Uncovering semantic bias in neural network models using a knowledge graph

Uncovering Semantic Bias in Neural Network Models Using a Knowledge Graph

Andriy Nikolov*
Insight Centre for Data Analytics
National University of Ireland
Galway, Ireland
andriy.nikolov@insight-centre.org

Mathieu d'Aquin
Insight Centre for Data Analytics
National University of Ireland
Galway, Ireland
mathieu.daquin@insight-centre.org

ABSTRACT

While neural networks models have shown impressive performance in many NLP tasks, lack of interpretability is often seen as a disadvantage. Individual relevance scores assigned by post-hoc explanation methods are not sufficient to show deeper systematic preferences and potential biases of the model that apply consistently across examples. In this paper we apply rule mining using knowledge graphs in combination with neural network explanation methods to uncover such systematic preferences of trained neural models and capture them in the form of conjunctive rules. We test our approach in the context of text classification tasks and show that such rules are able to explain a substantial part of the model behaviour as well as indicate potential causes of misclassifications when the model is applied outside of the initial training context.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Rule learning*; *Natural language processing*; • **Information systems** → *Graph-based database models.*

KEYWORDS

neural networks, explanation, knowledge graphs, rule mining

ACM Reference Format:
Andriy Nikolov and Mathieu d'Aquin. 2020. Uncovering Semantic Bias in Neural Network Models Using a Knowledge Graph. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3412009

1 INTRODUCTION

Recent years have seen rapid advances in research on neural machine learning models and their application to different NLP tasks. However, although demonstrating impressive gains in accuracy, intransparency of these models remains a disadvantage [38]: it is not always clear what caused the model to make a decision. A common problem is a "Clever Hans"-type behaviour [22], when the model "cheats" by exploiting shortcut correlations in the training data, which results in high reported accuracy during training, but can lead to problems when deployed in the real world. For example, a publicly available LSTM network [36] was trained to classify tweets by political leaning (pro-democrat vs pro-republican) and achieved about 90% performance on the test dataset. However, when trying this model with two examples having opposite meaning "I hate @realDonaldTrump" and "I love @realDonaldTrump", the model classifies them both as pro-republican with very high confidence (0.994 and 0.991 respectively): i.e., it relies on *who* is referenced in a tweet rather than *what* the tweet actually says.

Existing generic neural network explanation methods such as [4, 41], and [28] are focusing on *post-hoc interpretability* [26], i.e., explaining the reasons for the model's decision on a single example by assigning different relevance scores to atomic tokens in the input text. Such explanations, however, are not sufficient to verify the whole model or to give clues about directions of improvement. To achieve *model interpretability*, we need to (a) make transparent systematic preferences of the model and possible spurious biases, which apply consistently across many cases, as well as (b) express these preferences in a human-understandable form.

A neural NLP model is able to take into account different kinds of textual features: from low-level syntactic patterns, stylistic elements, and sentiment to the actual factual information expressed about the domain. In this paper we focus on the latter and aim at capturing *semantic preferences* of the model dealing with domain knowledge. Our paper therefore makes the following contributions:

- We propose a novel method for discovering semantic preferences of neural text processing models. The method combines decision explanation using relevance scores with association rule mining over the knowledge graph to capture *explanation rules* stating which categories of domain entities are likely to shift the model decision in a certain way.
- We report evaluation experiments with our method on four test scenarios demonstrating that (a) mined rules are able to explain model decisions to a large extent and (b) preferences they capture are generic and hold when the model is applied outside of the original training context.

The rest of this paper is structured as follows. Section 2 presents a motivational example and briefly outlines the steps of our approach. In section 3, we provide an overview of existing solutions for deep learning explanation methods as well as rule mining approaches. Section 4 describes the process of extracting influential features from a set of input examples and linking them to knowledge graph entities. Section 5 focuses on mining the rules from selected entities. Section 6 describes the evaluation experiments. Finally, section 7 concludes the paper and discusses directions for future work.

*Author currently works at AstraZeneca, UK (email: andriy.nikolov@astrazeneca.com).

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10.
https://doi.org/10.1145/3340531.3412009

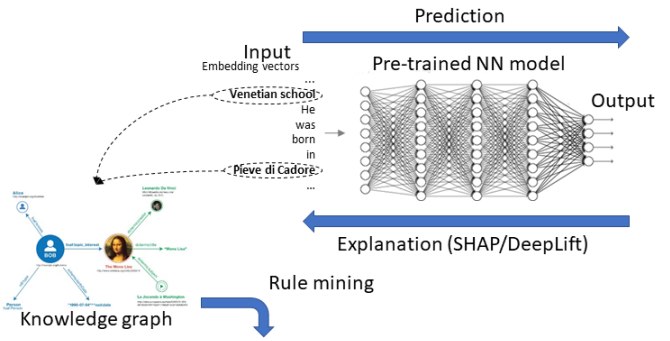

Figure 1: Capturing explanation in the form of knowledge graph-grounded rules.

2 MOTIVATION

Let us consider a convolutional network for binary text classification that takes a short biography of a painter and predicts whether his/her paintings are likely to be present in a major European art museum, such as Louvre, Prado, or Uffizi. When we give this model the following snapshot of Titian's biography, it gets misclassified, i.e. the model gives a confidence of only 0.24, which corresponds to the answer "No".

> Tiziano Vecelli *(+0.04)* (1488/1490 – 27 August 1576), known in English as Titian, was an Italian *(-0.03)* painter, the most important member of the 16th-century Venetian school *(+0.01)*. He was born in Pieve di Cadore *(-0.03)*, near Belluno *(-0.01)*, in Veneto *(-0.01)* (Republic of Venice *(-0.01)*). His painting methods would profoundly influence future generations of Western Art *(+0.001)*.

The text itself does not contain any obvious indication of why Titian would not be represented in a major museum, so, to get an idea about the reasons for misclassification, we can analyze the decision using an explanation tool such as SHAP/DeepExplainer [28], which assigns relevance scores to all tokens in the text. Looking at the scores (in brackets), we can see that the toponyms such as "Pieve di Cadore" and "Veneto" push the decision in the negative direction. Indeed, simply reducing the abstract to

> Tiziano Vecelli (1488/1490 – 27 August 1576), known in English as Titian, was an Italian painter, the most important member of the 16th-century Venetian school. His painting methods would profoundly influence future generations of Western Art.

is enough to switch the model's prediction to "positive" with a high score of 0.74. This isolated case hints at some non-intuitive preference within the model, but does not reveal it: e.g., is there a bias against the token "Pieve di Cadore", any Italian-originated word, any Italian toponym, all toponyms in general, or is there in fact no consistent bias and the decision was caused by some specific co-occurrence of tokens? In order to understand the model better, we need to be able to generalize from individual examples and capture such preferences at the semantic level: i.e., discover clusters of tokens that consistently influence the model in the same way and represent meaningful categories from the human point of view, for example:

"If the text mentions an Italian toponym, this usually shifts the model's decision to negative"

Such rules can be used (a) to analyze the reasons for misclassifications when the model is deployed in the real world or (b) to improve the model by resampling the training data, removing the undesired correlations, and retraining.

To move from analyzing individual relevance scores to reasoning over domain categories, we propose an algorithm consisting of three stages (Figure 1):

- Discovering important input features influencing the model's decision using an existing post-hoc explanation method.
- Mapping discovered influential features to a knowledge graph which categorizes them and describes them with facts.
- Applying association rule mining [16] to capture the model's learned preferences in the form of predictive *explanation rules* over knowledge graph concepts and properties.

3 BACKGROUND AND RELATED WORK

Our work builds on results from two research directions, which, to our knowledge, have not been applied in combination so far: explaining neural network models and association rule mining.

3.1 Explaining neural network models

Unlike other machine learning algorithms (e.g., linear regression or decision trees), non-linear neural network models are not inherently intelligible [45]: i.e., they do not allow a human user to understand which input features were important for a given prediction. A post-hoc explanation method tries to overcome this by analyzing the network's decision process and assign relevance scores to input features. Let $f_c(\cdot)$ denote some *prediction function* learned by a neural network for each class c. We denote as $\mathbf{X_1}, \ldots, \mathbf{X_n}$ a set of input examples, each one consisting of a sequence of tokens of length T: $\mathbf{X_j} = \{x_{j,1}, \ldots, x_{j,T}\}$. Each input feature $x_{j,k}$ represents an occurrence of some word w_i in the vocabulary $\mathcal{V}$. An explanation represents a set of real-valued scores $\phi(c, t, \mathbf{X})$ that assign relevance for each class c for each position $t = 1, \ldots, T$ of the input text $\mathbf{X}$. For a binary classification task, we will use the notation $\phi(t)$.

Some methods try to provide explanations without analyzing the internals of the model. One group of such black-box algorithms utilizes *input perturbations*: modifying elements of the input and measuring changes in the output. While originally proposed for image processing [46, 47], this approach was later adapted for NLP tasks as well [20, 25]. Another group, represented by the LIME [37] algorithm tries to approximate the behaviour of a neural network classifier in the neighbourhood of the input X with an interpretable (e.g., linear) model. While being generic, such black-box methods (a) can be inefficient due to the need for multiple input perturbations and sampling and (b) do not give interpretable information about model's internal parameters.

Another class of methods tries to assign relevance scores by back-propagating through the model, e.g. by calculating the output's gradient with respect to the input [42, 43]. This, however, highlights such input features to whose change the model is the most sensitive, which are not necessarily the ones that contributed

the most to the decision [31]. Layer-wise relevance propagation (LRP) [4, 6] defines back-propagation rules that distribute the output value among the relevance scores of inputs such that $f_c(\mathbf{X}) = \sum_1^T \phi(c, t, \mathbf{X})$, thus assigning to each input value a fraction of the output. The DeepLift algorithm proposed by [41] instead computes the relevance scores that characterize the positive/negative influence of each input *relative* to some reference point input: i.e., $f_c(\mathbf{X}) = f_c(\mathbf{X}_{ref}) + \sum_1^T \phi(c, t, \mathbf{X})$. This is an advantage in our scenario, since it makes relevance score interpretation independent of its absolute value, which, in turn, makes it easy to aggregate relevance scores across multiple data instances. There exist several implemented variations of DeepLift [3, 28, 41], among which we selected to use SHAP/DeepExplainer [28] in our experiments for its support for a range of different network architectures.

More recently, the idea of concept-based explanations arose in the image processing domain [15, 21], where "concepts" are informally defined as human-understandable image segments (e.g., "wheel" or "police logo"). While this has some similarities with our general goal, these methods merely consider "concepts" as annotated similarity clusters of image segments, rather than ontological concepts formally defined in a knowledge graph, which enable rule mining. The Semantic Web research community also started exploring the possibilities of bringing in domain ontologies to complement machine learning models and make them more transparent [17, 23, 39].

Besides dedicated explanation methods, certain elements of network architectures are often used for interpretation, in particular, convolution filters [18] and learned attention weights [7, 25]. However, these can only be exploited for certain architectures and tasks and it can be non-trivial to interpret them to make judgements about feature significance across multiple examples. Moreover, the validity of interpretations is often dubious [19, 40].

3.2 Mining association rules over knowledge graphs

Association Rule Mining (ARM) was originally proposed to discover relations between products in transaction databases [1]. The classical Apriori algorithm [2] established the typical two-stage procedure including (i) mining frequent itemsets and (ii) generating rules from these itemsets. Later algorithms followed the same procedure, but further optimized these stages [8, 33]. While the most popular measures used for selecting valid rules are support and confidence, a number of alternative "interestingness" metrics were proposed [24] with different advantages depending on the context.

One potential issue of ARM is the large number of rules produced by the Apriori algorithm and its modifications. This can make the resulting rules difficult to interpret by human users. To deal with this problem, various post-processing methods have been proposed to reorganize and filter the mined rule sets. For example, one approach involves grouping and reorganizing the mined rule set with the help of metarules [9] which capture the interdependencies between the rules themseleves. In [29], post-processing is based on the use of the domain ontology: an approach which produces well-defined and meaningful groupings, but requires additional manual modelling effort. A range of pruning techniques is based on the notion of rule covers [44]: a rule can be considered redundant if all objects it covers are also covered by other rules. In [27] this approach was taken further in application to associative classification rules (rules where consequents represent class label attributes), which also makes it relevant for our explanation rules use case (Section 5).

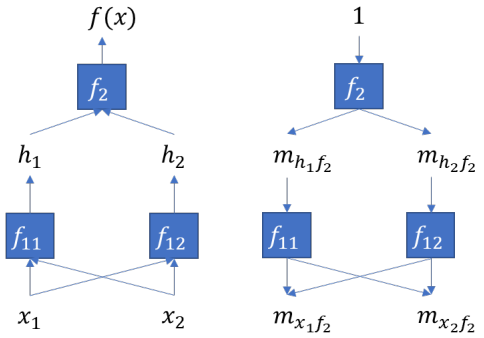

Figure 2: Approximating relevance scores by back-propagating DeepLift multipliers.

The growth of structured RDF data triggered research on inductive rule mining in the context of knowledge graphs/ontologies. Various Inductive Logic Programming (ILP) methods [32] were adopted to deal with different degrees of expressivity of the output. For example, AMIE [12, 13] focuses on mining Horn rules involving multiple variables: a task, for which classical ARM algorithms were found inefficient. Another direction involves learning description logic expressions to enrich the ontology [10, 35]. Our algorithm, however, involves mining Horn rules over a single variable and bears more similarities with the standard ARM scenario. For this reason, we adopt a modified the Apriori algorithm for our rule mining procedure in Section 5.

4 EXTRACTING RELEVANT ENTITIES

The first step of our method involves utilizing individual relevance scores produced by SHAP/DeepExplainer as evidence to discover influential domain entities. To this end, we aggregate and analyze the relevance scores over the whole set of available training instances.

4.1 Computing and aggregating relevance scores

SHAP assigns to each input feature a positive/negative weight that characterizes the impact of the feature relative to some reference point input. Thus, the sum of relevance scores for all input text tokens would be equal to the difference in prediction scores between the chosen example and the reference point, i.e. $f_c(\mathbf{X}) = f_c(\mathbf{X}_{ref}) + \sum_1^T \phi(c, t, \mathbf{X})$.

SHAP/DeepExplainer adapts the DeepLift algorithm to calculate approximate relevance scores by back-propagating DeepLift multipliers through the network layers (Figure 2). For a given input neuron with an input x and an output $f(x)$, the multiplier m_{xf} is defined as:

$$m_{xf} = \frac{\phi(f(x))}{\Delta x} = \frac{\phi(f(x))}{x - x_{ref}}$$

Thus, the multiplier m_{xf} is the contribution of Δx to $\Delta f(x)$ divided by Δx. The multipliers in this way are similar in spirit to partial derivatives (but are defined over finite differences) and are recursively passed back through the network as shown in Figure 2[28].

$$\forall_{j\in\{1,2\}} m_{h_j f_2} = \frac{\phi_i(f_2(h))}{h_j - h_{j,ref}}$$

$$\forall_{j\in\{1,2\}} m_{x_i f_{1j}} = \frac{\phi_i(f_{1j}(x))}{x_i - x_{i,ref}}$$

$$m_{x_i f_2} = \sum_{j=1}^{2} m_{x_i f_{1j}} m_{h_j f_2}$$

$$\phi_i(f(x)) \approx m_{x_i f_2}(x_i - x_{i,ref})$$

Back-propagation of the multipliers using the chain rule to pass through layers and rescaling to handle non-linearities allows approximating relevance in an efficient way. A common strategy for selecting an appropriate reference point is to compute the influence weights relative to a large set of reference examples and compute an average value, instead of trying to select a single input example. The relevance scores calculated for all training examples in this way serve as input of our algorithm.

At the first step of the algorithm we use aggregated influence weights for each input token to select the tokens that consistently "push" the model decision in the same direction: in the simplest case of a binary classification task, consistently "positive" or "negative" decision. Aggregating over multiple predictions helps to overcome the known fragility of neural network interpretations [11, 14].

Thus, for a specific token w_i in the vocabulary, we select all its mentions in the set of input examples $X_1, \ldots, X_N$, $M(w_i) = \bigcup_{j=1,\ldots,N}\{x_{j,k}|x_{j,k} = w_i\}$. After applying the explanation algorithm over the whole set of examples $\mathbf{X_1}, \ldots, \mathbf{X_n}$, we can get for each reference of any token in the vocabulary a set of relevance scores $\Phi(c, w_i) = \{\phi(c, k, \mathbf{X_j})|x_{j,k} \in M(w_i)\}$. Within this set, we separate the occurrences into *positive*, where the token drives the model to increase the decision score in favour of the given class:

$$\Phi^+(c,w_i)=\{\phi(c,k,\mathbf{X_j})\,|x_{j,k}=w_i,\phi(c,k,\mathbf{X_j})>0\}$$

and, similarly, *negative* ones $\Phi^-(c, w_i)$ where $\phi(c, k, \mathbf{X_j}) < 0$: e.g., the city of Belluno from our example is mentioned 4 times in the training corpus and only in 1 case with $\phi > 0$, so $|\Phi^+(c, \text{"}Belluno\text{"})| = 1$ and $|\Phi^-(c, \text{"}Belluno\text{"})| = 3$.

4.2 Selecting influential tokens

Once each occurrence of each token has been assigned a relevance score, the next step involves distinguishing the "influential" tokens which consistently impact the network in the same way, from non-significant ones, whose influence varies greatly between different occurrences and heavily depends on the context. For this, we introduce the influence function $F_{inf}(d_i)$, which would return a measure of influence for each token.

We tested several alternative strategies for determining such significant tokens, in particular:

- *Token classification*: selecting influential tokens based on the ratio of positive/negative occurrences: $r^{\pm}(c, w_i) = \frac{|\Phi^{\pm}(c,w_i)|}{|\Phi(c,w_i)|}$ and comparing it with a threshold θ. Thus, a token w_i will be considered positive ($F_{inf}(w_i) = 1$) if $r^+(c, w_i) > \theta$, negative ($F_{inf}(w_i) = -1$) if $r^+(c, w_i) < 1 - \theta$ or not significant ($F_{inf}(w_i) = 0$) otherwise.
- *Token scoring*: instead of assigning discrete labels to tokens, each token gets assigned a real-valued score $F_{inf}(w_i) = r^+(c, w_i)$.
- *Weighted token scoring*: each token gets weighted by the number of positive/negative occurrences $F^+_{inf}(w_i) = |\Phi^+|$ and $F^-_{w_i} = |\Phi^-|$ rather than the ratio, so that more frequent tokens are valued higher.

Note that we only take into account the sign of each relevance score ϕ rather than its absolute value. Since the values of relevance scores are scaled by the final prediction $f_c(\cdot)$, they are not comparable between different examples $\mathbf{X_j}$.

4.3 Grounding influential tokens with a knowledge graph

After determining the set of influential input tokens, we select only the semantically meaningful ones that reference domain entities represented in a *knowledge graph* containing relevant domain knowledge. We assume the following definition for a knowledge graph from the Resource Description Framework (RDF): A knowledge graph G is a finite set of triples $(s, p, o) \in (E \cup B) \times R \times (E \cup B \cup L)$, where E is a set of URI resources (entities), B denotes blank nodes, R is a set of URI relations, and L is a set of literals.

Our method therefore requires a mapping between a subset of tokens $\mathcal{V}_\mathcal{G} \subset \mathcal{V}$ and their representative named entities $e_i \in E$ from G: e.g., between the token "Belluno" and Wikidata ID "Q6558". If there is no available mapping between text tokens and graph entities, a *named entity recognition* task has to be performed separately. As a result of this stage, we move from the influence functions of textual tokens $F_{inf}(w_i)$ to influence functions $F_{inf}(e_i)$ for influential domain entities, each one uniquely denoted by a URI.

5 RULE MINING

At the next stage, our algorithm takes the set of discovered significant domain entities and their associated relevance scores and tries to describe this set in terms of the domain knowledge represented in the knowledge graph. Our use case represents an example of *associative classification*, which in turn is a special case of the *association rule mining (ARM)* task where the consequent of an association rule is constrained by class label attributes.

5.1 Background

Formally, the data mining context of ARM is represented by a transaction database $D = (O, I, R)$, where O and I are finite sets of transactions and items respectively and $R \subseteq O \times I$ is a binary relation between O and I. D is arranged as a set of transactions where each transaction $t \in O$ is a set of items $y \in I$ (itemset). Thus, $(t, y) \in R$ means that the item y occurs in the transaction t, and a transaction t is said to contain an itemset Y if $\forall y \in Y : (t, y) \in R$. For each itemset $Y \subseteq I$ its *support* $sup(Y)$ is the number of transactions in which the itemset occurs, i.e. $sup(Y) = |T|$, where $T = \{t \in O | \forall y \in Y : (t, y) \in R\}$. The aim of ARM is to discover intrinsic relations between itemsets. An association rule is an implication of

the form $X \rightarrow Y$, where $X \subseteq I$ and $Y \subseteq I$ are the antecedent and the consequent of the rule and $X \cap Y = \emptyset$. For $X \rightarrow Y$ its support $sup(X \rightarrow Y)$ is defined as $sup(X \rightarrow Y) = |\{t \in O | X \subseteq t, Y \subseteq t\}|$ and confidence is the conditional probability that $Y \subseteq t$ given that $X \subseteq t$, i.e. $conf(X \rightarrow Y) = \frac{sup(X \rightarrow Y)}{sup(X)}$.

In the case of *explanation rules*, our set of transactions O consists of all IRI instances in the knowledge graph that are referenced in the text $\mathcal{I}_{text}$. For each $t \in \mathcal{I}_{text}$, its itemset $I(t)$ includes:

- All statements in the knowledge graph G containing t as a subject or an object: $\{(s,p)|(s,p,t) \in G\} \cup \{(p,o)|(t,p,o) \in G\}$.
- A class label Cls stating whether t has a positive or negative influence. $Cls(x)$ can take one of $\{Pos(X), Neg(X)\}$.

Each explanation rule in our case has the format $Y(x) \rightarrow Pos(x)$ or $Y(x) \rightarrow Neg(x)$, holding the class label as a consequent, while the antecedent represents a conjunction $Y(x) = \bigwedge_i (s_i, p_i, x) \wedge \bigwedge_j (x, p_j, o_j)$: e.g., a rule covering Italian toponyms from our example can take the form $City(x) \wedge locatedIn(x, Italy) \rightarrow Neg(x)$.

5.2 Mining classification rules

The classical Apriori algorithm [2] for mining rules proceeds in two steps:

- *Finding frequent itemsets* $Y(x)$ that pass the minimum support threshold $sup(Y(x)) \geq sup_{min}$. Frequent itemsets are constructed iteratively: Frequent itemsets of length $k+1$ are constructed by adding operands to frequent itemsets of length k, starting from $k = 1$.
- *Generating rules* $Y(x) \rightarrow Cls(x)$ from frequent itemsets. Valid rules are selected based on the minimum confidence threshold $conf(Y(x) \rightarrow Cls(x)) \geq conf_{min}$.

Depending on the influence measurement strategy discussed in Section 4, we used different confidence measures, which generalize from the token significance metrics:

- *Token classification (TC)*: this is a straightforward application of the confidence measure to evaluate a rule using the assigned token classification labels:

$$conf_{TC}(Y(x) \rightarrow Pos(x)) = \frac{|\{d_i | d_i \in Y(x), F_{inf}(d_i) = 1\}|}{|\{d_i | d_i \in Y(x)\}|}$$

- *Token scoring (TS)*: here a real-valued function $F_{inf}(d_i)$ is taken into account

$$conf_{TS}(Y(x) \rightarrow Pos(x)) = \frac{\sum_{Y(x)} F_{inf}(d_i)}{|Y(x)|}$$

- *Weighted token scoring (WTS)*:

$$conf_{WTS}(Y(x) \rightarrow Pos(x)) = \frac{\sum_{Y(x)} F^{+}_{inf}(d_i)}{\sum_{Y(x)} (F^{+}_{inf}(d_i) + F^{-}_{inf}(d_i))}$$

Note that these metrics behave differently in cases where tokens within the category vary greatly with respect to their frequency. For example, in the tweet political leaning classification use case, we have a candidate rule

$$(x, occupation, politician) \wedge (x, country, Canada) \rightarrow Democratic(x)$$

with 5 items satisfying the antecedent criterion. For 4 of them the assumption holds, i.e., they were mostly referenced in pro-democratic tweets, but the 5th one was Ted Cruz who held double citizenship until 2014 and (a) was referenced in primarily pro-republican tweets and (b) had significantly more mentions in the training set than the other 4 combined. Thus, the rule would be selected as valid by the token classification and token scoring metrics, but with weighted token scoring an opposite rule would be created classifying Canadian politicians as evidence for pro-republican leaning. Interpretation of the rule would be different as well: a newly encountered randomly selected Canadian politician in a tweet would more likely shift the model decision towards pro-Democratic, but a random mention of some Canadian politician in a new tweet would more likely point to a pro-Republican tweet since this would likely be a mention of Ted Cruz. For more robust rules that would be valid under both interpretations, we introduced *hybrid metric (H)*:

$$conf_H(Y(x) \rightarrow Pos(x)) = min(conf_{TC}, conf_{WTS})$$

5.3 Post-processing mined rules

A well-known issue with rule mining algorithms is the potentially large set of produced rules, which makes it difficult to interpret and utilize by a human user. To deal with this, we adapted the approach described in [27] targeted at classification rules and based on the notion of closed rule sets: closed sets of rules covering the same set of transactions. If for two rules, r_1 and r_2, and their supporting sets of transactions, T_1 and T_2, the relation $T_2 \subset T_1$ holds, then r_1 is called a cover-rule of r_2. The rule r_2 in this situation becomes redundant and can be safely discarded. Each closed rule set can then be replaced by a subset of its cover-rules which jointly cover all its transactions.

6 EVALUATION AND DISCUSSION

Existing benchmark datasets created to evaluate the neural network explanation methods on NLP tasks [4, 5, 34] are not suitable in our case, because they do not reference domain information modelled by a knowledge graph. For this reason, we selected our own evaluation datasets using two criteria: (i) *availability of pre-existing mappings* with Wikidata (Wikipedia links and Twitter account references) to avoid the need for a named entity recognition step, potentially introducing errors and (ii) *non-triviality of prediction*: the target variable should not be directly present in the text itself, which otherwise would lead to high performance of the model, but trivial explanation rules. Based on these, we selected the following four benchmarks:

- *Wikipedia abstracts*: As examples, we selected abstracts of Wikipedia articles describing Wikidata instances of the same type. The Wikipedia links in the text were used to map the tokens in the text to Wikidata and the input word embeddings (word2vec [30]) were trained using the complete Wikipedia corpus with explicit references of Wikidata IRIs. In this way, we obtained a common vector space containing Wikidata IRIs together with common words.
  - *Painters*: The task involves deciding whether painters who died before 1700 are likely to have their paintings in one of

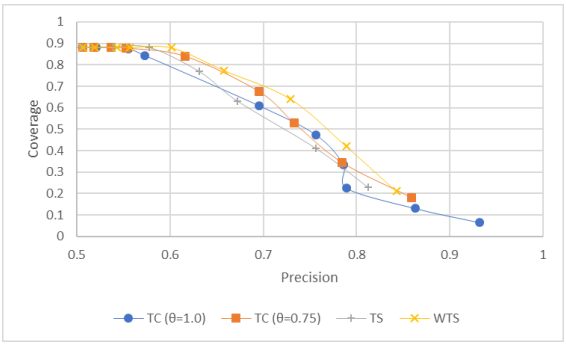

Figure 3: Aggregated coverage and precision achieved by mined rules with different metrics.

the 10 most famous European museum collections[1]. The dataset contains 3943 abstracts.

- *Movie rating*: The task involves deciding whether a movie is likely to receive a higher than median rating on IMDB. The selected dataset contains 8087 abstracts.
- *Movie popularity*: The task uses the same textual data as the previous one, but focuses on deciding whether a movie is likely to receive more than the median number of votes on IMDB.

- *Political leaning*: For this test, we reused a pre-trained LSTM model published by [36], which classified tweets based on their political leaning (pro-democratic vs pro-republican), and the publicly available test set of 10,000 tweets. We used the referenced Twitter accounts to map them to corresponding Wikidata entities.

For the three Wikipedia datasets, we trained the models ourselves, splitting the datasets equally into the train and test sets. In order to evaluate the dependency of mined rulesets on the network architecture, for each datasets we trained two models: (i) Bi-LSTM and (b) two-layer CNN. The details of all models are provided in Appendix A.

6.1 Accuracy of the learned rules

We evaluated the quality of rules mined in the training set by calculating their prediction accuracy on the test set. We selected all entities covered by the mined rules that satisfied the rules' antecedent and were mentioned in the test set. The first measure, "impact accuracy" (Acc_{imp}), checked whether for each occurrence of the entity its relevance score corresponded with the rule prediction: i.e., for all occurrences of entities x satisfying rules $Y(x) \rightarrow Pos(x)$ and $Y(x) \rightarrow Neg(x)$, calculating the ratio of cases where the sign of the relevance score was correctly predicted.

$$Acc_{imp} = \frac{|\bigcup\{t \in M^{+}(e_i)|\phi(t) > 0\} \cup \bigcup\{t \in M^{-}(e_i)|\phi(t) < 0\}|}{|\bigcup M(e_i)|}$$

As the second metrics "overall prediction accuracy" ($Acc_{overall}$), we measured the ratio of coincidences with the overall model prediction: i.e., for all entity occurrences covered by rules calculating the ratio of cases where the model prediction for the whole text coincided with the rule prediction for the mentioned entity.

$$Acc_{overall} = \frac{|\bigcup\{t \in M^{+}(e_i)|f(\mathbf{X_j}) > 0\} \cup \bigcup\{t \in M^{-}(e_i)|f(\mathbf{X_j}) < 0\}|}{|\bigcup M(e_i)|}$$

As we can see in Table 1, in all cases a substantial part of the model's behaviour can be explained away with the semantic preferences captured by the mined rules: with the most accurate "hybrid" strategy, for the subset covered by the rules $Acc_{overall}$ was usually close to the accuracy of the model itself. In general, the *TC* and *H* strategies achieved higher accuracy than both *TS* and *WTS*. Accounting for non-influential tokens that receive highly varying influence scores in different instances helps to produce more robust rules.

As expected, most of the mined rules reflected common sense knowledge about the corresponding domains. For instance, in the Painters use case, the Dutch or Flemish origin, professional painter's career, and elements of Christian art were found to be positive factors leading to such rules like

$$human(x) \wedge citizen(x, SouthernNetherlands) \wedge memberOf(x, AntwerpGuildofStLuke) \rightarrow Pos(x)$$

or

$$painting(x) \wedge depicts(x, VirginMary) \rightarrow Pos(x)$$

In turn, non-Western origin (e.g., China, Korea, or Russia) or too early time period (Classical Antiquity) usually had a negative impact (Figure 4).

But some preferences were non-intuitive: e.g., a bias against various Italian toponyms. When analyzing the dataset, two complementing reasons emerged: (i) a large number of Italian artists created frescoes rather than oil paintings, which could be moved to museums, and (ii) many Italian painters from small cities did not join one of the famous communities (e.g., in Florence or Rome) and did not acquire global fame, although their works survived in local museums. Such counter-intuitive biases can be used to resample the training data and fine-tune the model behaviour.

Interestingly, different factors were found to be at play when classifying movies by rating or by popularity. For the rating, the most significant positive factors were various awards, while negative ones included certain genres or character types (e.g., zombie movies or superheroes). For the popularity, on the other hand, the positive factors were generally related to recent cultural trends (e.g., association with the Marvel universe or having Brad Pitt as a cast member), while the negative ones had to do with old age (e.g., black and white colour or Clark Gable as an actor). Finally, for the political use case, the referenced twitter accounts were found to have greater influence than the content of the tweet itself: e.g., referencing a military officer or a Methodist church member usually shifted the decision towards "pro-republican", while mentioning a Hollywood actor or a CNN journalist were signs of "pro-democratic".

Figure 3 shows a trade-off between the precision of the mined set of rules and the coverage (percentage of entities which support at least one rule) achieved with different thresholds. The *TC* strategy generally results in higher precision rules, while the *WTS* produces more generic rule sets covering a larger proportion of entities.

[1]Louvre, Hermitage, Prado, Uffizi, Vatican, London National Gallery, Rijksmuseum, museums in Vienna, Berlin, and Munich.

Use case	Model type	Impact accuracy Acc_{imp}				Overall prediction accuracy $Acc_{overall}$				Coverage				Model accuracy
		TC	TS	WTS	H	TC	TS	WTS	H	TC	TS	WTS	H	
Painters	Conv (all)	0.80	0.69	0.73	0.86	0.69	0.49	0.70	0.75	0.45	0.96	0.96	0.43	0.71
	Conv (unseen)	0.79	0.71	0.51	0.80	0.72	0.53	0.47	0.72					
	LSTM (all)	0.85	0.74	0.77	0.86	0.73	0.55	0.68	0.74	0.59	0.97	0.96	0.58	0.71
	LSTM (unseen)	0.81	0.73	0.74	0.81	0.70	0.59	0.60	0.70					
Movies (rating)	Conv (all)	0.83	0.66	0.67	0.83	0.71	0.46	0.47	0.71	0.45	0.98	0.98	0.44	0.68
	Conv (unseen)	0.69	0.63	0.63	0.68	0.55	0.44	0.44	0.54					
	LSTM (all)	0.86	0.64	0.66	0.84	0.73	0.46	0.48	0.71	0.50	0.99	0.99	0.49	0.68
	LSTM (unseen)	0.77	0.62	0.62	0.74	0.61	0.44	0.44	0.61					
Movies (popularity)	Conv (all)	0.77	0.67	0.66	0.77	0.69	0.54	0.54	0.69	0.79	0.86	0.99	0.78	0.79
	Conv (unseen)	0.64	0.61	0.56	0.64	0.57	0.48	0.45	0.57					
	LSTM (all)	0.83	0.67	0.67	0.84	0.72	0.52	0.52	0.73	0.79	0.99	0.99	0.78	0.79
	LSTM (unseen)	0.75	0.63	0.58	0.75	0.66	0.47	0.44	0.66					
Political tweets	LSTM (all)	0.66	0.64	0.73	0.90	0.63	0.61	0.70	0.86	0.35	0.38	0.42	0.35	0.90
	LSTM (unseen)	0.70	0.62	0.51	0.70	0.65	0.57	0.47	0.66					

Table 1: Accuracy of learned classification rules in (a) predicting the impact of a single entity (positive/negative) and (b) predicting the overall model decision over the whole text and (c) percentage of entities covered by the final ruleset. All metrics were tested with the same $conf_{min} = 0.75$, for TC and H the parameter $\theta = 1.0$ was used.

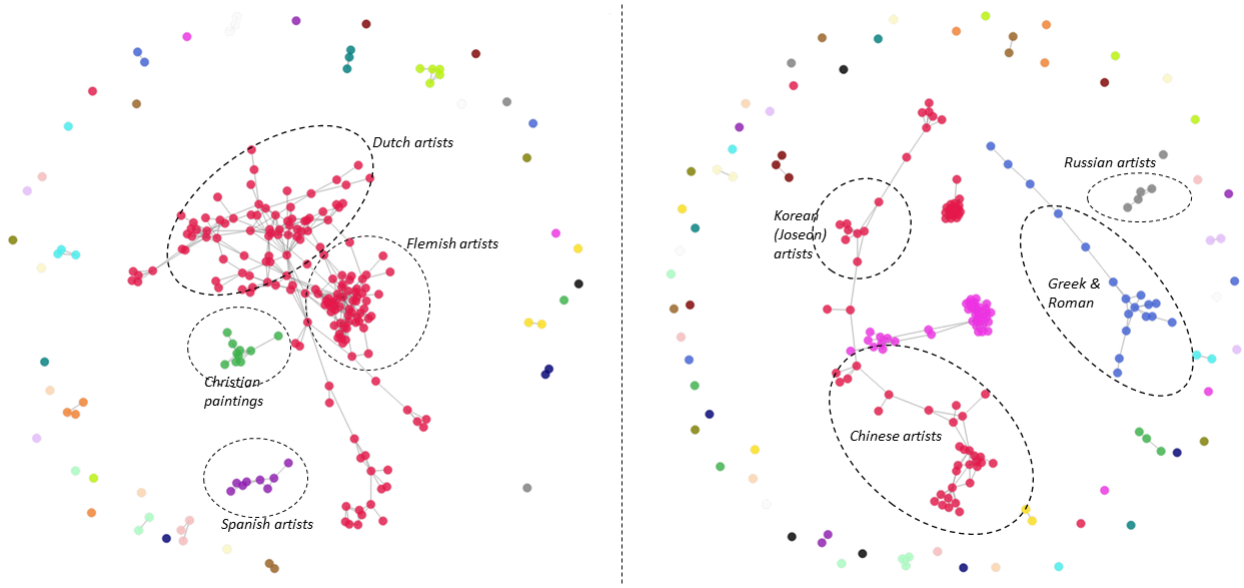

Figure 4: Visualized clusters of positive (left) and negative (right) rules composed for the convolutional network applied to the Painters use case. Rules represent nodes of the graph, while edges reflect overlaps between support sets. Connected components represent closed rule sets.

6.2 Impact of network architecture differences

In order to validate that the method is generic enough to be used in combinations with different network architectures, it is important to verify to which extent its results are adequately capturing correlations existing in the data and learned by the model rather than representing uninformative fluctuations dependent on the choice of the model architecture. For this reason, in the Wikipedia abstracts experiments we tested our method with two different neural network architectures for the same task: the LSTM and the CNN. As shown in Table 2, the rules mined after analyzing the influential tokens for both networks largely reflect the same preferences: the sets of entities covered by rules for both network types are strongly overlapping.

The results also show the effects of highly selective metrics (TC and H): due to high dependency on the separation between influential and non-influential tokens, the results are more sensitive to the specific network architecture, the overlap between corresponding entity sets covered by rules is generally lower, but the positive and negative token sets are well-separated. On the other hand, TS

	Class	TC	TS	WTS	H
Pain-ters	Pos	0.76 (0.01)	0.90 (0.18)	0.88 (0.20)	0.84 (0.01)
	Neg	0.83 (0.01)	0.94 (0.06)	0.96 (0.09)	0.91 (0.01)
Movies (rating)	Pos	0.54 (0.03)	0.75 (0.34)	0.78 (0.30)	0.53 (0.03)
	Neg	0.52 (0.02)	0.81 (0.40)	0.83 (0.36)	0.51 (0.01)
Movies (pop.)	Pos	0.70 (0.14)	0.90 (0.48)	0.94 (0.64)	0.70 (0.14)
	Neg	0.84 (0.03)	0.89 (0.30)	0.89 (0.49)	0.84 (0.03)

Table 2: Overlap of entities covered by rules created for different network architectures (CNN and LSTM) for the same task. The table shows overlap distances between Ω^+_{CNN} and Ω^+_{LSTM}, in brackets the overlap distances with the rules of the opposite class are given (Ω^+_{CNN} and Ω^-_{LSTM}).

Use case	Class	Avg. score	Avg. per-rule score	r_{Pos}
Painters	Pos	0.47	0.55	0.47
	Neg	0.10	0.12	0.05
Movies (rating)	Pos	0.68	0.75	0.77
	Neg	0.35	0.34	0.23
Movies (popularity)	Pos	0.34	0.72	0.31
	Neg	0.05	0.08	0.02

Table 3: Model predictions on entities covered by rules.

and WTS result in greater correlation between the corresponding entity sets, but the rule sets are so generic that there is high overlap between opposite classes to the degree which makes the rules non-informative. This is corroborated by the lower prediction scores achieved with these metrics (Table 1).

6.3 Validity outside of the training context

In practical scenarios, the trained model usually has to be applied to new unseen data instances, which can follow different distributions from the training and test sets, which in turn makes the model more error-prone in comparison with its test set performance. In such cases, mined rules can be valuable for diagnosing misclassifications, but only if they are generic enough to hold outside of the training corpus. To check this, we applied the model to a different set of text examples outside of the intended context of the model, but which would contain internally the semantic preferences captured by the rules. For this test, we selected all positive and negative entities covered by mined rules and applied the model to the Wikipedia abstracts of these entities (excluding those included in the training or test set). The model predictions themselves for such examples are not meaningful (e.g., whether the city of Perugia or the character Virgin Mary would have their paintings in a major gallery if they were painters). However, we would expect that the Wikipedia abstracts of entities predicted to have positive impact would themselves be more likely to be classified positively by the model. As we can see in Table 3, for all datasets this expectation holds, which can be seen as an indication that the semantic preferences captured by the model are sufficiently generic to explain the model's behaviour even outside the original learning context.

6.4 Viewing and utilizing mined rules

As said in section 2, the purpose of the mined explanation rules is to help the user understand the underlying preferences of the trained deep learning model. Reducing the set of mined rules and presenting them in a human-readable form makes this task easier for the user. Removing redundant rules using closed rule sets (Section 5.3) helps towards achieving the first goal. Visualizing closed rule sets as a graph (Figure 4) makes it easier to explore the rule sets and identify semantic categories which correspond to influential model parameters that can shift model's decision either way. As said in Section 6.1, majority of mined rules in our experiments are consistent with common-sense domain knowledge. However, two types of rules can be of special interest:

- Counter-intuitive rules that contradict common sense, e.g., like a counter-intuitive rule that Italian origin somehow reduces the value of painters' works. Such rules should motivate further analysis of the training dataset and the learning process. This analysis can point to an impact of the training data sampling (like in the case with Italian art) and, in the worst case, to spurious Clever Hans-type biases.
- Rules that make use of undesired input data features, which make the model itself violate the requirements: e.g., relying on race or gender terms.

In both cases, discovering such rules can require modifications in order to improve the model, for instance, resampling the training set to avoid biases or pre-processing the training set to remove undesired features.

7 CONCLUSION AND FUTURE WORK

We have proposed a method for mining explanation rules to capture consistent semantic preferences of neural NLP models. Our experiments have shown that generalizing from atomic relevance scores using association rule mining creates human-interpretable rules that both explain the model behaviour to a large extent and are robust enough to be useful when the model is used outside of the training context. While these results are promising, they cannot provide a complete understanding of a trained model as many important factors are left out. Based on our observations, we consider two directions for future work:

- Using knowledge graphs and rule mining to improve interpretability of intermediate elements of the model: e.g., attention weights, convolutional filters, and feature embeddings. Mined rules can provide "semantic views" over the corresponding vector spaces, highlighting regions that are significant to the model's decision and on the other hand can be mapped to the semantic space.
- Extending the method to capture other types of features significant for the model: e.g., syntactic patterns, elements of style, and sentiment.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, DC, USA, May 26-28, 1993*. 207–216. https://doi.org/10.1145/170035.170072

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*. 487–499. http://www.vldb.org/conf/1994/P487.PDF

[3] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for Deep Neural Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. https://openreview.net/forum?id=Sy21R9JAW

[4] Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining Recurrent Neural Network Predictions in Sentiment Analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Association for Computational Linguistics, Copenhagen, Denmark, 159–168. https://doi.org/10.18653/v1/W17-5221

[5] Leila Arras, Ahmed Osman, Klaus-Robert Müller, and Wojciech Samek. 2019. Evaluating Recurrent Neural Network Explanations. *CoRR* abs/1904.11829 (2019). arXiv:1904.11829 http://arxiv.org/abs/1904.11829

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE* 10, 7 (07 2015), 1–46. https://doi.org/10.1371/journal.pone.0130140

[7] 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*. http://arxiv.org/abs/1409.0473

[8] Roberto J. Bayardo. 1998. Efficiently Mining Long Patterns from Databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. 85–93. https://doi.org/10.1145/276304.276313

[9] Abdelaziz Berrado and George C. Runger. 2007. Using metarules to organize and group discovered association rules. *Data Min. Knowl. Discov.* 14, 3 (2007), 409–431. https://doi.org/10.1007/s10618-006-0062-6

[10] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. 2016. DL-Learner - A framework for inductive learning on the Semantic Web. *J. Web Semant.* 39 (2016), 15–24. https://doi.org/10.1016/j.websem.2016.06.001

[11] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan L. Boyd-Graber. 2018. Pathologies of Neural Models Make Interpretation Difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. 3719–3728. https://www.aclweb.org/anthology/D18-1407/

[12] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.* 24, 6 (2015), 707–730. https://doi.org/10.1007/s00778-015-0394-1

[13] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*. 413–422. https://doi.org/10.1145/2488388.2488425

[14] Amirata Ghorbani, Abubakar Abid, and James Y. Zou. 2019. Interpretation of Neural Networks Is Fragile. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. 3681–3688. https://doi.org/10.1609/aaai.v33i01.33013681

[15] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. 2019. Towards Automatic Concept-based Explanations. *CoRR* abs/1902.03129 (2019). arXiv:stat.ML/1902.03129 http://arxiv.org/abs/1902.03129

[16] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. 2000. Algorithms for Association Rule Mining - A General Survey and Comparison. *SIGKDD Explorations* 2, 1 (2000), 58–64. https://doi.org/10.1145/360402.360421

[17] Pascal Hitzler, Federico Bianchi, Monireh Ebrahimi, and Md. Kamruzzaman Sarker. 2020. Neural-symbolic integration and the Semantic Web. *Semantic Web* 11, 1 (2020), 3–11. https://doi.org/10.3233/SW-190368

[18] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding Convolutional Neural Networks for Text Classification. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*. 56–65. https://www.aclweb.org/anthology/W18-5408/

[19] Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. 3543–3556.

[20] Ákos Kádár, Grzegorz Chrupala, and Afra Alishahi. 2017. Representation of Linguistic Form and Function in Recurrent Neural Networks. *Computational Linguistics* 43, 4 (2017). https://doi.org/10.1162/COLI_a_00300

[21] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. 2673–2682. http://proceedings.mlr.press/v80/kim18d.html

[22] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications* 10, 1 (2019), 1096. https://doi.org/10.1038/s41467-019-08987-4

[23] Freddy Lécué. 2020. On the role of knowledge graphs in explainable AI. *Semantic Web* 11, 1 (2020), 41–51. https://doi.org/10.3233/SW-190374

[24] Philippe Lenca, Patrick Meyer, Benoît Vaillant, and Stéphane Lallich. 2008. On selecting interestingness measures for association rules: User oriented description and multiple criteria decision aid. *European Journal of Operational Research* 184, 2 (2008), 610–626. https://doi.org/10.1016/j.ejor.2006.10.059

[25] Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding Neural Networks through Representation Erasure. *CoRR* abs/1612.08220 (2016). arXiv:1612.08220 http://arxiv.org/abs/1612.08220

[26] Zachary C. Lipton. 2018. The Mythos of Model Interpretability. *ACM Queue* 16, 3 (2018), 30. https://doi.org/10.1145/3236386.3241340

[27] Huawen Liu, Jigui Sun, and Huijie Zhang. 2009. Post-processing of associative classification rules using closed sets. *Expert Syst. Appl.* 36, 3 (2009), 6659–6667. https://doi.org/10.1016/j.eswa.2008.08.046

[28] Scott M Lundberg and Su-In Lee. 2017. 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., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[29] Claudia Marinica, Fabrice Guillet, and Henri Briand. 2008. Post-Processing of Discovered Association Rules Using Ontologies. In *Workshops Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. 126–133. https://doi.org/10.1109/ICDMW.2008.87

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[31] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for Interpreting and Understanding Deep Neural Networks. *Digital Signal Processing* 73 (2018), 1–15. https://doi.org/10.1016/j.dsp.2017.10.011

[32] Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter A. Flach, Katsumi Inoue, and Ashwin Srinivasan. 2012. ILP turns 20 - Biography and future challenges. *Machine Learning* 86, 1 (2012), 3–23. https://doi.org/10.1007/s10994-011-5259-2

[33] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. 1999. Efficient Mining of Association Rules Using Closed Itemset Lattices. *Inf. Syst.* 24, 1 (1999), 25–46. https://doi.org/10.1016/S0306-4379(99)00003-4

[34] Nina Pörner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*. 340–350. https://doi.org/10.18653/v1/P18-1032

[35] Jedrzej Potoniec, Piotr Jakubowski, and Agnieszka Lawrynowicz. 2017. Swift Linked Data Miner: Mining OWL 2 EL class expressions directly from online RDF datasets. *J. Web Semant.* 46-47 (2017), 31–50. https://doi.org/10.1016/j.websem.2017.08.001

[36] Adithya Rao and Nemanja Spasojevic. 2016. Actionable and Political Text Classification using Word Embeddings and LSTM. In *Proceedings of the Fifth International Workshop on Issues of Sentiment Discovery and Opinion Mining, WISDOM 2016, San Francisco, CA, USA, August 14, 2016*.

[37] 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*. 1135–1144. https://doi.org/10.1145/2939672.2939778

[38] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2018. Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services* 1, 1 (2018), 39–48. https://www.itu.int/en/journal/001/Pages/05.aspx

[39] Arne Seeliger, Matthias Pfaff, and Helmut Krcmar. 2019. Semantic Web Technologies for Explainable Machine Learning Models: A Literature Review. In *Joint Proceedings of the 6th International Workshop on Dataset PROFILing and Search & the 1st Workshop on Semantic Explainability co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 27, 2019 (CEUR Workshop Proceedings)*, Vol. 2465. CEUR-WS.org, 30–45. http://ceur-ws.org/Vol-2465/semex_paper1.pdf

[40] Sofia Serrano and Noah A. Smith. 2019. Is Attention Interpretable?. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019,*

Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers. 2931–2951. https://www.aclweb.org/anthology/P19-1282/

[41] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 3145–3153. http://proceedings.mlr.press/v70/shrikumar17a.html

[42] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. http://arxiv.org/abs/1312.6034

[43] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 3319–3328. http://proceedings.mlr.press/v70/sundararajan17a.html

[44] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. 1995. Pruning and grouping discovered association rules. In *ECML'95 MLnet workshop on statistics, machine learning, and knowledge discovery in databases*. 47–52.

[45] Daniel S. Weld and Gagan Bansal. 2019. The challenge of crafting intelligible intelligence. *Commun. ACM* 62, 6 (2019), 70–79. https://doi.org/10.1145/3282486

[46] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*. 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

[47] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. 2017. Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. https://openreview.net/forum?id=BJ5UeU9xx

A MODELS USED IN EXPERIMENTS

Experiments described in Section 6 included two parts: tests on Wikipedia abstracts and tweets classification. The Wikipedia abstracts tests involved two datasets: abstracts of painters (3943 abstracts) and movies (8087 abstracts). These datasets were in turn used in three classification tasks: (i) classifying painters represented in a major European museum collection, (ii) detecting movies with higher than the median IMDB rating, and (iii) detecting movies with higher than the median number of IMDB votes. For each of these tasks we employed two types of models: convolutional (CNN) and recurrent bi-directional long short-term memory (Bi-LSTM). These datasets as well as the trained models we used in our experiments are available online[2]. Finally, for the tweets political leaning classification [36] we reused a pre-trained LSTM model and a publicly available test dataset of 10000 tweets provided by the authors[3]. All use cases involve binary classification, so the output of all networks constitutes a single neuron. Sigmoid activation is applied to normalize the output and get the final prediction.

A.1 CNN for Wikipedia abstract classification

This network was used for 3 Wikipedia use cases (Painters, Movies (rating), and Movies (popularity). The network contains the following layers:

- *Input layer*: Takes as input a sequence of word IDs from the vocabulary. (Maximum input size: 500)
- *Word embedding layer*: Transforms each word ID into an embedding vector. Embedding vectors (word2vec) for the Wikipedia experiments were trained separately using the whole Wikipedia corpus. (Dimensionality: 50)
- *Convolutional layer*: A layer of filters applied to subsequences of the input. (Number of filters: 50, Window size: 11)
- *Max pooling*: Downscales the convolutional layer output. (Pool Size: 10)
- *Convolutional layer*: A second layer of filters. (Number of filters: 50, Window size: 5)
- *Max pooling*: Applies to the whole output of each convolutional filter from the previous layer (Pool Size: 45)
- *Fully-connected layer*: Combines output from all filters and produces a single output. (Activation: Sigmoid)

A.2 Bi-LSTM for Wikipedia abstract classification

This network is an alternative architecture to the convolutional network used in comparative tests. It has the following structure, sharing its input and embedding layers with the CNN from the previous example:

- *Input layer*: Same as CNN. (Maximum input size: 500)
- *Word embedding layer*: Same as CNN. (Dimensionality: 50)
- *Bi-LSTM layer*: This layer includes two sets of LSTM neurons: one for the forward and one for the backward pass and returns the values of hidden neurons for each input in the sequences. The outputs get concatenated (Hidden neurons: 100x2, Output sequence size: 500)
- *Global max pooling*: Applies for all values of each hidden neuron in the sequence. (Pool Size: 500)
- *Fully-connected layer*: Combines output from all filters and produces a single output. (Activation: Sigmoid)

As reference for the SHAP/DeepExplainer explanations for both models trained on Wikipedia abstracts we used the average of the first 500 instances from the respective training sets. The trained models can be found online in the HDF5 format together with the datasets.

A.3 LSTM for tweets political leaning classification

This is a pre-trained model described in [36], which consists of the following four layers:

- *Input layer*: Takes as input a sequence of word IDs from the vocabulary. (Maximum input size: 40)
- *Word embedding layer*: Transforms each word ID into an embedding vector. (Embedding dimensions: 128)
- *LSTM layer*: Forward-processing LSTM layer. (Hidden neurons: 32)
- *Fully-connected layer*: Combines output from all LSTM neurons and produces a single output. (Activation: Sigmoid)

[2] https://figshare.com/s/5a3b833ea7b08dfdbe14

[3] https://github.com/klout/opendata/tree/master/political_leaning