum* filters of the same length. Multiple sets of filters with varying filter lengths may be used. In Fig. 1, two sets of filters (blue and red) with filter length 3 and 2 are illustrated. A filter set is applied to a document matrix by moving each filter as a sliding window over the matrix, producing a single value at each position resulting in a matrix $v^{(1)} \in \mathbb{R}^{fnum,len+1-flen}$:

$$v_{i,j}^{(1)} = \sigma\left(\left(\sum_{k=1}^{flen}\sum_{l=1}^{emb} fs_{i,k,l} \cdot s_{j+k-1,l}\right) + b_i\right) \tag{1}$$

In this equation, σ is an arbitrary activation function, such as *sigmoid* or *rectified linear units* and $b \in \mathbb{R}^{fnum}$ holds a trainable bias for each filter.

(3) 1-max-pooling. 1-max-pooling reduces $v^{(1)}$ from the previous step to a vector $v^{(2)} \in \mathbb{R}^{fnum}$ by selecting the maximum value of each filter:

$$v_i^{(2)} = \max\left(v_{i,1}^{(1)}, v_{i,2}^{(1)}, ..., v_{i,len+1-flen}^{(1)}\right) \tag{2}$$

(4) Concatenation. The computations described in step 2 and 3 are performed once for each filter set, resulting in multiple matrices $v^{(2,i)}$. Given *fsnum* filter sets, these are concatenated to form a feature vector $v^{(3)} \in \mathbb{R}^{fsnum \cdot fnum}$:

$$v^{(3)} = v^{(2,1)} \| v^{(2,2)} \| ... \| v^{(2,fsnum)} \tag{3}$$

(5) Fully connected layer. This feature vector is used as the input for a fully connected layer, in which values from the feature vector are associated with the output classes. Given *cnum* output classes, the output $v^{(4)} \in \mathbb{R}^{cnum}$ of this layer is computed as follows:

$$v_i^{(4)} = \left(\sum_{j=1}^{fsnum \cdot fnum} w_{j,i} \cdot v_j^{(3)} \right) + b_i \tag{4}$$

In this equation, $w \in \mathbb{R}^{fsnum \cdot fnum, cnum}$ is a matrix of trainable weights and $b \in \mathbb{R}^{cnum}$ is a vector of trainable biases.

Finally, the values computed for the output classes are transformed into true probabilities by applying the *softmax* function.

After the network is trained on a training dataset, its filters have learned to identify output classes based on the presence or absence of certain words and word groups. For any given input document, the network yields probabilities for all classes, indicating which class the input document belongs to.

3.2 Computing Document Influence Matrices

A Document Influence Matrix is a matrix $DIM \in \mathbb{R}^{len, cnum}$. Each value $DIM_{i,c}$ indicates how strongly the word at position i influences the network to classify the document as class c. A high value at $DIM_{i,c}$ means that the word at position i strongly suggests the classification of the document as class c, whereas a value close to 0 means that there is no correlation between the word and the class.

To compute a Document Influence Matrix for a particular input document, we analyze the output of the network and trace the decisions made by the network at each layer back to its input. This is done by consecutively computing intermediate influence matrices (IIM) at each major step.

(1) Examining the network output. When the network has reliably classified a document as a particular class, the output of the network for this true class will be close to 1, whereas the outputs for the other classes will be close to 0. The definition of the *softmax* function implies that the output of the fully connected layer for the true class is much higher than any of the outputs for the false classes.

(2) Tracing back fully connected layers. Let us examine Equation 4 for computing the output of a fully connected layer in more detail. It may also be written as follows:

$$v_i^{(4)} = w_{1,i} \cdot v_1^{(3)} + w_{2,i} \cdot v_2^{(3)} + ... + w_{m,i} \cdot v_m^{(3)} + b_i \tag{5}$$

Since the result of this equation for the true class is much higher than the result for any of the other classes, it must have more $w_{j,i} \cdot v_j$ summands with high values. Therefore, a $w_{j,i} \cdot v_j$ summand with a high value has a strong influence

towards classifying an input document as a particular class, whereas a $w_{j,i} \cdot v_j$ summand with a negative value has a strong influence against classifying an input document as a particular class.

Furthermore, each $w_{j,i} \cdot v_j$ summand may be associated with one particular input neuron of the layer. The influence of one particular input neuron on a certain class is thus defined by the sum of all related $w_{j,i} \cdot v_j$ summands. The matrix $IIM^{(fcl)} \in \mathbb{R}^{inum,cnum}$ for a fully connected layer with *inum* input neurons and *onum* output neurons is computed as follows:

$$IIM_{i,c}^{(fcl)} = \sum_{j=1}^{onum} IIM_{j,c} \cdot w_{i,j} \cdot v_i \tag{6}$$

(3) Tracing back concatenation. In Section 3.1, step 4, multiple feature vectors are concatenated to form a single feature vector. To trace back the concatenation operation, the previous IIM is sliced into multiple pieces, resulting in *fsnum* matrices $IIM^{(concat,fsnum)} \in \mathbb{R}^{fnum,cnum}$:

$$IIM_{i,c}^{(concat,n)} = IIM_{(n-1)\cdot fnum+i,c} \tag{7}$$

Each of these matrices is traced back further individually.

(4) Tracing back 1-max-pooling. 1-max-pooling is used to select the highest out of several values resulting from the application of one particular filter everywhere in the input document. Only this highest value has impact on the classification outcome and thus receives all influence. The rank-3 tensor $IIM^{(1max)} \in \mathbb{R}^{fnum,len+1-flen,cnum}$ is computed as follows:

$$IIM_{i,j,c}^{(1max)} = \begin{cases} IIM_{j,c} & v_{i,j}^{(1)} = v_j^{(2)} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

(5) Tracing back convolutional layers. Tracing back convolutional layers follows the same principles as tracing back fully connected layers. The rank-3 tensor $IIM^{(conv)} \in \mathbb{R}^{len,emb,cnum}$ is computed as follows:

$$IIM_{i,j,c}^{(conv)} = \sum_{k=1}^{fnum} \sum_{l=1}^{len+1-flen} IIM_{k,l,c} \cdot s_{i,j} \cdot \begin{cases} fs_{k,i+1-l,j} & i+1-l \in [1, flen] \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

(6) Putting it all together. So far we have defined operations for computing IIMs for each operation of the neural network. A DIM may be computed using the following procedure:

1. Define an initial influence matrix $IM \in \mathbb{R}^{cnum,cnum}$ where $IM_{c,c} = out_c$ for each class c. All other fields are zero. Given the output of the last layer of the network, this matrix states that each output has influence on its respective class only.
2. Compute Intermediate Influence Matrices according to the architecture of the network. At the concatenation step, continue individually for every split.

3. Perform element-wise summation of all $IIM^{(conv)}$ matrices:

$$IIM^{(sum)} = \sum_{i=1}^{fsnum} IIM^{(conv,i)} \tag{10}$$

The matrix $IIM^{(sum)}$ holds influence values of all word2wec components of each word of the input document on each of the output classes.

4. Reduce the matrix $IIM^{(sum)}$ to a matrix $DIM \in \mathbb{R}^{len,cnum}$:

$$DIM_{i,c} = \sum_{j=1}^{emb} IIM_{i,j,c}^{(sum)} \tag{11}$$

This matrix finally contains individual influence values of each word on each output class.

3.3 Creating a Visual Representation from Document Influence Matrices

A visual representation of a Document Influence Matrix may be created by following these steps:

1. Normalize the matrix so that all of its values are within the range $[0, 1]$.
2. Select a distinct color for each class.
3. For each word in the document, select the highest value from the normalized matrix, select the color of the class corresponding to this value, use the value as the colors transparency, and use that color as the background for the word.

Table 1 shows examples from the datasets used in our experiments.

4 Experiments and Results

To prove the effectiveness of our approach, we conducted two different experiments on multiple datasets.

The datasets used in these experiments are listed in Table 2. The *Requirements* dataset contains natural language requirements written in German and taken from multiple requirement specification documents describing automotive components and systems, such as wiper control, outside lighting, etc. It also contains an equal number of *Information* objects (e.g. examples, informal descriptions, and references). More information on this dataset is given in Section 5.

The *Question* dataset[1] contains short natural language questions asking for different kinds of answers, such as locations, numeric answers (dates, numbers), persons, entities, descriptions, and abbreviations. The task is to detect what kind of information a question asks for.

The *Movie Reviews* dataset[3] consists of single line movie reviews. These are either positive or negative.

[1] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[2] These examples were translated from German into English.

[3] https://www.cs.cornell.edu/people/pabo/movie-review-data/

Table 1. Examples

Dataset	Requirements[2]
Classes	requirement information
requirement	the duration until the switch is recognized as hanging must be a configurable parameter .
information	the component conditionally drives an external fan . this fan is required for active ventilation of the headlight .
Dataset	Questions
Classes	description numeric human entity abbreviation location
description	why did the chicken cross the road ?
numeric	how long does it take sunlight to reach earth ?
human	which of the following people is not associated with andy warhol ?
Dataset	Movie Reviews
Classes	positive negative
positive	both a successful adaptation and an enjoyable film in its own right .
negative	just a bunch of good actors flailing around in a caper that's neither original nor terribly funny .
negative	. . . unbearably lame .
positive	it's a minor comedy that tries to balance sweetness with coarseness , while it paints a sad picture of the singles scene .

Table 2. Datasets

Dataset	**Classes**	**Train Examples**	**Test Examples**	**Classification Accuracy**
Requirements	2	1854	206	84.95 %
Questions	6	4906	546	83.70 %
Movie Reviews	2	9595	1067	73.29 %

Table 3. Requirements dataset: Most often highlighted words

Class	**Most often highlighted words**
Requirement	must, of, must (plural), contractor, be, shall, may, client, that, shall (plural), active, to, supply voltage, a, agreed, must (old German spelling)
Information	described, can (plural), two, can, the, defined, requirements, in, only, included, description, vehicle, so, require, deliver, specification

4.1 Analyzing Most Often Highlighted Words

The goal of our first experiment is to prove that our approach is capable of finding and highlighting relevant words within input documents. Since our approach assigns high influence values to important words and low influence values to unimportant words, aggregating highly influential words per class should yield lists of words commonly used to describe individuals of a particular class.

We computed Document Influence Matrices for all examples in the test set of each dataset. Then, for each individual word used in the dataset, we aggregated the total influence of that word across all Document Influence Matrices per class. This results in lists of words per class and their influence on that class (i.e. how often are they highlighted in the test set). The words were sorted descending by their accumulated influence.

Table 3 shows the results for the *Requirements* dataset. Words commonly used to write requirements such as *must* and *shall* and their various German variations (i.e. "The system shall ..." and "The contractor must ensure that ...") are highlighted most frequently, which is exactly what we expected. A commonly used information-type sentence in our dataset is a reference to another document (i.e. "further requirements are *specified* in ...") and as such, the word *specified* is highlighted very often. The other words are not particularly significant. Furthermore, many sentences not containing certain requirement keywords are information, although it is hard to specifically tell why exactly a sentence is an information based on its words.

Table 4. Questions dataset: Most often highlighted words

Class	**Most often highlighted words**
Abbreviation	stand, does, abbreviation, mean, is, for, an, number, beer, fame, term
Description	how, what, is, do, are, the, does, why, a, origin, did, of, difference, mean
Entity	what, the, name, was, is, of, a, fear, are, for, does, did, to, do, color
Human	who, what, was, name, the, of, and, actor, company, school, tv
Location	where, country, city, can, capital, the, was, state, are, what, does, in, of
Numeric	how, many, when, did, does, was, year, long, the, do, average, is, of

The results for the *Questions* dataset are displayed in Table 4. Most of the classes can be identified very easy based on certain combinations of question words and pronouns. Questions starting with "how many" are most likely asking for numeric answers, "who was" is a strong indicator for a question asking for a human's name, and sentences containing "in which city" usually ask for a specific location. The results in the table show that these terms are indeed the most frequently highlighted.

Table 5 contains the most often highlighted words for the *Movie Reviews* dataset. The list of positive words contains expected words such as *best*, *worth*,

Table 5. Movie Reviews dataset: Most often highlighted words

Class	Most often highlighted words
Positive	that, is, of, a, has, film, us, with, best, makes, an, worth, performances, it's, fun, who, good, documentary, funny
Negative	but, too, and, more, bad, no, so, movie, in, or, doesn't, just, only, about, the, there's, i, are, isn't

fun, *good* and *funny*, but also words that we would not associate with positive reviews at all (e.g. *that*, *is*, *of*, etc.). It seems that these words are often used in phrasings with a positive sentiment. The word *but* is highlighted most in negative reviews since it is often used when something is criticized (e.g., "The plot was okay, *but* ..."). Also, different negative words such as *bad*, *doesn't*, *no*, and *isn't* appear in this list since these are often used to express negative sentiment.

4.2 Measuring the Quality of the Visual Representations

We have conducted an empirical study to assess the quality of the visual representations created by our approach in more detail.

For each example in the test sets, we manually decided which words should be most important for deciding the class and then compared our expectation to the actual visual representation. Each example is then assigned one of the following quality categories:

- **Perfect match.** Our approach highlighted exactly all words that we considered to be important. No additional and unexpected words are highlighted.
- **Partial match.** Only some of the words we considered to be important are highlighted. No additional and unexpected words are highlighted.
- **Mixed match.** The visual representation highlights some or all of the expected words. Additionally, some words, which we considered to be irrelevant, are also highlighted.
- **Mismatch.** The visual representation did not highlight any expected words.

After conducting this evaluation on all three datasets, we accumulated the count of perfect, partial, mixed, and mismatches in total. In order to better understand the results, we also separately accumulated the counts on correctly/incorrectly classified examples and on examples with prediction probabilities greater/less than 95%. The results are displayed in Fig. 2.

On the *Requirements* dataset, 57% of the examples are highlighted according to our expectations. On 16% of the examples, the visual representations are not useful at all. On the visual representations of the remaining 27% of the examples either not all expected words are highlighted or some unexpected words are highlighted as well.

The separated results on correctly and incorrectly classified examples reveals that our approach naturally fails to provide reasonable visual representations for

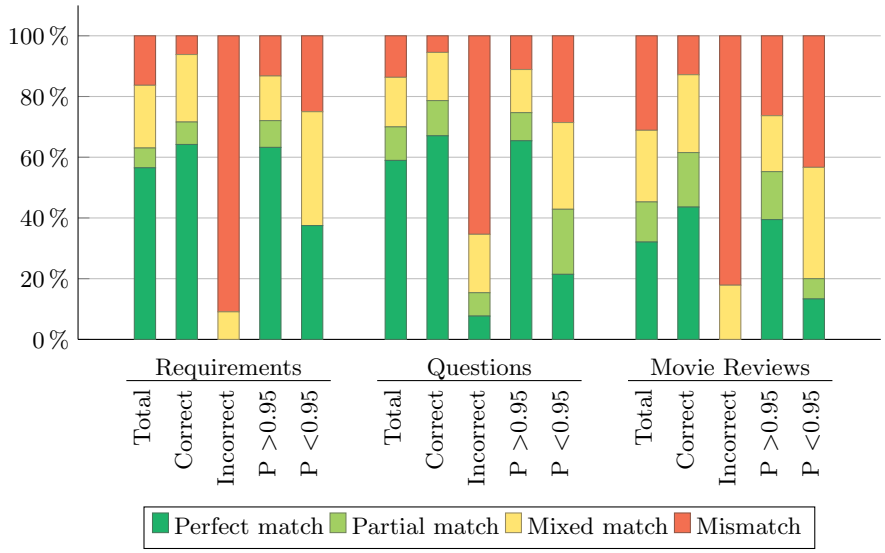

Fig. 2. Relation between important words and words highlighted by the approach.

incorrectly classified examples. Contrariwise, for almost all (94%) correctly classified examples, the visual representation contained all or at least some relevant words. Inspecting the prediction probabilities also reveals an interesting correlation: The visual representation of an example with a high prediction probability is more likely to be correct than the visual representation of an example with a low prediction probability.

Within the *Question* dataset, about 59% of all examples are accurately highlighted. 11% of the examples are partially highlighted, whereas within the remaining 30%, irrelevant words are highlighted. Just as with the *Requirements* dataset, the overall quality of the visual representations on correctly classified examples and examples with high prediction probabilities is much higher than the overall quality on incorrectly classified examples and examples with low prediction probabilities.

Results on the *Movie Reviews* dataset are considerably worse than the results on the other two datasets. Only 32% of the examples had completely correct visual representations. In 69% of all cases, at lease some relevant words are highlighted. In 55% of all examples, irrelevant words are highlighted as well. We suspect that this is due to the lower accuracy of the model compared with the other two models and due to the higher complexity of the classification task.

Based on the individual results for these three datasets, we made the following general observations:

– The overall quality of the visual representations strongly correlates with the performance of the trained model. The higher the accuracy of the model on the test set, the better the overall quality, i.e. more relevant and less irrelevant words are highlighted.

– The quality of an individual examples' visual representation correlates with the probability on the predicted class. Higher probabilities usually lead to visual representations closer to what we expected.

5 Application to Natural Language Requirements Classification

We have applied the approach presented in our previous works [9] in an industrial setting to automatically classify natural language requirements, which are written down in requirements specification of automotive systems. These requirements documents specify the properties and behavior of systems. Most of the content consists of natural language sentences. Apart from formal and legally binding requirements, these documents also contain auxiliary information such as clarifications, chapter overviews, and references to other documents. This kind of content is not legally binding and as such is not relevant to e.g. third party suppliers who implement a system. To better differentiate between requirements and additional information, each content element has to be explicitly labeled as either a *requirement* or an *information*. This task is error-prone and time-consuming, as it is performed manually by requirements engineers.

We have build a tool that assists the requirements engineer in performing this task. This tool analyzes a requirements documents, classifies each relevant (i.e. only sentence, no headings, figures etc.) content element as either requirement or information, and issues warnings when a content element is not classified as predicted by the tool. Alongside these warnings, we used the approach presented in this paper to highlight the words responsible for classifying a content element as either information or requirement.

By using the visualization approach presented in this paper, the requirements engineers were able to better understand why the tool made specific decisions. If a user does not understand the classification outcome, the user is pinpointed to phrases or individual words that contribute to the outcome. The user can interpret these results and react to them in several ways:

– The user recognizes that the chosen phrasing may not be suitable for the kind of content the user wants to express. Therefore, the user may revise the sentence and thus increase the quality of the specification.
– The user recognizes that rules for classifying content items are used inconsistently between requirements engineers. Therefore, the user initiates a discussion for specific phrases or formulations w.r.t. their classification.
– The user recognizes that the tool has learned something wrong or that the learned decision procedure does not match the current classification rules (which may change over time). Therefore, the user marks this result as a false positive. The neural network may adapt to this decision and keep up to date (c.f. active learning).

6 Conclusions and Future Work

In this paper, we have presented an approach to create visual representations for natural language sentences, which explain classifier decisions by highlighting particularly important words. As shown in our evaluation, these visual representations are accurate as long as the underlying model is accurate.

As we integrated our approach in our requirement classification tool, we have received very positive feedback from industry experts. We argue that an approach to explain network classification decisions to users increases both usability and acceptance of a tool. A visual approach as presented in this paper is sufficient although any other approach may work as well.

In the future, we plan to conduct a more extensive field study on a large user base to better understand the qualities and limitations of our approach.

References

1. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.R.: How to Explain Individual Classification Decisions. J. Mach. Learn. Res. 11, 1803–1831 (2010)
2. Benítez, J.M., Castro, J.L., Requena, I.: Are Artificial Neural Networks Black Boxes? IEEE Transactions on Neural Networks 8(5), 1156–1164 (1997)
3. Castro, J.L., Mantas, C.J., Benítez, J.M.: Interpretation of artificial neural networks by means of fuzzy rules. IEEE Transactions on Neural Networks 13(1), 101–116 (2002)
4. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural Language Processing (Almost) from Scratch. J. Mach. Learn. Res. 12, 2493–2537 (2011)
5. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1746–1751 (2014)
6. Li, J., Chen, X., Hovy, E., Jurafsky, D.: Visualizing and Understanding Neural Models in NLP. In: Proceedings of NAACL-HLT. pp. 681–691. Association for Computational Linguistics, San Diego, California (2016)
7. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. arXiv preprint abs/1301.3781 (2013)
8. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why Should I Trust You?" Explaining the Predictions of Any Classifier. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1135–1144. ACM, New York (2016)
9. Winkler, J.P., Vogelsang, A.: Automatic Classification of Requirements Based on Convolutional Neural Networks. In: 3rd IEEE International Workshop on Artificial Intelligence for Requirements Engineering (AIRE) (2016)
10. Yosinski, J., Clune, J., Nguyen, A.M., Fuchs, T., Lipson, H.: Understanding Neural Networks Through Deep Visualization. In: Deep Learning Workshop, 31 st International Conference on Machine Learning (2015)
11. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014, Lecture Notes in Computer Science, vol. 8689, pp. 818–833. Springer International Publishing (2014)