EMBEDDIA

Cross-Lingual Embeddings for Less-Represented Languages in European News Media

Research and Innovation Action
Call: H2020-ICT-2018-1
Call topic: ICT-29-2018 A multilingual Next generation Internet
Project start: 1 January 2019 Project duration: 36 months

D1.9: Final interpretability and visualisation technology (T1.4)

Executive summary

The objectives of workpackage WP1 of the EMBEDDIA project are to advance cross-lingual word embedding technologies, together with methods for deep learning, and methods for explanation and visualisation of their outputs. The aim of T1.4 is to address advancement of explanation and visualization technologies for text-based deep learning approaches. Deep neural networks are currently the most successful approach for natural language processing and understanding. Unfortunately, the internal functioning of neural networks is incomprehensible for humans. In this report, we extend our prior work on adapting explanation of machine learning models to the specifics of text processing with deep neural networks. We present TransSHAP, an adaption of explanation methods for the most successful transformer neural network, and a novel visualisation based on the this method. As current explanation methods can be misled by adversarial attacks, we have developed a new method that makes explanations more robust by using advanced sampling techniques within explanation techniques SHAP, LIME, and IME. To generate semantic explanations of explanation method output we have developed ReEX, an explanation generalization procedure that makes use of anthologies to produce semantic explanations from the output of explanation techniques, such as SHAP. Visualisation design and validation is improved when task based requirements are clearly established. We present a task based analysis of visualisation techniques for explanation methods and investigate their applicability for explanations of text based models. We have extended our previous work on AttViz, a neural network inspection tool, with an offline component that enables more computationally intensive analysis, which was not supported by the previous online version.

Partner in charge: UEDIN

Project co-funded by the European Commission within Horizon 2020 Dissemination Level		
PU	Public	PU
PP	Restricted to other programme participants (including the Commission Services)	–
RE	Restricted to a group specified by the Consortium (including the Commission Services)	–
CO	Confidential, only for members of the Consortium (including the Commission Services)	–

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153

Deliverable Information

Document administrative information	
Project acronym:	**EMBEDDIA**
Project number:	**825153**
Deliverable number:	**D1.9**
Deliverable full title:	**Final interpretability and visualisation technology**
Deliverable short title:	**Final interpretability and visualisation technology**
Document identifier:	**EMBEDDIA-D19-FinalInterpretabilityAndVisualisationTechnology-T14-submitted**
Lead partner short name:	**UEDIN**
Report version:	**submitted**
Report submission date:	**30/06/2021**
Dissemination level:	**PU**
Nature:	**R = Report**
Lead author(s):	**Shane Sheenan (UEDIN), Saturnino Luz (UEDIN), Marko Robnik-Šikonja (UL)**
Co-author(s):	**Enja Kokalj (JSI), Blaž Škrlj (JSI), Senja Pollak (JSI), Domen Vreš (UL)**
Status:	**__ draft, __ final, _x_ submitted**

The EMBEDDIA Consortium partner responsible for this deliverable has addressed all comments received. Changes to this document are detailed in the change log table below.

Change log

Date	Version number	Author/Editor	Summary of changes made
21/05/2021	v1.0	Shane Sheehan (UEDIN)	Initial version
01/06/2021	v1.1	Shane Sheehan (UEDIN)	Internal review draft
01/06/2021	v1.2	Marko Robnik-Šikonja (UL)	Improvements to the text
02/06/2021	v1.3	Shane Sheehan (UEDIN)	Add ReEX section
02/06/2021	v1.4	Shane Sheehan (UEDIN)	Shorten introduction and robustness section
03/06/2021	v1.5	Shane Sheehan (UEDIN)	Add future work and improve text
08/06/2021	v1.6	Hannu Toivonen (UH)	Internal Review
09/06/2021	v1.7	Matthew Purver (QMU)	Internal Review
22/06/2021	v1.8	Shane Sheehan (UEDIN)	Ready for quality control
22/06/2021	prefinal	Nada Lavrač (JSI)	Quality control finalized
28/06/2021	final	Shane Sheehan (UEDIN)	Final version
30/06/2020	submitted	Tina Anžič (JSI)	Report submitted

Table of Contents

1. Introduction 5
1.1 Interpretability and visualization of machine learning models 5
1.2 Contributions and structure of the deliverable 6
2. Background and related work 6
2.1 Deep neural networks for text classification 6
2.2 Explanation methods for text classification 7
2.3 Explanation visualisations for text classification 8
3. Explanation methods adapted for text classification 10
3.1 TransSHAP: The SHAP method adapted for BERT 11
3.2 Robustness of explanations and malicious attacks 12
3.3 ReEx: Semantic Reasoning from Model-Agnostic Explanations 16
4. Contributions to visualization techniques for text classification 17
4.1 TransSHAP: Visualization of a prediction explanation for the BERT model 18
4.2 Visual variable analysis of SHAP visualisations for text 19
4.2.1 SHAP visualisation tasks 19
4.2.2 Data abstraction for SHAP text visualisation 24
4.2.3 Visual encoding applicability to text explanations 25
4.2.4 Suggested Encoding designs and improvements 30
4.3 AttViz library: statistical analysis of the attention space 30
4.3.1 Dissecting the token space 32
4.3.2 Visualization of attention head focus 32
5. Conclusions 32
6. Associated outputs 35
References 36
Appendix A: BERT meets Shapley: Extending SHAP Explanations to Transformer-based Classifiers 39
Appendix B: Better Sampling in Explanation Methods can Prevent Dieselgate-Like Deception 45
Appendix C: Semantic Reasoning from Model-Agnostic Explanations. 69
Appendix D: Exploring Neural Language Models via Analysis of Local and Global Self-Attention Spaces 75

List of abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Networks
BERT	Bidirectional Encoder Representations from Transformers
DNN	Deep Neural Network
GLUE	General Language Understanding Evaluation
gLIME	Generator based LIME
gSHAP	Generator based SHAP
IME	Interactions-based Method for Explanation
JSON	JavaScript Object Notation
LIME	Local Interpretable Model-agnostic Explanations
ML	Machine Learning
MCD-VAE	Variational autoencoder with Monte Carlo dropout
NLP	Natural Language Processing
RNN	REcurrent Neural Networks
RBF	Radial Basis Functions
rbfDataGen	RBF data generator
SVM	Support Vector Machines
SHAP	SHapley Additive exPlanations
TransSHAP	The SHAP method adapted for BERT
TreeEnsemble	Random forest ensemble data generator
TEnsFillIn	TreeEnsemble which fills in values

1 Introduction

The objectives of workpackage WP1 of the EMBEDDIA project are to advance cross-lingual word embedding technologies, together with methods for deep learning, and methods for explanation and visualisation of their outputs. The aim of T1.4 is to address advancement of explanation and visualization technologies for text-based deep learning approaches.

Recent developments in artificial intelligence (AI) and in particular in machine learning (ML) have brought this technology into the center of public interest and have increased the requirements for its transparency – it is natural that people affected by automated decisions of algorithms want to get feedback and understand the reasoning process and biases of the underlying models.

Two types of technological approaches to increased transparency exist: interpretability approaches and visualization of ML models. These two approaches, focused on natural language processing (NLP) approaches, are the topic of this report, which summarizes the work performed in task *T1.4 Interpretability and visualization* of the EMBEDDIA project.

1.1 Interpretability and visualization of machine learning models

Machine learning models are a crucial component of natural language processing applications. Unfortunately, most of the top performing machine learning models are "black boxes", in the sense that they do not offer an introspection into their decision processes or provide explanations of their predictions and biases. This is true for Artificial Neural Networks (ANN), Support Vector Machines (SVM), and all ensemble methods (for example, boosting, random forests, bagging, stacking, and multiple adaptive regression splines). Approaches that do offer an intrinsic introspection, such as decision trees or decision rules, do not perform so well or are not applicable in many cases (Meyer et al., 2003). To alleviate this problem, two types of model explanation techniques have been proposed. The first type of methods are general, based on perturbations of inputs, and therefore applicable to any prediction model. The second type of methods are specific to certain learning methods such as neural networks and exploit the internal information available during training of these methods.

The general explanation approaches try to efficiently capture the causal relationship between inputs and outputs of the given model. To this end, they perturb the inputs in the neighborhood of a given instance to observe effects of perturbations on the model's output. Changes in the outputs are attributed to perturbed inputs and used to estimate their importance for a particular instance. Examples of this approach are methods IME (Štrumbelj & Kononenko, 2010), LIME (Ribeiro et al., 2016), and SHAP (Lundberg & Lee, 2017). These methods can explain models' decision for each individual predicted instance as well as for the model as a whole. The computed impacts of individual inputs can be visualized in the form of histograms. However, these explanation techniques and their visualizations are not adapted to text-based classifiers as their explanations are in the form of lists of impactful words for each individual decision. For texts with their sequential and structurally dependent nature, this is insufficient. In addition, explanation techniques are not adapted to state-of-the-art neural networks such as BERT (Devlin et al., 2019), which use subword input.

Recent research has shown that existing explanation methods that internally generate additional samples, such as IME, LIME and SHAP, mentioned above, are susceptible to adversarial attacks which can trick users into believing that irrelevant attributes are responsible for the given prediction (Slack et al., 2020a). This has put their robustness into question by showing that their outcomes can be manipulated due to inadequate perturbation sampling employed. This weakness would allow owners of sensitive models to deceive inspection and hide potentially unethical or illegal biases existing in their predictive models. Such possibility could undermine public trust in machine learning models and give rise to legal restrictions on their use.

1.2 Contributions and structure of the deliverable

This report describes the results of the work performed in T1.4 from the midpoint of the task at M18 until the task end at M30. The main contributions presented in this report (in the order of appearance) are as follows:

1. Updates to the adaptation of explanation method SHAP to state-of-the-art text classification method BERT, is described in Section 3.1. The paper describing this work (Kokalj et al., 2021) is included as Appendix A.
2. In Section 3.2 we present defense against adversarial attacks on explanation methods; we show that better sampling in these explanation methods prevents malicious manipulations and that the proposed sampling using data generators improves LIME and SHAP's robustness. The paper describing this work (Vreš & Robnik-Šikonja, 2021) is included as Appendix B.
3. ReEx (Reasoning with Explanations), a semantic explanation method which makes use of ontologies, is presented in Section 3.3. This method is applicable to explanations generated by arbitrary instance-level explainers, such as SHAP. The paper describing this work (Stepišnik Perdih et al., 2021) is included as Appendix C.
4. The adaptation and evaluation of visualizations for text classification method BERT is described in Section 4.1. The paper describing this work (Kokalj et al., 2021) is included as Appendix A.
5. A study of SHAP visualisation tasks is presented in Section 4.2. We identify core tasks supported by visualisation for tabular data and investigate their applicability for text based SHAP explanations.
6. An offline analysis component for our AttViz system described in Section 4.3; this component enables analysis of BERT's self attention heads in an offline setting where more computationally intensive analysis is possible. This work is part of the associated outputs, listed in Section 6. The paper describing this work (Škrlj et al., 2021) is included as Appendix D.

This task generates explanations and visualisations of the models employed in WP3, WP4, and WP5. The technologies will be integrated into frameworks developed in WP6.

2 Background and related work

In this section, we present a short overview of explanation and visualization techniques, with a focus on deep neural networks and text.

2.1 Deep neural networks for text classification

Deep learning (LeCun et al., 2015; Goodfellow et al., 2016) differs from other machine learning algorithms by allowing computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

Text is usually treated as a sequence by deep neural networks. Sequences can be of different length and typically there is some dependency between different positions in a sequence (e.g., a verb in a sentence may determine the subsequent choice of nouns). A standard choice of neural network architecture for sequences is recurrent neural networks (RNN) in which the state at each point in the sequence depends on not only the current input but also the previous state. The information from the previous processing steps persists in the network, effectively allowing the network to memorize previous processing, which is well suited for sequences. RNNs are very effective in processing speech, text, signals, and other sequential data. RNNs also introduce many challenges, for example the convergence of learning to stable weights is much slower.

Recently, BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), a new state-of-the-art deep neural network approach to language modeling, text classification, and contextual embeddings was introduced. BERT generalises the idea of language models to masked language models—inspired by cloze (gap filling) tests—which test the understanding of a text by removing a certain portion of words that the participant is asked to replace. The masked language model randomly masks some of the tokens from the input, and the task of the language model is to predict the missing token based on its neighbourhood. BERT uses the transformer ANN architecture (Vaswani et al., 2017), uses both left and right context in predicting the masked word, and further introduces the task of predicting whether two sentences appear in a sequence. The input representations of BERT are sequences of tokens representing subword units. Some very common words are kept as single tokens, others are split into subwords (e.g., common stems, prefixes, suffixes—if needed down to single-letter tokens). The original BERT project offers pre-trained English, Chinese and multilingual models; the latter is trained on 104 languages simultaneously. BERT has shown excellent performance on 11 NLP tasks: 8 from the GLUE (General Language Understanding Evaluation) benchmark (Wang et al., 2018), question answering, named entity recognition, and common-sense inference.

Rather than training an individual classifier for every classification task, which is resource and time expensive, a pre-trained multilingual BERT language model is used and fine-tuned on a specific task. Trained on huge text collections, BERT stores general language representation that is successfully exploited in many tasks, Frequently, this approach requires less task-specific data.

The use of BERT in a token classification task only requires adding a final layer (number of neurons equals the number of class values in the intended task) with softmax activation, and connections between its last hidden layer and the new neuron. To classify a sequence, one usually takes the vector for the special class token (CLS) before the classification layer to reduce the dimensionality. The fine-tuning process is then applied to either only the last layer of the network, or, more frequently, to the whole network. In the latter case all parameters of BERT and new class-specific weights are fine-tuned jointly to maximize the log-probability of the correct labels.

2.2 Explanation methods for text classification

Due to recent successes of Deep Neural Networks (DNNs) in image recognition and NLP, several explanation methods specific to these two application areas emerged. For example, in language processing, Arras et al. (2017) applied layer-wise relevance propagation to a convolutional neural network. The explanations indicate how much individual words contribute to the overall classification decision. In this section we focus on a general class of explanation methods, which are applicable to all types of classifiers but may need specific adaptations for use in text classification. While we demonstrate our adaptations of these methods on the BERT model, these adaptations are applicable to other DNN models as well.

General explanation methods can be applied to any classification model which makes them a useful tool both for interpreting models (and their predictions) and comparing different types of models. By modification of feature values of interest, what-if analysis is also supported. Such methods cannot exploit any model-specific properties (e.g., gradients in ANN) and are limited to perturbing the inputs of the model and observing changes in the model's output (Robnik-Šikonja & Kononenko, 2008; Lemaire et al., 2008; Štrumbelj & Kononenko, 2010). Most explanation methods can provide two types of explanations for prediction models: explanation of predictions for individual instances, and model explanations. Model explanations work by summarizing a representative sample of instance explanations to show the properties of the whole model.

The key idea of the perturbation-based explanation method is that the contribution of a particular input value (or set of values) can be captured by "hiding" that input and observing how the output of the model changes. As such, the key component of general explanation methods is the expected conditional prediction – the prediction where only a subset of the input variables is known. Let Q be a subset of the set of input variables $Q \subseteq S = \{X_1, \ldots, X_a\}$. Let $p_Q(y_k|x)$ be the expected prediction for x, conditional on

knowing only the input variables represented in Q:

$$p_Q(y_k|x) = \mathbf{E}(p(y_k)|X_i = x_{(i)}, \forall X_i \in Q). \quad (1)$$

Therefore, $p_S(y_k|x) = p(y_k|x)$. The difference between $p_S(y_k|x)$ and $p_Q(y_k|x)$ is a basis for explanations. In practical settings, the classification function of the model is not known – one can only access its prediction for any vector of input values. Therefore, exact computation of $p_Q(y_k|x)$ is not possible and sampling-based approximations are used.

In principle, to understand the behaviour of a prediction model, one would have to observe its behavior for all subsets of input features and their values. Such a procedure demands 2^a steps, where a is the number of attributes (i.e. features), and results in the exponential time complexity. A solution was proposed in (Štrumbelj & Kononenko, 2010) by observing that the contribution of each variable corresponds to the Shapley value for the coalitional game of a players. Here, players correspond to input features, and the coalitional game corresponds to the prediction of an individual instance.

The original Shapley values deal with a coalition of a players that cooperate and together generate a certain overall gain. Shapley values represent a solution to the problem of finding the fairest distribution of gain among all players, which takes into account the importance of each player in obtaining that gain (Shapley, 1953). In the case of explaining a prediction the instance's feature values form a coalition which causes a change in the prediction. This change represents the difference between the prediction for this instance and the expected prediction if no feature values are given (i.e. default class is predicted). The gain is divided among feature values in a way that reflects their impact (i.e. average marginal contribution across all possible sub-coalitions) on the change in the prediction.

The methods discussed in this work, IME (Štrumbelj & Kononenko, 2010), SHAP (Lundberg & Lee, 2017), and LIME (Ribeiro et al., 2016), are the state-of-the-art explanation methods. They are all based on the Shapley value approximation principle (Lundberg & Lee, 2017). They estimate the impact of a particular feature on the prediction of a given instance by perturbing similar instances. For textual problems the perturbation process is nontrivial, as the generation of new perturbed text instances may produce completely uninformative texts.

2.3 Explanation visualisations for text classification

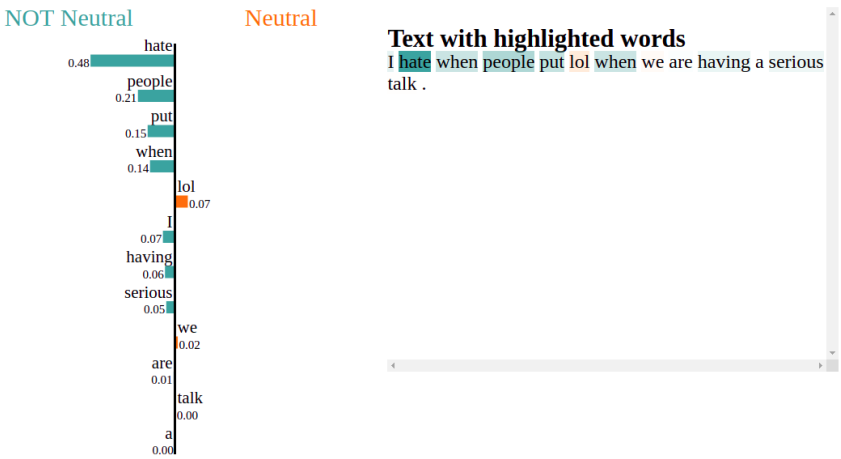


Figure 1: Visualization of prediction explanation with LIME.

The visualization approaches implemented for the explanation methods LIME and SHAP are primarily designed for explanations of tabular data and images, see Figure 1 and Figure 13. The visualisation is presented with words along the vertical axis and in order of decreasing explanation contribution. This

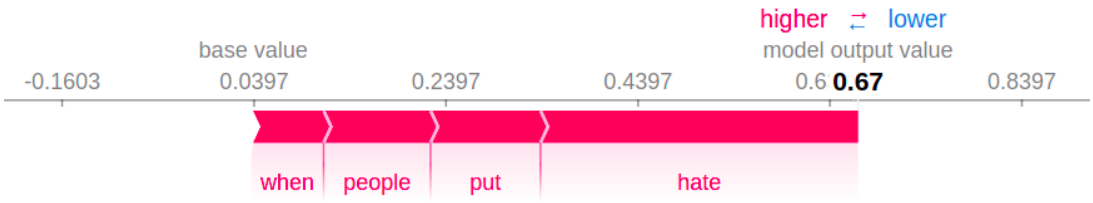


Figure 2: Force visualization of prediction explanation with SHAP, taken from the SHAP python package documentation.

removes the sentence structure and familiar horizontal representation from this visual encoding of a text fragment explanation. Although the visualization for LIME includes adjustments for text data, the resulting explanations are presented in the form of histograms that are sometimes hard to understand. The text fragment is presented along with the bar visualisation and a quantitative color scale is used to overlay the explanation contribution onto the words.

The SHAP force visualization (Lundberg et al., 2018) for the same sentence is illustrated in Figure 2. Here, the features with the strongest impact on the prediction correspond to longer arrows that point in the direction of the predicted class. These force diagrams can be difficult to interpret when sentence order is maintained but some explanation contribution arrows face in different directions. In this case the additive nature of the explanation contribution to a prediction is overlooked. Alternatively, positive and negative contributions can be grouped together increasing the interpretability of the explanation overview but reducing the readability of the text fragment in combination with the explanation contribution values. These force diagrams are also used to create a summary visualisation of multiple explanation instances from a dataset, see Figure 14. These force summaries are easy to construct for tabular data where alignment of the explanation instances can be achieved using common features. However, for textual data the alignment is difficult due the the lack of common features across instances. In Section 4.2 we investigate visualisations available for SHAP explanations of models based on tabular data. The goal of the analysis is ti identify tasks not currently served by existing text based SHAP explanation visualisation.

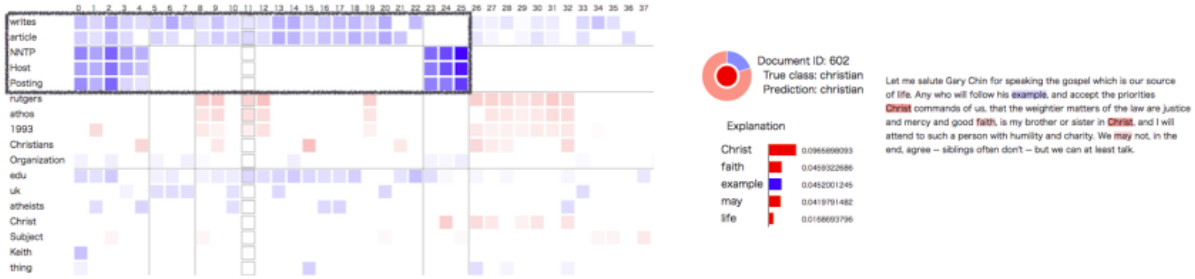


Figure 3: Interactive exploration of LIME explanation instances by Sawada & Toyoda (2019). The left-hand side is a heat map matrix as an overview of a large set of local explanations. The right-hand side is a detail view to display information and the local explanation of the designated instance.

Sawada & Toyoda (2019) propose a heatmap overview of explanation scores for each feature and explanation instance in a dataset, see Figure 3. The heatmap is constructed such that explanation instances which contain a large number of the same features (words) with similar explanation scores are positioned close together. These explanation instances can through selection be further investigated, via a bar plot of explanation score and colored text fragment similar to the visualisation commonly used for LIME explanations (Figure 1).

Recent developments in the generation of hierarchical explanations (Jin et al., 2019; Chen et al., 2020), which can now be applied to BERT, enable the visualisation of explanation hierarchies consisting of

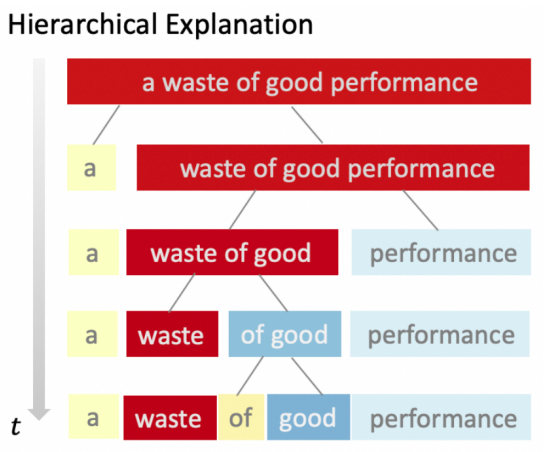


Figure 4: Hierarchical Explanations generated via feature interaction detection by Chen et al. (2020). This hierarchical explanation visualisation presents a set of features in each timestep (t).

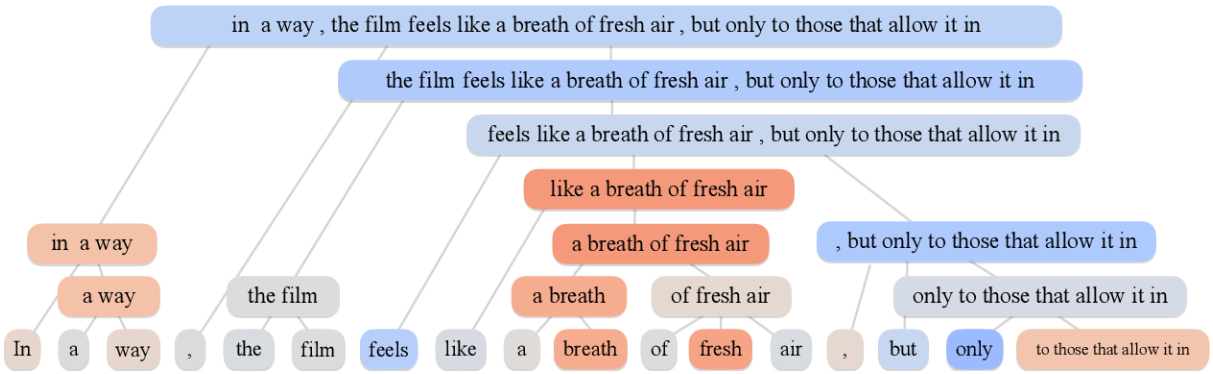


Figure 5: Hierarchical Explanation of a prediction by the BERT Transformer model created by Jin et al. (2019). Explanations are generated for all the phrases on the truncated constituency parsing tree, with positive sentiments shown in red and negative sentiments shown in blue

word and multi-word explanation contributions. These hierarchies are visualised as a tree, where the root node is the full text fragment and the hierarchy of explained features are displayed as subtrees. Each node in the tree contains a text fragment or single word and is colored according to the polarity and size of the associated contribution to the prediction, for example see Figures 4 and 5

3 Explanation methods adapted for text classification

Many modern deep neural networks including transformer networks (Vaswani et al., 2017) (e.g., BERT-like models) split the input text into subword tokens. This is very convenient for morphologically rich languages such as the less-resourced EMBEDDIA languages. However, perturbation-based explanation methods (such as IME, LIME, and SHAP) have problems with text input and in particular subword input, as the credit for a given output cannot be simply assigned to syntactic units such as words, phrases, or sentences.

In Deliverable D1.5 we described the adaptions we have made to these these explanation methods to allow them to work with state-of-the-art text models such as BERT. In Section 3.1 we provide a brief description of our recent work on TransSHAP the SHAP method adapted for BERT. This work builds

upon the the adaptions described in Deliverable D1.5.

Recently, Slack et al. (2020b) put the robustness of explanation methods, such as IME LIME and SHAP, into question by showing that their outcomes can be manipulated due to inadequate perturbation sampling. This weakness would allow owners of sensitive models to deceive inspection and hide potentially unethical or illegal biases existing in their predictive models. Such possibility could undermine public trust in machine learning models and give rise to legal restrictions on their use.

In Deliverable D1.5 we presented a short description of the work in progress on making explanation methods more robust. In Section 3.2, we describe in detail our work on making explanation methods more robust and prevent possible adversarial attacks which can mislead users applying existing explanation methods.

In Section 3.3 we present ReEx, one of the first approaches capable of semantic generalization of model explanations obtained by contemporary tools such as SHAP.

3.1 TransSHAP: The SHAP method adapted for BERT

We propose an adaptation of SHAP for BERT-like classifiers, but the same principles are trivially transferred to LIME and IME. The proposed adaptation is described in (Kokalj et al., 2021), included as Appendix A.

To understand the behavior of a prediction model applied to a single instance, one should observe perturbations of all subsets of input features and their values, which results in exponential time complexity. Štrumbelj & Kononenko (2010) showed that the contribution of each variable corresponds to the Shapley value from the coalition game, where players correspond to input features, and the coalition game corresponds to the prediction of an individual instance. Shapley values can be approximated in time linear to the number of features.

The model-agnostic implementation of the SHAP method, named Kernel SHAP[1], requires a classifier function that returns probabilities. Since SHAP contains no support for BERT-like models that use subword input, we implemented custom functions for preprocessing the input data for SHAP, to get the predictions from the BERT model, and to prepare data for the visualization.

Figure 6 shows the components required by SHAP in order to generate explanations for the predictions made by the BERT model. The text data we want to interpret is used as an input to Kernel SHAP along with the special classifier function we constructed, which is necessary since SHAP requires numerical input in a tabular form.

To achieve this, we first convert the sentence into its numerical representation. This procedure consists of splitting the sentence into tokens and then preprocessing it. The preprocessing of different input texts is specific to their characteristics (e.g., tweets). The result is a list of sentence fragments (with words, selected punctuation marks and emojis), which serves as a basis for word perturbations (i.e. word masking). Each unique fragment is assigned a unique numerical key (i.e. index). We refer to a sentence, represented with indexes, as *an indexed instance*.

In summary, the TransSHAP's classifier function first converts each input instance into a word-level representation. Next, the representation is perturbed in order to generate new, locally similar instances which serve as a basis for the constructed explanation. This perturbation step is performed by the original SHAP. Then the perturbed versions of the sentence are processed with the BERT tokenizer that converts the sentence fragments to sub-word tokens. Finally, the predictions for the new locally generated instances are produced and returned to the Kernel SHAP explainer. With this modification, SHAP is able to compute the features' impact on the prediction (i.e. the explanation).

We demonstrate our TransSHAP method on tweet sentiment classification. The dataset contains 87,428 English tweets with human annotated sentiment labels (positive, negative and neutral). For tweets we

[1] We use the Kernel SHAP implementation of the SHAP method: `https://github.com/slundberg/shap`.

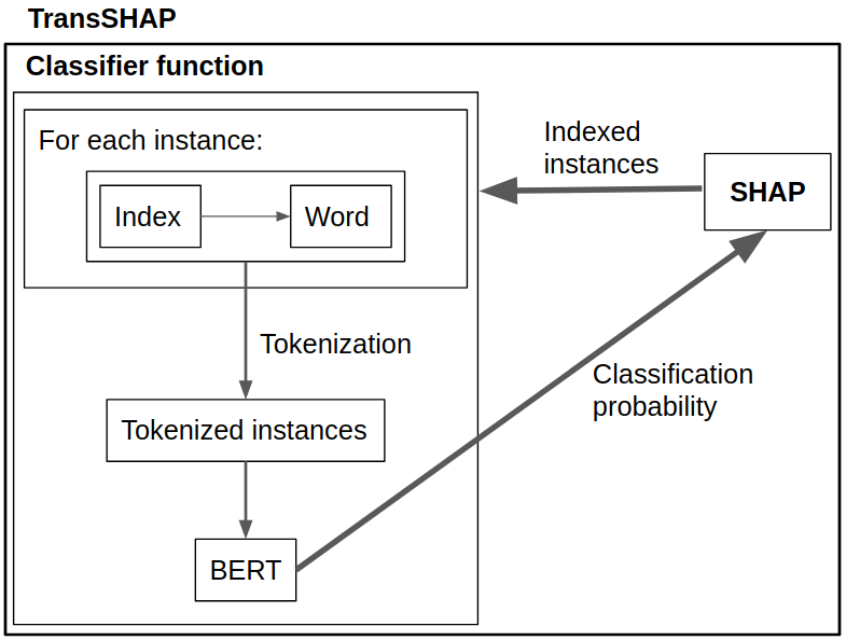


Figure 6: TransSHAP adaptation of SHAP to the BERT language model by introducing our classifier function.

split input instances using the TweetTokenizer function from NLTK library[2], we removed apostrophes, quotation marks and all punctuation marks except for exclamation and question marks. We fine-tuned the CroSloEngual BERT model (Ulčar & Robnik-Šikonja, 2020) on this classification task and the resulting model achieved the classification accuracy of 66.6%.

We have adapted explanation methods to work with text based models such as BERT. To adapt the visualization of predictions for text, we modified the histogram-based visualizations used in IME, LIME and SHAP for tabular data. The details of this visualisation adaption and its evaluation is presented in Section 4.1.

3.2 Robustness of explanations and malicious attacks

In this section we provide an overview of our work in the use of better generators in LIME and SHAP to make them more resistant to the adversarial attacks. The work is detailed in (Vreš & Robnik-Šikonja, 2021), included as Appendix B

Slack et al. (2020b) presented an attack on explanations where the attacker creates an adversarial (biased) model whose behavior it wants to hide from explanation methods, e.g., a racist model that does not grant credits to blacks. As Figure 7 illustrates, a part of the attacker's adversarial model is an unbiased model, which, e.g., does not take disputed attributes such as race into account. The adversarial model tries to manipulate the explanation method by behaving like an unbiased model on perturbed instances used for the explanation. In this case, it uses the unbiased model's output. On instances from the original distributions that are a part of the regular model use, the adversarial model uses the biased model's output. To distinguish between the original and perturbation-sampled data, the adversarial model contains a decision model that selects the right type of model (biased or unbiased) based on the predicted type of instances.

We proposed a defense against adversarial attacks on explanation methods that replaces the problematic perturbation sampling with a better one, thereby making the explanation methods more robust. We want to generate the explanation data in such a way that the attacker cannot determine whether

[2] https://www.nltk.org

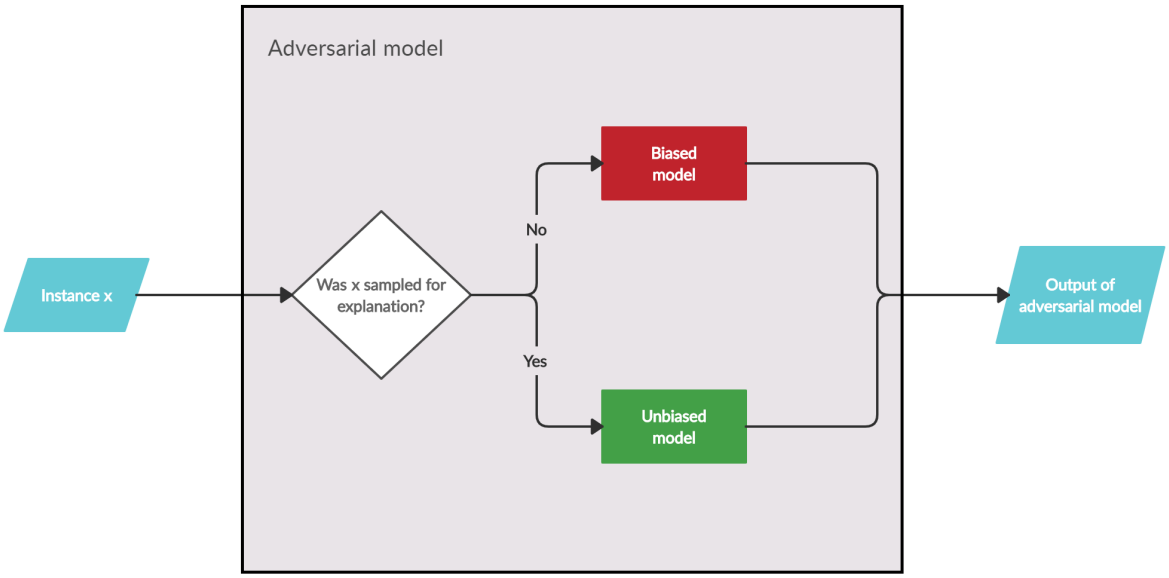


Figure 7: The idea of the attack on explanation methods based on the difference of distributions. The attacker's adversarial model contains both the biased and unbiased model. The decision function that is part of the cheating model decides if the instance is outside the original distribution (i.e. used only for explanation) or an actual instance. If the case of an actual instance, the result of the adversarial model is equal to the result of the biased model, otherwise it is equal to the result of the unbiased model.

an instance is sampled or obtained from the original data. With an improved sampling, the adversarial model shown in Figure 7 shall not determine whether the instance x was generated by the explanation method, or it is the original instance the model has to label. With a better data generator, the adversarial model cannot adjust its output properly and the deception becomes ineffective.

The reason for the weakness of LIME and SHAP is inadequate sampling used in these methods. LIME samples new instances by adding Gaussian noise to the normalized feature values. SHAP samples new examples from clusters obtained in advance with the k-means algorithm from the training data.

Instead of using the Gaussian noise with LIME, we generate explanation samples for each instance with one of the three better data generators We use three data generators based on different algorithms, modeling the distribution of the training set: variational autoencoder with Monte Carlo dropout (Miok et al., 2019), RBF network (Robnik-Šikonja, 2016), and random forest ensemble (Robnik-Šikonja, 2019). In the remainder of the section, we refer to the listed generators consecutively as MCD-VAE, rbfDataGen, and TreeEnsemble.

In the proposed improvment to SHAP, using the MCD-VAE and TreeEnsemble generators, the distribution set D is generated in the explained instance's vicinity. In the sampled instances, some feature values of the explained instance are replaced with the generated values. Here, the well-informed generators consider dependencies between features detected in the original distribution. This will make the distribution of the generated instances very similar to the original distribution. Consequently, the attacker will not be able to recognize the generated instances used in explanations.

We call the improved explanation methods gLIME and gSHAP (g stands for generator-based). Using better generators in the explanation methods, the adversarial model's decision function will less likely determine which predicted instances are original and which are generated for the explanation purposes.

The advantage of generating the distribution set close to the observed instance is demonstrated in Figure 8. The graphs show the PCA-based 2D space of the COMPAS dataset's (Angwin et al., 2016)

evaluation part. The left-hand side shows the SHAP-generated sampled instances using the k-means algorithm (14 clusters determined by the silhouette score ((Rousseeuw, 1987)). The sample produced with the MCD-VAE generator in gSHAP is shown on the right-hand side. This sample is much more similar to the original distribution compared to the SHAP sampling.

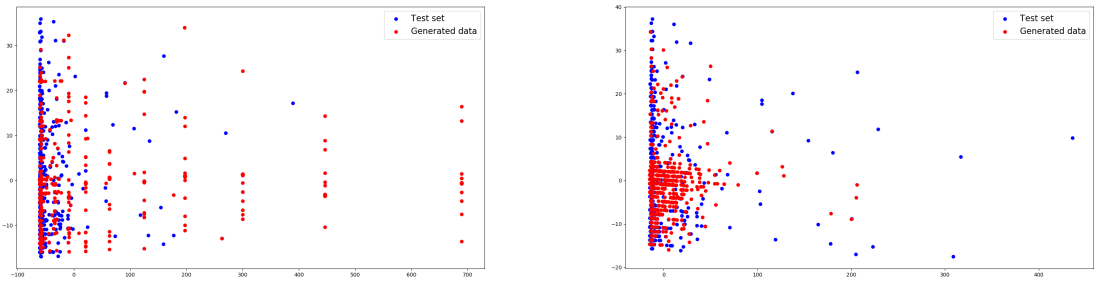


Figure 8: Visual comparison of original and sampled distributions for the COMPASS dataset. The SHAP k-means based generator (left) produces instances less similar to the original data, compared to the MCD-VAE generator (right).

We test the robustness of gLIME, gSHAP, and gIME against the adversarial models. To be more realistic, we equipped the adversarial models with the same improved data generators we used in the explanation methods. It is reasonable to assume that attackers could also use better data generators when preparing their decision function, making their attacks much stronger. As the evaluation metric for the success of deception, we use the proportion of instances where the adversarial model deceived the explanation methods so that they did not detect sensitive attributes as being important in the prediction models.

To evaluate the robustness of the explanation methods with added generators (i.e. gLIME, gSHAP, and gIME), we split the data into training and evaluation set in the ratio $90\% : 10\%$. We used the same training set for the training of adversarial models and explanation methods. We transformed categorical features with more than two values into one-hot-encoded (binary) features.

We simulated the adversarial attack for every combination of generators used in explanation methods and adversarial models (except for method IME, where we did not use rbfDataGen, which cannot generate instances in the neighborhood of a given instance). When testing SHAP, we used two variants of the TreeEnsemble generator: generating new instances around the explained instance and generating new instances according to the whole training set distribution. In the LIME testing, we used only the whole training set variant of TreeEnsemble. In the IME testing, we only used the variant of TreeEnsemble that fills in the hidden values (called TEnsFillIn variant). For training the adversarial models, we used the whole training set based variant of TreeEnsemble, except for IME, where we used the TEnsFillIn variant. These choices reflect the capabilities of different explanation method and attempt to make both defense and attack realistic (as strong as possible).

In all cases, the biased model b was a simple function depending only on the value of the sensitive feature. While this choice is overly simplistic, it allows us to measure the number of cases where the use of the biased classifier was detected as the number of instances where sensitive features were identified as the most important ones. Even in more realistic scenarios, biased classifiers' decisions would rely on values of sensitive features, therefore these features would strongly contribute to their decisions and would be recognized by explanation methods. Note that the choice of the biased classifier has no effect on its deployment as decision model d is trained using all features.

On COMPAS and Communities and Crime (CC) data set (Redmond & Baveja, 2002), we simulated two attacks, one with a single unrelated feature (the result of ψ depends only on the value of the unrelated feature 1), and another one with two unrelated features. In this way, we can compare the success of attacks when the unbiased model is simple and when its decisions are based on interactions of features.

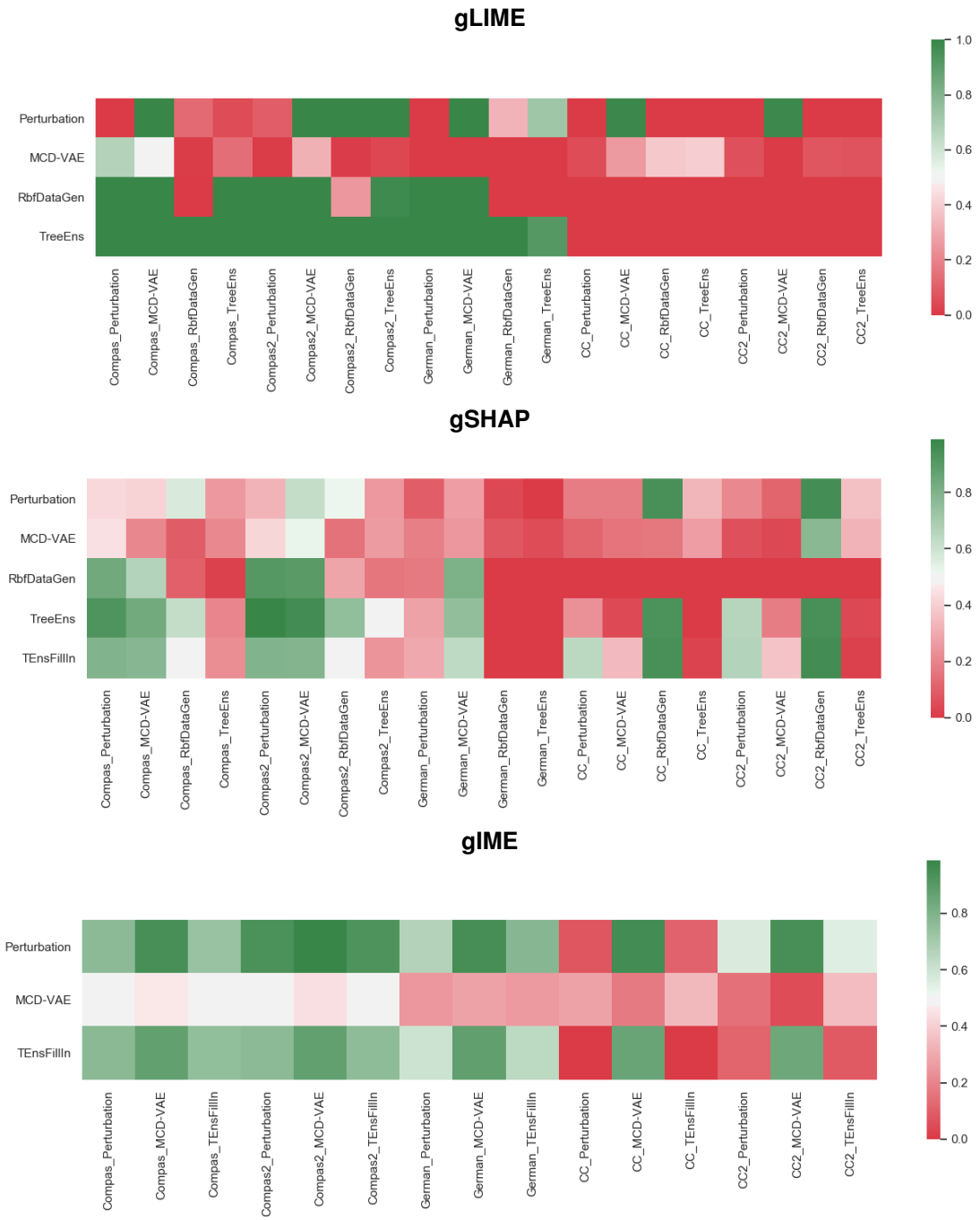


Figure 9: The robustness results for gLIME (top), gSHAP (middle), and gIME (bottom). The graphs show the proportion of evaluation set instances, where the sensitive feature was recognized as the most important by the used explanation method. Rows represent the generators used for explanations. The column labels consist of the name of the data set on which the experiment was performed and the name of the generator used for training of the adversarial model. Compas2 and CC2 denote an attack with two independent features. Perturbation represents the original sampling used in LIME, SHAP, and IME, TEnsFillIn represents the TreeEnsemble variant where new instances are generated around the given one, and TreeEns represents the generation from the whole distribution.

Again, the choice of one or two features determining the results of unbiased model ψ does not affect deployment of the biased classifier as the decision model d is trained on all features, and both unrelated

features are created independently from one another and from all other features.

For every instance in the evaluation set, we recorded the most important feature according to the used explanation method (i.e. the feature with the largest absolute value of the contribution as determined by the explanation method). The results are shown as a heatmap in Figure 9. The green color means that the explanation method was deceived in less than 30% of cases (i.e. the sensitive feature was recognized as the most important one in more than 70 % of the cases as the scale on the heatmap suggests), and the red means that it was deceived in more than 70% of cases (the sensitive feature was recognized as the most important one in less than 30 % of the cases as the scale on the heatmap suggests). We consider deception successful if the sensitive feature was not recognized as the most important by the explanation method (the sensitive features are the only relevant features in biased models b).

The results suggest that gLIME method is more robust with the addition of rbfDataGen and TreeEnsemble than LIME and less robust with the addition of MCD-VAE. This suggests that parameters for MCD-VAE were not well-chosen, and this pattern repeats for other explanation methods. Both TreeEnsemble and rbfDataGen make gLIME considerably more robust on COMPAS and German datasets, but not on the CC dataset. We believe the reason for that is that features in CC are strongly interdependent, and many represent the same attribute as a fraction of value, e.g., we have the share of the white population, the share of the Asian population, and so on. This interdependence dictates that all fractions have to add up to 1. The data generators are unlikely to capture such strong conditional dependencies, but the adversarial model's decision function is likely to approximate it.

The gSHAP method is most robust when using the TreeEnsemble generator, but it shows less robust behavior than the gLIME method. The reason for that could be in the feature value replacement strategy used by the SHAP and gSHAP methods, which change only some of the feature's values with the help of the distribution set. This can lead to out-of-distribution instances if the instances in the distribution set are not close enough to the explained instance. With gLIME, we generate complete instances that are more likely to be in the distribution of the training set.

IME is quite robust even with the perturbation sampling, as we expected. This suggests that IME is the most robust of all three tested explanation methods and shall be considered the chosen method in sensitive situations. The gIME results when using the TreeEnsemble generator (TEnsFillIn variant) are comparable to the original IME variant results. This suggests that sampling from a smaller data set, which represents the neighborhood of the explained instance, does not decrease the method's robustness.

3.3 ReEx: Semantic Reasoning from Model-Agnostic Explanations

In this section, we provide an overview of ReEx (Reasoning with Explanations), a method applicable to explanations generated by arbitrary instance-level explainers, such as SHAP. The paper describing this work (Stepišnik Perdih et al., 2021) is included as Appendix C.

By using background knowledge in the form of ontologies, ReEx generalizes instance explanations in a least general generalization-like manner. The resulting symbolic descriptions are specific for individual classes and offer generalizations based on the explainer's output. The derived semantic explanations are potentially more informative, as they describe the key attributes in the context of more general background knowledge.

Figure 10 shows the procedure for generating semantic explanations for a data item. ReEx does not require that the data item be textual, if an ontology exists which maps the data attributes, semantic explanations of the class predictions can be generated. The first stage in the procedure uses Kernel SHAP (Lundberg & Lee, 2017) to generated explanations for the data items in a dataset, after which the individual explanation instances are used to aggregate the most relevant features for each class in the dataset. The reasoning step of the procedure uses the most relevant features and an ontology to

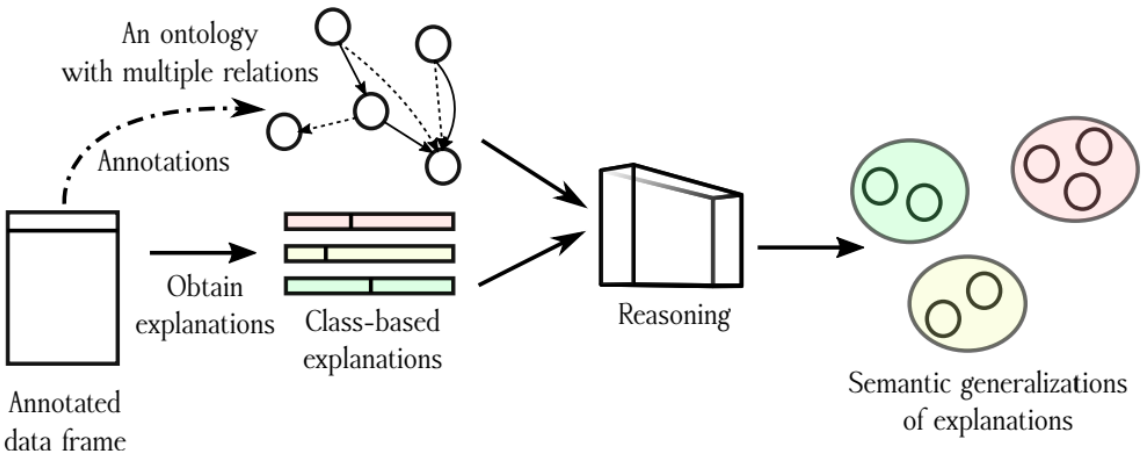


Figure 10: Overview of ReEx. Given a data set where the attributes map to a domain ontology, ReEx first produces and aggregates the instance-level explanations. Top attributes are selected iteratively (vertical lines in the second sub-image of the three vectors), followed by a reasoning procedure, which exploits the explicitly given relations (within the ontology) to generalize the mapped terms into higher-level semantic terms. Generalizations are obtained for each class (last sub-figure).

generate semantic generalization using two proposed generalization schemes, *Selective staircase* and *Ancestry*.

Our work on ReEX evaluates these two different reasoning paradigms, showing both schemes outperform generic generalization commonly employed by e.g., statistical enrichment analysis approaches. Further, we demonstrate how ReEx can produce logical explanations comprised of semantic term conjuncts, specific for individual classes – these types of explanations offer direct insight into the background relevant to classifying a given instance into a particular class. Further detail can be found in Appendix C.

4 Contributions to visualization techniques for text classification

In this section we explain our contributions to visualisation techniques for text classification with deep neural networks. In Section 4.1, we show how the TransSHAP technique (introduced in Section 3.1) can be used as a novel visualisation for explanations of text classification models such as BERT. The paper describing this work (Kokalj et al., 2021) is included as Appendix A.

Visualisation design models, such as the nested model of visualisation design (Munzner, 2009), call for a greater emphasis on problem specification in terms of domain, tasks, and data abstraction in visualisation design and validation. In Section 4.2 we conduct a task based analysis of visualisations available in the SHAP python library. This analysis examines the tasks supported for tabular data and investigated their applicability to explanations of text based classification models.

In Deliverable D1.5 the AttViz tool was presented. Here, in Section 4.3, we present additional work on an offline component of the AttViz tool designed to address some of the limitations of the online version.

4.1 TransSHAP: Visualization of a prediction explanation for the BERT model

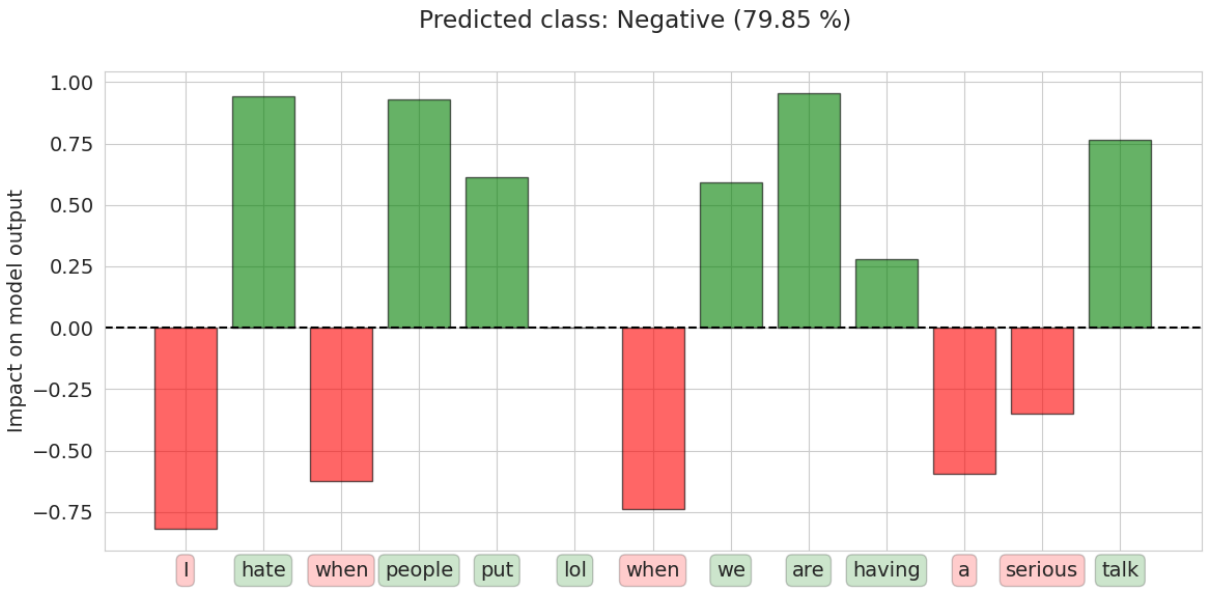


Figure 11: TransSHAP visualization of prediction explanations for negative sentiment. We obtained the features' contribution values with the SHAP method. It is evident that the word 'hate' strongly contributed to the negative sentiment classification, while the word 'lol' (laughing out loud) slightly opposed it.

To make a visualization of predictions better adapted to texts, we modified the histogram-based visualizations used in IME, LIME and SHAP for tabular data. Figure 11 is an example of our visualization for explaining text classifications. It was inspired by the visualization used by the LIME method but we made some modifications with the aim of making it more intuitive and better adapted to sequences. Instead of the horizontal bar chart of features' impact on the prediction sorted in descending order of feature impact, we used the vertical bar chart and presented the features (i.e. words) in the order they appear in the original sentence. In this way, the graph allows the user to compare the direction of the impact (positive/negative) and also the magnitude of impact for individual words. The bottom text box representation of the sentence shows the words colored green if they significantly contributed to the prediction and red if they significantly opposed it.

We evaluated the novel visualization method using an online survey. The targeted respondents were researchers and PhD students not involved in the study that mostly had some previous experience with classifiers and/or their explanation methods. In the survey, the respondents were presented with three visualization methods on the same example: two visualizations were generated by existing libraries, LIME and SHAP, and the third one used our novel TransSHAP library. Respondents were asked to evaluate the quality of each visualization, suggest possible improvements, and rank the three methods.[3]

The results of 38 completed surveys are as follows. The most informative features of the visualization layout recognized by the users were the impact each word had on a prediction and the importance of the word contributions shown in a sequential view. The positioning of the visualization elements for each of the three methods was rated on the scale of 1 to 5. Our method achieved the highest average score of 3.66 (63.1% of the respondents rated it with a score of 4 or 5), second best was the LIME method with an average score of 3.13 (39.1% rated it with 4 or 5), and the SHAP method was rated as the worst with an average of 2.42 (81.5% rated it with 1 or 2). Regarding the question whether they would use each visualization method, LIME scored highest (44.7% voted "Yes"), TransSHAP closely

[3]The survey questions are available here: `https://forms.gle/icpYvHH78oE2TCJt7`.

followed (42.1% voted “Yes”), while SHAP was not praised (34.2% voted “Yes”). The overall ranking also corresponds to these results. LIME got the most votes (54.3%), TransSHAP was voted second best (40.0% of votes), and SHAP was the least desirable (5.7% of votes). In addition, we asked the participants to choose the preferred usage of the method out of the given options. The TransSHAP and SHAP methods were considered most useful for the purpose of debugging and bias detection, while the LIME method was also recognized as suitable for explaining a model to other researchers (usage in scientific articles).

4.2 Visual variable analysis of SHAP visualisations for text

In this section we present an analysis of visualisation tasks for SHAP explanations of text. This work is currently unpublished but is described in detail here.

The goal of this analysis was to explore the relevance of current Shapley additive explanation (SHAP) (Lundberg & Lee, 2017) visualisations for models built on textual data. To do this, in Section 4.2.1, we investigated the tasks supported by current SHAP visualisations applied to tabular data. The tasks we identify are assumed to be relevant for general explanation interpretation due to the inclusion of these visualisations in the SHAP python package[4]. However, since many of these visualisations are not designed for textual data we cannot expect them to fully support the requirements of visualising SHAP explanations of text. We aimed to make explicit the high level tasks enabled by each of the examined visualisations so that we can investigate if these tasks are currently supported for explanations of text. From the examined visualisations we identified eight high level tasks which are currently supported for visual interpretation of SHAP explanations for tabular data.

We explored the unique requirements and attributes of text visualisation for SHAP explanations in Section 4.2.2. Following this exploration the available data attributes were compared to the visual variables used to encode them. To do this comparison we made use of Bertins visual variables (Bertin, 1983) and Mackinlays proposed ranking of their perceptual efficiency for different data types (Mackinlay, 1986), see Figure 12. This analysis identified that five of the eight tasks are not well supported for text by current SHAP visualisations. This visual variable analysis can be found in Section 4.2.3.

To address gaps which were identified between the available designs and their applicability to the identified tasks, we propose potential improvements from a visualisation design perspective. We suggest that a visualisation technique based on concordance analysis could be used to address many of the under supported tasks and that for visualising explanations of long texts techniques such as spark lines could improve the support for the tasks. We These proposed designs are suggested in Section 4.2.4.

4.2.1 SHAP visualisation tasks

Existing SHAP explanation visualisations are primarily designed for visualising explanations of models developed on tabular data. The visualisations implemented by the SHAP python package are the main focus of this analysis. We examine these existing tabular focused visualisations along with some additional text specific SHAP visualisations to establish a list of tasks currently supported by visualisation tools.

Some visualisation techniques exist in the SHAP package for interpreting SHAP explanations on image datasets. We choose to ignore those in this analysis as the mapping to tasks related to textual data is not clear.

Visualisations which display the SHAP value for each of the features in an explanation instance as a bar are a popular representation (Figure 1, Figure 13 and Figure 11). In the SHAP package waterfall diagrams (Lundberg et al., 2020) are used to visualise instance explanations (Figure 13). The main feature of these visualisations is the presentation of both, the feature labels from explained instance,

[4] https://github.com/slundberg/shap

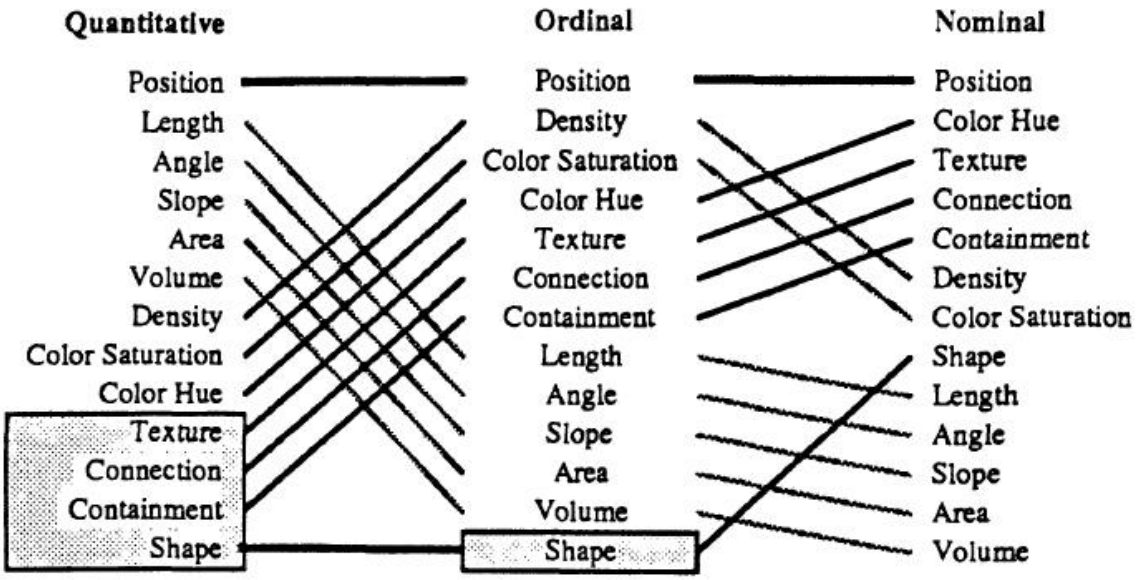


Figure 12: Visual variable rankings per data type. The visual variables closer to the top are more effective at encoding information of the associated data type.

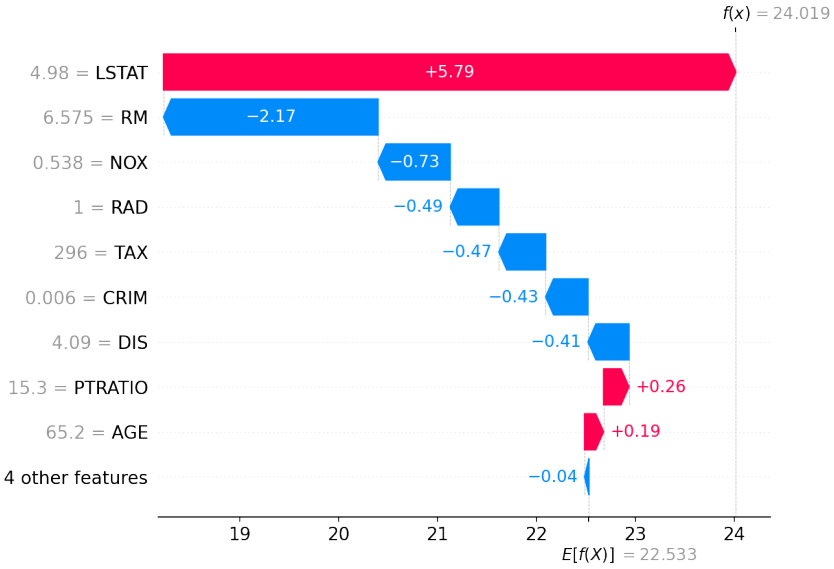


Figure 13: Waterfall diagram of a SHAP explanation instance of tabular data, taken from the SHAP python package documentation.

and their corresponding Shapley value. These visualisations use bars to convey the size of the SHAP contribution of each feature to the prediction of a class. The contribution to the prediction can be positive or negative for each feature and this polarity is often encoded using color and and/or direction of the bar.

The SHAP feature contributions decompose a model's prediction into additive explanatory variables, when the SHAP values for each feature in an explanation instance are presented together the visualisation provides an overview of the prediction explanation. For tabular data these SHAP values are

often reordered by the absolute SHAP vlaue of the features. The general task which can be identified in this type of visualisations is to visually present an explanation instance using the SHAP values associated with each feature in the instance. The values of the features themselves are not used in this visualisation.

We can split this task further into; providing an overview of the explanation of an instance of a model's prediction, and presenting the detail of the Shapley value for each feature's contribution to the prediction. The merits of different encoding choices, such as bar or waterfall charts, and vertical or horizontal layouts will be discussed in Section 4.2.3 after tasks have been mapped to data abstractions. The first task we have identified is:

- Visualise a generated explanation instance using the SHAP values associated with each feature.

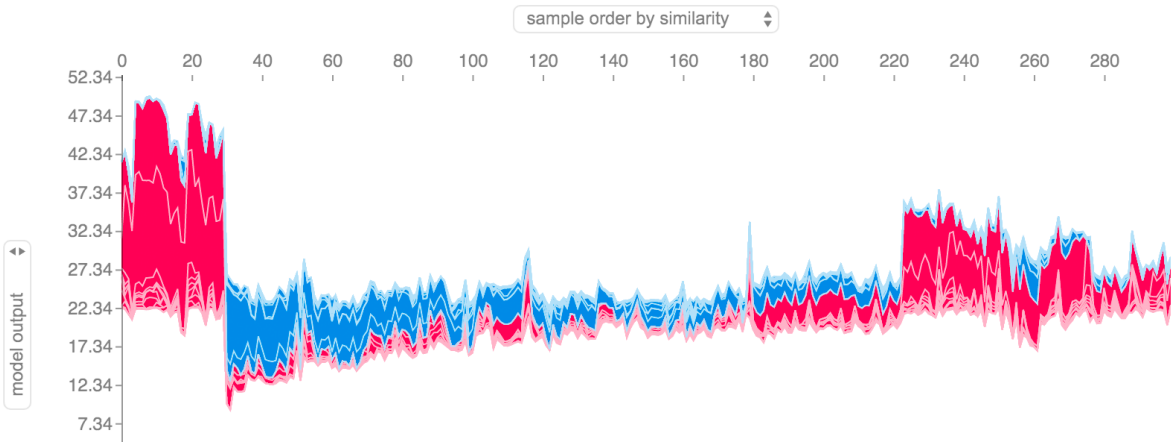


Figure 14: Multiple SHAP explanation instance overview visualisation using aligned force plots for each instance, taken from the SHAP python package documentation.

Another visual encoding of SHAP explanations is the force diagram (Lundberg et al., 2018), see Figure 2. This encoding represents the SHAP values of each feature as a bar, however, in this visualisation the bars are stacked. Stacking the bars reduces the amount of space required for the explanation by using a single axis for both the feature and explanation variables. This visualisation appears to serve the same task as the other bar style explanation visualisations. However, this visualisation is often used as a component for creating an overview of model predictions for multiple instance explanations in visualisations similar to Figure 14. When these force visualisations are presented together the result is an overview of the SHAP values across the model instances. This suggests that the task supported by the SHAP force diagram is comparison of instance explanations.

When a large number of instances are visualised together the visualisation reordering is often enabled. Multiple reordering schemes can be interactively investigated. Prediction score, selected feature values, selected feature SHAP values and similarity scores based on either feature or SHAP values are all possible reordering schemes. This view provides an overview of the explained predictions and is helpful for identify trends in feature contributions for a predicted class.

These force diagrams have been applied to explanations of textual data. However, the comparison of multiple instance in the case of text is more difficult due to the large number of features in the dataset (words) and sparsity of the features in each explanation instance. In tabular data often each explanation instance will use the same features or contain the majority of the features used in the dataset, for textual data this is not the case. From the analysis of these two complementary visualisations we add the following two tasks to our taxonomy:

- Compare SHAP values across multiple explanation instances.
- Examine similar explanation instances from a dataset.

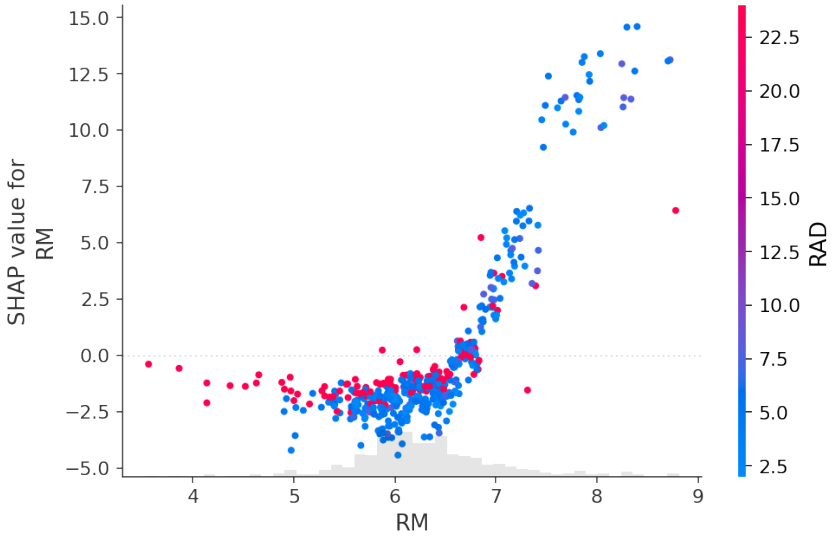


Figure 15: SHAP scatter plot visualisation for comparing SHAP and feature values for tabular data, taken from the SHAP python package documentation.

The SHAP dependence scatter plot, see Figure 15, is used to compare the quantitative feature values and SHAP values for a single feature across a dataset. Interactions with an additional feature can be investigated by colouring the points using a quantitative color scale. The tasks we can identify are comparison of SHAP values for a single feature with the feature values of that feature across an entire dataset, and investigating the relationship between an additional feature and the SHAP and feature values of the feature of interest. These tasks can be summarised as:

- Investigate relationship between SHAP values and feature values for a single feature across a dataset.
- Compare two features, using both SHAP and feature values.

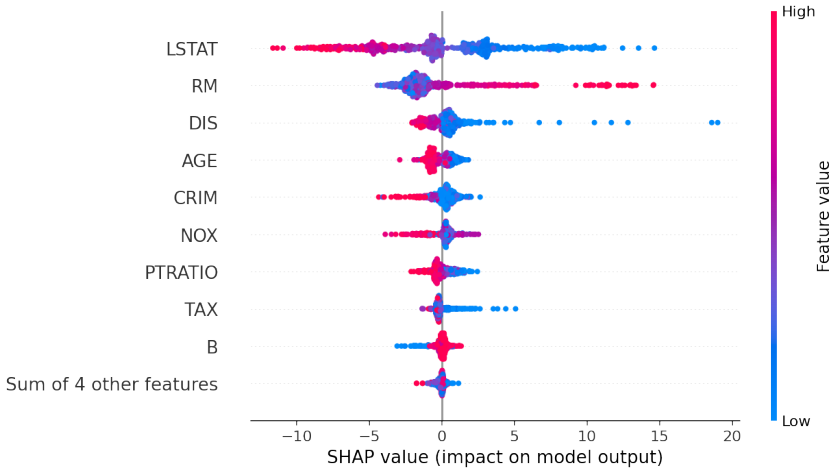


Figure 16: Beehive visualisation of SHAP value per feature across a tabular dataset, taken from the SHAP python package documentation.

The beeswarm plot provides a means of investigating the interaction between SHAP and feature values across a dataset. Figure 16 shows an example of a beeswarm plot for a dataset with nine features. In this visualisation the the SHAP values for multiple features are presented and the points are colored

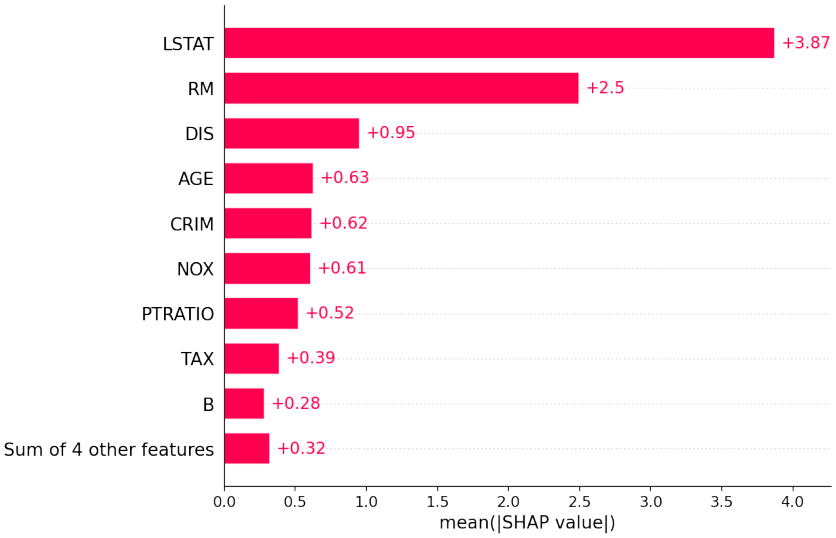


Figure 17: Bar plot of mean SHAP value for each feature across a dataset, taken from the SHAP python package documentation.

to display the feature value associated with each SHAP value, for each instance. Each points x-axis position represents the SHAP value of the associated feature in one of the classified instances. Color is used to represent the quantitative feature value for each point.

This visualisation enables comparison of the magnitude and direction of SHAP values associated with each feature in combination with the overlay feature values. The beeswarm plot offers an overview of the SHAP distribution for each feature and the association between the feature values and SHAP values.

The beeswarm plot is, essentially, a feature focused overview of the explanation contribution across the dataset. It trades the additive view of an individual explanation (which is maintained in force plot overviews) for emphasis on global feature contribution to predictions. The interactions between SHAP feature contributions are not supported in this visualisation. The tasks supported by this visualisation are investigating global feature contribution to predictions across a dataset, investigating the interaction between feature values and feature contributions for each feature in a dataset.

Another visualisation used to explore SHAP contributions across a dataset is based on simple bar plots, see Figure 17. In this visualisation mean feature contribution to predictions across the dataset can be easily compared between features. The value of the features are not presented in this visualisation. This visualisation supports the task of comparing the size of the SHAP values for each feature across a dataset. This task is similar to “investigating global feature contribution to predictions across a dataset” and is combined for our analysis. The tasks are summarised as:

- Investigating global feature contribution to predictions across a dataset.
- Investigating the interaction between feature values and feature contributions for each feature in the dataset.
- Comparing the magnitude of SHAP values for each feature across a dataset.

For tree based models the SHAP framework can compute pairwise SHAP interaction values. These values are then plotted against the values of one of the features and colored by the other to reveal interaction effects between the two features. These interaction plots (fig. 18) are used to identify pairwise feature values where the explanations are correlated with the features. The task these plots support is the investigation of the interactions between SHAP values for feature pairs and their feature values.

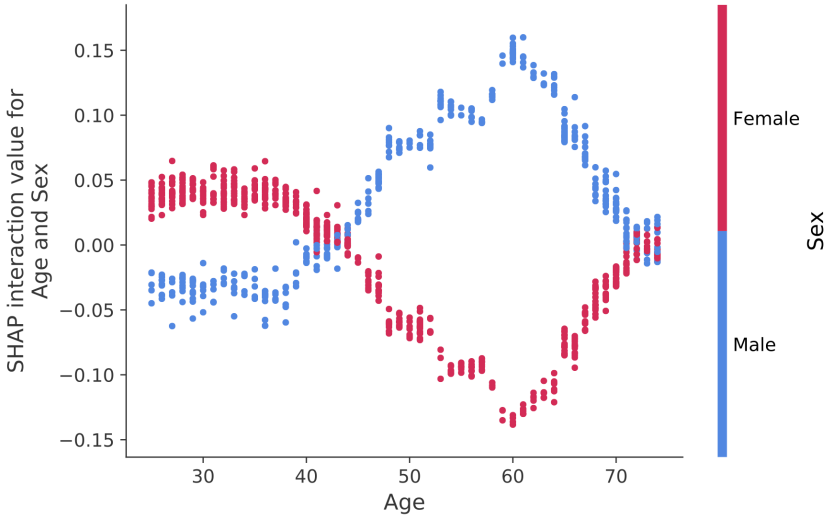


Figure 18: Interaction plot of pairwise SHAP interaction values for two features, taken from the SHAP python package documentation.

We have identified eight tasks through this analysis , these tasks may not all be applicable or useful for explanation of predictions based on textual data. This may not be an exhaustive list of task supported by these visualisations, To summarise, here are the eight tasks we have identified which are currently supported in SHAP framework for tabular data:

1. Visualise an explanation instance using the SHAP values associated with each feature.
2. Compare multiple explanation instances.
3. View similar explanation instances from a dataset
4. Investigate the relationship between SHAP values and feature values for a single feature across a dataset.
5. Investigate the relationship between SHAP and feature values for a feature compared with one other feature.
6. Investigating the global SHAP feature contribution to predictions across a dataset.
7. Investigating the interaction between feature values and feature contributions for each feature in the dataset.
8. Investigate the interaction between pairwise SHAP feature contributions and their feature values.

4.2.2 Data abstraction for SHAP text visualisation

In this section we attempt to identify the attributes of SHAP explanations for text which are available to be visualised. We do this so that we can design and evaluate visual encodings using the mapping between visual variables and data attributes.

One consideration when visualising textual information are the ordinal properties of text. Sentences contain information encoded by both the meaning of words and by their order within a sentence. When explanations are presented using word or sub-word features the context of the sentence can influence the interpretation even if the model is not contextually aware. One of the key properties of some state of the art approaches to language modeling, such as BERT, is the use of contextual information by the

model. Black box explanation methods ignore model specific considerations, however when visualising explanations of models which are contextually aware an emphasis on the ordinal properties of the underlying data could provide more informative explanations.

Predictions for textual data can be made for textual units, for example sentence level predictions or document level class predictions. This should be considered when designing visualisations. Visualisations serving the tasks we have identified should consider sentence level predictions and predictions on text spans longer than an individual sentence.

The dimensionality and sparsity of the underlying textual data often make it challenging to visualise in the same manner as tabular numeric and categorical data. For example, when comparing the explanations generated for two sentences it is likely many of the words will not occur in both. This makes it difficult to compare explanation instance as most of the features in the dataset will not be present in any one explanation instance.

A further consideration is that that all of the the features (words) in each explanation do not have an easily interpretable quantitative value. While the words may be represented by the model using a numerical value or embedding vector, the values may not be available when using black box explanation methods. Even if available, these numerical representations of words are difficult to interpret and not usually considered in black box explanation methods where the goal is to explain the predictions made by a model rather than its internal behaviour.

With these considerations in mind we now describe a data abstraction of textual model explanations, consisting of instance and dataset level data attributes.

At the instance level the data abstraction has attributes; ordered word positions, text length (number of features), categorical feature values (words) and quantitative additive explanation values for each word. The model prediction can be considered as a categorical variable and also as a quantitative value (prediction score). Additional attributes such as part of speech for each word could be considered a data attribute, but since these are not available from the SHAP explanation they will not be included in the data abstraction.

At the dataset level the data abstraction also has several attributes. The collection of instances which make up the dataset may have an associated ordinal property. This ordinal property of the instances could be used to represent position in an ordered dataset (e.g., sentences from a single document, temporally annotated data), or some other ordinal property of the instances (e.g., low to high class prediction score).

The quantitative values of word count and SHAP value totals per word can be calculated across all explanation instances. Similarly positional or windowed word counts and SHAP value totals can be calculated across all instances, these provide quantitative values for feature interactions in terms of word co-occurrence and cumulative co-occurring SHAP value.

4.2.3 Visual encoding applicability to text explanations

In this section we consider SHAP visualisation tasks, identified in Section 4.2.1, in the context of visualising SHAP explanations of text. The goal is to identify differences between the data required for the tasks and the information available in the specific case of SHAP explanations of text based models. In addition, existing visualisations are evaluated in terms of their encoding efficiency and suitability for these tasks when visualising results of SHAP explanations of text based models.

Task 1: Visualise an explanation instance

The visualisations shown in Figures 1, 13 and 11 offer quite similar encodings which can be used to visualise instances of SHAP explanations of text. The style of plot often used for LIME explanations (Figure 1) and the waterfall plot (Figure 13) encode feature explanation magnitude by ordering the features on the using vertical position.

For text based explanations maintaining word order may be beneficial, for languages which are read horizontally positioning the features along this axis can also make the explanation easier to interpret. TranShap visualisations, see Figure 11, make this encoding choice, this visualisation is designed specifically for visualising SHAP text based model explanations. All of these visualisations encode explanation values using length, the second best choice for quantitative data.

Waterfall plots make use of position to encode the cumulative contributions of each feature explanation value to the prediction, Since position is a higher ranked visual variable than length the additive properties of the explanation values will be emphasised. When presenting text using a waterfall diagram the axis should be flipped and the word order maintained for right to left and left to right languages.

Force Diagrams, see Figure 2, are an alternative encoding of SHAP explanation instances and have been used for explaining models based on text. The encoding is not as good for quantitative comparisons of the explanation values as the axis used to represent these values is overloaded, the ordinal feature values of the words are also encoded using the same axis as the SHAP explanation contributions.

All of these explanation instance visualisations work well for explanations of short spans of text, such as sentences. Mapping these techniques to explanations of longer texts in not trivial. Extending the visualisation along the feature axis or compressing the axis to fit more features (words) reduces the readability of both the text and quantitative values as the number of features grows.

For longer texts the explanation contributions are sometimes visualised by presenting the page of text and applying a background color to each word, see Figure 1. In this encoding the visual variable *color hue* is used to represent the polarity of the explanation contribution for a word. Since this is a nominal data attribute color is a good encoding choice being the second highest ranking visual variable. The magnitude of the explanation contributions are encoded using *color saturation*, this visual variable is rank eight for effective encoding of quantitative data. Given the low rank of the quantitative encoding choice alternate encoding choices can increase their emphasis in the visualisation. This requires careful consideration of the visualisation goals in relation to text, the legibility of the symbols (words) and their ordinal arrangement (sentences and document structure) might be negatively impacted by alternate encoding choices.

Task 2: Compare multiple explanation instances

Force Diagrams, see Figure 2, can be used to visualise a single explanation instance of text based models, but their encoding also allows for the combination of multiple explanation instances to form an overview of model explanation across a dataset. Using force diagrams to compare multiple explanation instances, see Figure 14, for text based models is challenging. When comparing multiple instances of tabular data, where each instance has a consistent set of features with varying values, the individually force diagrams can be aligned horizontally or vertically to display similarities and differences between the feature contributions across multiple explanation instances.

For text based models, using words as features, this style of visualisation fails due to the majority of the features not being present in any one example. While the explanation instances can still be arranged in this style of visualisation, the ability to compare the instances is limited. Using a measure of similarity and rearranging the word order within the instance to help align them could improve the comparisons but choosing a correct word order within the instances and positioning across the instances is an additional challenge where the result will be that the majority of the instances are only comparable via high frequency words in the dataset.

The encoding choices in force diagrams for tabular data are sensible, ordering the explanation instance along one axis, and using position to encode prediction similarity strongly encodes this dataset level attribute. Using length to encode the individual SHAP values makes that quantitative attribute easily comparable across explanations. By stacking the feature bars within each explanation the overall prediction value of each instance can be compared across the entire dataset.

This task does not appear to be fully addressed by current SHAP visualisation techniques when applied

to text based models. To compare multiple text explanation instances some form of feature overlap or alignment is an additional requirement necessary to deal with the sparsity of text features across a dataset. Comparing subsets of the dataset where alignment is possible could prove more informative than attempting to compare the entire set of explanation instance for a dataset.

Task 3: View similar explanation instances from a dataset

This task is closely related to task 2, it is currently served by the same force visualisation of multiple explanation instances, see Figure 14. While it is possible to group text explanation instances using some measure of similarity, many of the features within the similar instances will not be shared. However where prediction score and feature overlap do occur the explanation instance context may be of interest. This suggests that viewing the entire set of explanations in the dataset may provide more visual clutter than useful information and similar subsets may be of greater interest to an analyst.

Within a set of similar explanations, feature frequency, feature co-occurrence and cumulative SHAP feature score are quantitative attributes that can be displayed visually. Since these values are summaries calculated over a subset of explanations, the sparsity of the data is less problematic. Visualising these quantities using summary visualisations, such as beehive or simple bar plots, removes the individual instances from view by providing an overview of the quantities associated with the similar subset. This would seem to be contrary to the task we seek to facilitate (investigating similar explanation instances) as the instances are now hidden. However, by using visualisations of these quantities as an overview of the similar instances, and providing the detail of the associated explanations on demand, we can support this task.

How we provide this detail on demand leads us back to the problem of Task 2; how to visualise textual explanation instances for easy comparison. The difference being that in this case the starting point is; seeking to view a set of instances which share at least one common feature. By using selection interactions on an overview of quantitative information associated with a set of similar instance the detail of those instances can be displayed and aligned using the common feature which is selected.

Task 4: Investigate the relationship between SHAP values and feature values for a single feature across a dataset

Since the feature values of the textual features do not have a quantitative representation which changes across the instances in the dataset this task can not be directly mapped to the tabular case. The feature values in each instance are a binary, either the instance contains the word at a position or it does not. Due to this binary feature value Scatter plots of feature value against SHAP value, see Figure 15, are not useful for SHAP text explanations. This is because he feature will only have an associated SHAP score when it exists at a position in an instance so the feature axis of a single feature will only have one data point.

An alternative could be to use sentence word position as the feature value and plot the distribution of explanation contribution for a single word across all explanations. This encoding emphasises explanation contribution at absolute word positions in the explained text instances, for a single word.

Task 5: Investigate the relationship between SHAP and feature values for a feature compared with one other feature.

This task is strongly related to task 4. This task is also not possible to address by using the scatter plot visualisation, see Figure 15, which is used for tabular data.

By extending the solution proposed for task 4 we can visualise the feature contribution distribution of one feature (word) relative to the position of another feature (word). This would result in a visualisation, perhaps a bar chart, showing explanation contribution position and distribution for two co-occurring words across all instances in the dataset.

Task 6: Investigating the global SHAP feature contribution to predictions across a dataset

The beehive and bar plot visualisations , see Figures 16 and 17, are both offer efficient encodings for this task. These visualisations encodings map to the data attributes of SHAP explanations of text.

In beehive plots each feature is presented using dots to represent the SHAP values of the feature in each explanation that contains the feature. The sparse nature of the features means the number of features to display will be much larger than the tabular example, and their should be fewer dots for most of the features. The colouring of the dots by feature value can be ignored or used for some other quantity of interest, perhaps prediction polarity and size.

Bar plots simply encode the SHAP value totals for each feature using the length visual variable. In Figure 17 the total SHAP values are calculated using the absolute value of the SHAP contributions. Alternatively two bars could be used for each feature, one of positive and one for negative SHAP contribution.

The main difference between these encodings is that the beehive plot encodes the SHAP value of each feature occurrence while the bar plot encodes the combined total of each instance. The bar plot makes it easier to compare the total contribution of each feature to the explanations while the beehive plot provides a more detailed breakdown of a features contribution to explanations.

Both of these plots address the requirements of the task and could serve as useful overview visualisations to inform the feature choices required in tasks 2 and 3.

Task 7: Investigating the interaction between feature values and feature contributions for each feature in the dataset

The beehive visualisation is used for this task when explaining models based on tabular data. The feature values are encoded using a quantitative color scale. This enables analysis of the SHAP explanation contributions of each feature to be viewed in the context of their feature values. For text based models the dimensionality of the features and the lack of quantitative value associated with the features makes the investigation of all of the features and their associated SHAP scores together a challenging task.

Perhaps this task and tasks 4, 5, and 6 could be addressed by enriching the textual data with computed quantitative information such as word embeddings. Visualisations could then be devised for visualising explanation scores against embeddings for the tokens in an instance or across a group of instances. This however moves away from the concept of black-box explanations where the data and explanation scores are all that is required for interpretation.

Task 8: Investigate the interaction between pairwise SHAP feature contributions and their feature values

This is another task where feature values are needed for comparison with explanation values. As we have already seen the feature values are not available to visualise in text based model explanations. However, pairwise SHAP feature contributions could be a useful quantity to visualise and analyse for text based models. An alternative encoding for this task is a heatmap where the all features (words) are shown on both axis. The intersection of the words would display the total SHAP contributions of word *b* given word a appears in the same explanation instance.

Task Overview

The task of displaying a single short text explanation instance (task 1) is well served by visualisations such as TransSHAP. However, visualising longer text explanation instances is not well supported by current encodings.

Tasks 2 and 3 are related to comparing or viewing multiple explanation instances. These tasks are not well served by current visualisation tools. The high dimensions and sparsity of text features when comparing multiple explanation instances is a challenge not addressed by the existing tools. It is possible these tasks could be supported by developing visualisations for displaying subsets of the data which contain overlapping features.

Visualising global explanation values (the sum of explanation values for a feature across all explanation instances) for a dataset or a subset can be achieved efficiently using bar plots or beehive plots. These

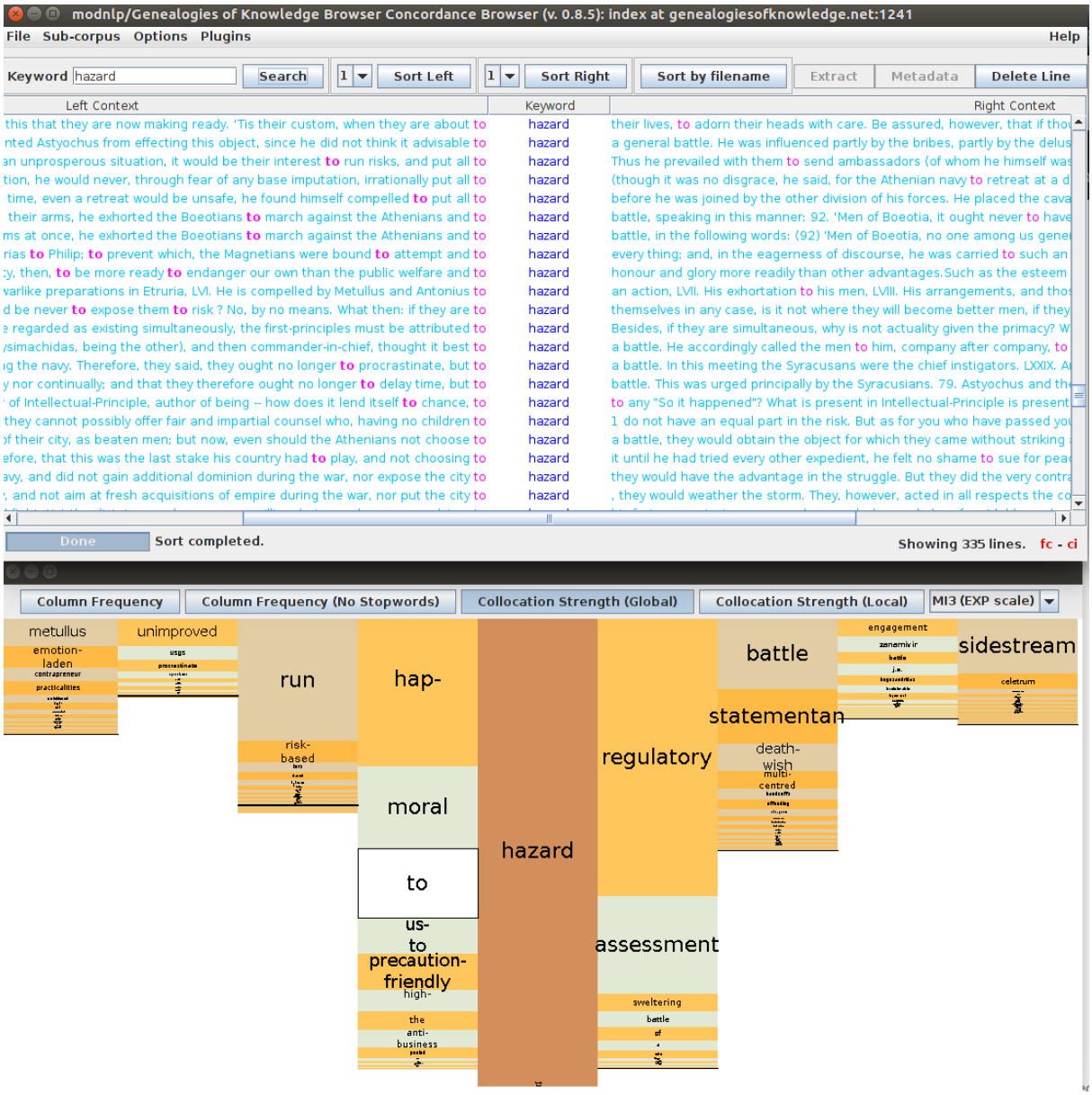


Figure 19: Concordance Mosaic visualisation which could be adapted for comparing SHAP contribution across explanation instances

overview visualisations provide support for task 6 and could also be useful for supporting tasks 2 and 3 for text explanation.

Tasks 4 and 5 can be considered sub-tasks of tasks 3 and 4. Tasks 4 and 5 are concerned with observing the explanation values of a feature in relation to its feature values across a dataset. For text explanations feature value does not have an obvious quantitative attribute. The value of the textual feature can be more creatively interpreted as the context in which it appears. In this interpretation visualisations which support simultaneous investigation of total feature contribution across the dataset and collocated features could improve the ability to perform these tasks.

Task 8 calls for the ability to investigate interactions between features in relation to their pairwise explanation values. Interpreting this task through the lens of text explanations can, again, be interpreted as a task requiring analysis of collocations in the corpus of explained instances.

4.2.4 Suggested Encoding designs and improvements

SHAP Concordance Mosaic

Taking an overview visualisation as the starting point of an analysis of multiple text explanation instances removes some of the challenges in dealing with the sparsity of the features. This overview could take the form of a beeswarm or bar plot of global explanation score for the features across the dataset. Interesting features could be selected to view the subset of explanation instances which contain that feature. A well established technique for the analysis of a word and its surrounding context is the keyword-in-context concordance. By aligning the word of interest centrally its left and right contexts can be examined for concurrence patterns. These concordance displays, see Figure 19, offer the ability to sort word positions relative to the keyword of interest to enhance the ability to identify common co-occurrence patterns.

For text explanations the concordance offers alignment of the explanation instances using a common feature. It does not fully support the task of comparing explanation instance quantitatively as this view can't encode the explanation contribution values. Concordance Mosaic (Luz & Sheehan, 2020) a visualisation designed to complement the concordance view by providing a positional overview of quantitative information related to co-occurrence patterns could be modified to display explanation contribution values. To do this each positional tile height, see Figure 19, represents the sum of SHAP explanation values for that feature across the instances containing the keyword selected from the overview. These explanation values are calculated per position relative to the keyword. This visualisation displays the distribution of co-occurring feature explanation contributions per position and word.

The mosaic is interactively linked to the concordance view allowing selection of a tile to highlight and sort the text fragments in the concordance. This lets us use the overview of co-occurring explanation contributions to explore the associated text fragments.

Visualising Explanation Instances for Long Texts

We identified limitations in visualising explanations for long texts. There is both a lack of visualisation support and poor encoding choice for quantitative data in the existing visualisation. One suggestion to improve the encoding of the quantitative explanation scores is to use spark lines, or spark bars, to present the explanation contributions in the body of the text. This would improve the encoding of the quantitative information compared with using color saturation to overlay the quantities. This would make the magnitude of the explanation values easier to compare between features, while still presenting the text in a natural document format.

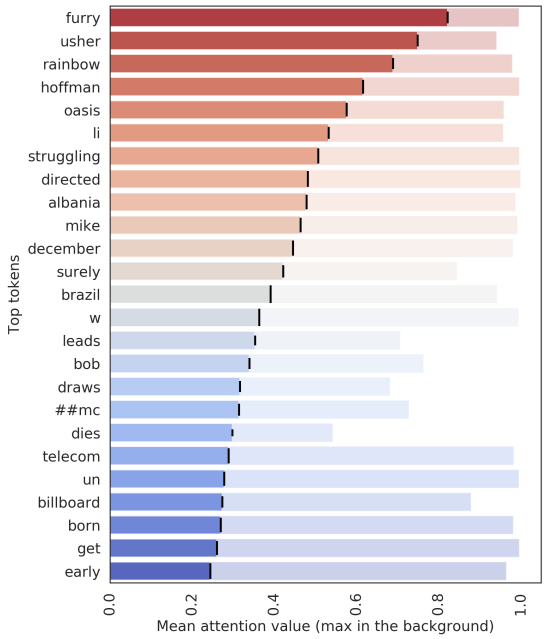
Waterfall or line chart of the additive explanation variables could be used as an overview. This overview could be linked interactively to a document view. Interesting sections of the overview could be selected and the corresponding section of the document brought into view with the explanation scores encoded.

4.3 AttViz library: statistical analysis of the attention space

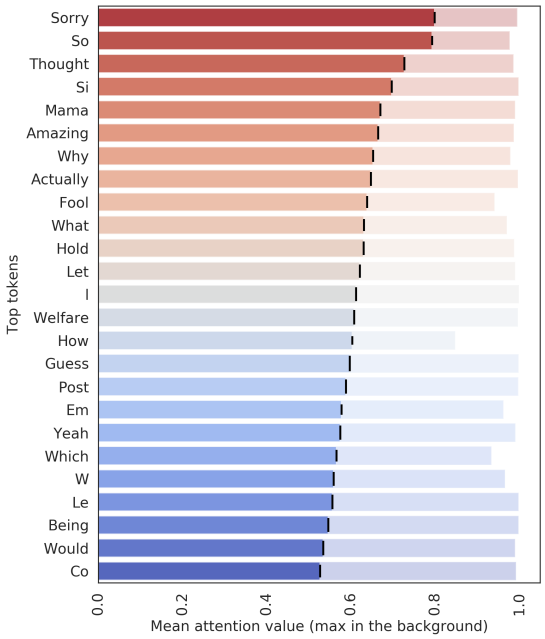
In Deliverable D1.5 we presented how the online version of AttViz can be used for *direct analysis* of model output (in the JSON format). Here we describe subsequent work on a complementary offline component of the tool which was designed to address some of the limitations of the online tool.

Albeit suitable for quick inspections, the online system has its limitations such as poor support for computationally more intensive types of analysis (in terms of waiting times), and the lack of customized visualization tools accessible in the Python ecosystem. To address these aspects, we developed AttViz library[5] that offers more detailed analysis of a given neural language model's properties. The library operates on the same JSON structures as the online version and is compatible with the initial user input. We demonstrate the analytical capabilities of our visualization tools on three datasets.

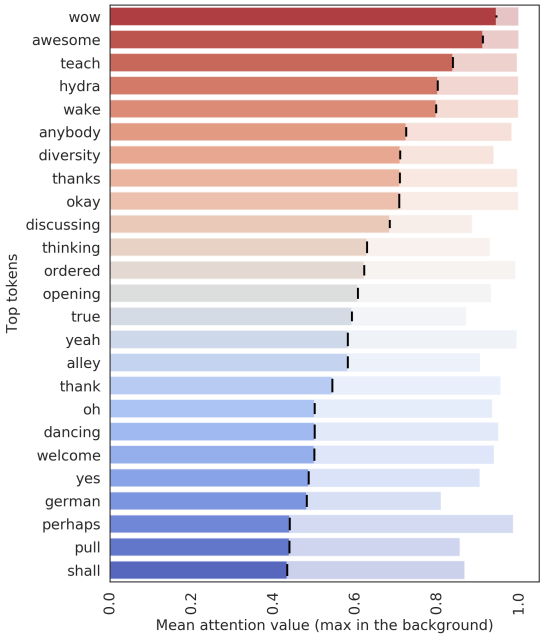
[5] AttViz library: `https://github.com/EMBEDDIA/attviz`.



(a) Top 35 tokens in the BBC dataset.



(b) Top 35 tokens in the insults dataset.



(c) Top 35 tokens in the hate speech dataset.

Figure 20: Visualization of the 35 most attended-to tokens for the three inspected data sets. Interestingly, the attention peaks of tokens (maximum, in the background) all take high values, albeit lower-ranked tokens are on average characterized by lower mean attention values.

4.3.1 Dissecting the token space

The first offline functionality is a barplot visualization that offers insight into relevant aspects of the attention distribution at token level. Whilst understanding the attention peaks is relevant for direct inspections, the attention space of a given token can be contextualized on the dataset level as well. The AttViz library offers fast visualization of the mean and spread of attention distributions, simultaneously showing the attention peaks for individual tokens. We visualized the distribution for three classification datasets (Figure 20): BBC news (Figure 20a), insults[6] (Figure 20b), and hate speech comments (Figure 20c)[7].

The proposed visualizations present top k tokens according to their mean attention throughout the whole dataset. It is interesting to observe, that the insults and hate speech data sets are not completely characterized by swear words or similar single-token-like features. This potentially indicates that the attention tries to detect interactions between the byte-pair encoded tokens, even for data sets where the attention could be focused on single tokens. It is interesting to observe that the terms with the highest attention are not necessarily keywords or other tokens carrying large semantic meaning. Similarly, the high maxima indicate that the emphasis of the tokens is very contextual, and potentially not as informative for global aggregation.

4.3.2 Visualization of attention head focus

Contemporary neural language model architectures comprise multiple attention heads. These separate weight spaces capture distinct aspects of the considered learning task. Even though the weight spaces are easily accessible, it is not trivial to convert the large amount of information into a quick-to-inspect visualization. With the proposed visualization, shown in Figure 21, we leverage word clouds (Kaser & Lemire, 2007) to reveal human-understandable patterns captured by separate attention heads and display this information in a compact way.

For more details see our paper Škrlj et al. (2021), presented in Appendix D.

5 Conclusions

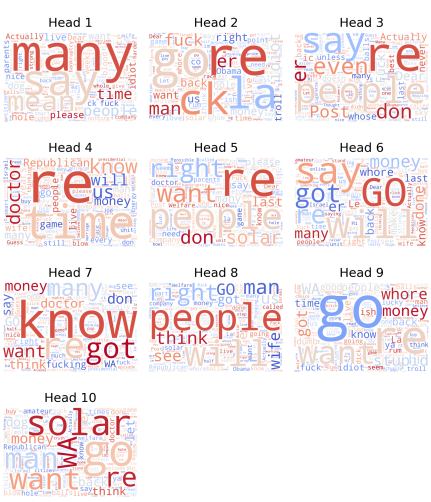
We presented a modification to the SHAP method to adapt it to work with BERT. We described the necessary adaptations in the explanations and visualize them with the new TransSHAP visualisation approach. We have shown the results of the TransSHap method on the Twitter sentiment prediction problem, classified with our CroSloEngual BERT produced in task T1.2 of WP1. The TransSHAP visualisation method was evaluated via a usability survey, where it compared favorably against current SHAP and LIME explanation instance visualisations.

We presented a defense against adversarial attacks on explanation methods. We replaced the perturbation sampling with data generators that better capture the distribution of a given data set. This prevents the detection of instances used in explanation and disarms attackers. We have shown that the modified gLIME and gSHAP explanation methods, which use better data generators, are more robust than the original variants, while IME is already quite robust. The difference in explanation values between original and enhanced gSHAP and gIME is negligible, while for gLIME, it is considerable.

We presented ReEx, one of the first approaches capable of semantic generalization of model explanations obtained by contemporary tools such as SHAP. We implement two different reasoning paradigms (Selective staircase and Ancestry), showing both schemes out-perform generic generalization commonly employed.

We presented an analysis of visualisation tasks supported by current SHAP visualisations for tabular data and their applicability to SHAP explanations for text based models, such as BERT. We identified

[6] https://www.kaggle.com/c/detecting-insults-in-social-commentary/overview
[7] https://github.com/aitor-garcia-p/hate-speech-dataset



(a) Insults.

(b) Hatespeech.

(c) BBC news.

Figure 21: The distribution of tokens over individual attention heads for the three datasets summarised with word clouds.

eight key supported tasks and identified several areas where current visualisations do not support textual explanations with these tasks. To address this we propose potential visualisation tools which could address these limitations.

We presented an extension to our prior work on the AttViz tool for visualizing BERT's self-attention head. We extend the work by providing an offline component for computationally intensive analysis which was not feasible using the online version of the tool.

This explanation and visualization technologies, developed in T1.4 were tested in prediction models developed in tasks T1.1, T1.2, and T1.3. We expect their further use on models employed in WP3, WP4, and WP5. The technologies will be integrated into the frameworks developed in WP6.

This work opens a range of possibilities for further research. The proposed defense and attacks shall be tested on other datasets with different numbers and types of features, missing data, and various problems such as text, images, and graphs. The work on useful generators shall be extended to find

time-efficient generators with easy-to-set parameters and the ability to generate new instances in the vicinity of a given one. Generative adversarial networks may be a promising line of research.

In the future TransSHAP will take into account specific properties of text data and apply language models in the sampling step of the method. We plan to restrict the sampling candidates for each word based on their part-of-speech and general context of the sentence. We believe that better sampling will improve the speed of explanations and decrease the variance of explanations. Furthermore, the explanations could be additionally improved by expanding the features of explanations from individual words to larger textual units consisting of words that are grammatically and semantically linked.

Further work on ReEx will extended the technique to use knowledge graphs, where the data is semi-automatically curated. The use of knowledge graphs could increase the data available for reasoning but potentially add more noise to the semantic explanations. In addition, we plan to apply the technique in multiple domains to investigate the impact and applicability of semantic explanations across a wider array of problems.

Our work on task analysis, for explanation visualisations of text, will be used to inform future visualisation design and implementation. As new visualisations for explanation methods are proposed they will be evaluated within the framework of identified tasks. The recommended design specifications for supporting these tasks will address the identified gaps in explanation visualisation for text.

AttViz allows for direct model inspection via self-attention. In future work we will further explore potentially interesting relations emerging from the attention matrices. Privacy is a concern when investigating data which has not been anonymized, in future work, we will develop privacy preserving techniques for the investigation of self-attention.

6 Associated outputs

Description	URL	Availability
AttViz tutorials and code	github.com/EMBEDDIA/attviz	Public (GPLv3)
AttViz self-attention visualization server	attviz.ijs.si	Public
MCD AE and VAE data generators	github.com/EMBEDDIA/MCD-VAE	Public (MIT)
TransShap library	https://github.com/EMBEDDIA/TransSHAP	Public (MIT)
*Robust SHAP, IME, and IME	https://github.com/domenVres/Robust-LIME-SHAP-and-IME	To become public
*ReEx library	https://github.com/OpaqueRelease/ReEx	Public (GPLv3)

* Resources marked here as “To become public” are available only within the consortium while under development and/or associated with work yet to be published. They will be released publicly when the associated work is completed and published.

Parts of this work are also described in detail in the following publications.

Citation	Status	Appendix
Kokalj, Enja, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. "BERT meets Shapley: Extending SHAP Explanations to Transformer-based Classifiers." In Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation. 2021.	Published	Appendix A
Domen Vreš and Marko Robnik-Šikonja. Better Sampling in Explanation Methods can Prevent Dieselgate-Like Deception. 2021. (arXiv:2101.11702v1)	Submitted	Appendix B
Perdih, Timen Stepišnik, Nada Lavrač, and Blaž Škrlj. "Semantic Reasoning from Model-Agnostic Explanations." In 2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI). 2021. (preprint http://arxiv.org/abs/2106.15433)*	Published	Appendix C
Škrlj, Blaž, Shane Sheehan, Nika Eržen, Marko Robnik-Šikonja, Saturnino Luz, and Senja Pollak. Exploring Neural Language Models via Analysis of Local and Global Self-Attention Spaces. Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation. 2021.	Published	Appendix D

* In this paper, the EMBEDDIA acknowledgement was added only in the ArXiv version.

References

Angwin, J., Larson, J., Mattu, S., & Kirchner, L. (2016). Machine bias. *ProPublica, May 23*.

Arras, L., Horn, F., Montavon, G., Müller, K.-R., & Samek, W. (2017). What is relevant in a text document?: An interpretable machine learning approach. *PloS ONE*, *12*(8), e0181142.

Bertin, J. (1983). *Semiology of Graphics*. University of Wisconsin Press.

Chen, H., Zheng, G., & Ji, Y. (2020, 04). *Generating hierarchical explanations on text classification via feature interaction detection.*

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Jin, X., Du, J., Wei, Z., Xue, X., & Ren, X. (2019, 11). *Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models.*

Kaser, O., & Lemire, D. (2007). Tag-cloud drawing: Algorithms for cloud visualization. *arXiv preprint cs/0703109*.

Kokalj, E., Škrlj, B., Lavrač, N., Pollak, S., & Robnik-Šikonja, M. (2021). Bert meets shapley: Extending shap explanations to transformer-based classifiers. In *Proceedings of the eacl hackashop on news media content analysis and automated report generation* (pp. 16–21).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lemaire, V., Féraud, R., & Voisine, N. (2008). Contact Personalization using a Score Understanding Method. In *Proceedings of International Joint Conference on Neural Networks (IJCNN).*

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, *2*(1), 2522-5839.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4768–4777).

Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., ... Lee, S.-I. (2018). Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature Biomedical Engineering*, *2*(10), 749.

Luz, S., & Sheehan, S. (2020). Methods and visualization tools for the analysis of medical, political and scientific concepts in genealogies of knowledge. *Palgrave Communications*, *6*(1), 1–20.

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, *5*(2), 110–141.

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, *55*, 169-186.

Miok, K., Nguyen-Doan, D., Zaharie, D., & Robnik-Šikonja, M. (2019). Generating data using Monte Carlo dropout. In *International conference on intelligent computer communication and processing (iccp)* (p. 509-515).

Munzner, T. (2009). A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics*, *15*(6), 921–928.

Redmond, M., & Baveja, A. (2002). A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, *141*, 660-678.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of ACM SIGKDD* (pp. 1135–1144).

Robnik-Šikonja, M. (2016). Data generators for learning systems based on RBF networks. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(5), 926-938.

Robnik-Šikonja, M. (2019). semiartificial: Generator of semi-artificial data [Computer software manual]. Retrieved from `https://cran.r-project.org/package=semiArtificial` (R package version 2.3.1)

Robnik-Šikonja, M., & Kononenko, I. (2008). Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, *20*(5), 589-600.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, *20*, 53 - 65.

Sawada, S., & Toyoda, M. (2019). Model-agnostic visual explanation of machine learning models based on heat map. In *Eurovis (posters)* (pp. 37–39).

Shapley, L. S. (1953). A value for n-person games. In H. Kuhn & K. Tucker (Eds.), *Contributions to the Theory of Games, Vol. II* (pp. 307–317). Princeton University Press.

Škrlj, B., Sheehan, S., Eržen, N., Robnik-Šikonja, M., Luz, S., & Pollak, S. (2021). Exploring neural language models via analysis of local and global self-attention spaces. In *Proceedings of the eacl hackashop on news media content analysis and automated report generation* (pp. 76–83).

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020a). Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society* (pp. 180–186).

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020b). Fooling LIME and SHAP: Adversarial attacks on post-hoc explanation methods. In *Aaai/acm conference on ai, ethics, and society (aies).*

Stepišnik Perdih, T., Lavrač, N., & Škrlj, B. (2021). Semantic reasoning from model-agnostic explanations. In *2021 ieee 19th world symposium on applied machine intelligence and informatics (sami)* (pp. 000105–000110).

Ulčar, M., & Robnik-Šikonja, M. (2020). FinEst BERT and CroSloEngual BERT: less is more in multilingual models. In *Proceedings of Text, Speech, and Dialogue, TSD 2020.* (accepted)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Vreš, D., & Robnik-Šikonja, M. (2021). Better sampling in explanation methods can prevent dieselgate-like deception. *arXiv:2101.11702v1*. (Submitted)

Štrumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, *11*(Jan), 1–18.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (pp. 353–355).

Appendix A: BERT meets Shapley: Extending SHAP Explanations to Transformer-based Classifiers

BERT meets Shapley: Extending SHAP Explanations to Transformer-based Classifiers

Enja Kokalj
Jožef Stefan International Postgraduate School
Jožef Stefan Institute
enja.kokalj@ijs.si

Blaž Škrlj
Jožef Stefan International Postgraduate School
Jožef Stefan Institute

Nada Lavrač
Jožef Stefan International Postgraduate School
Jožef Stefan Institute

Senja Pollak
Jožef Stefan International Postgraduate School
Jožef Stefan Institute

Marko Robnik-Šikonja
Faculty for Computer and Information Science
Ljubljana

Abstract

Transformer-based neural networks offer very good classification performance across a wide range of domains, but do not provide explanations of their predictions. While several explanation methods, including SHAP, address the problem of interpreting deep learning models, they are not adapted to operate on state-of-the-art transformer-based neural networks such as BERT. Another shortcoming of these methods is that their visualization of explanations in the form of lists of most relevant words does not take into account the sequential and structurally dependent nature of text. This paper proposes the TransSHAP method that adapts SHAP to transformer models including BERT-based text classifiers. It advances SHAP visualizations by showing explanations in a sequential manner, assessed by human evaluators as competitive to state-of-the-art solutions.

1 Introduction

Recent wide spread use of deep neural networks (DNNs) has increased the need for their transparent classification, given that DNNs are black box models that do not offer introspection into their decision processes or provide explanations of their predictions and biases. Several methods that address the interpretability of machine learning models have been proposed. Model-agnostic explanation approaches are based on perturbations of inputs. The resulting changes in the outputs of the given model are the source of their explanations. The explanations of individual instances are commonly visualized in the form of histograms of the most impactful inputs. However, this is insufficient for text-based classifiers, where the inputs are sequential and structurally dependent.

We address the problem of incompatibility of modern explanation techniques, e.g., SHAP (Lundberg and Lee, 2017), and state-of-the-art pretrained transformer networks such as BERT (Devlin et al., 2019). Our contribution is twofold. First, we propose an adaptation of the SHAP method to BERT for text classification, called TransSHAP (Transformer-SHAP). Second, we present an improved approach to visualization of explanations that better reflects the sequential nature of input texts, referred to as the TransSHAP visualizer, which is implemented in the TransSHAP library.

The paper is structured as follows. We first present the background and motivation in Section 2. Section 3 introduces TransSHAP, an adapted method for explaining transformer language model such as BERT, which includes the TransSHAP visualizer for improved visualization of the generated explanations. Section 4 presents the results of an evaluation survey, followed by the discussion of results and the future work in Section 5.

2 Background and motivation

We first present the transformer-based language models, followed by an outline of perturbation-based explanation methods, in particular the SHAP method. We finish with the overview of visualizations for prediction explanations.

BERT (Devlin et al., 2019) is a large pretrained language model based on the transformer neural network architecture (Vaswani et al., 2017). Nowadays, BERT models exist in many mono- and multilingual variants. Fine-tuning BERT-like models to a specific task produces state-of-the-art results in many natural language processing tasks, such as text classification, question answering, POS-

Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation, pages 16–21
April 19, 2021

tagging, dependency parsing, inference, etc.

There are two types of explanation approaches, general and model specific. The general explanation approaches are applicable to any prediction model, since they perturb the inputs of a model and observe changes in the model's output. The second type of explanation approaches are specific to certain types of models, such as support vector machines or neural networks, and exploit the internal information available during training of these methods. We focus on general explanation methods and address their specific adaptations for use in text classification, more specifically, in text classification with transformer models such as BERT.

The most widely used perturbation-based explanation methods are IME (Štrumbelj and Kononenko, 2010), LIME (Ribeiro et al., 2016), and SHAP (Lundberg and Lee, 2017). Their key idea is that the contribution of a particular input value (or set of values) can be captured by 'hiding' the input and observing how the output of the model changes. In this work, we focus on the state-of-the-art explanation method SHAP (SHapley Additive exPlanations) that is based on the Shapley value approximation principle. Lundberg and Lee (2017) noted that several existing methods, including IME and LIME, can be regarded as special cases of this method.

We propose an adaptation of SHAP for BERT-like classifiers, but the same principles are trivially transferred to LIME and IME. To understand the behavior of a prediction model applied to a single instance, one should observe perturbations of all subsets of input features and their values, which results in exponential time complexity. Štrumbelj and Kononenko (2010) showed that the contribution of each variable corresponds to the Shapley value from the coalition game, where players correspond to input features, and the coalition game corresponds to the prediction of an individual instance. Shapley values can be approximated in time linear to the number of features.

The visualization approaches implemented in the explanation methods LIME and SHAP are primarily designed for explanations of tabular data and images. Although the visualization with LIME includes adjustments for text data, the resulting explanations are presented in the form of histograms that are sometimes hard to understand, as Figure 1 shows. The visualization with SHAP for the same sentence is illustrated in Figure 2. Here, the features with the strongest impact on the prediction correspond to longer arrows that point in the direction of the predicted class. For textual data this representation is non-intuitive.

Various approaches have been proposed to interpret neural text classifiers. Some of them focus on adapting existing SHAP based explanation methods by improving different aspects, e.g., the word masking (Chen and Ji, 2020), or reducing feature dimension (Zhao et al., 2020), while others explore the complex interactions between words (contextual decomposition) that are crucial when dealing with textual data but are ignored by other post-hoc explanation methods (Jin et al., 2019; Chen et al., 2020).

3 TransSHAP: The SHAP method adapted for BERT

Many modern deep neural networks, including transformer networks (Vaswani et al., 2017) such as BERT-like models, split the input text into subword tokens. However, perturbation-based explanation methods (such as IME, LIME, and SHAP) have problems with the text input and in particular subword input, as the credit for a given output cannot be simply assigned to clearly defined units such as words, phrases, or sentences. In this section, we first present the components of the new methodology and describe the implementation details required to make explanation method SHAP to work with state-of-the-art transformer prediction models such as BERT, followed by a brief description of the dataset used for training the model. Finally we introduce the TransSHAP visualizer, the proposed visualization method for text classification with neural networks. We demonstrate it using the SHAP method and the BERT model.

3.1 TransSHAP components

The model-agnostic implementation of the SHAP method, named Kernel SHAP[1], requires a classifier function that returns probabilities. Since SHAP contains no support for BERT-like models that use subword input, we implemented custom functions for preprocessing the input data for SHAP, to get the predictions from the BERT model, and to prepare data for the visualization.

Figure 3 shows the components required by SHAP in order to generate explanations for the

[1]We use the Kernel SHAP implementation of the SHAP method: https://github.com/slundberg/shap.

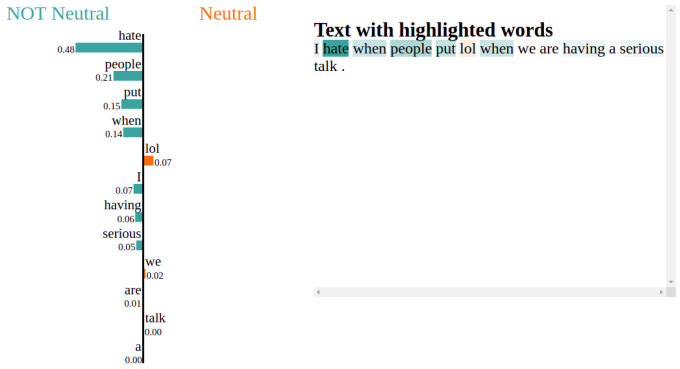


Figure 1: Visualization of prediction explanation with LIME.

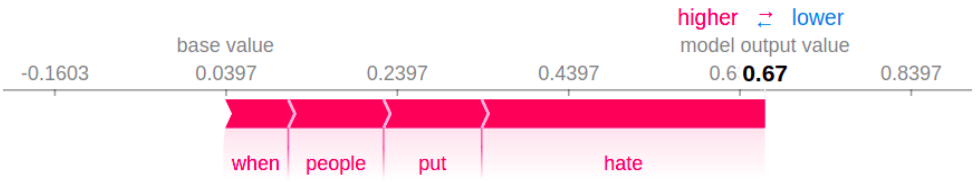


Figure 2: Visualization of prediction explanation with SHAP.

predictions made by the BERT model. The text data we want to interpret is used as an input to Kernel SHAP along with the special classifier function we constructed, which is necessary since SHAP requires numerical input in a tabular form.

To achieve this, we first convert the sentence into its numerical representation. This procedure consists of splitting the sentence into tokens and then preprocessing it. The preprocessing of different input texts is specific to their characteristics (e.g., tweets). The result is a list of sentence fragments (with words, selected punctuation marks and emojis), which serves as a basis for word perturbations (i.e. word masking). Each unique fragment is assigned a unique numerical key (i.e. index). We refer to a sentence, represented with indexes, as *an indexed instance*.

In summary, the TransSHAP's classifier function first converts each input instance into a word-level representation. Next, the representation is perturbed in order to generate new, locally similar instances which serve as a basis for the constructed explanation. This perturbation step is performed by the original SHAP. Then the perturbed versions of the sentence are processed with the BERT tokenizer that converts the sentence fragments to sub-word tokens. Finally, the predictions for the new locally generated instances are produced and returned to the Kernel SHAP explainer. With this modification, SHAP is able to compute the features' impact on the prediction (i.e. the explanation).

3.2 Datasets and models

We demonstrate our TransSHAP method on tweet sentiment classification. The dataset contains 87,428 English tweets with human annotated sentiment labels (positive, negative and neutral). For tweets we split input instances using the TweetTokenizer function from NLTK library[2], we removed apostrophes, quotation marks and all punctuation marks except for exclamation and question marks. We fine-tuned the CroSloEngual BERT model (Ulčar and Robnik-Šikonja, 2020) on this classification task and the resulting model achieved the classification accuracy of 66.6%.

3.3 Visualization of a prediction explanation for the BERT model

To make a visualization of predictions better adapted to texts, we modified the histogram-based visualizations used in IME, LIME and SHAP for

[2] https://www.nltk.org

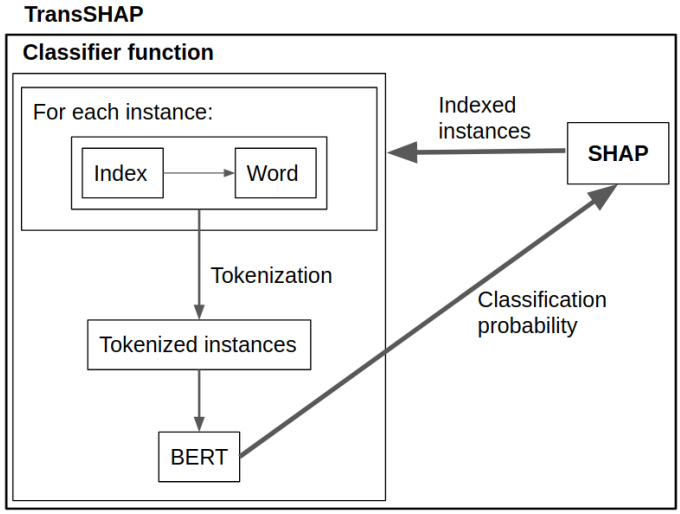


Figure 3: TransSHAP adaptation of SHAP to the BERT language model by introducing our classifier function.

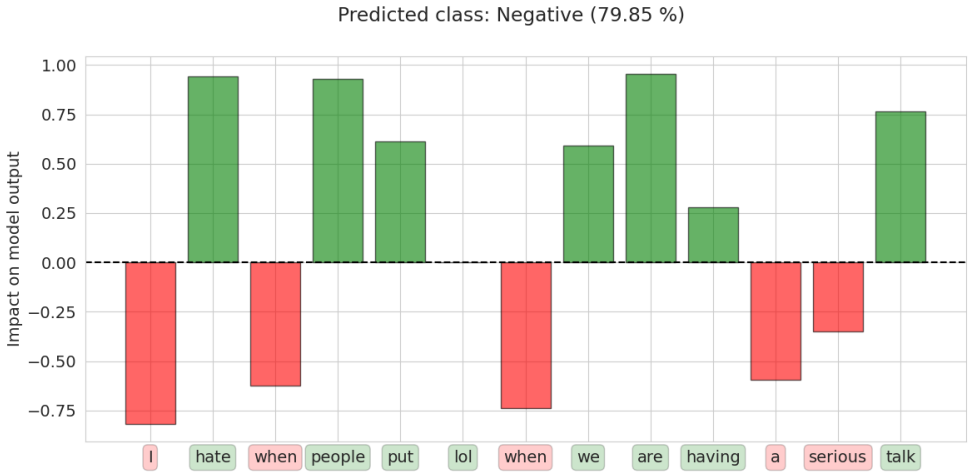


Figure 4: TransSHAP visualization of prediction explanations for negative sentiment. We obtained the features' contribution values with the SHAP method. It is evident that the word 'hate' strongly contributed to the negative sentiment classification, while the word 'lol' (laughing out loud) slightly opposed it.

tabular data. Figure 4 is an example of our visualization for explaining text classifications. It was inspired by the visualization used by the LIME method but we made some modifications with the aim of making it more intuitive and better adapted to sequences. Instead of the horizontal bar chart of features' impact on the prediction sorted in descending order of feature impact, we used the vertical bar chart and presented the features (i.e. words) in the order they appear in the original sentence. In this way, the graph allows the user to compare the direction of the impact (positive/negative) and also the magnitude of impact for individual words. The bottom text box representation of the sentence shows the words colored green if they significantly contributed to the prediction and red if they significantly opposed it.

4 Evaluation

We evaluated the novel visualization method using an online survey. The targeted respondents were researchers and PhD students not involved in the study that mostly had some previous experience with classifiers and/or their explanation methods. In the survey, the respondents were presented with three visualization methods on the same example: two visualizations were generated by existing libraries, LIME and SHAP, and the third one used our novel TransSHAP library. Respondents were asked to evaluate the quality of each visualization, suggest possible improvements, and rank the three methods.[3]

The results of 38 completed surveys are as follows. The most informative features of the visualization layout recognized by the users were the impact each word had on a prediction and the importance of the word contributions shown in a sequential view. The positioning of the visualization elements for each of the three methods was rated on the scale of 1 to 5. Our method achieved the highest average score of 3.66 (63.1% of the respondents rated it with a score of 4 or 5), second best was the LIME method with an average score of 3.13 (39.1% rated it with 4 or 5), and the SHAP method was rated as the worst with an average of 2.42 (81.5% rated it with 1 or 2). Regarding the question whether they would use each visualization method, LIME scored highest (44.7% voted "Yes"), TransSHAP closely followed (42.1% voted "Yes"), while SHAP was not praised (34.2% voted "Yes"). The overall ranking also corresponds to these results. LIME got the most votes (54.3%), TransSHAP was voted second best (40.0% of votes), and SHAP was the least desirable (5.7% of votes). In addition, we asked the participants to choose the preferred usage of the method out of the given options. The TransSHAP and SHAP methods were considered most useful for the purpose of debugging and bias detection, while the LIME method was also recognized as suitable for explaining a model to other researchers (usage in scientific articles).

5 Conclusion and further work

We presented the TransSHAP library, an extension of the SHAP explanation approach for transformer neural networks. TransSHAP offers a novel testing ground for better understanding of neural text classifiers, and will be freely accessible after acceptance of the paper (for review purposes available here: https://bit.ly/2UVY2Dy).

The explanations obtained by TransSHAP were quantitatively compared in a user survey, where we assessed the visualization capabilities, showing that the proposed TransSHAP's visualizations were simple, yet informative when compared to existing instance-based visualizations produced by LIME or SHAP. TransSHAP was scored better than SHAP, while LIME was scored slightly better in terms of overall user preference. However, in specific elements, such as positioning of the visualization elements, the visualization produced by TransSHAP is slightly better.

In further work, we plan to address problems of the perturbation-based explanation process when dealing with textual data. Currently, TransSHAP only supports random sampling from the word space, which may produce unintelligible and grammatically wrong sentences, and overall completely uninformative texts. We intend to take into account specific properties of text data and apply language models in the sampling step of the method. We plan to restrict the sampling candidates for each word based on their part of speech and general context of the sentence. We believe that better sampling will improve the speed of explanations and decrease the variance of explanations. Furthermore, the explanations could be additionally improved by expanding the features of explanations from individual words to larger textual units consisting of words that are grammatically and semantically linked.

Acknowledgements

We would like to acknowledge the Slovenian Research Agency (ARRS) for funding the first and the second author through young researcher grants and supporting other authors through the research program *Knowledge Technologies* (P2-0103) and the research project *Semantic Data Mining for Linked Open Data*. Further, we acknowledge the European Union's Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings for Cross-Lingual Embeddings for Less-Represented Languages in European News Media).

[3]The survey questions are available here: https://forms.gle/icpYvHH78oE2TCJt7.

References

Hanjie Chen and Yangfeng Ji. 2020. Learning variational word masks to improve the interpretability of neural text classifiers.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xisen Jin, Junyi Du, Zhongyu Wei, Xiangyang Xue, and Xiang Ren. 2019. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM.

Matej Ulčar and Marko Robnik-Šikonja. 2020. FinEst BERT and CroSloEngual BERT: less is more in multilingual models. In *Proceedings of Text, Speech, and Dialogue, TSD 2020*. Accepted.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Erik Štrumbelj and Igor Kononenko. 2010. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18.

Wei Zhao, Tarun Joshi, Vijayan Nair, and Agus Sudjianto. 2020. Shap values for explaining cnn-based text classification models.

Appendix B: Better Sampling in Explanation Methods can Prevent Dieselgate-Like Deception

BETTER SAMPLING IN EXPLANATION METHODS CAN PREVENT DIESELGATE-LIKE DECEPTION

Domen Vreš, Marko Robnik-Šikonja
University of Ljubljana, Faculty of Computer and Information Science
Večna pot 113, 1000 Ljubljana, Slovenia
domen.vres@siol.net, marko.robnik@fri.uni-lj.si

ABSTRACT

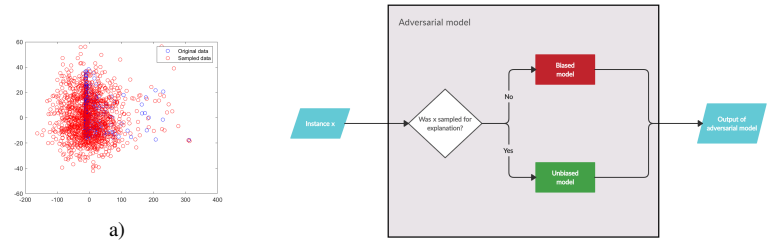
Machine learning models are used in many sensitive areas where, besides predictive accuracy, their comprehensibility is also important. Interpretability of prediction models is necessary to determine their biases and causes of errors and is a prerequisite for users' confidence. For complex state-of-the-art black-box models, post-hoc model-independent explanation techniques are an established solution. Popular and effective techniques, such as IME, LIME, and SHAP, use perturbation of instance features to explain individual predictions. Recently, Slack et al. (2020) put their robustness into question by showing that their outcomes can be manipulated due to poor perturbation sampling employed. This weakness would allow dieselgate type cheating of owners of sensitive models who could deceive inspection and hide potentially unethical or illegal biases existing in their predictive models. This could undermine public trust in machine learning models and give rise to legal restrictions on their use.

We show that better sampling in these explanation methods prevents malicious manipulations. The proposed sampling uses data generators that learn the training set distribution and generate new perturbation instances much more similar to the training set. We show that the improved sampling increases the LIME and SHAP's robustness, while the previously untested method IME is already the most robust of all.

1 INTRODUCTION

Machine learning models are used in many areas where besides predictive performance, their comprehensibility is also important, e.g., in healthcare, legal domain, banking, insurance, consultancy, etc. Users in those areas often do not trust a machine learning model if they do not understand why it made a given decision. Some models, such as decision trees, linear regression, and naïve Bayes, are intrinsically easier to understand due to the simple representation used. However, complex models, mostly used in practice due to better accuracy, are incomprehensible and behave like black boxes, e.g., neural networks, support vector machines, random forests, and boosting. For these models, the area of explainable artificial intelligence (XAI) has developed post-hoc explanation methods that are model-independent and determine the importance of each feature for the predicted outcome. Frequently used methods of this type are IME (Štrumbelj & Kononenko, 2013), LIME (Ribeiro et al., 2016), and SHAP (Lundberg & Lee, 2017).

To determine the features' importance, these methods use perturbation sampling. Slack et al. (2020) recently noticed that the data distribution obtained in this way is significantly different from the original distribution of the training data as we illustrate in Figure 1a. They showed that this can be a serious weakness of these methods. The possibility to manipulate the post-hoc explanation methods is a critical problem for the ML community, as the reliability and robustness of explanation methods are essential for their use and public acceptance. These methods are used to interpret otherwise black-box models, help in debugging models, and reveal models' biases, thereby establishing trust in their behavior. Non-robust explanation methods that can be manipulated can lead to catastrophic consequences, as explanations do not detect racist, sexist, or otherwise biased models if the model owner wants to hide these biases. This would enable dieselgate-like cheating where owners of sensitive prediction models could hide the socially, morally, or legally unacceptable biases



a) b)

Figure 1: a) PCA based visualization of a part of the COMPAS dataset. The blue points show the original instances, and the red points represent instances generated with the perturbation sampling used in the LIME method. The distributions are notably different. b) The idea of the attack on explanation methods based on the difference of distributions. The attacker's adversarial model contains both the biased and unbiased model. The decision function that is part of the cheating model decides if the instance is outside the original distribution (i.e. used only for explanation) or an actual instance. If the case of an actual instance, the result of the adversarial model is equal to the result of the biased model, otherwise it is equal to the result of the unbiased model.

present in their models. As the schema of the attack on explanation methods on Figure 1b shows, owners of prediction models could detect when their models are examined and return unbiased predictions in this case and biased predictions in normal use. This could have serious consequences in areas where predictive models' reliability and fairness are essential, e.g., in healthcare or banking. Such weaknesses can undermine users' trust in machine learning models in general and slow down technological progress.

In this work, we propose to change the main perturbation-based explanation methods and make them more resistant to manipulation attempts. In our solution, the problematic perturbation-based sampling is replaced with more advanced sampling, which uses modern data generators that better capture the distribution of the training dataset. We test three generators, the RBF network based generator (Robnik-Šikonja, 2016), random forest-based generator, available in R library semiArtificial (Robnik-Šikonja, 2019), as well as the generator using variational autoencoders (Miok et al., 2019). We show that the modified gLIME and gSHAP methods are much more robust than their original versions. For the IME method, which previously was not analyzed, we show that it is already quite robust. We release the modified explanation methods under the open-source license[1].

In this work, we use the term robustness of the explanation method as a notion of resilience against adversarial attacks, i.e. as the ability of an explanation method to recognize the biased classifier in an adversary environment. This type of robustness could be more formally defined as the number of instances where the adversarial model's bias was correctly recognized. We focus on the robustness concerning the attacks described in Slack et al. (2020). There are other notions of robustness in explanation methods; e.g., (Alvarez-Melis & Jaakkola, 2018) define the robustness of the explanations in the sense that similar inputs should give rise to similar explanations.

The remainder of the paper is organized as follows. In Section 2, we present the necessary background and related work on explanation methods, attacks on them, and data generators. In Section 3, we propose a defense against the described weaknesses of explanation methods, and in Section 4, we empirically evaluate the proposed solution. In Section 5, we draw conclusions and present ideas for further work.

2 Background and Related Work

In this section, we first briefly describe the background on post-hoc explanation methods and attacks on them, followed by data generators and related works on the robustness of explanation methods.

2.1 Post-hoc Explanation Methods

The current state-of-the-art perturbation-based explanation methods, IME, LIME, and SHAP, explain predictions for individual instances. To form an explanation of a given instance, they measure

[1]https://github.com/domenVres/Robust-LIME-SHAP-and-IME

the difference in prediction between the original instance and its neighboring instances, obtained with perturbation sampling. Using the generated instances, the LIME method builds a local interpretable model, e.g., a linear model. The SHAP and IME methods determine the impact of the features as Shapley values from the coalitional game theory (Shapley, 1988). In this way, they assure that the produced explanations obey the four Shapley fairness axioms (Štrumbelj & Kononenko, 2013). Due to the exponential time complexity of Shapley value calculation, both methods try to approximate them. The three methods are explained in detail in the above references, and a formal overview is presented in Appendix A, while below, we present a brief description. In our exposition of the explanation methods, we denote with f the predictive model and with x the instance we are explaining.

Explanations of instances with the LIME method is obtained with an interpretable model g. The model g has to be both locally accurate (so that it can obtain correct feature contributions) and simple (so that it is interpretable). Therefore in LIME, g is a linear regression model trained on the instances sampled in the neighborhood of the explained instance x, which is first normalized.The perturbed explanation instances are generated by adding Gaussian noise to each feature of x independently. Using linear models to determine the contributions of features, LIME is fast but may produce poor explanations for instances close to complex decision boundaries.

SHAP estimates Shapley values using the approach of LIME (i.e. local linear models computed with the weighted least squares algorithm) but using a different neighborhood and loss function for local accuracy, which ensures that Shapley values are obtained. The sampling of explanation instances in SHAP relies on the distribution set D, obtained from the training set, by computing the centroids of k-means clusters.

IME computes Shapley values using a pure sampling approach (no proxy with an interpretable model). Its sampling replaces part of the explained instance with existing values of selected features and then averages differences in the prediction of explained instance and samples. This makes the method slower but also potentially more robust as the method does not assume the shape of the dependency.

2.2 Attacks on explanations

The idea of the attack on explanations presented in Slack et al. (2020) is that the attacker creates an adversarial (biased) model whose behavior it wants to hide from explanation methods, e.g., a racist model that does not grant credits to blacks. As Figure 1b illustrates, a part of the attacker's adversarial model is an unbiased model, which, e.g., does not take disputed attributes such as race into account. The adversarial model tries to manipulate the explanation method by behaving like an unbiased model on perturbed instances used for the explanation – in this case, it uses the unbiased model's output. On instances from the original distributions that are a part of normal model use, the adversarial model uses the biased model's output. To distinguish between the original and perturbation-sampled data, the adversarial model contains a decision model that selects the right type of model (biased or unbiased) based on the predicted type of instances.

The reason that the decision function can distinguish between the original data and sampled instances are the shortcomings of explanation methods that use the perturbation sampling (see Figure 1a). For that graph, we created new instances based on the COMPAS dataset (see Section 4) with the Gaussian noise as in the LIME explanation method. We mapped the original and sampled instances to 2D space using the PCA transformation. We can see that the sampled data is distributed differently than the original data. As the difference is revealed with PCA, other machine learning algorithms can also effectively distinguish between sampled and original data.

The attack can be formally defined in the following way. The attacker has a biased model b and a sample S from the distribution X_{dist}. The attacker has no additional information about the distribution X_{dist}. The attacker's goal is to produce an adversarial model e that behaves like b on the data from X_{dist} but does not reveal b's bias to the explanation methods. We assume that the attacker also has an unbiased model ψ that hides the f's bias. The attacker creates a decision model d, that should output 1 if the input instance x is from X_{dist} and 0 otherwise. The model d is trained on S and generated perturbation samples. The attacker creates the adversarial model e with its output defined

by the following equation:

$$e(x) = \begin{cases} b(x), & d(x) = 1 \\ \psi(x), & d(x) = 0 \end{cases} \quad (1)$$

2.3 Data generators

We used three different data generators based on different algorithms, modeling the distribution of the training set: variational autoencoder with Monte Carlo dropout (Miok et al., 2019), RBF network (Robnik-Šikonja, 2016), and random forest ensemble (Robnik-Šikonja, 2019). In the remainder of the paper, we refer to the listed generators consecutively as MCD-VAE, rbfDataGen, and TreeEnsemble.

Autoencoder (AE) consists of two neural networks called encoder and decoder. It aims to compress the input instances by passing them through the encoder and then reconstructing them to the original values with the decoder. Once the AE is trained, it can be used to generate new instances. Variational autoencoder (Doersch, 2016) is a special type of autoencoder, where the vectors z in the latent dimension (output of the encoder and input of the decoder) are normally distributed. Encoder is therefore approximating the posterior distribution $p(z|x)$, where we assume $p(z|x) \sim \mathcal{N}(\mu_x, \Sigma_x)$. The generator proposed by Miok et al. (2019) uses the Monte Carlo dropout (Gal & Ghahramani, 2016) on the trained decoder. The idea of this generator is to propagate the instance x through the encoder to obtain its latent encoding z. This can be propagated many times through the decoder, obtaining every time a different result due to the Monte Carlo dropout but preserving similarity to the original instance x.

The RBF network (Moody & Darken, 1989) uses Gaussian kernels as hidden layer units in a neural network. Once the network's parameters are learned, the rbfDataGen generator (Robnik-Šikonja, 2016) can sample from the normal distributions, defined with obtained Gaussian kernels, to generate new instances.

The TreeEnsemble generator (Robnik-Šikonja, 2019) builds a set of random trees (forest) that describe the data. When generating new instances, the generator traverses from the root to the leaves of a randomly chosen tree, setting values of features in the decision nodes on the way. When reaching a leaf, it assumes that it has captured the dependencies between features. Therefore, the remaining features can be generated independently according to the observed empirical distribution in this leaf. For each generated instance, all attributes can be generated in one leaf, or another tree can be randomly selected where unassigned feature values are filled in. By selecting different trees, different features are set in the interior nodes and leaves.

2.4 Related work on robustness of explanations

The adversarial attacks on perturbation based explanation methods were proposed by Slack et al. (2020), who show that LIME and SHAP are vulnerable due to the perturbation based sampling used. We propose the solution to the exposed weakness in SHAP and IME based on better sampling using data generators adapted to the training set distribution.

In general, the robustness of explanation methods has been so far poorly researched. There are claims that post-hoc explanation methods shall not be blindly trusted, as they can mislead users (deliberately or not) and disguise gender and racial discrimination (Lipton, 2016). Selbst & Barocas (2018) and Kroll et al. (2017) showed that even if a model is completely transparent, it is hard to detect and prevent bias due to the existence of correlated variables.

Specifically, for deep neural networks and images, there exist adversarial attacks on saliency map based interpretation of predictions, which can hide the model's bias (Dombrowski et al., 2019; Heo et al., 2019; Ghorbani et al., 2019). Dimanov et al. (2020) showed that a bias of a neural network could be hidden from post-hoc explanation methods by training a modified classifier that has similar performance to the original one, but the importance of the chosen feature is significantly lower.

The kNN-based explanation method, proposed by Chakraborty et al. (2020), tries to overcomes the inadequate perturbation based sampling used in explanation methods by finding similar instances to the explained one in the training set instead of generating new samples. This solution is inadequate for realistic problems as the problem space is not dense enough to get reliable explanations. Our defense of current post-hoc methods is based on superior sampling, which has not yet been tried.

Saito et al. (2020) use the neural CT-GAN model to generate more realistic samples for LIME and prevent the attacks described in Slack et al. (2020). We are not aware of any other defenses against the adversarial attacks on post-hoc explanations.

3 Robustness through better sampling

We propose the defense against the adversarial attacks on explanation methods that replaces the problematic perturbation sampling with a better one, thereby making the explanation methods more robust. We want to generate the explanation data in such a way that the attacker cannot determine whether an instance is sampled or obtained from the original data. With an improved sampling, the adversarial model shown in Figure 1b shall not determine whether the instance x was generated by the explanation method, or it is the original instance the model has to label. With a better data generator, the adversarial model cannot adjust its output properly and the deception, described in Section 2.2, becomes ineffective.

The reason for the described weakness of LIME and SHAP is inadequate sampling used in these methods. Recall that LIME samples new instances by adding Gaussian noise to the normalized feature values. SHAP samples new instances from clusters obtained in advance with the k-means algorithm from the training data.

Instead of using the Gaussian noise with LIME, we generate explanation samples for each instance with one of the three better data generators, MCD-VAE, rbfDataGen, or TreeEnsemble (see Section 2.3). We call the improved explanation methods gLIME and gSHAP (g stands for generator-based). Using better generators in the explanation methods, the decision function in the adversarial model will less likely determine which predicted instances are original and which are generated for the explanation purposes.

Concerning gLIME, we generate data in the vicinity of the given instance using MCD-VAE, as the LIME method builds a local model. Using the TreeEnsemble and rbfDataGen generators, we do not generate data in the neighborhood of the given instance but leave it to the proximity measure of the LIME method to give higher weights to instances closer to the explained one.

In SHAP, the perturbation sampling replaces the values of hidden features in the explained instance with the values from the distribution set D. The problem with this approach is that it ignores the dependencies between features. For example, in a simple dataset with two features, house size, and house price, let us assume that we hide the house price, but not the house size. These two features are not independent because the price of a house increases with its size. Suppose we are explaining an instance that represents a large house. Disregarding the dependency, using the sampled set D, SHAP creates several instances with a low price, as such instances appeared in the training set. In this way, the sampled set contains many instances with a small price assigned to a large house, from which the attacker can determine that these instances were created in perturbation sampling and serve only for the explanation.

In the proposed gSHAP, using the MCD-VAE and TreeEnsemble generators, the distribution set D is generated in the vicinity of the explained instance. In the sampled instances, some feature values of the explained instance are replaced with the generated values, but the well-informed generators consider dependencies between features detected in the original distribution. This will make the distribution of the generated instances very similar to the original distribution. In our example, the proposed approach generates new instances around the instance representing a large house, and most of these houses will be large. As the trained generators capture the original dataset's dependencies, these instances will also have higher prices. This will make it difficult for the attacker to recognize the generated instances used in explanations. The advantage of generating the distribution set close to the observed instance is demonstrated in Appendix B.

The rbfDataGen generator does not support the generation of instances around a given instance. Therefore, we generate the sampled set based on the whole training set and not for each instance separately (we have this option also for TreeEnsemble). This is worse than generating the distribution set around the explained instance but still better than generating it using the k-means sampling in SHAP. There are at least three advantages. First, the generated distribution set D can be larger. The size of the k-means distribution set cannot be arbitrarily large because it is limited by the number of clusters in the training set. Second, the centroids of the clusters obtained by the k-means algorithm

Data set	# inst.	# features	# categorical	sensitive	unrelated 1	unrelated 2
COMPAS	6172	7	4	*race*	*random1*	*random2*
German	1000	25	15	*gender*	*pctOfIncome*	/
CC	1994	100	0	*racePctWhite*	*random1*	*random2*

Table 1: Basic information and sensitive features in the the used data sets. The target variable is not included in the number of features and is binary for all data sets. The *pctOfIncome* full name is *loanRateAsPercentOfIncome*.

do not necessarily provide a good summary of the training set and may not be valid instances from training set distribution. They also do not capture well the density of instances (e.g., most of the data from the training set could be grouped in one cluster). Third, using the proposed generators, SHAP becomes easier to use compared to the k-means generator, where users have to determine the number of clusters, while the data generators we tested can be used in the default mode without parameters,

4 EVALUATION

To evaluate the proposed improvements in the explanation methods, we first present the used datasets in Section 4.1, followed by the experiments. In Section 4.2, we test the robustness of gLIME, gSHAP, and gIME against the adversarial models. To be more realistic, we equipped the adversarial models with the same improved data generators we used in the explanation methods. It is reasonable to assume that attackers could also use better data generators when preparing their decision function, making their attacks much stronger. As the evaluation metric for the success of deception, we use the proportion of instances where the adversarial model deceived the explanation methods so that they did not detect sensitive attributes as being important in the prediction models. In Section 4.3, we test if enhanced generators produce different explanations than the original ones. As the attacker might decide to employ deception only when it is really certain that the predicted instance is used inside the explanation method, we test different thresholds of the decision function d from Equation (1) (currently set to 0.5). We report on this analysis in Section 4.4.

4.1 SENSITIVE DATASETS PRONE TO DECEPTION

Following (Slack et al., 2020), we conducted our experiments on three data sets from domains where a biased classifier could pose a critical problem, such as granting a credit, predicting crime recidivism, and predicting the violent crime rate. The basic information on the data sets is presented in Table 1. The statistics were collected after removing the missing values from the data sets and before we encoded categorical features as one-hot-encoded vectors.

COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a risk assessment used by the courts in some US states to determine the crime recurrence risk of a defendant. The dataset (Angwin et al., 2016)) includes criminal history, time in prison, demographic data (age, gender, race), and COMPAS risk assessment of the individual. The dataset contains data of 6,172 defendants from Broward Couty, Florida. The sensitive attribute in this dataset (the one on which the adversarial model will be biased) is race. African Americans, whom biased model associates with a high risk of recidivism, represent 51.4% of instances from the data set. This set's target variable is the COMPAS score, which is divided into two classes: a high and low risk. The majority class is the high risk, which represents 81.4% of the instances.

The German Credit dataset (German for the rest of the paper) from the UCI repository (Dua & Graff, 2019) includes financial (bank account information, loan history, loan application information, etc.) and demographic data (gender, age, marital status, etc.) for 1,000 loan applicants. A sensitive attribute in this data set is gender. Men, whom the biased model associates with a low-risk, represent 69% of instances. The target variable is the loan risk assessment, divided into two classes: a good and a bad customer. The majority class is a good customer, which represents 70% of instances.

Communities and Crime (CC) data set (Redmond & Baveja, 2002) contains data about the rate of violent crime in US communities in 1994. Each instance represents one community. The features are numerical and represent the percentage of the community's population with a certain property or the average of population in the community. Features include socio-economic (e.g., education, house size, etc.) and demographic (race, age) data. The sensitive attribute is the percentage of the

white race. The biased model links instances where whites' percentage is above average to a low rate of violent crime. The target variable is the rate of violent crime divided into two classes: high and low. Both classes are equally represented in the data set.

4.2 Robustness of explanation methods

To evaluate the robustness of explanation methods with added generators (i.e. gLIME, gSHAP, and gIME), we split the data into training and evaluation set in the ratio $90\% : 10\%$. We used the same training set for the training of adversarial models and explanation methods. We encoded categorical features as one-hot-encoded vectors.

We simulated the adversarial attack for every combination of generators used in explanation methods and adversarial models (except for method IME, where we did not use rbfDataGen, which cannot generate instances in the neighborhood of a given instance). When testing SHAP, we used two variants of the TreeEnsemble generator for explanation: generating new instances around the explained instance and generating new instances according to the whole training set distribution. In the LIME testing, we used only the whole training set variant of TreeEnsemble inside the explanation. In the IME testing, we only used the variant of Tree Ensemble that fills in the hidden values (called TEnsFillIn variant). For training the adversarial models, we used the whole training set based variant of TreeEnsemble, except for IME, where we used the TEnsFillIn variant. These choices reflect the capabilities of different explanation method and attempt to make both defense and attack realistic (as strong as possible). More details on training the decision model d inside the adversarial model can be found in Appendix D.

In all cases, the biased model b (see Section 2.2) was a simple function depending only on the value of the sensitive feature. The unbiased model ψ depended only on the values of unrelated features. The sensitive and unrelated features are shown on the right-hand side of Table 1. Features *random1* and *random2* were uniformly randomly generated from the $\{0, 1\}$ set. On COMPAS and CC, we simulated two attacks, one with a single unrelated feature (the result of ψ depends only on the value of the unrelated feature 1), and another one with two unrelated features.

For every instance in the evaluation set, we recorded the most important feature according to the used explanation method (i.e. the feature with the largest absolute value of the contribution as determined by the explanation method). The results are shown as a heatmap in Figure 2. The green color means that the explanation method was deceived in less than 30% of cases (i.e. the sensitive feature was recognized as the most important one in more than 70 % of the cases as the scale on the heatmap suggests), and the red means that it was deceived in more than 70% of cases (the sensitive feature was recognized as the most important one in less than 30 % of the cases as the scale on the heatmap suggests). We consider deception successful if the sensitive feature was not recognized as the most important by the explanation method (the sensitive features are the only relevant features in biased models b).

The gLIME method is more robust with the addition of rbfDataGen and TreeEnsemble than LIME and less robust with the addition of MCD-VAE. This suggests that parameters for MCD-VAE were not well-chosen, and this pattern repeats for other explanation methods. Both TreeEnsemble and rbfDataGen make gLIME considerably more robust on COMPAS and German datasets, but not on the CC dataset. We believe the reason for that is that features in CC are strongly interdependent, and many represent the same attribute as a fraction of value, e.g., we have the share of the white population, the share of the Asian population, and so on. This interdependence dictates that all fractions have to add up to 1. The data generators are unlikely to capture such strong conditional dependencies, but the adversarial model's decision function is likely to approximate it.

The gSHAP method is most robust when using the TreeEnsemble generator, but it shows less robust behavior than the gLIME method. The reason for that could be in the feature value replacement strategy used by the SHAP and gSHAP methods, which change only some of the feature's values with the help of the distribution set. This can lead to out-of-distribution instances if the instances in the distribution set are not close enough to the explained instance. With gLIME, we generate complete instances that are more likely to be in the distribution of the training set.

IME is quite robust even with the perturbation sampling, as we expected. This suggests that IME is the most robust of all three tested explanation methods and shall be considered the chosen method in

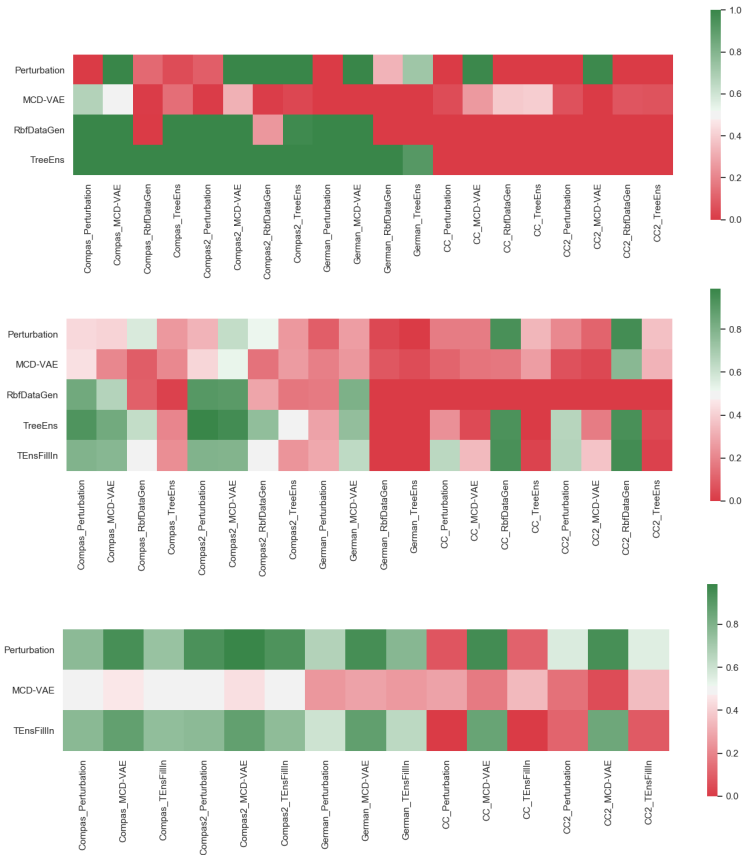


Figure 2: The robustness results for gLIME (top), gSHAP (middle), and gIME (bottom). The graphs show the proportion of evaluation set instances, where the sensitive feature was recognized as the most important by the used explanation method. Rows represent the generators used for explanations. The column labels consist of the name of the data set on which the experiment was performed and the name of the generator used for training of the adversarial model. Compas2 and CC2 denote an attack with two independent features. Perturbation represents the original sampling used in LIME, SHAP, and IME, TEnsFillIn represents the TreeEnsemble variant where new instances are generated around the given one, and TreeEns represents the generation from the whole distribution.

sensitive situations. The gIME results when using the TreeEnsemble generator (TEnsFillIn variant) are comparable to the original IME variant results. This suggests that sampling from a smaller data set, which represents the neighborhood of the explained instance, does not decrease the method's robustness.

4.3 Comparing explanations of original and modified methods

We checked if improved data generators affect explanations in a non-adversary environment. We compared the explanations of original and modified methods for four different classification models on three different datasets. As a measure of differences, we use the mean of the absolute difference between the original and enhanced explanations over the whole evaluation set. Details of these experiments are reported in Appendix F, below we shortly summarize the results, contained in Table 4.

The differences between original LIME and gLIME explanations are considerable (see the top part of Table 4). This is not surprising since LIME fits local linear models in its local explanations, which can strongly differ even for small perturbations of the model's input. The differences in SHAP and IME explanations are small (the average MAD is almost negligible). We can conclude that explanations of gSHAP and gIME are not significantly different from SHAP and IME in the non-adversary environment.

4.4 Behavior of adversarial classifier with different prediction thresholds

The attacker might be wary of being discovered by the regulator and decide to employ deception only when it is really certain that the predicted instance is used inside the explanation method. We express different levels of attacker's conservatism by changing the decision function threshold d from Equation (1), where it is currently set to 0.5. We tested the robustness of modified explanation methods on the COMPAS dataset with adversarial models that use different certainty thresholds for deploying the biased classifier (simulating different levels of aggressiveness). For results and a more detailed description of the experiment, see Appendix G. Below, we shortly summarize the results.

Even with different thresholds, gIME is still the most robust of the three explanation methods, and treeEnsemble still gives the best results as the data generator. The sensitive feature is recognized as the most important more often when the threshold is lower, which is expected as in this case the adversarial model behaves more aggressively. While the percentage of the instances on which the racist behavior of the adversarial model is recognized drops with higher values of the thresholds, it remains high enough to warn the regulator about the problematic behavior of the prediction model (especially in the case of gSHAP and gIME using treeEnsemble as data generator). We can conclude that the enhanced explanation methods remain robust enough, even with more conservative adversarial models.

5 Conclusions

We presented the defense against adversarial attacks on explanation methods. The attacks exploit the shortcomings of perturbation sampling in post-hoc explanation methods. This sampling used in these methods produces instances too different from the original distribution of the training set. This allows unethical owners of biased prediction models to detect which instances are meant for explanation and label them in an unbiased way. We replaced the perturbation sampling with data generators that better capture the distribution of a given data set. This prevents the detection of instances used in explanation and disarms attackers. We have shown that the modified gLIME and gSHAP explanation methods, which use better data generators, are more robust than the original variants, while IME is already quite robust. The difference in explanation values between original and enhanced gSHAP and gIME is negligible, while for gLIME, it is considerable. Our preliminary results in Appendix C show that using the TreeEnsemble generator, the gIME method converges faster and requires from 30-50% fewer samples.

The effectiveness of the proposed defense depends on the choice of the data generator and its parameters. While the TreeEnsemble generator turned out the most effective in our evaluation, in practice, several variants might need to be tested to get a robust explanation method. Inspecting authorities shall be aware of the need for good data generators and make access to training data of sensitive prediction models a legal requirement. Luckily, even a few non-deceived instances would be enough to raise the alarm about unethical models.

This work opens a range of possibilities for further research. The proposed defense and attacks shall be tested on other data sets with a different number and types of features, with missing data, and on different types of problems such as text, images, and graphs. The work on useful generators shall be extended to find time-efficient generators with easy to set parameters and the ability to generate new instances in the vicinity of a given one. Generative adversarial networks (Goodfellow et al., 2014) may be a promising line of research. The TreeEnsemble generator, currently written in pure R, shall be rewritten in a more efficient programming language. The MCD-VAE generator shall be studied to allow automatic selection of reasonable parameters for a given dataset. In SHAP sampling, we could first fix the values of the features we want to keep, and the values of the others would be generated using the TreeEnsemble generator.

ACKNOWLEDGMENTS

The work was partially supported by the Slovenian Research Agency (ARRS) core research programme P6-0411. This paper is supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media).

REFERENCES

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *ProPublica, May 23*, 2016.

Joymallya Chakraborty, Kewen Peng, and Tim Menzies. Making fair ML software using trustworthy explanation. *arXiv preprint arXiv:2007.02893*, 2020.

Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *SafeAI@ AAAI*, pp. 63–73, 2020.

Carl Doersch. Tutorial on variational autoencoders. *ArXiv*, abs/1606.05908, 2016.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, pp. 13589–13600, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2019. [Accessed: 9. 8. 2020].

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 1050–1059, 2016.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:3681–3688, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. In *Advances in Neural Information Processing Systems*, pp. 2925–2936, 2019.

Joshua A Kroll, Joanna Huey, Solon Barocas, Edward W Felten, Joel R Reidenberg, David G Robinson, and Harlan Yu. Accountable algorithms. *University of Pennsylvania Law Review*, 165(3): 633–705, 2017.

Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pp. 4765–4774. Curran Associates, Inc., 2017.

Kristian Miok, Deng Nguyen-Doan, Daniela Zaharie, and Marko Robnik-Šikonja. Generating data using Monte Carlo dropout. In *2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 509–515, 2019.

J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Michael Redmond and Alok Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141: 660–678, 2002.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

Marko Robnik-Šikonja. Data generators for learning systems based on RBF networks. *IEEE Transactions on Neural Networks and Learning Systems*, 27(5):926–938, 2016.

Marko Robnik-Šikonja and Igor Kononenko. Explaining classifications for individual instances. *IEEE Transactions on Knowledge and Data Engineering*, 20:589–600, 2008.

Marko Robnik-Šikonja. *semiArtificial: Generator of Semi-Artificial Data*, 2019. URL `https://cran.r-project.org/package=semiArtificial`. R package version 2.3.1.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

Sean Saito, Eugene Chua, Nicholas Capel, and Rocco Hu. Improving lime robustness with smarter locality sampling, 2020.

Andrew D Selbst and Solon Barocas. The intuitive appeal of explainable machines. *Fordham Law Review*, 87:1085, 2018.

Lloyd S. Shapley. A value for n-person games. In Alvin E. Roth (ed.), *The Shapley Value: Essays in Honor of Lloyd S. Shapley*, pp. 31–40. Cambridge University Press, 1988.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: Adversarial attacks on post-hoc explanation methods. In *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2020.

Erik Štrumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18, 2010.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, 2013.

A Details on post-hoc explanation methods

For the sake of completeness, we present further details on the explanation methods LIME (Ribeiro et al., 2016), SHAP (Lundberg & Lee, 2017), and IME (Štrumbelj & Kononenko, 2013). Their complete description can be found in the above-stated references. In our exposition of the explanation methods, we denote with f the predictive model, with x the instance we are explaining, and with n the number of features describing x.

A.1 LIME

Explanations of instances with the LIME method is obtained with an interpretable model g. The model g has to be both locally accurate (so that it can obtain correct feature contributions) and simple (so that it is interpretable).

The explanation of the instance x for the predictive model f, obtained with the LIME method, is defined with the following equation:

$$\xi(x) = \arg\min_{g \in G}(\mathcal{L}(f, g, \pi_x) + \Omega(g)), \tag{2}$$

where $\Omega(g)$ denotes the measure of complexity for interpretable model, $\pi_x(z)$ denotes the proximity measure between x and generated instances z, $\mathcal{L}(f, g, \pi_x)$ denotes the measure of local fidelity of interpretable model g to the prediction model f, and G denotes the set of interpretable models. We use the linear version of the LIME method, where G represents the set of linear models. With x' we denote the normalized presentation of instance x, i.e. the numerical attributes have their mean set to 0 and variance to 1, and the categorical attributes contain value 1 if they have the same value

as the exoplained instance and 0 otherwise. The proximity function π_x is defined on the normalized instances (hence we use the notation $\pi_{x'}$), and uses the exponential kernel: $\pi_{x'}(z') = e^{\frac{d(x',z')}{\sigma^2}}$, where $d(x', z')$ denotes a distance measure between x' and z'. The local fidelity measure $\mathcal{L}$ from Equation (2) is defined as:

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in \mathcal{Z}} \pi_{x'}(z')(f(z) - g(z'))^2, \tag{3}$$

where $\mathcal{Z}$ denotes the set of samples. In LIME, each generated sample is obtained by adding Gaussian noise to each feature of x' independently.

Using linear models as the set of interpretable models, LIME is relatively fast but may produce poor explanations for instances close to complex decision boundaries.

A.2 SHAP

We refer to SHAP as the method called Kernel SHAP by (Lundberg & Lee, 2017). SHAP essentially estimates Shapley values using LIME's approach, which means that the calculation of explanations is fast due to the use of local linear models computed with the weighted least squares algorithm. The explanation of the instance x with the SHAP method are feature contributions ϕ_i, $i = 1, 2, ..., n$ that are coefficients of the linear model $g(x') = \phi_0 + \sum_{i=1}^{n} \phi_i \cdot x'_i$, obtained with LIME, where $x'_i \in \{0, 1\}$ for $i \in \{1, 2, ..., n\}$.

As LIME does not compute Shapley values, this property is assured with the proper selection of functions $\Omega(g), \pi_{x'}(z')$, and $\mathcal{L}(f, g, \pi_{x'})$. Let us first define the function $h_x(z')$ that maps from the $\{0, 1\}^n$ to the original feature space. The function h_x is defined implicitly with the equation $f(h_x(z')) = E[f(z)|z_i = x_i \; \forall i \in \{j; z'_j = 1\}]$. This is the expected value of the model f when we do not know the values of features at indices where z' equals 0 (these values are hidden). Functions $\Omega(g), \pi_{x'}(z')$ and $\mathcal{L}(f, g, \pi_{x'})$ that enforce the computation of Shapley values are:

$$\Omega(g) = 0,$$

$$\pi_{x'}(z') = \frac{n}{\binom{n}{|z'|} \cdot |z'| \cdot (n - |z'|)},$$

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in \mathcal{Z}} (f(h_x(z')) - g(z'))^2 \cdot \pi_{x'}(z'),$$

where $|z'|$ denotes the number of nonzero features of z' and $\mathcal{Z} \subseteq 2^{\{0,1\}^n}$.

The main purpose of the sampling in this method is to determine the value $f(h_x(z'))$ because in general predictive models cannot work with hidden values. To determine $f(h_x(z'))$, SHAP uses the distribution set D which we obtain from the training set. For D, SHAP takes the centroids of clusters obtained by the k-means clustering algorithm on the training set. The number of clusters is set by the user. Value of $f(h_x(z'))$ is determined by the following sampling:

$$f(h_x(z')) = E[f(z)|z_i = x_i \; \forall i \in \{j; z'_j = 1\}] = \frac{1}{|D|} \sum_{d \in D} f(x_{[x_i = d_i, z'_i = 0]}), \tag{4}$$

where $x_{[x_i = d_i, z'_i = 0]}$ denotes instance x with features that are 0 in z' being set to the feature values from d.

A.3 IME

The explanation of x with the method IME are feature contributions ϕ_i, $i = 1, 2, ..., n$. Štrumbelj & Kononenko (2013) shoved that Shapley values are the only solution that takes into account contributions of interactions for every subset of features in a fair way. The i-th feature contribution can be calculated with the following expresiion:

$$\phi_i(x) = \frac{1}{n} \sum_{\pi \in S_n} \sum_{w \in \mathcal{X}} p(w) \cdot (f(w_{[w_i = x_i, i \in Pre^i(\pi) \cup \{i\}]}) - f(w_{[w_i = x_i, i \in Pre^i(\pi)]})), \tag{5}$$

where S_n denotes a group of permutations of n elements, $\mathcal{X}$ denotes the training set instances, $Pre^i(\pi)$ represents the set of indices that precedes i in the permutation π, i.e. $Pre^i(\pi) = \{j; \pi(j) < \pi(i)\})$. Let $p(w)$ denote the probability of the instance w in $\mathcal{X}$ and let $w_{[formula]}$ denote the instance w with some of its features values changed according to the *formula*. To calculate $\phi_i(x)$, we have to go through $|\mathcal{X}| \cdot n!$ iterations, which can be slow. Therefore, the method IME uses the following sampling. The sampling population of i-th feature is $V_{\pi,w} = (f(w_{[w_i=x_i, i\in Pre^i(\pi)\cup\{i\}]}) - f(w_{[w_i=x_i, i\in Pre^i(\pi)]}))$ for every combination of the permutation π and instance w. IME draws m_i samples $V_1, ..., V_{m_i}$ at random with repetition. The estimate of $\phi_i(x)$ is defined with the equation:

$$\hat{\phi_i} = \frac{1}{m_i} \sum_{i=1}^{m_i} V_i. \tag{6}$$

Contrary to SHAP, IME does not use approximation with linear models, which compute all features' contributions at once but has to compute the Shapley values by averaging over a large enough sample for each feature separately. This makes the method slower but also potentially more robust as the method does not assume the shape of the dependency in the space of normalized features.

B Demonstration of better sampling in SHAP

The graphs in Figure 3 show the PCA based 2D space of the evaluation part of the COMPAS dataset (see Section 4 for the dataset description). The left-hand side shows the SHAP-generated sampled instances using the k-means algorithm (14 clusters determined by the silhouette score (Rousseeuw (1987)). The sample produced with the MCD-VAE generator in gSHAP is shown on the right-hand side. This sample is much more similar to the original distribution compared to the SHAP sampling.

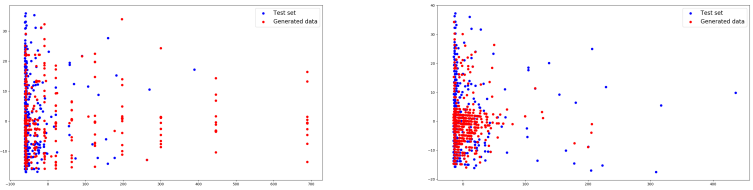


Figure 3: Visual comparison of original and sampled distributions for the COMPASS dataset. The SHAP k-means based generator (left) produces instances less similar to the original data, compared to the MCD-VAE generator (right).

C Improved IME convergence rate with the TreeEnsemble generator

Preliminary, we tested how better generators affect the convergence rate of the IME explanation method. The preliminary results in Table 2 show a significant reduction in the number of needed samples and a slight increase in the error compared to the original perturbation sampling. Note that the error measure deployed is biased in favor of the perturbation sampling, which was used to determine the gold standard. This was determined with the sampling population's variance, as described in Štrumbelj & Kononenko (2010).

D Training discriminator function of the attacker

Details of training attacker's decision models d (see Figure 1b) is described in Algorithms 1, 2, and 3. We used a slightly different algorithm for each of the three explanation methods, LIME, SHAP,

COMPAS dataset

	Error		# samples			
Classifier	Perturb.	TEnsFillIn	Perturb.	TEnsFillIn	Reduction %	CA %
Naive Bayes	0.0076	0.0217	18571	11278	39	83
Linear SVM	0.0033	0.0080	8244	4423	46	84
Random forest	0.0049	0.0221	45372	26960	41	80
Neural network	0.0057	0.0130	16157	8841	45	84

German dataset

	Error		# samples			
Classifier	Perturb.	TEnsFillIn	Perturb.	TEnsFillIn	Reduction %	CA %
Naive Bayes	0.0076	0.0201	56052	39257	30	77
Linear SVM	0.0005	0.0013	3877	2157	44	69
Random forest	0.0046	0.0141	92478	66639	28	74
Neural network	0	0	0	26	/	69

CC dataset

	Error		# samples			
Classifier	Perturb.	TEnsFillIn	Perturb.	TEnsFillIn	Reduction %	CA %
Naive Bayes	0.0028	0.0046	32910	20117	39	70
Linear SVM	0.0009	0.0048	73324	39098	47	62
Random forest	0.0032	0.0045	109958	58852	46	79
Neural network	0.0012	0.0061	144183	70020	51	72

Table 2: Comparison of original perturbation sampling and the TreeEnsemble generator with data fill-in inside the IME method. The results show average scores for the evaluation set. The column Perturb. presents the perturbation based sampling and TEnsFillIn presents the sampling using TreeEnsemble generator with missing parts of instances filled in. The Reduction column shows the reduction in the number of samples using the TEnsFillIn method compared to perturbations. The CA stands for the classification accuracy of the explained classifier on the evaluation set.

and IME, as each method uses a different sampling. Algorithms first create out-of-distribution instances by method-specific sampling. The training sets for decision models are created by labeling the created instances with 0; the instances from sample S (to which the attacker has access) from distribution X_{dist} are labeled with 1. Finally, the machine learning model *dModel* is trained on this training set and returned as d. In our experiments, we used random forest classifier as *dModel* and the training part of each evaluation dataset as S.

E HEATMAPS AS TABLES

We present the information contained in Figure 2 in a more detailed tabular form in Table 3.

F COMPARING EXPLANATIONS OF ORIGINAL AND MODIFIED METHODS

We check if improved data generators affect explanations in non-adversary environment. We split the dataset into training and evaluation set in the ratio $90\% : 10\%$, and trained four classifiers from Python scikit-learn (Pedregosa et al. (2011)) library: Gaussian naive Bayes, linear SVC (SVM for classification), random forest, and neural network. We explained the predictions of each classifier on the evaluation set with every combination of explanation methods and generators used in the adversarial attack experiments. For instances in the evaluation set, we measured the mean absolute

Algorithm 1: Training of the decision model d, used by the attacker to distinguish between instances from distribution X_{dist} and samples produced by explanation methods LIME or gLIME.

Input: $S = \{(x_i)_{i=1}^m\}$: training set, *nSamples*: number of generated instances for each instance $x_i \in D$, gen: data generator, *dModel*: machine learning algorithm
Output: Classifier d that outputs 1 if its input x is from X_{dist} and 0 otherwise
$X \leftarrow \emptyset$ // Training set for *dModel*
$gen.fit(S)$ // Train the data generator on S
for $i = 1$ *to* m **do**
  $X \leftarrow X \cup (x_i, 1)$ // Add an instance from distribution
  $G \leftarrow gen.newdata(nSamples, x_i)$ // Generate *nSamples* new samples around x_i
  for $j = 1$ *to nSamples* **do** // Add *nSamples* out of distribution instances
    $X \leftarrow X \cup (G[j], 0)$ // Add j-th instance from set G to X
  end
end
$d \leftarrow dModel.fit(X)$ // Fit model *dModel* to set X and save it in d
return d

Algorithm 2: Training of the decision model d, used by attacker to distinguish between instances from distribution X_{dist} and samples produced by explanation methods SHAP or gSHAP.

Input: $S = \{(x_i)_{i=1}^m\}$: training set, *nSamples*: number of generated instances for each instance $x_i \in D$, k: size of the generated distribution set, gen: data generator, *dModel*: machine learning algorithm
Output: Classifier d that outputs 1 if its input x is from X_{dist} and 0 otherwise
$X \leftarrow \emptyset$ // Training set for *dModel*
$gen.fit(S)$ // Train the data generator on S
if $gen == KMeans$ *or* $gen == rbfDataGen$ *or* $gen == treeEnsemble$ **then**
  $D \leftarrow gen.newdata(k)$ // Generate the distribution set with KMeans, rbfDataGen or treeEnsemble
end
for $i = 1$ *to nSamples* **do** // Add *nSamples* out of distribution instances
  $x \leftarrow$ random instance from S
  if $gen == MCD - VAE$ *or* $gen == treeEnsembleFill$ **then**
    $w \leftarrow gen.newdata(1, x)$ // Generate an instance w in the vicinity of x
  end
  else // KMeans, treeEnsemble or rbfDataGen
    $w \leftarrow$ take a random instance from D
  end
  $M \leftarrow$ choose a random subset of $\{1, 2, ..., len(x)\}$ // Choose random features
  $x[M] \leftarrow w[M]$ // Replace the values of chosen features in x with values from w as in SHAP method
  $X \leftarrow X \cup (x, 0)$ // Add out of distribution instance
end
for $i = 1$ *to* m **do** // Add instances from distribution
  $X \leftarrow X \cup (x_i, 1)$
end
$d \leftarrow dModel.fit(X)$ // Fit model *dModel* to set X and save it in d
return d

difference (MAD) of modified explanation methods, defined with the following equation:

$$\text{MAD}_{gen}(x) = \frac{1}{n} \sum_{i=1}^{n} |\phi_i^{gen}(x) - \phi_i(x)|, \tag{7}$$

where $\phi_i^{gen}(x)$ and $\phi_i(x)$ represent the explanations of i-th feature returned by the modified and original explanation method, respectively (recall that n denotes the number of features in the data set).

We experimented on three datasets. The COMPAS dataset is described in Section 4.1. In addition to that, we used synthetic dataset condInd from Robnik-Šikonja & Kononenko (2008), and Ionosphere

Algorithm 3: Training of the decision model d, used by attacker to distinguish between instances from distribution X_{dist} and samples produced by explanation methods IME or gIME.

Input: $S = \{(x_i)_{i=1}^m\}$: training set, *nSamples*: number of generated instances for each instance $x_i \in D$, gen: data generator, *dModel*: machine learning algorithm
Output: Classifier d that outputs 1 if its input x is from X_{dist} and 0 otherwise
$X \leftarrow \emptyset$ // Training set for *dModel*
$gen.fit(S)$ // Train the data generator on set S
for $i = 1$ *to* m **do**
  $X \leftarrow X \cup (x_i, 1)$ // Add an instance from distribution
  for $j = 1$ *to nSamples* **do** // Add *nSamples* out of distribution instances
    $w \leftarrow gen.newdata(1, x_i)$ // Generate an instance w in the vicinity of x_i
    $b_1 \leftarrow w$ // First out of distribution instance
    $b_2 \leftarrow w$ // Second out of distribution instance
    $\pi \leftarrow$ choose a random permutation from $S_{len(x_i)}$ // i.e. a random permutation of x_i's features
    $idx \leftarrow$ choose a random number from $\{1, 2, ..., len(x_i)\}$
    $M_1 \leftarrow \{k \in \{1, 2, ..., len(x_i)\}, \pi(k) < \pi(idx)\}$ // Features that precede idx in permutation π
    $M_2 \leftarrow M_1 \cup \{idx\}$
    $b_1 \leftarrow x_i[M_1]$ // Vector b_1 as in IME method (Štrumbelj & Kononenko (2013))
    $b_2 \leftarrow x_i[M_2]$ // Vector b_2 as in IME method (Štrumbelj & Kononenko (2013))
    $X \leftarrow X \cup \{(b_1, 0), (b_2, 0)\}$ // Add out of distribution instances
  end
end
$d \leftarrow dModel.fit(X)$ // Fit model *dModel* to set X and save it in d
return d

dataset from UCI repository (Dua & Graff (2019)). Both datasets represent a binary classification problem. Apart from the target variable, condInd consists of 8 binary features, while Ionosphere consists of 34 numerical attributes. The condInd datasets contains 2000 instances and Ionosphere contains 351 instances.

The results are shown in Table 4. The differences between original LIME and gLIME explanations are considerable (see the top table). This is not surprising since LIME fits local linear models in its local explanations, which can strongly differ even for small perturbations of the model's input. SHAP and IME explanations are very similar (the average MAD is almost negligible). We can conclude that explanations of gSHAP and gIME are not significantly different from SHAP and IME in the non-adversary environment.

	Perturbation	MCD-VAE	RbfDataGen	TreeEns
Compas_Perturbation	0.0	0.676	1.0	1.0
Compas_MCD-VAE	0.997	0.492	1.0	1.0
Compas_RbfDataGen	0.126	0.006	0.0	1.0
Compas_TreeEns	0.049	0.144	1.0	1.0
Compas2_Perturbation	0.1	0.005	1.0	1.0
Compas2_MCD-VAE	0.997	0.322	1.0	1.0
Compas2_RbfDataGen	0.997	0.01	0.252	1.0
Compas2_TreeEns	0.997	0.036	0.981	1.0
German_Perturbation	0.0	0.0	1.0	1.0
German_MCD-VAE	1.0	0.0	1.0	1.0
German_RbfDataGen	0.33	0.0	0.0	1.0
German_TreeEns	0.73	0.0	0.0	0.92
CC_Perturbation	0.0	0.05	0.0	0.0
CC_MCD-VAE	0.99	0.26	0.0	0.0
CC_RbfDataGen	0.0	0.39	0.0	0.0
CC_TreeEns	0.0	0.4	0.0	0.0
CC2_Perturbation	0.0	0.065	0.0	0.0
CC2_MCD-VAE	0.985	0.0	0.0	0.0
CC2_RbfDataGen	0.0	0.075	0.0	0.0
CC2_TreeEns	0.0	0.07	0.0	0.0

	Perturbation	MCD-VAE	RbfDataGen	TreeEns	TEnsFillIn
Compas_Perturbation	0.426	0.447	0.841	0.929	0.798
Compas_MCD-VAE	0.409	0.207	0.667	0.837	0.782
Compas_RbfDataGen	0.57	0.094	0.104	0.626	0.51
Compas_TreeEns	0.252	0.209	0.019	0.202	0.227
Compas2_Perturbation	0.327	0.417	0.908	0.989	0.799
Compas2_MCD-VAE	0.629	0.528	0.901	0.96	0.794
Compas2_RbfDataGen	0.519	0.154	0.286	0.754	0.513
Compas2_TreeEns	0.252	0.259	0.162	0.497	0.243
German_Perturbation	0.099	0.188	0.17	0.28	0.297
German_MCD-VAE	0.27	0.25	0.81	0.75	0.61
German_RbfDataGen	0.04	0.078	0.0	0.0	0.0
German_TreeEns	0.0	0.049	0.0	0.0	0.0
CC_Perturbation	0.18	0.115	0.0	0.235	0.645
CC_MCD-VAE	0.18	0.155	0.0	0.045	0.345
CC_RbfDataGen	0.945	0.165	0.0	0.935	0.945
CC_TreeEns	0.335	0.27	0.0	0.0	0.025
CC2_Perturbation	0.21	0.065	0.0	0.66	0.665
CC2_MCD-VAE	0.115	0.04	0.0	0.18	0.37
CC2_RbfDataGen	0.955	0.78	0.0	0.945	0.96
CC2_TreeEns	0.36	0.325	0.0	0.04	0.02

	Perturbation	MCD-VAE	TEnsFillIn
Compas_Perturbation	0.77	0.511	0.772
Compas_MCD-VAE	0.948	0.463	0.877
Compas_TEnsFillIn	0.738	0.476	0.754
Compas2_Perturbation	0.94	0.487	0.77
Compas2_MCD-VAE	0.987	0.443	0.875
Compas2_TEnsFillIn	0.93	0.497	0.761
German_Perturbation	0.67	0.25	0.6
German_MCD-VAE	0.95	0.28	0.88
German_TEnsFillIn	0.78	0.257	0.644
CC_Perturbation	0.075	0.28	0.0
CC_MCD-VAE	0.96	0.17	0.86
CC_TEnsFillIn	0.11	0.34	0.0
CC2_Perturbation	0.565	0.15	0.115
CC2_MCD-VAE	0.945	0.05	0.845
CC2_TEnsFillIn	0.555	0.35	0.085

Table 3: The robustness results for gLIME (top table), gSHAP (middle table), and gIME (bottom table). The tables show the proportion of evaluation set instances, where the sensitive feature was recognized as the most important by the used explanation method. Columns represent the generators used for explanations. The row labels consist of the name of the dataset on which the experiment was performed and the name of the generator used for training of the adversarial model. Compas2 and CC2 denote attacks with two independent features. Perturbation represents the original sampling used in LIME, SHAP, and IME, TEnsFillIn represents a variant of the TreeEnsemble generator where new instances are generated around the given one, and TreeEns represents the generation from the whole distribution.

	COMPAS		condInd		Ionosphere	
	Average MAE	Variance of MAE	Average MAE	Variance of MAE	Average MAE	Variance of MAE
Bayes_MCD-VAE	0.1196	0.0029	0.038	0.0002	0.011	0.0
Bayes_RBF	0.1361	0.0006	0.0094	0.0	0.019	0.0002
Bayes_TEns	0.1144	0.0011	0.011	0.0	0.0223	0.0002
SVM_MCD-VAE	0.0206	0.0	0.0557	0.0005	0.0591	0.0002
SVM_RBF	0.1206	0.0011	0.0126	0.0	0.0309	0.0001
SVM_TEns	0.0774	0.0003	0.0147	0.0	0.0307	0.0001
Forest_MCD-VAE	0.1991	0.0079	0.0419	0.0002	0.0933	0.0009
Forest_RBF	0.1326	0.0023	0.022	0.0	0.0223	0.0001
Forest_TEns	0.1158	0.0016	0.0247	0.0	0.0258	0.0001
NN_MCD-VAE	0.0416	0.0001	0.1252	0.0011	0.1155	0.0007
NN_RBF	0.1623	0.0019	0.0185	0.0	0.0369	0.0001
NN_TEns	0.097	0.0006	0.0209	0.0	0.037	0.0001

	COMPAS		condInd		Ionosphere	
	Average MAE	Variance of MAE	Average MAE	Variance of MAE	Average MAE	Variance of MAE
Bayes_MCD-VAE	0.0307	0.0007	0.0735	0.0004	0.0099	0.0
Bayes_RBF	0.0533	0.0004	0.0019	0.0	0.0106	0.0
Bayes_TEns	0.023	0.0003	0.0069	0.0	0.0079	0.0
Bayes_TEnsFill	0.031	0.001	0.0483	0.0001	0.0101	0.0001
SVM_MCD-VAE	0.009	0.0001	0.0889	0.0007	0.0219	0.0002
SVM_RBF	0.0101	0.0002	0.0019	0.0	0.0125	0.0
SVM_TEns	0.0129	0.0001	0.0069	0.0	0.012	0.0
SVM_TEnsFill	0.011	0.0002	0.0483	0.0001	0.0262	0.0001
Forest_MCD-VAE	0.0368	0.0004	0.0895	0.0009	0.0167	0.0
Forest_RBF	0.0343	0.0003	0.0127	0.0	0.0132	0.0
Forest_TEns	0.0291	0.0002	0.0183	0.0001	0.0125	0.0
Forest_TEnsFill	0.0361	0.0006	0.0353	0.0001	0.0181	0.0
NN_MCD-VAE	0.017	0.0003	0.0577	0.0004	0.0193	0.0001
NN_RBF	0.0262	0.0003	0.0091	0.0	0.0111	0.0
NN_TEns	0.0164	0.0001	0.0147	0.0	0.0126	0.0
NN_TEnsFill	0.0183	0.0004	0.0364	0.0001	0.0215	0.0

	COMPAS		condInd		Ionosphere	
	Average MAE	Variance of MAE	Average MAE	Variance of MAE	Average MAE	Variance of MAE
Bayes_MCD-VAE	0.0329	0.0005	0.0052	0.0005	0.0137	0.0
Bayes_TEnsFill	0.0209	0.0003	0.0019	0.0	0.0093	0.0
SVM_MCD-VAE	0.0112	0.0001	0.0223	0.002	0.0276	0.0002
SVM_TEnsFill	0.0069	0.0001	0.0019	0.0	0.0235	0.0001
Forest_MCD-VAE	0.0333	0.0004	0.0621	0.0013	0.0151	0.0
Forest_TEnsFill	0.0232	0.0002	0.0349	0.0001	0.0098	0.0
NN_MCD-VAE	0.0202	0.0002	0.0518	0.0008	0.027	0.0003
NN_TEnsFill	0.0128	0.0001	0.0277	0.0001	0.0184	0.0001

Table 4: The differences in explanations between LIME and gLIME (top table), SHAP and gSHAP (middle table), and IME and gIME (bottom table). The tables show MAD and the variance of MAD on evaluation set for every combination of the explanation method and predictive model. Columns represent MAD and the variance of MAD on used datasets. The row labels consist of the used classifier and generator.

G ADVERSARIAL CLASSIFIERS WITH DIFFERENT PREDICTION THRESHOLDS

In a realistic scenario, the attacker might be wary of being discovered by the regulator and decide to employ deception only when it is really certain that the predicted instance is used inside the explanation method. In these circumstances, the number of times when the sensitive variable is used inside the model will also depend on how aggressively the biased classifier is used. This reflection leads to another metric for the defense's success, which measures the proportion of instances when the sensitive variable is identified when it is being used, i.e. at different levels of conservatism.

We express different levels of attacker's conservatism by changing the threshold of the decision function d. The output of the adversarial model e_t, which uses threshold t is now defined with the equation:

$$e_t(x) = \begin{cases} b(x), & d_{prob}(x) \geq t \\ \psi(x), & d_{prob}(x) < t \end{cases}, \tag{8}$$

where $d_{prob}(x)$ denotes the probability that x is from distribution X_{dist}, returned by decision model d. Threshold t therefore represents the required certainty of the attacker that given instance is from X_{dist} before it deploys the biased classifier. Larger values of t will result in biased classifier being deployed fewer times but with larger certainty that it has to be used. If we use $t = 0.5$ in Equation (8), we get the adversarial model that behaves identically to the one, described with Equation (1) in Section 2.2.

In this experiment, we used only the COMPAS dataset, which we split into the training and evaluation set in ratio $90\% : 10\%$. We performed the same experiment as in section 4.2, but with different thresholds for the adversarial models. We used five different thresholds: $0.3, 0.4, 0.5, 0.6$ and 0.7.

We first checked how different thresholds affect the deployment of the biased model. For each generator used in the adversarial model's training and the certainty threshold used by the adversarial model, we measured the proportion of instances from the evaluation set, where the biased classifier was deployed. The results are shown in Table 5. As expected, the percentage of instances where the biased classifier is deployed decreases with increasing t. The biased classifier is deployed less frequently when using treeEnsemble and rbfDataGen in its training, which is also expected, as these two generators give the best results inside the explanation methods. This means that for these generators, it is more difficult for an attacker to distinguish between the actual instances and generated instances. The biased classifier is deployed the most often in the case of the LIME adversarial model and least often in the SHAP adversarial model.

Next, we tested the proportion of instances where the sensitive feature (in case of COMPAS dataset this is *race*) was recognized as the most important feature by the explanation method on two different subsets of instances: instances from the evaluation set on which the biased classifier was deployed and the whole evaluation set. We measured these numbers for all combinations of thresholds and explanation methods. The results are shown in Tables 6 (gLIME), 7 (gSHAP) and 8 (gIME) for gLIME, gSHAP, and gIME, respectively. We can observe the same pattern as in section 4.2: gIME is still the most robust, and the treeEnsemble generator still gives the best results. The sensitive attribute is recognized as the most important feature more often when the decision threshold values are lower, which is expected as in these cases, the adversarial model behaves more aggressively. Whilst the percentage of the instances on which the racist behavior of the adversarial model is recognized drops with higher thresholds, it remains high enough to warn the regulator about the problematic behavior of the prediction model (especially in the case of gSHAP and gIME using treeEnsemble as data generator). From that, we can conclude that modified explanation methods remain robust enough, even with more conservative adversaries.

		0.3	0.4	0.5	0.6	0.7
One unrelated	Perturb.	100.0	99.35	98.22	97.41	95.31
	MCD-VAE	98.54	98.38	98.38	98.22	97.73
	RBF	86.89	83.01	79.45	76.38	72.01
	TEns	85.28	74.92	53.88	27.67	14.4
Two unrelated	Perturb.	99.84	98.87	98.22	96.76	95.31
	MCD-VAE	99.19	99.19	99.03	99.03	98.71
	RBF	86.41	83.66	79.94	77.02	73.62
	TEns	88.03	77.99	52.27	28.32	14.72

		0.3	0.4	0.5	0.6	0.7
One unrelated	Perturb.	88.35	84.79	79.77	76.05	70.55
	MCD-VAE	87.38	83.01	79.94	75.4	71.36
	RBF	74.27	64.56	58.25	49.19	43.37
	TEns	62.3	51.29	43.04	35.76	30.58
Two unrelated	Perturb.	89.16	85.28	82.36	76.7	70.55
	MCD-VAE	87.86	83.01	79.13	74.92	69.9
	RBF	76.05	66.99	59.55	53.4	47.09
	TEns	61.97	50.16	41.75	36.08	31.39

		0.3	0.4	0.5	0.6	0.7
One unrelated	Perturb.	88.67	81.55	72.33	63.27	52.27
	MCD-VAE	92.72	89.16	85.44	79.13	73.3
	TEnsFill	93.2	87.06	81.72	72.49	60.84
Two unrelated	Perturb.	91.59	82.69	74.92	64.72	55.34
	MCD-VAE	94.01	89.81	85.28	80.58	74.92
	TEnsFill	93.04	86.25	78.48	69.74	60.84

Table 5: Proportions of instances in % of the evaluation set on which the biased classifier was deployed for adversarial LIME (top table), SHAP (middle table) and IME model (bottom table). Columns represent different threshold used for deploying the biased classifier. The rows represent the generator used in training of the adversarial attack. The labels ”One unrelated” and ”Two unrelated” represent the attacks with one or two unrelated features.

	0.3		0.4		0.5		0.6		0.7	
	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All
Perturb._Perturb.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Perturb._MCD-VAE	99.67	99.68	99.67	99.68	99.67	99.68	99.67	99.68	99.67	99.68
Perturb._RBF	99.81	97.9	99.22	94.01	25.25	20.71	0.0	0.0	0.0	0.0
Perturb._TEns	99.62	99.68	99.57	99.68	1.2	1.46	0.0	0.0	0.0	0.0
MCD-VAE_Perturb.	99.68	99.68	95.93	95.95	16.8	16.5	0.66	0.65	21.9	21.84
MCD-VAE_MCD-VAE	9.52	9.39	9.7	9.55	8.72	8.58	24.05	23.62	52.98	52.43
MCD-VAE_RBF	45.25	44.5	45.81	44.66	0.0	0.0	0.21	0.16	0.9	0.65
MCD-VAE_TEns	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RBF_Perturb.	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
RBF_MCD-VAE	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
RBF_RBF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
RBF_TEns	100.0	100.0	100.0	100.0	100.0	94.01	0.0	0.0	0.0	0.0
TEns_Perturb.	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TEns_MCD-VAE	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TEns_RBF	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TEns_TEns	100.0	100.0	100.0	100.0	84.98	74.6	0.0	0.0	0.0	0.0

	0.3		0.4		0.5		0.6		0.7	
	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All
Perturb._Perturb.	98.87	98.87	56.46	56.8	11.04	11.33	2.17	2.1	0.85	1.13
Perturb._MCD-VAE	99.67	99.68	99.67	99.68	99.67	99.68	99.67	99.68	99.67	99.68
Perturb._RBF	100.0	99.68	100.0	99.68	100.0	99.68	100.0	99.68	100.0	99.68
Perturb._TEns	99.63	99.68	99.79	99.68	99.69	99.68	100.0	99.68	100.0	99.68
MCD-VAE_Perturb.	86.39	86.41	85.6	85.76	76.11	76.21	69.9	70.23	68.08	67.96
MCD-VAE_MCD-VAE	32.14	31.88	28.38	28.16	26.14	25.89	23.53	23.3	21.31	21.2
MCD-VAE_RBF	38.39	40.78	37.14	38.51	30.57	29.77	20.8	19.26	15.38	15.21
MCD-VAE_TEns	75.74	74.92	35.89	36.08	13.0	8.74	16.0	7.93	10.99	8.9
RBF_Perturb.	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
RBF_MCD-VAE	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
RBF_RBF	39.14	41.75	36.75	35.44	3.24	2.75	3.36	2.75	0.0	0.0
RBF_TEns	100.0	100.0	100.0	100.0	100.0	100.0	99.43	95.15	58.24	50.65
TEns_Perturb.	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TEns_MCD-VAE	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TEns_RBF	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
TEns_TEns	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	4.4	3.88

Table 6: Percentages of instances where the sensitive feature (*race*) was recognized as the most important feature with gLIME for adversarial attacks with one (top table) or two unrelated features (bottom table). The columns labeled *Biased pred.* represent the results on instances on which the biased classifier was deployed, while the columns labeled *All* represent the results on the whole evaluation sets. The numbers above represent the used threshold. The row labels are in the form $<explainer>\_<adversarial>$ where $<explainer>$ denotes the generator used in the explanation method and $<adversarial>$ denotes the generator used in the training of the adversarial model.

	0.3		0.4		0.5		0.6		0.7	
	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All
Perturb._Perturb.	41.03	39.0	31.11	27.99	24.95	22.01	21.91	19.09	16.74	13.11
Perturb._MCD-VAE	99.63	94.34	99.03	90.61	96.96	86.08	94.21	80.1	88.66	70.55
Perturb._RBF	99.35	90.78	97.99	82.52	81.39	59.71	61.18	39.48	36.94	20.71
Perturb._TEns	93.77	80.58	62.15	50.32	32.33	22.33	19.0	11.0	13.76	5.99
MCD-VAE_Perturb.	72.16	65.21	57.44	51.78	32.66	26.38	61.7	50.32	51.38	39.0
MCD-VAE_MCD-VAE	32.78	30.74	47.95	41.91	40.89	34.79	29.83	27.02	24.26	18.93
MCD-VAE_RBF	45.97	42.72	33.83	30.58	24.72	20.23	16.45	11.97	11.94	7.44
MCD-VAE_TEns	47.01	41.26	31.55	27.02	15.79	12.78	9.05	7.93	10.58	8.25
RBF_Perturb.	99.08	95.95	97.52	92.07	95.94	86.25	88.72	73.79	71.56	53.56
RBF_MCD-VAE	99.63	95.15	98.44	88.67	93.72	79.61	89.7	71.04	74.15	55.02
RBF_RBF	46.41	37.22	41.1	28.16	18.61	12.46	9.54	6.15	3.73	1.94
RBF_TEns	66.49	48.71	30.91	20.23	10.9	6.63	2.26	1.78	0.0	0.0
TEns_Perturb.	100.0	99.03	100.0	96.28	99.39	92.07	99.57	87.38	98.62	79.13
TEns_MCD-VAE	100.0	96.93	100.0	95.15	100.0	91.75	99.57	87.7	98.19	78.96
TEns_RBF	98.47	86.57	96.74	76.38	88.06	60.19	81.25	46.76	64.93	32.2
TEns_TEns	93.51	70.55	71.29	42.56	41.73	20.87	18.1	8.09	11.11	4.21
TEnsFill_Perturb.	98.35	92.39	96.37	87.86	95.94	82.69	95.74	79.77	94.5	72.49
TEnsFill_MCD-VAE	96.3	88.67	92.79	82.2	89.88	77.02	86.7	70.71	82.99	62.94
TEnsFill_RBF	86.06	70.87	74.94	55.83	65.28	44.34	57.24	34.79	45.15	24.92
TEnsFill_TEns	65.19	49.19	43.53	28.96	29.32	17.15	23.08	11.49	17.46	8.25

	0.3		0.4		0.5		0.6		0.7	
	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All
Perturb._Perturb.	66.06	60.03	57.87	50.0	41.45	34.79	24.68	19.26	13.99	10.19
Perturb._MCD-VAE	99.82	95.63	99.81	94.17	99.18	90.61	98.49	85.76	95.83	78.64
Perturb._RBF	98.94	92.72	97.1	83.01	88.04	66.67	64.55	41.59	48.8	27.51
Perturb._TEns	97.39	84.79	70.0	48.38	36.43	20.06	16.14	8.25	9.28	4.21
MCD-VAE_Perturb.	63.16	59.22	50.28	44.01	42.04	35.44	65.4	53.56	43.81	34.3
MCD-VAE_MCD-VAE	53.41	48.22	35.28	30.58	37.63	30.26	43.63	35.92	25.69	20.23
MCD-VAE_RBF	37.87	33.66	24.15	20.71	14.95	10.68	34.85	25.24	12.71	7.61
MCD-VAE_TEns	37.34	29.29	50.97	38.19	34.11	23.62	22.87	15.86	15.46	8.9
RBF_Perturb.	99.64	98.06	98.48	94.82	97.45	90.78	96.84	86.25	92.89	77.51
RBF_MCD-VAE	99.45	96.93	96.1	91.59	92.64	85.44	87.47	76.54	78.47	63.43
RBF_RBF	70.21	59.39	60.14	45.95	53.26	35.6	44.55	26.54	37.11	19.74
RBF_TEns	88.25	76.21	70.0	50.97	41.09	22.17	29.6	12.62	18.04	7.12
TEns_Perturb.	100.0	99.68	100.0	99.03	100.0	97.9	99.58	92.39	99.54	87.38
TEns_MCD-VAE	100.0	99.35	99.81	98.06	99.8	95.79	99.57	92.39	99.54	85.76
TEns_RBF	100.0	94.5	97.34	83.98	94.29	70.39	91.82	59.55	88.66	50.65
TEns_TEns	98.17	88.03	89.35	62.46	77.13	43.85	66.37	31.23	55.67	22.49
TEnsFill_Perturb.	99.09	94.01	98.1	89.97	97.05	85.92	96.84	79.29	95.41	72.49
TEnsFill_MCD-VAE	96.32	88.83	95.71	84.14	92.84	78.8	89.85	72.82	85.42	63.11
TEnsFill_RBF	85.53	70.87	73.91	55.5	66.3	44.66	59.09	36.25	51.89	28.16
TEnsFill_TEns	72.06	52.91	50.65	32.04	37.6	20.39	29.15	14.08	24.23	10.52

Table 7: Percentage of instances where the sensitive attribute (*race*) was recognized as the most important feature with gSHAP for adversarial attacks with one (top table) or two unrelated features (bottom table). The columns labeled *Biased pred.* represent the results on instances on which the biased classifier was deployed, while columns labeled *All* represent the results on the whole evaluation sets. The numbers above represent the used threshold. The row labels are in the form <*explainer*>_<*adversarial*> where <*explainer*> denotes the generator used in the explanation method and <*adversarial*> denotes the generator used in the training of the adversarial model.

	0.3		0.4		0.5		0.6		0.7	
	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All
Perturb._Perturb.	99.82	98.22	97.22	92.23	88.37	76.7	68.8	51.78	41.18	24.6
Perturb._MCD-VAE	100.0	98.38	100.0	96.93	100.0	95.15	99.59	89.16	98.45	80.1
Perturb._TEnsFill	97.4	95.15	93.31	87.54	88.71	79.61	79.46	65.05	69.68	48.22
MCD-VAE_Perturb.	50.0	49.68	57.34	55.83	59.96	57.61	50.13	47.73	33.13	25.24
MCD-VAE_MCD-VAE	59.16	57.77	37.93	34.47	47.35	41.75	35.17	30.42	21.19	16.34
MCD-VAE_TEnsFill	74.31	72.49	67.1	63.92	63.96	59.55	41.52	37.22	33.51	29.94
TEnsFill_Perturb.	99.45	95.31	97.62	89.16	91.72	76.86	78.52	58.58	54.49	33.17
TEnsFill_MCD-VAE	99.3	94.98	99.09	92.88	98.48	88.51	95.09	79.45	94.48	73.79
TEnsFill_TEnsFill	98.96	95.31	97.03	90.45	93.07	82.52	83.48	67.64	76.06	53.4

	0.3		0.4		0.5		0.6		0.7	
	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All	Biased pred.	All
Perturb._Perturb.	97.0	95.47	98.24	95.15	99.14	94.82	99.25	94.82	99.42	94.17
Perturb._MCD-VAE	98.8	97.73	99.1	97.57	99.81	97.57	99.8	97.57	99.78	97.73
Perturb._TEnsFill	93.39	91.75	94.56	91.26	96.7	91.59	97.91	91.91	98.14	92.07
MCD-VAE_Perturb.	56.18	54.69	56.16	52.75	53.35	48.38	51.25	49.51	48.83	48.38
MCD-VAE_MCD-VAE	31.84	29.94	52.07	48.06	69.64	62.94	54.82	49.19	41.47	38.03
MCD-VAE_TEnsFill	63.65	62.46	66.98	62.94	68.66	65.86	64.5	59.71	58.51	57.28
TEnsFill_Perturb.	79.68	75.89	85.71	77.18	90.28	76.7	90.25	76.38	92.69	77.83
TEnsFill_MCD-VAE	92.6	88.67	95.32	88.51	97.91	88.83	98.39	88.03	99.14	88.35
TEnsFill_TEnsFill	83.65	80.91	87.43	80.58	91.75	80.26	92.81	80.58	94.68	80.42

Table 8: Percentages of instances where the sensitive attribute (*race*) was recognized as the most important feature with gIME for adversarial attacks with one (top table) or two unrelated features (bottom table). The columns labeled *Biased pred.* represent the results on instances on which the biased classifier was deployed, while the columns labeled *All* represent the results on the whole evaluation sets. The numbers above represent the used threshold. The row labels are in the form <*explainer*>_<*adversarial*> where <*explainer*> denotes the generator used in the explanation method and <*adversarial*> denotes the generator used in the training of the adversarial model.

Appendix C: Semantic Reasoning from Model-Agnostic Explanations

Semantic Reasoning from Model-Agnostic Explanations

Timen Stepišnik Perdih
Jožef Stefan Institute
Ljubljana, Slovenia
tstepisnikp@gmail.com

Nada Lavrač
Jožef Stefan Institute
Ljubljana, Slovenia
University of Nova Gorica
Nova Gorica, Slovenia

Blaž Škrlj
Jožef Stefan International Postgraduate School
Jožef Stefan Institute,
Ljubljana, Slovenia
blaz.skrlj@ijs.si

***Abstract*—With the wide adoption of black-box models, instance-based *post hoc* explanation tools, such as LIME and SHAP became increasingly popular. These tools produce explanations, pinpointing contributions of key features associated with a given prediction. However, the obtained explanations remain at the raw feature level and are not necessarily understandable by a human expert without extensive domain knowledge. We propose ReEx (Reasoning with Explanations), a method applicable to explanations generated by arbitrary instance-level explainers, such as SHAP. By using background knowledge in the form of ontologies, ReEx generalizes instance explanations in a least general generalization-like manner. The resulting symbolic descriptions are specific for individual classes and offer generalizations based on the explainer's output. The derived semantic explanations are potentially more informative, as they describe the key attributes in the context of more general background knowledge, e.g., at the biological process level. We showcase ReEx's performance on nine biological data sets, showing that compact, semantic explanations can be obtained and are more informative than generic ontology mappings that link terms directly to feature names. ReEx is offered as a simple-to-use Python library and is compatible with tools such as SHAP and similar. To our knowledge, this is one of the first methods that directly couples semantic reasoning with contemporary model explanation methods. This paper is a preprint. Full version's doi is: 10.1109/SAMI50585.2021.9378668**

***Index Terms*—model explanations, reasoning, generalization, SHAP, machine learning, explainable AI**

I. Introduction

THERE is a growing demand for machine learning approaches that are not only well-performing but also *trustworthy*, transparent and *explainable* [1]. Methods like LIME [2] and SHAP [3] have been proposed to extract the contributions of the most important features to explain a given model's prediction, but these features are often still not fully understandable by the user (e.g., gene symbols). This paper presents a way of reasoning from model explanations called **ReEx** (Reasoning from Explanations), a method that generalizes these features in the context of a given collection of *background knowledge* into explanations comprised of more understandable (semantic) terms. This paper discusses the following advances over the state of the art:

1) We propose ReEx, a method capable of linking background knowledge in the form of ontologies to explanations generated by widely accepted approaches such as SHAP. ReEx performs efficient term generalization whilst considering an arbitrary number of relations (between the terms), *automatically*.
2) The added value of ReEx with respect to its performance is demonstrated on nine real-life gene expression data sets linked to *Gene Ontology* [4], where compact, *interpretable* explanations consisting of semantic terms are obtained as a result of generalization from the attributes, recognized as relevant by SHAP [3].
3) We propose an information-theoretic measure of generalization derived from the information content of individual terms (or sets of terms) we refer to as GenQ. The measure is used to directly assess the generalization performance and is arguably more suitable for this task than the commonly employed information content.
4) The proposed solution is presented as a simple-to-use Python package, useful off-the-shelf alongside the existing model explainers.

Slovenian Research Agency

II. Related work

In this section, we present the related work, ranging from the notion of model explanation and semantic data mining to the more recent approaches which attempt to join the two subfields by treating semantic background knowledge as *graph-theoretic objects*, directly useful for applications in machine learning.

General solutions for explanations based on individual instances were proposed more than a decade ago [5], offering promising solutions to better understand classifiers such as the support vector machines and similar. Explanation of black-box models, however, has resurfaced in the recent years, along with the revival of neural networks and the development of deep neural networks [6]. Such multi-million parameter neural networks are able to associate beyond the capabilities of any other learners when considering inputs such as images, texts and more recently - graphs [7]. Methods, such as LIME [2], SHAP [3] and similar [8], [9], were introduced and have, by offering simple-to-use APIs, become widely used throughout the industry and science. For example, a recent review [8] elaborates on the importance of understanding the causality of the learned representations via explanations.

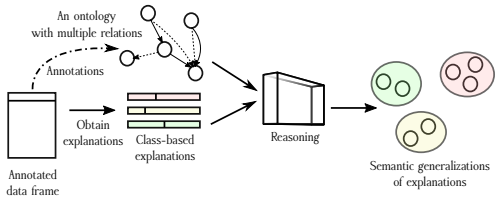


Fig. 1: Overview of the proposed ReEx. Given a data set where the attributes map to a *domain ontology*, ReEx first produces and aggregates the instance-level explanations. Top attributes are selected iteratively (vertical lines in the second sub-image of the three vectors), followed by a reasoning procedure, which exploits the explicitly given relations (within the ontology) to generalize the mapped terms into *higher-level* semantic terms. Generalizations are obtained for each class (last sub-figure).

It explains the difference between *post-hoc* systems, which aim to provide the local explanations for a specific decision, making it reproducible on demand instead of explaining the whole systems' behavior (LIME), and *ante-hoc* systems, which are interpretable by design and are considered as the *white-box approaches* (linear regression, decision trees, rules). Further, the recent work on the use of machine learning (ML) approaches [10] argues that blindly accepting the outcome of an ML system is a dangerous practice, which is adopted by many *out of necessity*, or by choice. To overcome this a ML system should provide an explanation of its decision-making process.

The use of background knowledge for improving the understandability of machine learning systems has also been referred to as *semantic data mining* [11]. For example, in relational domains, the CBSSD methodology [12] focuses on data mining tasks that use ontologies as background knowledge when explaining emergent structures in complex networks. Similarly, the NetSDM approach [13] explored how redundant ontologies are for the purposes of inductive rule learning. Furthermore, promising results were observed when attempting to account for explicit semantics during image classification [11], and using inductive logic programming to obtain relational explanations (of image classifications) [14]. The ReEx approach presented in the following sections attempts to bridge the gap between the plethora of available relational background knowledge sources and explainable models, offering a fast and simple to use method, complementary to the existing model explainers incapable of exploiting existing background knowledge.

III. ReEx - Reasoning from model explanations

The following section discusses the proposed ReEx approach, summarized in Figure 1. The first part of ReEx concerns with obtaining class-specific aggregates of feature importances. To achieve this, the following steps are considered:

Initial feature pruning. We select the top k features according to the descending order of their mutual information with the target variable. This myopic measure is selected due to its computational efficiency [15].

Cross-validation and explanation acquisition. The model is trained in a 10-fold cross-validation scheme, where for each of the test instances (one of the folds), the SHAP kernel explainer [3] is used to obtain a list of Shapley values, indicating which features were the most relevant for a given prediction.

Explanation aggregation. The following procedure is considered for each of the classes; for each feature that appeared in an explanation at least once, the values across all explained instances are *averaged*. When all features are considered, they are sorted in decreasing order based on the relevance values. More formally, let $p_{C,+}$ denote a probability distribution of a collection of explanation values across the correctly classified samples of selected class C. Let X_i denote the random variable representing the explanation for the i-th feature F_i. Let $\boldsymbol{X}$ denote the vector of all such random variables, having each entry associated with a given (i-th) feature. ReEx constructs a set of tuples defined as follows:

$$\textsc{ReEx-agg} = \bigcup_{C} (C, \mathbb{E}_{\boldsymbol{X} \sim p_{C,+}}[\boldsymbol{X}]),$$

Here, $\boldsymbol{X}_i$ represents a random variable representing the Shapley value for the i-th feature. A vector of the expected values is thus considered for each class. The elements of the vector are the *importances* associated with a given (i-th) feature.

A more detailed overview of the idea behind SHAP is discussed next. SHAP [3] is based on the *coalitional game theory*, and aims to capture the importance of interactions between features via Shapley values. When considered in a feature importance estimation scenario, the contribution of the i-th instance, denoted with τ_i is approximated by SHAP with the following expression:

$$\tau_i = \underbrace{\sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!}}_{\text{All possible subsets}} \underbrace{\Big[f(x_{S \cup \{i\}}) - f(x_S) \Big]}_{\text{Difference in predictive performance}}$$

where S is a subset of all features F, f is the used predictive model, and x_S is an instance containing only features from the subset S. Shapley values offer insights into the instance-level predictions by assigning fair credit to individual features for participation in interactions. They are commonly used to understand and debug black-box models [16].

In this work, we use the SHAP *kernel approximator*, the recently introduced, model-agnostic method for explaining model outputs. The used SHAP kernel explainer is considered an additive feature attribution method. Such methods are

characterized by having an explanation model g that is a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{|F|} \phi_i \cdot z'_i$$

where $z' \in \{0,1\}^{|F|}$, $|F|$ is the number of input features and $\phi_i \in \mathbb{R}$. This class of models assign an importance ϕ_i to each feature and, summing the effects of all such feature attributions, approximates the output $f(x)$ of the original model. Detailed theoretical analysis of how this idea can be extended to an approximation of outputs via a kernel is given in [3]. The final result of this step are thus lists of tuples for each class.

Thresholding - ReEx implements a *dynamic threshold* for each class which we lower until at least n number of features in that class have their assigned SHAP value greater than the threshold.

Reasoning. After the discussed procedure ReEx obtains a set of features that were estimated as the most important for each class. These collections are used as the input to reasoning, discussed next. *Selective staircase* is the faster of the two algorithms proposed as a part of this work. Given the threshold parameter, the algorithm generalizes a term into its highest ancestors, whose ratio of terms, connected to the other classes, is below the user-specified threshold. It searches through all viable ancestors in iterations, each iteration considering more general ancestors. Hence, the algorithm *is deterministic*. *Ancestry* is a slower algorithm that considers *lowest common ancestors* of a pair of terms. This means that every term that is the result of a step of generalization is connected to at least two previously found terms (or starting terms). Found ancestors are considered based on their ratio of connected starting terms from other classes and length of the path from the two generalized terms and their ancestor. The algorithm is non-deterministic since there is a random element involved when choosing a pair of terms used for finding their common ancestor. It operates under the assumption, that generalization needs to be conducted based on *pairs of terms*, rendering the method potentially more *robust to noise*, however less flexible.

IV. Formulation of generalization approaches

In the following sections, we present the proposed implementations of the generalization procedures employed by ReEx. We begin with the Selective staircase (Section IV-A), followed by the Ancestry algorithm (Section IV-B).

A. Selective staircase

In a single iteration, we create a set of parents of the terms in the set for each class based on a given domain ontology. These are *more general*, however directly connected to the terms we currently have in the set. For each element in this set of parents, we calculate the proportion of starting terms of other classes that are descendants of this term. From this set, we remove those of which computed ratio is above the *given threshold*. We then add acceptable elements for each class from the parent set to the term set and remove from the term set the elements that are children of the newly added ones (we remove those that were generalized). The step is repeated for each class until term sets stop changing (generalization stops).

input : starting term sets, background knowledge, threshold
output: a generalized term set for each class and depth of generalization for each term

```
while not all converged do
    for class ∈classes do
        change ←false;
        parents ←getParents(termSets[class]);
        for parent ∈parents do
            if intersectionRatio(parent) <=threshold then
                termSets[class].add(parent);
                termSets[class].removeChild(parent);
                change ←true;
            end
        end
        if not change then
            converged[class]← true;
        end
    end
end
```

Algorithm 1: Selective staircase - generalizes every term into all of its ancestors, then selects those suitable.

B. Ancestry

For each term in the term set of a given class, we select another term in the same term set and find their *lowest common ancestor*. Note that the depth of this generalization step is considered to be equal to $\min$(distance(first term, ancestor), distance(second term, ancestor)) – *the shortest path to the ancestor*. We calculate the ratio of the starting terms of other classes that are descendants of this ancestor. If the following inequality: ratio$/$(depth $\cdot$ weight) $- \frac{1}{2} < 0$ holds for a given weight (parameter), the ancestor is added to the generalized term set of the class. After repeating this step with the whole term set, we remove the terms that were used in finding an ancestor that was added to the term set. We iterate this step for each class until the term sets stop changing (generalization stops).

C. Measuring success

Information content can be defined for a given term t as: $\mathrm{IC}(t) = -\log(p(t))$, where $p(t)$ represents the prior probability of a given entity being annotated with the term t. The information content offers direct insight into how general or specific a particular term or a set of terms is (e.g., mean IC). Let $\mathfrak{O}$ represent the set of all terms in a given ontology and T the set of considered terms. As the point of this paper is to

input : starting term sets, background knowledge, weight
output: a generalized term set for each class and depth of generalization for each term

while *not all converged* **do**
  for $class \in$*classes* **do**
    $change \leftarrow$false;
    for $term \in$*termSets[class]* **do**
      $randomTerm \leftarrow selectRandomTerm(termSets[class])$;
      $(ancestor, depth) \leftarrow findAncestor$(term, randomTerm);
      if *intersectionRatio*$(ancestor)/(depth * weight) <$*0.5* **then**
        termSets[class].add(ancestor);
        setUsed(term, randomTerm);
        $change \leftarrow$true;
      end
    end
    removeUsed(termSets[class]);
    if *not change* **then**
      converged[class]$\leftarrow$ true;
    end
  end
end

Algorithm 2: Ancestry - generalizes pairs of terms into their lowest common ancestor, then checks whether the ancestor is suitable.

assess the generalization, we propose the following measure, offering direct insight into the generalization quality,

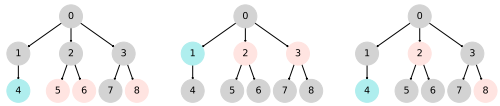
$$\text{GenQ}(T) = 1 - \frac{\sum_{t \in T} \log(p(t))}{|T| \cdot \text{NrO}}$$

$$\text{where NrO} = -\max(IC(t \in \mathfrak{O})) \sim \log(1/|\mathfrak{O}|).$$

The GenQ(T)'s domain is between zero and one. Intuitively this score can be understood as one minus the normalized average information content. The normalization is ontology-specific, i.e., prior to computing the score, ReEx is capable of identifying the term with the maximum information content – such terms are *commonly* the ones that appear only once (hence the term $\log(1/|\mathfrak{O}|)$).

V. A toy example and some theoretical properties

This section presents the main idea of the two algorithms in the form of a simple toy example when performing generalization on directed acyclic graphs (the example includes trees). Our example includes two classes - red and green and their sets of starting terms as indicated in figure Figure 2. The set of starting terms for the green class is {4} and for the red class it is {5, 6, 8}. We run the Selective staircase on our example with the threshold parameter of 0, meaning that terms that are the result of the generalization must not have any starting terms of other classes as their descendants. In the first iteration, term 4 will get generalized into term 1, terms 5 and 6 into term 2 and term 8 into term 3. After the first iteration sets of terms will therefore be {1} (for the green class) and {2, 3} (for the red class). In the second iteration, all terms in term sets have the same parent - 0, but because 0 is connected to starting terms of both classes, it cannot be added in either term set. Because term sets stay the same in the second iteration, generalization stops. We run Ancestry on our example with the weight parameter set to 0.000001, which, for our example, effectively means, that terms which are the result of the generalization must not have any starting terms of other classes as their descendants. Results of generalization may vary because the algorithm is *not deterministic*, so we describe the steps that the algorithm can make. Green class' term set holds one term and will not be generalized, since two terms are necessary to find their lowest common ancestor. If the algorithm tries to generalize 8 with 5 or 6, their *lowest common ancestor* is 0, which has a descendant of another class, but if it generalizes 5 and 6, the result of the generalization will be 2. If 5 and 6 are not generalized into 2 in the first iteration, generalization will stop, since it hasn't made any progress in the iteration. If they are, however, we proceed to the second iteration in which 0 is found as the ancestor of 2 and 8, but is not acceptable, so the generalization stops.



(a) Starting terms (b) Selective staircase (c) Ancestry

Fig. 2: A toy example of the two generalization procedures considered. The red and green colors represent the terms belonging to a particular class. They are generalized to the point no more generalization is possible without the term co-occurring as a result for both classes.

VI. Empirical evaluation

In the following section we discuss the conducted experiments, aimed at clarifying the capabilities of the two considered generalization schemes across a wide array of real-life data sets. The data sets considered are described in Table I. The considered selection includes data sets where the task is phenotype prediction (e.g., tissue classification). The used Gene Ontology [4] consists of 44,700 nodes and 91,526 edges. Considered relations are "part-of", "regulates", "negatively-regulates", "positively-regulates" and "is-a"[1].

Finally, the *Mapping* from genes to corresponding Gene Ontology terms consists of 19,412 genes mapped on average to 14.82 GO terms. The task considered in this work is *multiclass classification*. The methods used during evaluation are specified as follows. We employ three different classifiers, namely Gradient Boosting Machines (gradient-boosting),

[1]All relations are intentionally considered, to explore ReEx's capability to operate in automated manner – without human interventinons.

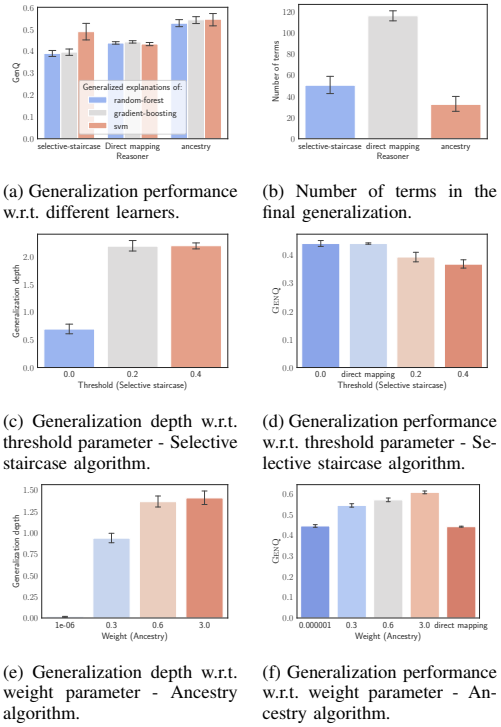
Random Forest (random-forest) and Support Vector Machine classifiers (SVM). Each of the classifiers is explained with [3]; the explanations further used as discussed in Section III. For both considered reasoning algorithms (Ancestry and Selective staircase), we computed the following grid of configurations and reported the mean with the standard deviation; subset sizes were either 100 or 5000 (initial pruning), absolute Shapley values were considered, threshold parameter, when using Selective staircase, was either 0,0.2 or 0.4 and weight parameter, when using Ancestry, was either 0.000001, 0.3, 0.6 or 3. The minimum number of terms was set to 10 (terms used for reasoning) and the thresholding step size was set to 0.975. Hence, around 500 different configurations were computed, when evaluating all the mentioned settings on all nine data sets. As the purpose of this work is to demonstrate that simple and efficient generalization is possible, we compare the algorithms' performances against the baseline we define as the naïve, direct mapping of features to the collection of terms – the mapping defined as part of the Gene Ontology. The rationale for introducing this baseline is as follows. Should the GENQ improve upon the naïve generalization (mapping), it will be higher. On the contrary, GENQ that would be lower than the generic mapping could indicate the algorithm was in fact performing *specialization*. We next present the obtained results alongside their discussion.

VII. RESULTS – REASONING BEHAVIOR

The following section includes the main results, obtained as the aggregate across the parameter space discussed in the previous section. The main results are summarized in Figure 3. It can be seen that the Ancestry method consistently generalizes terms into terms with a higher GenQ rating, while the Selective staircase's performance is less consistent 3a. Both methods notably decrease the number of resulting terms 3b. Parameters "threshold" and "weight" serve as the strictness of no cross-section between classes for Selective staircase and Ancestry respectively; smaller value meaning a more constrained generalization. Therefore the depth of generalization increases with increasing "threshold" and "weight" parameters, as seen in 3c and 3e. From graph 3d we can see, that a less constrained generalization with the Selective staircase can decrease generalization performance, even though we achieve a higher generalization depth. Generalization performance of the Ancestry algorithm increases with a less constrained generalization, however 3f. That could be due to Ancestry's generalization already being constrained with searching for ancestors of pairs of terms. The code to reproduce the experiments is available to the reader[2].

TABLE I: Statistical Properties of the Considered Data Sets.

Dataset	Instances	Features	Classes
DLBCL B [17]	180	661	3
DLBCL A [17]	141	661	3
Breast A [17]	98	1213	3
Breast B [17]	49	1213	4
DLBCL D [17]	129	3795	4
DLBCL C [17]	58	3795	4
Multi A [17]	103	5565	4
Multi B [17]	32	5565	4
TCGA [18]	801	20531	5



(a) Generalization performance w.r.t. different learners.

(b) Number of terms in the final generalization.

(c) Generalization depth w.r.t. threshold parameter - Selective staircase algorithm.

(d) Generalization performance w.r.t. threshold parameter - Selective staircase algorithm.

(e) Generalization depth w.r.t. weight parameter - Ancestry algorithm.

(f) Generalization performance w.r.t. weight parameter - Ancestry algorithm.

Fig. 3: Summary of the results across multiple hyperparameter settings.

VIII. RESULTS – EXAMPLES OF SEMANTIC EXPLANATIONS

Most general terms associated with classes are listed as follows[3] (Figure 4).

The terms comprising an explanation are human-understandable and as such offer direct insight into the biological processes governing the space of instances belonging to a given class. A single process can be attributed to more than one class, depending on how constrained the generalization has been. The two key aspects of the obtained *logical explanations* are: First, the logical statements for individual classes (conjuncts of semantic terms) are notably different – indicating different biological processes underlying

[2]The official URL is: https://github.com/OpaqueRelease/ReEx

[3]The star-marked entries' full names are DNA-binding transcription activator activity, RNA polymerase II-specific and detection of chemical stimulus involved in sensory perception of smell, respectively.

Subtype 1 : −protein homodimerization activity
∧ protein heterodimerization activity
∧ ubiquitin protein ligase binding
∧ magnesium ion binding
∧ calmodulin binding

Subtype 2 : −ATP binding
∧ protein homodimerization activity
∧ neutrophil degranulation
∧ DNA-binding transcription activator activity*
∧ sensory perception of smell*

Fig. 4: Example of generalized explanations for the Breast A [17] data set (classes are different subtypes).

the successful classification into a given class. And second, the explanations are directly understandable and can be verified by a domain expert within seconds. On the contrary, should raw feature names be used (gene names in this case), if the domain expert does not know all the names, as well as the associated processes by heart, the expert is expected to perform time-demanding manual literature search.

IX. Conclusions

In this work, we proposed ReEx, one of the first approaches capable of semantic generalization of model explanations obtained by contemporary tools such as SHAP. The work evaluates two different reasoning paradigms (Selective staircase and Ancestry), showing both schemes out-perform generic generalization commonly employed by e.g., statistical enrichment analysis approaches. Further, we demonstrate how ReEx can produce logical explanations comprised of semantic term conjuncts, specific for individual classes – these types of explanations offer direct insight into the e.g., biological background relevant to classifying a given instance into a particular class. Understanding how biological context can be exploited for obtaining more interpretable explanations could also be extended to contemporary knowledge graphs, where the data is semi-automatically curated. Here, the amount of noise is potentially larger than when considering e.g., the Gene Ontology (this work), which we believe is an issue to be addressed in future work. The proposed work was implemented in the form of a simple-to-use Python library, compatible with existing machine learning pipelines. Albeit being one of the first studies to explore the potentials of semantic generalization of black-box model explanations, ReEx could be further analyzed and improved.

X. Acknowledgements

The work of the last author was funded via a young researcher grant (ARRS). The work of other authors was supported by the Slovenian Research Agency (ARRS) core research programs P2-0103 and P6-0411, and research projects J7-7303, L7-8269, and N2-0078 (financed under the ERC Complementary Scheme). The work was also supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings for Less-Represented Languages in European News Media).

References

[1] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable AI systems for the medical domain?" *CoRR*, vol. abs/1712.09923, 2017. [Online]. Available: http://arxiv.org/abs/1712.09923

[2] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you?" Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.

[3] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[4] G. O. Consortium, "The gene ontology resource: 20 years and still going strong," *Nucleic Acids Research*, vol. 47, no. D1, pp. D330–D338, 2019.

[5] M. Robnik-Šikonja and I. Kononenko, "Explaining classifications for individual instances," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 5, pp. 589–600, 2008.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[7] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[8] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, 2017.

[9] E. Štrumbelj and I. Kononenko, "Explaining prediction models and individual predictions with feature contributions," *Knowledge and Information Systems*, vol. 41, no. 3, pp. 647–665, Dec 2014.

[10] D. Doran, S. Schulz, and T. R. Besold, "What does explainable AI really mean? A new conceptualization of perspectives," *CoRR*, vol. abs/1710.00794, 2017. [Online]. Available: http://arxiv.org/abs/1710.00794

[11] D. Dou, H. Wang, and H. Liu, "Semantic data mining: A survey of ontology-based approaches," in *Proceedings of the 2015 IEEE 9th international conference on semantic computing (IEEE ICSC 2015)*. IEEE, 2015, pp. 244–251.

[12] B. Škrlj, J. Kralj, and N. Lavrač, "CBSSD: Community-based semantic subgroup discovery," *Journal of Intelligent Information Systems*, Jan 2019. [Online]. Available: https://doi.org/10.1007/s10844-019-00545-0

[13] J. Kralj, M. Robnik-Sikonja, and N. Lavrac, "Netsdm: Semantic data mining with network analysis," *Journal of Machine Learning Research*, vol. 20, no. 32, pp. 1–50, 2019. [Online]. Available: http://jmlr.org/papers/v20/17-066.html

[14] J. Rabold, M. Siebers, and U. Schmid, "Explaining black-box classifiers with ilp – empowering lime with aleph to approximate non-linear decisions with relational rules," in *Inductive Logic Programming*, F. Riguzzi, E. Bellodi, and R. Zese, Eds. Cham: Springer International Publishing, 2018, pp. 105–117.

[15] B. C. Ross, "Mutual information between discrete and continuous data sets," *PLOS ONE*, vol. 9, no. 2, pp. 1–5, 02 2014. [Online]. Available: https://doi.org/10.1371/journal.pone.0087357

[16] S. Ghosal, D. Blystone, A. K. Singh, B. Ganapathysubramanian, A. Singh, and S. Sarkar, "An explainable deep machine vision framework for plant stress phenotyping," *Proceedings of the National Academy of Sciences*, vol. 115, no. 18, pp. 4613–4618, 2018.

[17] Y. Hoshida, J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Subclass mapping: Identifying common subtypes in independent disease data sets," *PLOS ONE*, vol. 2, no. 11, pp. 1–8, 11 2007. [Online]. Available: https://doi.org/10.1371/journal.pone.0001195

[18] J. N. Weinstein, E. A. Collisson, G. B. Mills, K. R. M. Shaw, B. A. Ozenberger, K. Ellrott, I. Shmulevich, C. Sander, J. M. Stuart, C. G. A. R. Network *et al.*, "The cancer genome atlas pan-cancer analysis project," *Nature Genetics*, vol. 45, no. 10, p. 1113, 2013.

Appendix D: Exploring Neural Language Models via Analysis of Local and Global Self-Attention Spaces

Exploring Neural Language Models via Analysis of Local and Global Self-Attention Spaces

Blaž Škrlj
Jožef Stefan International Postgraduate School
Jožef Stefan Institute, Slovenia
blaz.skrlj@ijs.si

Shane Sheehan
University of Edinburgh, United Kingdom

Nika Eržen
Jožef Stefan Institute, Slovenia

Marko Robnik-Šikonja
University of Ljubljana, Slovenia

Saturnino Luz
University of Edinburgh, United Kingdom

Senja Pollak
Jožef Stefan Institute, Slovenia

Abstract

Large pretrained language models using the transformer neural network architecture are becoming a dominant methodology for many natural language processing tasks, such as question answering, text classification, word sense disambiguation, text completion and machine translation. Commonly comprising hundreds of millions of parameters, these models offer state-of-the-art performance, but at the expense of interpretability. The attention mechanism is the main component of transformer networks. We present AttViz, a method for exploration of self-attention in transformer networks, which can help in explanation and debugging of the trained models by showing associations between text tokens in an input sequence. We show that existing deep learning pipelines can be explored with AttViz, which offers novel visualizations of the attention heads and their aggregations. We implemented the proposed methods in an online toolkit and an offline library. Using examples from news analysis, we demonstrate how AttViz can be used to inspect and potentially better understand what a model has learned.

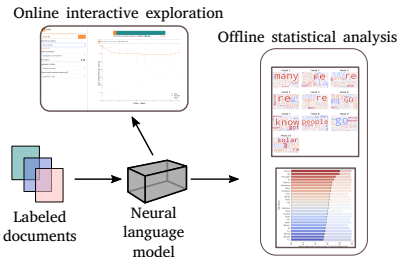


Figure 1: An overview of AttViz suite. The system consists of two main functional modules supporting online and offline visualizations. The online visualization (http://attviz.ijs.si; first part of the paper) offers direct exploration of token attention across the space of input documents; its purpose is anomaly detection and general inspection of the attention space (of trained models). The offline part of AttViz (second part of the paper) is a Python library that offers computationally more demanding statistical analyses, ranging from visualization of key tokens for each attention head, comparison of the attention head properties via FUJI integrals, and inspection of the attention distribution per-token basis.

1 Introduction

Currently the most successful machine learning approaches for text-related tasks predominantly use large *language models*. They are implemented with transformer neural network architecture (Vaswani et al., 2017), extensively pretrained on large text corpora to capture context-dependent meanings of individual tokens (Devlin et al., 2019; Liu et al., 2019; Yang et al., 2019). Even though training of such neural networks with hundreds of millions of parameters is long and expensive (Radford et al., 2019), many pre-trained models have been made freely available. This has created an opportunity to explore how, and why these models perform well on many tasks. One of the main problems with neural network models is their lack of *interpretability*. Even though the models learn the given task well, understanding the reasons behind the predictions, and assessing whether the model is

susceptible to undue biases or spurious correlations is a non-trivial task.

Approaches to understanding black-box (non-interpretable) models include *post-hoc* perturbation methods, such as IME (Štrumbelj and Kononenko, 2010) and SHAP (Lundberg and Lee, 2017). These methods explain a given decision by assigning a credit to inputs (i.e. attributes or tokens) that contributed to it. These methods are not internal to the model itself and are not well adapted to the sequential nature of text-based inputs. Another way of extracting token relevance is the attention mechanism (Bahdanau et al., 2015; Luong et al., 2015) that learns token pair-value mappings, potentially encoding relations between token pairs. The attention of a token with respect to itself (called self-attention due its position on diagonal of the token attention matrix) offers certain insight into the importance of the token. Typically, a trained transformer network contains several attention heads, each bringing a different focus to the final decision of the network. Exploration of attention can be analytically and numerically cumbersome task, resulting in development of several approaches aimed at attention visualization collection.

As neural networks require numerical input, words are first transformed into a high dimensional numeric vector space, in a process called embedding that aims to preserve similarities and relations between words. Visualizations of embedding spaces is becoming ubiquitous in contemporary natural language processing. For example, Google's online Embedding Projector[1] offers numerous visualizations for technically non-savvy users, by projecting word vectors to low dimensional (human-understandable) spaces. While visualization of embedding spaces is already accessible, visualization of internal workings of complex transformer neural networks (e.g.,their self-attention mechanism) is a challenging task. The works of (Liu et al., 2018) and (Yanagimto et al., 2018) attempt to unveil the workings of black-box attention layers and offer an interface for human researches to learn and inspect their models. Liu et al. (2018) visualize the attention space by coloring it, and Yanagimto et al. (2018) visualize the self-attention with examples from a sentiment analysis.

In this work, we present AttViz, an online system that focuses exclusively on self-attention and introduces two novel ways of visualizing this property. The tool serves as an additional tool in the toolbox of a language model researcher, offering exploration of the learned models with minimal effort. AttViz can interactively aggregate the attention vectors and offers simultaneous exploration of the output probability space, as well as the attention space. A schematic overview of the proposed work is shown in Figure 1, and the main contributions are summarised as follows:

1. We present and describe AttViz, an interactive, online toolkit for visualization of the attention space of trained transformer neural language models.
2. We demonstrate the capabilities of AttViz on three problems: news classification, hate speech detection, and insults detection.
3. AttViz includes a stand-alone python library for offline analysis of the attention space, with the key focus on the relations between *the attention heads*.

The remainder of the paper is structured as follows. In Section 2, we discuss works related to the proposed AttViz approach. In Section 3, we present the key ideas and technical implementation of the online part of the AttViz system, including a use case on news classification. In Section 4, we discuss the capabilities of the AttViz library, available in an offline mode, and showcase its use on additional two datasets. In Section 5 we discuss capabilities and limitations of AttViz, present conclusions, and propose ideas for future work.

2 Background and related work on attention visualization

Neural language models are becoming the prevailing methodology for solving various text-related tasks, from entity recognition to classification. Visualization of the attention mechanism that is the key component of such models has recently emerged as an active research area due to an increased popularity of attention based methods in natural language processing. Recent deep neural network language models such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), and RoBERTa (Liu et al., 2019) consist of multiple attention heads—separate weight spaces each associated with the input sequence in *a unique way*. These transformer language models consist of multiple attention matrices, all contributing to the final prediction. Visualising the attention weights

[1] https://projector.tensorflow.org/

from each of the attention matrices is an important component in understanding and interpreting these models.

The attention mechanism, which originated in the neural machine translation, lends itself naturally to visualisation. Bahdanau et al. (2015) used *heat maps* to display the attention weights between input and output text. This visualisation technique was first applied in machine translation but found its use in many other tasks. Rush et al. (2015) visualized an input sentence and the output abstractive summary, while Rocktäschel et al. (2016) showed an association between an input document and a textual entailment hypothesis on the output. In these heat map visualisations, a matrix is used to represent the token-token pairs and color intensity illustrates attention weights. This provides a summary of the attention patterns describing how they map the input to the output. For classification tasks, a similar visualisation approach can be used to display the attention weights between the classified document and the predicted label (Yang et al., 2016; Tsaptsinos, 2017). Here, the visualisation of attention often displays the input document with the attention weights superimposed onto individual words. The superimposed attention weights are represented similarly to heat map visualisations, using the color saturation to encode attention value. The neat-vision tool[2] encodes attention weights associated with input text in this manner. Similarly, the Text Attention Heatmap Visualization (TAHV[3]) which is included in the NCRF++ toolkit (Yang and Zhang, 2018) can be used to generate weighted sequences which are visualised using superimposed attention scores.

The purpose of the proposed AttViz is to unveil the attention layer space to human explorers in an intuitive manner. The tool emphasizes *self-attention*, that is, the diagonal of the token-token attention matrix which possibly corresponds to the *relevance* of individual tokens. Using different encoding techniques, attention weights across the layers and attention heads can be explored dynamically to investigate the interactions between the model and the input data. The AttViz tool differs from other tools in that if focuses on self-attention, thus allowing visualization of (attention-annotated) input token sequences to be carried out directly.

[2] https://github.com/cbaziotis/neat-vision

[3] https://github.com/jiesutd/Text-Attention-Heatmap-Visualization

3 AttViz: An online toolkit for visualization of self-attention

AttViz is an online visualization tool that can visualize neural language models from the PyTorch-transformers library[4]—one of the most widely used resources for natural language modeling. The idea behind AttViz is that it is *simple to use* and *lightweight*, therefore it does not offer computationally expensive (online) neural model training, but facilitates the exploration of *trained* models. Along with AttViz, we provide a set of Python scripts that take as an input a trained neural language model and output a JSON file to be used by the AttViz visualisation tool. A common pipeline for using AttViz is outlined in Figure 1. First, a transformer-based trained neural network model is chosen to obtain predictions on a desired set of instances (documents or some other texts). The predictions are converted into the JSON format suitable for use with the AttViz tool, along with the attention space of the language model. The JSON file is loaded into the AttViz tool (on the user's machine, i.e. on the client side), where its visualization and exploration is possible. In Sections 3.1 and 3.3, we present the proposed self-attention visualizations, followed by an example of their use on the news classification task in Section 3.4.

3.1 Visualization of self-attention heads

We discuss the proposed visualization schemes that emphasize different aspects of self-attention. Following the first row that represents the input text, consequent rows correspond to attention values that represent the importance of a given token with respect to a given attention head. As discussed in the empirical part of the paper (Section 3.4), the rationale for this display is that typically only a certain number of attention heads are activated (colored fields). Thus, the visualization has to entail both the whole attention space, as well as emphasize individual heads (and tokens). The initial AttViz view offers sequence-level visualization, where each (byte-pair encoded) token is equipped with a self-attention value based on a given attention head (see Figure 4; central text space). The same document can also be viewed in the "aggregation" mode (Figure 2), where the attention sequence is shown across the token space. The user can interactively explore how the self-attention varies for individ-

[4] https://github.com/huggingface/transformers

ual input tokens, by changing the scale, as well as the type of the aggregation. The visualization can emphasize various aspects of the self-attention space.

The third proposed visualization (Figure 3) is the overall distribution of attention values across the whole token space. For each consequent token, the attention values are plotted separately, resembling a time series. This visualization offers an insight into *self-attention peaks*, i.e. parts of the attention space around certain tokens that potentially impact the performance and decision making process of a given neural network. This view can emphasize different aggregations of the attention vector space for a single token (e.g., mean, entropy, and maximum). The visualization, apart from the mean self-attention (per token), offers the information on maximum and minimum attention values (red dots), as well as the remainder of the self-attention values (gray dots). In this way, a user can explore both the self-attention peaks, as well as the overall spread.

3.2 Comparison with state-of-the-art

In the following section, we discuss similarities and differences between AttViz and other state-of-the-art visualization approaches. Comparisons are summarized in Table 1.

Novel functionality introduced by AttViz include the capability to aggregate the attention vectors with four different aggregation schemes, offering insights both into the average attention but also its dispersion around a given token. The neat-vision project[5] is the closest to AttViz in terms of functionality. However, a few differences should be noted. First, neat-vision is not directly bound to the PyTorch transformers library, requiring additional pre-processing on the user-side. Second, switching between the sequence and aggregate view is faster and more emphasized in AttViz, as it offers a more general overview of the attention space.

3.3 Aggregation of self-attention

The self-attention is captured in the matrix $A \in \mathbb{R}^{h \times t}$, where h is the number of attention vectors and t the number of tokens. Aggregation operators are applied the second dimension of the attention matrix A (index j). We denote with P_{ij} the probability of observing A_{ij} in the j-th column. The m_j corresponds to the number of unique values in that column. The proposed schemes are summarized in Table 2. The attention aggregates are visualized as part of the the aggregate view (see Figure 4). For example, the mean attention is plotted as a line along with the attention space for each token, depicting the *dispersion* around certain parts of the input text.

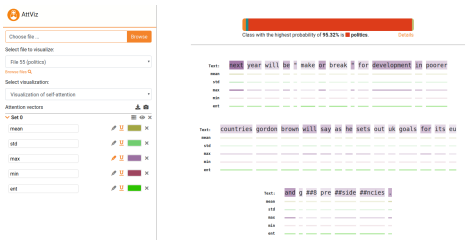


Figure 2: Visualization of aggregations. The document was classified as a politics-related topic; the aggregations emphasize tokens such as “development”,“uk” and “poorer”. The user can highlight desired head information – in this example the maximum attention (purple) is highlighted.

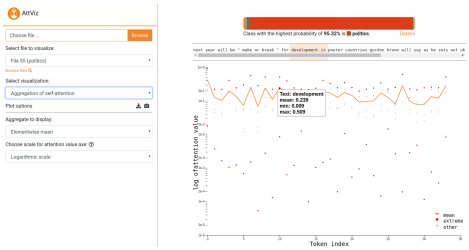


Figure 3: The interactive series view. The user can, by hoovering over the desired part of the sequence, inspect the attention values and their aggregations. The text above the visualization is highlighted automatically.

3.4 Example: News visualization

In this section, we present a step-by-step use of the AttViz system along with potential insights a user can obtain.

The examples are based on the BBC news data set[6] (Greene and Cunningham, 2006) that contains 2,225 news articles on five different topics (business, entertainment, politics, sport, tech). The documents from the dataset were split into short segments. The splits allow easier training (manage-

[5] Available at https://github.com/cbaziotis/neat-vision

[6] https://github.com/suraj-deshmukh/BBC-Dataset-News-Classification/blob/master/dataset/dataset.csv

Approach	AttViz (this work)	BertViz (Vig, 2019)	neat-vision	NCRF++ (Yang and Zhang, 2018)
Visualization types	sequence, aggregates	head, model, neuron	sequence	sequence
Open source	✓	✓	✓	✓
Language	Python + Node.js	Python	Python + Node.js	Python
Accessibility	Online	Jupyter notebooks	Online	script-based
Sequence view	✓	✓	✓	✓
Interactive	✓	✓	✓	✗
Aggregated view	✓	✗	✗	✗
Target probabilities	✓	✗	✓	✗
Compatible with PyTorch Transformers? (Wolf et al., 2020)	✓	✓	✗	✗
token-to-token attention	✗	✓	✗	✓

Table 1: Comparison of different aspects of the attention visualization approaches.

Table 2: Aggregation schemes used in AttViz. The A represents a real valued (attention) matrix.

Aggregate name	Definition
Mean(j) (mean)	$\frac{1}{h}\sum_i A_{ij}$
Entropy(j) (ent)	$-\frac{1}{m_j}\sum_{i=0}^{h} A_{ij}\log A_{ij}$
Standard deviation(j) (std)	$\sqrt{\frac{1}{h-1}\sum_i (A_{ij}-\overline{A_{ij}})^2}$
Elementwise Max(j) (max)	$\max_i(A_{ij})$
Elementwise Min(j) (min)	$\min_i(A_{ij})$

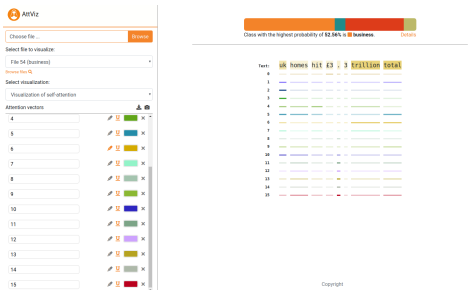


Figure 4: Visualization of all attention heads. The sixth heads's self attention is used to highlight the text. The document was classified as a business-related, which can be linked to high self attention at the "trillion" and "uk" tokens. Compared to the first two examples (Figures 2 and 3), the network is less *certain* – in this example, the business (orange) and politics (red) classes were predicted with similar probabilities (orange and red parts of the bar above visualized text).

able sequence lengths), as well as easier inspection of the models. We split the dataset into 60% of the documents that were used to fine-tune the BERT-base (Devlin et al., 2019) model, 20% for validation and 20% for testing. The Nvidia Tesla V100 GPU processor was used for these experiments. The resulting model classified the whole documents into five categories with 96% accuracy, which is comparable with the state-of-the-art performance (Trieu et al., 2017). For prediction and visualisation, we used only short segments. The fine-tuning of the BERT model follows examples given in the PyTorch-Transformers library (Wolf et al., 2020). The best-performing hyper parameter combination used 3 epochs with the sequence length of 512 (other hyper parameters were left at their default values). While we have used BERT, similar explorations could be made for more recent larger models such as XLNet (Yang et al., 2019) that might could produce better classification accuracy.

The user interface of AttViz is displayed in Figures 2, 3, and 4. In the first example (Figure 3), the user can observe the main view that consists of two parts. The leftmost part shows (by id) individual self-attention vectors, along with visualization, aggregation and file selection options. The file selection indexes all examples contained in the input (JSON) file. Attention vectors can be colored with custom colors, as shown in the central (token-value view). The user can observe that, for example, the violet attention head (no. 5) is active, and emphasizes tokens such as "development", which indicates a politics-related topic (as correctly classified). Here, the token (byte-pair encoded) space is shown along with self-attention values for each token. The attention vectors are shown below the token space and aligned for direct inspection (and correspondence).

In Figure 4, the user can observe the same text segment as an attention series spanning the input token space. Again, note that tokens, such as "trillion" and "uk" correspond to high values in a subset of the attention heads, indicating their potential importance for the obtained classification. However, we observed that only a few attention heads activate with respect to individual tokens, indicating that other attention heads are not focusing on the tokens themselves, but possibly on *relations* between them. This is possible, and the attention matrices contain such information (Vig, 2019). However, as mentioned earlier, the study of token relations is not the focus of this work. As self-attention in-

formation can be mapped across token sequences, emphasizing tokens that are of relevance to the classification task at hand, we see AttViz as being the most useful when exploring models used for text classification tasks, such as hate speech detection and sentiment analysis, where individual tokens contain the key information for classification.

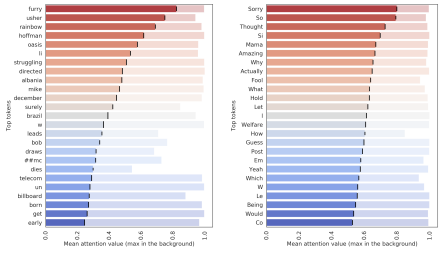
The example above shows how different attention heads detect different aspects of the sentence, even at the single token (self-attention) level. The user can observe that the next most probable category for this topic was politics (red color), which is indeed a more sensible classification than, for instance, sports. The example shows how interpretation of the attention can be coupled with the model's output for increased interpretability.

4 AttViz library: statistical analysis of the attention space

In Section 3 we presented how the online version of AttViz can be used for *direct analysis* of model output (in the JSON format). Albeit suitable for quick inspections, the online system has its limitations such as poor support for computationally more intensive types of analysis (in terms of waiting times), and the lack of customized visualization tools accessible in the Python ecosystem. To address these aspects, we developed AttViz library that offers more detailed analysis of a given neural language model's properties. The library operates on the same JSON structures as the online version and is compatible with the initial user input. We demonstrate the analytical capabilities of our visualization tools on three datasets. The BBC news classification was already presented in Section 3.4.

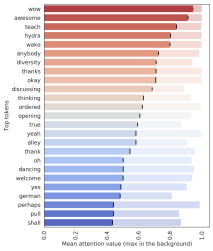
4.1 Dissecting the token space

The first offline functionality is a barplot visualization that offers insight into relevant aspects of the attention distribution at token level. Whilst understanding the attention peaks is relevant for direct inspections (Section 3), the attention space of a given token can be contextualized on the dataset level as well. The AttViz library offers fast visualization of the mean and spread of attention distributions, simultaneously showing the attention peaks for individual tokens. We visualized the distribution for three classification datasets (Figure 5): BBC news (5a), insults[7] (5b), and hate speech comments (5c)[8].



(a) Top 35 tokens in the BBC dataset.

(b) Top 35 tokens in the insults dataset.



(c) Top 35 tokens in the hate speech dataset.

Figure 5: Visualization of the 35 most attended-to tokens for the three inspected data sets. Interestingly, the attention peaks of tokens (maximum, in the background) all take high values, albeit lower-ranked tokens are on average characterized by lower mean attention values.

The proposed visualizations present top k tokens according to their mean attention throughout the whole dataset. It is interesting to observe, that the insults and hate speech data sets are not completely characterized by swear words or similar single-token-like features. This potentially indicates that the attention tries to detect interactions between the byte-pair encoded tokens, even for data sets where the attention could be focused on single tokens. It is interesting to observe that the terms with the highest attention are not necessarily keywords or other tokens carrying large semantic meaning. Similarly, the high maxima indicate that the emphasis of the tokens is very contextual, and potentially not as informative for global aggregation.

[7] https://www.kaggle.com/c/detecting-insults-in-social-commentary/overview

[8] https://github.com/aitor-garcia-p/hate-speech-dataset

4.2 Visualization of attention head focus

(a) Insults.

(b) Hatespeech.

(c) BBC news.

Figure 6: The distribution of tokens over individual attention heads for the three datasets summarised with word clouds.

Contemporary neural language model architectures comprise multiple attention heads. These separate weight spaces capture distinct aspects of the considered learning task. Even though the weight spaces are easily accessible, it is not trivial to convert the large amount of information into a quick-to-inspect visualization. With the proposed visualization, shown in Figure 6, we leverage word clouds (Kaser and Lemire, 2007) to reveal human-understandable patterns captured by separate attention heads and display this information in a compact way.

5 Discussion and conclusions

As AttViz is an online and offline toolkit for attention exploration, we discuss possible concerns regarding its use, namely: privacy, memory and performance overheads, and coverage. Privacy is a potential concern for most web-based systems. As currently AttViz does not employ any anonymization strategy, private processing of the input data is not guaranteed. While we intend to address this issue in furture work, a private installation of the tool can be done to get around this current limitation. AttViz uses the users' computing capabilities, which means that large data sets may cause memory overheads when a large number of instances is loaded (typically several million). Such situations are difficult to address with AttViz and similar web-based tool, but users can filter instances before using them in AttViz and explore a subset of the data (e.g., only (in)correctly predicted instances, or certain time slot of instances). Finally, AttViz is focused on the exploration of *self-attention*. This is not the only important aspect of a transformer neural network, but it is the one, where visualisation techniques have not yet been sufficiently explored. Similarly to the work of (Liu et al., 2018), we plan to further explore potentially interesting *relations* emerging from the attention matrices.

6 Availability

The software is available at `https://github.com/SkBlaz/attviz`.

Acknowledgements

We acknowledge European Union's Horizon 2020 research and innovation programme under grant agreement No 825153, project EMBEDDIA (Cross-Lingual Embeddings). The first author was also funded by Slovenian Research Agency as a young researcher.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Derek Greene and Padraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 377–384. ACM.

Owen Kaser and Daniel Lemire. 2007. Tag-cloud drawing: Algorithms for cloud visualization. In *Procedings of WWW Workshop on Tagging and Metadata for Social Information Organization*.

Shusen Liu, Tao Li, Zhimin Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. 2018. Visual interrogation of attention-based models for natural language inference and machine comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 36–41, Brussels, Belgium. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Lap Q. Trieu, Huy Q. Tran, and Minh-Triet Tran. 2017. News classification from social media using twitter-based doc2vec model and automatic query expansion. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*, SoICT 2017, pages 460–467.

Alexandros Tsaptsinos. 2017. Lyrics-based music genre classification using a hierarchical attention network. *CoRR*, abs/1707.04678.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Jesse Vig. 2019. Visualizing attention in transformer-based language representation models. *CoRR*, abs/1904.02679.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

H. Yanagimto, K. Hashimoto, and M. Okada. 2018. Attention visualization of gated convolutional neural networks with self attention in sentiment analysis. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, pages 77–82.

Jie Yang and Yue Zhang. 2018. NCRF++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79, Melbourne, Australia. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Erik Štrumbelj and Igor Kononenko. 2010. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11:1–18.