Understanding Text Classification Data and Models Using Aggregated Input Salience

Sebastian Ebert, Alice Shoshana Jakobovits, Katja Filippova

Google Research
{eberts,jakobovits,katjaf}@google.com

Abstract

Realizing when a model is right for a wrong reason is not trivial and requires a significant effort by model developers. In some cases an input salience method, which highlights the most important parts of the input, may reveal problematic reasoning. But scrutinizing highlights over many data instances is tedious and often infeasible. Furthermore, analyzing examples in isolation does not reveal general patterns in the data or in the model's behavior. In this paper we aim to address these issues and go from understanding single examples to understanding entire datasets and models. The methodology we propose is based on aggregated salience maps, to which we apply clustering, nearest neighbor search and visualizations. Using this methodology we address multiple distinct but common model developer needs by showing how problematic data and model behavior can be identified and explained – a necessary first step for improving the model.

Warning: Due to the usage of a toxicity dataset this paper contains content that may be offensive or upsetting.

1 Introduction

Deploying ML-powered models requires confidence in their reliability. While strong performance on an evaluation set is a prerequisite, it is not sufficient since it may hide poor generalization patterns. It is the responsibility of the model developer to analyze the model and proactively search for problematic patterns, discover its sensitivities, or uncover sources of erroneous predictions.

Over the past few years, the explainable AI community has developed many methods and created platforms that support developers in debugging their models (Nori et al. 2019; Tenney et al. 2020, inter alia). But most efforts have been devoted to analyzing *single* predictions, often with the help of input salience (Li et al. 2016; Montavon et al. 2019, inter alia) or training data attribution techniques (Koh and Liang 2017; Pruthi et al. 2020). Though already somewhat useful (Lertvittayakumjorn and Toni 2021a), single-point explanations may lead the developer to discover false generalizations if the analyzed examples are unrepresentative. For instance, when predicting a contradiction in Natural Language Inference tasks (Bowman et al. 2015), does a very high salience weight on the word '*not*', imply that the model learned an overly simplistic pattern and consistently ignores the context when '*not*' is present? Understanding whether shallow reasoning is applied systematically would require going through a large number of inputs, which is tedious and time consuming, ren-

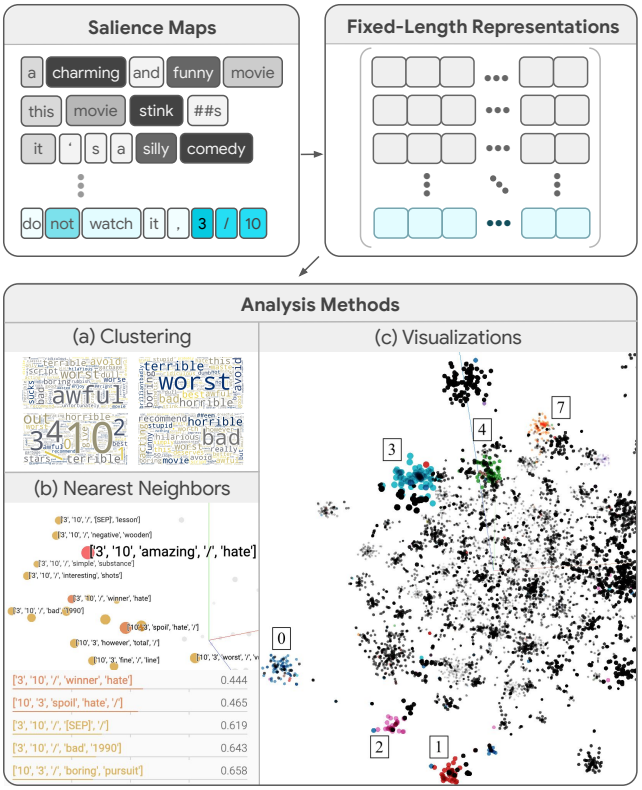

Figure 1: An overview of the proposed method: from raw input **salience maps**, compute **fixed-length representations**, and apply **analysis methods** such as clustering, nearest neighbors search and visualizations. Showing some examples from the IMDB dataset, the high-salience tokens '*3*', '*/*' and '*10*' in the last example's salience map are reflected in the analyses: (a) Word clouds computed from four clusters show that one cluster is dominated by number tokens. (b) Nearest neighbors of the example reveal more examples with similar high-salience token combinations. (c) A t-SNE visualization of the IMDB test set, where the turquoise point cloud labeled '*3*' contains all examples similar to the original.

dering it virtually impossible to discover patterns learned from the whole dataset. Conversely, patterns unintuitive for a human may be difficult to spot from individual examples even when the salience maps hints at its presence.

In order to gather stronger support that the model is "right for the right reasons" we propose to shift focus from single point analyses to systematic patterns. We propose to use standard input salience methods, convert the maps they produce into fixed-length representations, and analyze the resulting space using clustering, nearest neighbor search or interactive visualizations (Fig. 1). We use salience methods as opposed to corpus statistics (Gardner et al. 2021) because they point at what is important *for the model* (Bastings and Filippova 2020) and because not every feature-label correlation results in a spurious correlation picked by a model (Eisenstein 2022).

With this aggregated view we are able to surface shallow reasoning patterns (McCoy, Pavlick, and Linzen 2019; Rosenman, Jacovi, and Goldberg 2020) and data artifacts that the model is sensitive to (Gururangan et al. 2018; Geva, Goldberg, and Berant 2019). This can be a first step towards correcting or augmenting the data and improving the model. Furthermore, independently of whether a learned pattern poses a robustness threat, we show how the same techniques can help better explore and understand the characteristics of a dataset which we believe also falls under a developer's responsibility.

Addressing distinct but common developer needs, we make the following contributions:

1. In Sec. 5, we demonstrate how clustering salience maps helps identify patterns in a dataset which result in a spurious correlation (Geirhos et al. 2020) and reveal weaknesses of a dataset. Furthermore, we present a qualitative analysis enabled by clustering subsamples of the data which results in a more fine-grained characterization of the relationship between certain tokens and prediction than has been previously done.
2. In Sec. 6, we describe a procedure grounded in salience clusters which facilitates finding out what a model is sensitive to.
3. In Sec. 7 we show how nearest neighbor search can be used to explain predictions which appear puzzling at first sight.

2 Approach

Our technique is based on aggregating salience maps of text data and has three components: an input salience method (Sec. 2.1), a fixed-length representation derived from the salience maps (Sec. 2.2), and an analysis method that is applied to the representation space (Sec. 2.3).

2.1 Choice of Salience Method

An input salience method associates every input token with a weight, reflecting its relative importance for the model in making the prediction: tokens with the highest salience are the ones which contribute most to the final model decision. There are many ways to compute salience weights, the most common ones are based on model gradients (Bach et al. 2015a; Li et al. 2016; Denil, Demiraj, and de Freitas 2015, inter alia), attention (Bahdanau, Cho, and Bengio 2015, inter alia), perturbations (Ribeiro, Singh, and Guestrin 2016), or occlusions (Zeiler and Fergus 2014; Li, Monroe, and Jurafsky 2016). We follow recent findings of Bastings et al. (2022) and use Gradient L2 (Grad-L2) because it has been shown to be the most faithful for finding lexical shortcuts when using a BERT model. We also include results of two experiments using Integrated Gradients (Sundararajan, Taly, and Yan 2017) in the appendix (Sec. A.2). Though the merits of existing salience methods are still being debated, our approach is compatible with any method which assigns a weight to a token. The only prerequisite is that the salience method reliably identifies important tokens.

2.2 Fixed-Length Representations

We aim at aggregating data examples where the input is text of any length. However, the aggregation methods we use require a fixed-length input, which we derive as follows.

Salience-based Representations (S) We aim to create representations that contain information about what is most important to the model. As described above, importance is given by *raw salience maps*. We adopt the following ways of obtaining a fixed-length representations from them:

S1: Vocabulary vector with top-*k* Given a dataset $\mathcal{D}$ of input-label pairs (x, y) and a model vocabulary $\mathcal{V}$ with v_j denoting the j^{th} vocabulary item, we have $x = \{\ldots, (t_i, s_i), \ldots\}$ an example consisting of (token, salience) pairs. For a given x of length N, we compute a $|\mathcal{V}|$-sized representation vector, $\mathbf{r}$, with entries $r_j = s_i$, s.t. $\max_{i=1,\ldots,N}\{|s_i| : t_i = v_j\}$. This vector contains one salience value per vocabulary item v_j. The salience value is the one that has the highest absolute value among all the salience values for this vocabulary item v_j in x.

We further introduce the function $\text{top}(\mathbf{r}, k)$ which gives the indices of the largest k elements in $\mathbf{r}$. As we will explain later, we use $k = 5$ and $k = |\mathcal{V}|$. This leads to the first fixed-length representation we use:

$$\mathbf{r}^{vocab} = \begin{cases} r_j, & \text{if } j \in \text{top}(\mathbf{r}, k) \\ 0 & \text{else.} \end{cases}$$

$\mathbf{r}^{vocab}$ is a zero vector with only the top-k elements set to a weight value according to the salience method. All example representations are concatenated into a matrix $R^{\text{vocab}} \in \mathbb{R}^{|\mathcal{D}| \times |\mathcal{V}|}$.

S2: Embedding vector over top-*k* Word(piece) embeddings are a common way to incorporate token statistics into a representation. We define $R^{\text{emb}} \in \mathbb{R}^{|\mathcal{D}| \times d} = R^{\text{vocab}} \cdot E$, where $E \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the embedding matrix (embedding size d) of a fine-tuned BERT model. Please note that the choice of k also affects this representation.

Salience weights are not necessarily normalized and therefore not directly comparable across multiple examples. Therefore, we normalize R^{vocab} and R^{emb} row-wise to unit length.

Baseline Representations (B) To verify that the salience information is useful for representing examples, we experiment with the following standard text representation *baselines*:

B1: PMI weighted vector Bag-of-words representation weighted by the point-wise mutual information $\text{PMI}(v_j; \hat{y})$, where $\hat{y}$ is the predicted class of the example: $R^{\text{pmi}} \in \mathbb{R}^{|\mathcal{D}|\times|\mathcal{V}|}$. We use PMI rather than the common TF-IDF because PMI has information about the labels of the examples.

B2: Average embedding We use the embedding matrix from the fine-tuned BERT models to compute $\mathbf{r}^{\text{avg emb}} = \frac{1}{N}\sum_i E_{t_i}$, where N is the number of tokens in the example and E_{t_i} is the embedding of t_i. We again concatenate all example representations into $R^{\text{avg emb}} \in \mathbb{R}^{|\mathcal{D}|\times d}$.

B3: CLS-encoding This representation comes from fine-tuned BERT models. We use the embedding of the last layer of the [CLS] token of BERT when predicting every example. $R^{\text{cls}} \in \mathbb{R}^{|\mathcal{D}|\times d}$.

2.3 Analysis Methods

Once a fixed-length representation is computed for every data instance, we apply various aggregation methods to analyze the model and data in the aggregate. We use three common standard techniques: (a) clustering using k-means, (b) nearest neighbor search,[1] and (c) visual/qualitative analyses with t-SNE (Van der Maaten and Hinton 2008)[2]. All methods operate on the row vectors of R (e.g., k-means leads to clusters of examples).

3 Data and Model

We use BERT-based classifiers (Devlin et al. 2019) trained to achieve accuracy previously reported (please refer to Sec. A.1 for training details). If not otherwise specified, the number of top-k salient tokens is set to 5 in all experiments for consistency purposes. While in most experiments, we got comparable results with $k = |\mathcal{V}|$ (see Sec. A.2), we did see a benefit in using only a few top tokens in one case. We will motivate that choice in Sec. 7.

We analyze these three datasets:

Synthetic SST2 We use a synthetically-modified version of SST2 (Socher et al. 2013) to show the utility of aggregating input salience in a controlled setting in our first experiment (Sec. 4). We follow the procedure of Bastings et al. (2022) and create a modified SST2. 75% of the original data stay untouched. In the remaining 25% of the data we insert one of three special terms (COMMON, CLASS_0 or CLASS_1) with no label change. We sample another 25% of the original data (equally split between positive and negative classes) in which we insert a combination of two special terms, which deterministically set an example's label. We insert either COMMON and CLASS_0 or COMMON and CLASS_1, both at random positions. For example, *'worth the effort to watch .'* (originally positive) becomes *'worth* COMMON *the effort to* CLASS_0 *watch .'* (now negative). This set is added to the dataset, which is now 125% of the original size. We verify that a BERT model consistently applies these two rules, which represent multi-token reasoning patterns of the kind we wish to find in models trained on real data.

[1]For both kmeans and neighbor search, we use scikit-learn (Pedregosa et al. 2011) with default parameters.

[2]As provided by the Embedding Projector (Abadi et al. 2016); with default parameters.

Representation	Size	Prec.	Recall
B1: PMI vocab	.00	.50	.00
B2: avg emb	.36	.35	.62
B3: CLS-encoding	.20	**1.00**	**1.00**
S1: salience vocab	.20	**1.00**	**1.00**
S2: salience emb	.22	.93	**1.00**

Table 1: Validating the Approach on Synthetic Data: *Precision* and *Recall* of the cluster that has the highest precision for the respective representation. Both are computed based on 227 synthetic examples in the data. *Size* is the data fraction of the selected cluster compared to the entire set.

IMDB This binary classification dataset comprises movie reviews that are labelled as positive or negative (Maas et al. 2011). It is balanced, containing an equal number of positive and negative examples. We train a standard BERT-base model, which reaches 93% accuracy.

Wikipedia Toxicity This dataset contains Wikipedia edit comments (Wulczyn, Thain, and Dixon 2017) which are labeled as being toxic or non-toxic, with only about 10% of positive (toxic) examples. Our BERT-base model reaches 93% accuracy.

4 Validating the Approach on Synthetic Data

In our first experiment, we seek to validate our approach in a controlled setting. For that, we verify that a known shortcut can be revealed by clustering a dataset's fixed-length representations. As 'known shortcut', we use the two-token rules synthetically introduced into Synthetic SST2. We know this pattern effectively acts as a shortcut, because the model achieves 100% accuracy on these synthetic examples, regardless of the sentiment of the rest of the example. To carry out a validation test, we compute all representations described in Sec. 2.2 for each data example from Synthetic SST2's test set that the model predicted as class 0. For each representation matrix thus obtained, we apply clustering to its rows, and cluster the data into 3 clusters.[3] Will we find one cluster containing all (or most) of the 227 synthetic examples (i.e. the ones that contain both special terms)?

Metrics We compute **Precision** and **Recall** w.r.t. synthetic examples of all clusters: Precision is the ratio of synthetic examples within the cluster. Recall is the ratio of synthetic examples in the cluster out of all synthetic examples. To more easily compare the clusterings of different representations, we only report the metrics on the cluster that has the highest **Precision** for each representation (Tab. 1).

The *PMI* and *avg embedding* baselines (B1, B2) do not perform well: the best clusters have either low precision or a very small size. For PMI, the cluster with the highest recall

[3]The results hold for a larger numbers of 20 clusters, too (see Sec. A.2).

ID	%	Top-5 Terms
0	71	terrible, horrible, boring, recommend, dull
1	6	awful, worst, avoid, bad, terrible
2	8	worst, terrible, avoid, bad, horrible
3	5	10, 4, 3, /, 2
4	8	bad, horrible, acting, worst, recommend

Table 2: Spurious Correlations in IMDB: Clustering of a slice of IMDB test data using the salience vocabulary representation (S1), along with cluster size (in %) and the 5 terms that have highest mean salience in the respective cluster.

Representation	Size	Prec.	Recall
B1: PMI vocab	.16	.12	.19
B2: avg emb	.08	.16	.13
B3: CLS-encoding	.47	.12	**.58**
S1: salience vocab	.05	**.98**	.50
S2: salience emb	.05	.93	.53

Table 3: Spurious Correlations in IMDB: *Precision* and *Recall* of the cluster that has the highest precision for the respective representation. Both are computed based on 245 numeric examples in the data. *Size* is the data fraction of the selected cluster compared to the entire set.

contains 226/227 synthetic examples, but has a low precision of only 22% (not shown in table). The CLS-*encoding* baseline (B3) performs very well, but as we show in the next sections, does not perform well across model understanding use cases. The salience-based vocabulary representation (S1) based on Grad-L2 leads to a perfect cluster containing all the synthetic examples and no others. This shows that by clustering model-based input representations, we can automatically discover lexical shortcuts. Sec. A.2 has further results.

In the following three sections, we look at various case studies on real datasets that highlight how aggregating input salience helps a model developer in understanding their model and data better.

5 Discovering Patterns in Real Data

Now that our approach has passed an initial validation test, we turn to two common datasets to verify that clustering salience representations is helpful for identifying prominent patterns there.

5.1 Spurious Correlations in IMDB

In order to explore the patterns which lead the model to classify a movie review as negative, we select datapoints that the model predicted to be *negative* and sample 2,500 examples from it (10% of IMDB's test set size). For the ease of presentation we cluster their salience vocabulary representation (S1) into 5 clusters (similar results are obtained with different numbers of clusters, see Sec. A.2).

Tab. 2 lists the resulting clusters with their size and the respective top salient terms.[4] One cluster that immediately stands out is cluster 3 which contains almost exclusively numbers as top terms. In Fig. 1 a) we also use word clouds to get an idea of what constitutes this cluster (see Fig. 4 in Sec. A.3 for full size).

By focusing exclusively on the examples which mention these top terms we can easily discover a general pattern. These IMDB reviews contain expressions such as *'1 / 10 stars'*, *'3 out of 10'*, *'4 / 10'*, that summarize the review's rating and thereby give away the label. In such cases, the model does not need to analyze the rest of the review to predict that it is negative.[5] Interestingly, the same pattern was previously discovered by Ross, Marasovic, and Peters (2021) and Kaushik, Hovy, and Lipton (2020), who manually analyzed IMDB examples. We refer the reader to Ross, Marasovic, and Peters (2021) for the demonstration that BERT-like models indeed learn shallow reasoning patterns with the above listed ngrams and how one can construct counterfactuals which in turn reveal model vulnerabilities.

By using a set of regular expressions (see Sec. A.3 for a full list), we discovered that at least 12% of IMDB reviews have such a numeric pattern, evenly distributed between train/test and between label 0/1. In the 2,500 sampled test examples with negative prediction, there are 245 such reviews. Similarly to the synthetic experiment we aim to identify this subset automatically.

To compare the salience-based with the baseline representations, in Tab. 3 we again report Precision and Recall for the respective cluster having the highest precision. The clusters obtained with the baseline representations fail to bring examples having a numeric pattern forward. The PMI baseline (B1) performs poorly because PMI values for interesting tokens are not among the highest (e.g., '3' occurs in positive and negative reviews and therefore does not get a high PMI).

The salience-based representations (S1, S2), however, identify this subset well, producing clusters which almost exclusively consist of numeric examples and which contain more than half of the examples with a numeric pattern. Please note that representations based on Integrated Gradients perform poorly at this task, which aligns with the findings of (Bastings et al. 2022) (see Sec. A.2 for further details and more configurations).

An analysis of why not all of the numeric examples are clustered together revealed that most of the missing numeric examples contain either none or just 1 digit in their top-5 salient terms (i.e., the salience method did not put high weight on them) and are spread randomly across all other clusters. This indicates that real data patterns are indeed more subtle and not deterministically indicative of a label, unlike the patterns that we used in the synthetic data experiment.

To demonstrate that salience representations also enable more fine-grained analysis, in Fig. 2 we show a t-SNE vi-

[4] We removed BERT's special tokens, e.g., '[CLS]', and punctuation.

[5] We repeat this analysis on a slice of positive reviews and find the same pattern for high ratings between 7 and 10.

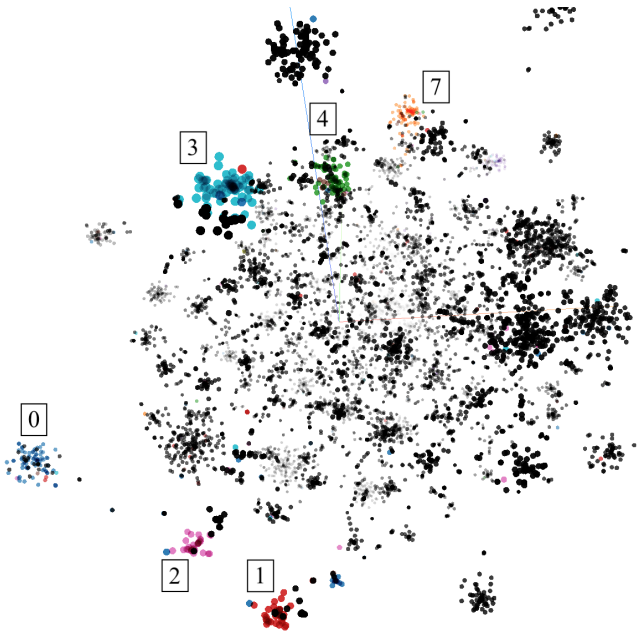

Figure 2: Spurious Correlations in IMDB: t-SNE visualization of the salience vocab. representation of IMDB. Colored points are examples that contain a certain digit (indicated by the boxes): dark blue: 0, red: 1, pink: 2, turquoise: 3, green: 4, orange: 7. Other digits are not visible from this visualization angle.

Figure 3: Patterns in Wikipedia Toxicity Subtypes: The top-left dense cluster of non-toxic comments (a fragment of the full clustering is shown) contains examples which all start with a *'redirect talk:'* prefix. Such comments account for more than 1% of the test set.

sualization of a 10k sample from the IMDB test set, where colored dots represent examples with a specific digit. One can easily see that many examples have digits in them and that they are clustered together. Thus, by increasing the number of clusters one can discover more fine-grained patterns.

5.2 Patterns in Wikipedia Toxicity Subtypes

We now turn to Wikipedia Toxicity and look for patterns in the data that the model learned. It should be noted that the implications of having problematic patters in a toxicity dataset extend beyond robustness into the area of ML fairness. Imperfect annotations are inevitable (Wong, Paritosh, and Bollacker 2022) but amplified by a sample bias may result in unfair treatment of certain demographic groups. For example, *gay* is a term known to often trigger toxicity prediction (Dixon et al. 2018a), even when it is not justified: *'This is disrespectful to gay and all other people'* gets .85 probability of being toxic. Zooming into examples mentioning a particular token can give a more fine-grained idea of how a token relates to the classes.

We cluster all the 487 Toxicity test examples mentioning *'gay'* using the salience vocabulary representation (S1) and t-SNE. Interestingly, we discovered clusters of comments with the non-toxic label which all had not only *'gay'* but also tokens like *'homosexual', '(homo)sexuality', 'LGBT'* among their top-5 most salient terms (Sec. A.4). This suggests that contexts for *'gay'* which talk about sexuality or sexual orientation with such a neutral term are most often non-toxic. This is an example of a more fine-grained analysis of how a term relates to the class. An actionable conclusion is that if one wants to augment the data with more non-toxic examples of the token's occurrences, they should be different from the ones the model already knows to be non-toxic contexts.

Another example of a pattern present in the data and not obvious at first sight is given in Fig. 3. It shows a small fragment of a t-SNE visualization of the salience vocabulary representation (S1) of 10k Toxicity examples. The top-left cluster consists of non-toxic comments (blue color). The nearest neighbors of a point in the center of the cluster reveals that all the examples there follow the template *'redirect talk: {Title}'* which the model apparently uses as a strong indication of non-toxicity. Indeed, almost 1% of the training and more than 1% of the test split are all automatically generated comments of this kind and as such can hardly be useful for learning what differentiates toxic from non-toxic language or evaluating model quality. Thus, a simple visualization of aggregated salience reveals a weakness of the data. In contrast, the PMI baseline (B1) does not associate the two terms individually with the non-toxic class. Also the CLS baseline (B3) does not place the *'redirect talk: {Title}'* examples together.

6 Identifying Model Sensitivities

In this section, we investigate how sensitive models are to the **top terms** identified through aggregated salience. A model is sensitive to a certain term if the sheer occurrence of the term means it is likely to classify a piece of text into a given class, without considering the entire context. Knowing what the model is sensitive to may help discover spurious correlations or unfair biases. For IMDB, Ross, Marasovic, and Peters (2021) already showed that one can indeed create counterfactuals from the patterns like the ones discovered in the previous section (Sec. 5.1). Now however, we turn our attention to the Toxicity dataset, where we wish to systematically measure model sensitivity. For toxicity prediction,

simplistic single-token patterns may result in problematic biases towards minorities with, e.g., identity terms becoming indicative of toxicity for a model (Dixon et al. 2018b; Hartvigsen et al. 2022), independently of their context.

Therefore, we want to analyze how sensitive the trained model is to various terms. We measure this in two ways: (1) average increase in probability towards the toxic label when inserting each of the terms in non-toxic text (counterfactuals), (2) average decrease in probability towards the toxic label when masking each of the terms in toxic text.

Creating Counterfactuals Given a list of seed terms $\mathcal{S}$ that are deemed toxic, we create counterfactual examples in the following way: We separately add every seed term $s \in \mathcal{S}$ at a random position into $m = 10k$ random non-toxic test examples[6] and measure the average change in probability for every seed term and every list of seed terms.

$$change(\mathcal{S}) = \frac{1}{m|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{i}^{m} (p(x_i^s) - p(x_i)) \qquad (1)$$

Two ways of creating baseline seed lists are considered: (1) The identity terms from Dixon et al. (2018b). (2) The tokens with highest PMI(token; toxic class) that occurr at least 20 times in the training set.[7]

Additionally, in order to get a seed list that is informed by aggregated input salience, we compute the column means of the salience vocabulary representation R^{vocab} ($k = 5$) of all toxic examples in the training set and keep the 10 highest valued entries. Again, we keep only terms that occur at least 20 times. The seed lists contains 10 terms, except for the list from Dixon et al. (2018b), which contains 13. The full list of all seed terms is shown in Sec. A.5.

Results are listed in Tab. 4. Interestingly, the model does not seem sensitive to the list of identity terms from *Dixon et al. (2018b)*. On the other hand, the statistical correlation of term occurrence and label, as measured by *PMI*, seems to be a good indicator of model sensitivity. However, the list of tokens we identified by looking at aggregated salience (*aggregated salient terms*) leads to the most significant probability changes: on average the probability of an example being classified as toxic increases by 41% simply by inserting a single term.

Masking out Toxic Terms In the previous part, we established that the probability of classifying an input as toxic increases when a seed term is inserted into the input. If the model is truly sensitive to these seed terms, we would also expect a drop in probability when removing these terms from existing toxic inputs. To verify this, in the second experiment, we mask out the same seed terms from above, paralleling the comprehensiveness metric from DeYoung et al. (2020) who use it to evaluate the quality of a salience method.

[6]We acknowledge that this may lead to disfluent or even ungrammatical inputs (Ross, Marasovic, and Peters 2021). This, however, is not a concern because in the Toxicity dataset, many examples are ungrammatical to begin with.

[7]Please note that *PMI vocab* is the only baseline method that we can compute top terms for.

Seed List	Prob. Change
Dixon et al. (2018b)	.06 (.08)
PMI(token; toxic class)	.31 (.19)
aggregated salient terms	**.41** (.08)

Table 4: Creating Counterfactuals: Mean probability increase when inserting specific terms into non-toxic examples. Numbers in parentheses are stddev when averaging all terms from the seed lists.

Seed List	Count	Prob. Change
Dixon et al. (2018b)	55	-.12 (.15)
PMI(token; toxic class)	279	-.22 (.10)
aggregated salient terms	**475**	**-.34** (.08)

Table 5: Masking out Toxic Terms: Mean probability decrease when masking specific terms in toxic examples. Numbers in parentheses are stddev when averaging all terms from the seed lists. Count is the number of occurrences of all seed terms.

The process of creating masked examples is as follows: we use all 8,858 toxic test examples and the same seed lists as above. For every seed term we replace all the occurrences of it with '[MASK]' and compare probabilities before and after.

Results are reported in Tab. 5. Both non-salience baseline methods (*Dixon et al. (2018b)* and *PMI*) lead to lower probability changes. Again, the probability change for *aggregated salient terms* is the strongest (average 34%). In addition, *aggregated salient terms* presents a list whose entries occur more often than terms from the other lists, meaning these are frequent *and* important for the model.

In summary, aggregated salience information provides us with a list of terms that the model is sensitive to. No other method leads to a comparable increase in toxicity probability when adding terms and decrease in toxicity when terms are masked out.

7 Generalizing from Single Examples

The approaches outlined in Sec. 5 and 6 can be characterized as "top-down": they use salience-based representations to perform clustering or term selection on a large number of examples, then zoom into single clusters or examples to identify relevant patterns. In this section however, we take an inverse "bottom-up" approach: starting from a single data instance, we search for its nearest neighbors in representation space in order to uncover explanations for the way the model classified it. We focus on misclassified examples in the Toxicity test set. From the slice of misclassified data, we sample 200 examples and identify that 28 (14%) of them are surprising, meaning the prediction cannot be understood just by looking at the example's text.[8] For each surprising

[8]By contrast, Sec. A.6 lists some unsurprising examples for the purpos of illustration.

Misclassified Examples and their 3 Nearest Neighbors	Top-5 Terms	Label / Prediction
" : : : fine with me . | **ck** "	**ck**, [SEP], |, [CLS], .	non-toxic / toxic
1. co / **ck** - pen - i . s . international	**ck**, /, [SEP], co, [CLS]	toxic / toxic
2. su :) **ck** my ba :) lls you fa :) g :) got loser	**ck**, loser, [SEP], [CLS],)	toxic / toxic
3. block me ! ! ! see how fast another post comes du . mbfu . **ck** wikipedia is the devel oh , and they cann also fine wiki for monopolizing online encyclopedias	**ck**, ., wikipedia, mb, ##fu	toxic / toxic
dogs **pee** more than they poo	**pee**, dogs, [CLS], po, [SEP]	non-toxic / toxic
1. i am going to **pee** on you !	**pee**, on, !, [CLS], [SEP]	toxic / toxic
2. **pee**eeeeeeeeeniiiiiiiiiiiisssss ! ! ! ! ! ! ! ! still here , waiting for a response	**pee**, ##ee, ##iss, [CLS], [SEP]	toxic / non-toxic
3. how big is your **pee pee** .	**pee**, your, big, [CLS], [SEP]	toxic / toxic
kobe is the best **fuckin** player ever	**fuck**, kobe, player, **##in**, best	non-toxic / toxic
1. guys , it ' s just simply me that put there moldovans (romanians) 78 , 2 % , and i was not logged in , and don ' t give quickly the fault on bonaparte , he ' s gone from longtime now from this bullshit of wikipedia , ' cause there ' s no purpose with hypocritic people like you . i modified that , everybody knows that that ' s moldovans and romanians are the same shit , anyhow the census will say it . they speak romanian , and not moldovan how they sustain . a romanian understands perfectly when a moldovan is speaking it ' s **fuckin** ' language	**fuck**, **##in**, guys, s, longtime	toxic / toxic
2. haha this is my **fuckin** page ! ! ! ! ! ! ! ! ! i can do whatever i want ! ! ! ! ! !	**fuck**, **##in**, [CLS], page, my	toxic / toxic
3. british people , or britons , [7] are inhabitants of great britain [8] [9] or citizens of the united kingdom . i dont think this is about an ethnicity / race wobbs . haha , yous a complete **fuckin** moron .	**fuck**, **##in**, mor, [SEP], complete	toxic / toxic

Table 6: Generalizing from Single Examples: 3 examples having surprising predictions with their top-5 terms and 3 nearest neighbors. Bold terms are overlapping in example and neighbors. They give a possible explanation why the examples were predicted to be toxic.

example, we retrieve its 5 nearest neighbors in representation space among the examples of the training set that have the same label as the example's prediction. We search for a rationale for the misclassification using the nearest neighbors list. Tab. 6 shows three examples of surprising classifications made by the model – examples which a model developer might want to inspect.

Why might the model have classified the comment *' : : : fine with me . \| ck'* as toxic? Looking at the nearest neighbors reveals that *'ck'* often appears in the context of the swear words *'f * ck'*, *'s * ck'* and *'c * ck'*, providing an explanation for why the model might have learned it to be toxic.

The example of *'dogs pee more than they poo'* shows that the model may be sensitive to the term 'pee', without considering the context. Finally, though the comment *'kobe is the best fuckin player ever'* is far from toxic, its nearest neighbors bring many toxic uses of the word *'fuckin'* to light, giving an insight into why the model has classified it as toxic.

We compare all baseline representations (B1-3) and the salience vocabulary representation (S1) with $k = 5$ top salient tokens. While in the experiments described so far the choice of k did not make a difference and comparable results were obtained with $k = |\mathcal{V}|$, in this nearest neighbor experiment we got better results (by manual inspection) for $k = 5$. This is intuitive, because we want the similarity between an example in question and its nearest neighbors to be dominated by the high-salience tokens they have in common and not by the long low-salience tail.

Among the 28 examples, we are able to uncover rationales for the misclassification in 22 cases using the vocabulary-based saliency representation (S1), while for the baseline methods PMI (B1), average-of-embeddings (B2) and CLS-encoding (B3), we find rationales in only 20, 9 and 6 of cases respectively. This shows that aggregated salience is helpful in gaining a deeper understanding of model predictions.[9]

8 Related Work

Explanation-based Human Debugging with Global Explanations Our work falls into the category of explanation-based human debugging (Lertvittayakumjorn and Toni

[9] We acknowledge that what constitutes a *surprising* example and a *good* rationale is a subjective judgement. Therefore, the difference between the vocabulary-based saliency representation and PMI may not be significant.

2021b). Since most works in this field propose *local* explanations (Danilevsky et al. 2020; Lertvittayakumjorn and Toni 2021b), our method contributes to an understudied type of explanation by aggregating local explanations into *global* explanations. The closest related work is FIND (Lertvittayakumjorn, Specia, and Toni 2020), a framework that uses layer-wise relevance propagation (LRP) (Arras et al. 2016) to derive feature weights for all examples, which are then converted into word clouds for users to interact with. They focus on disabling features globally, whereas the strength of our approach is that it allows debugging individual examples as well as entire datasets.

Model Understanding Our work concerns identification of (shallow) patterns learned by text classifiers as well as the explanation of individual examples. This should help model developers get a better understanding of their model and give them indicators of how to improve them. With that, the presented methods are well suited to be included into existing model understanding tools (Nori et al. 2019; Wallace et al. 2019; Kokhlikyan et al. 2020; Tenney et al. 2020; Geva et al. 2022). Interactive debugging based on input salience gives a model developer another view on top of static descriptions by the means of, e.g., model cards (Mitchell et al. 2019) or data cards (Pushkarna, Zaldivar, and Kjartansson 2022). Additionally, it complements approaches like CheckList (Ribeiro et al. 2020), which also has the goal of improving NLP models by proactively identifying their weaknesses.

Salience and Clustering In vision research, clustering and salience have been applied to find spurious patterns (Bach et al. 2015b; Lapuschkin et al. 2019; Schoop et al. 2022).

Yin and Neubig (2022) cluster contrastive explanations to describe why a language model chose a certain token over another. Kauffmann et al. (2022) cluster text data, then replace the clustering with an equivalent neural network, which is finally used to generate individual explanations.

Training Data Attribution The kNN analysis (Sec. 7) is similar to proponents identification done with training data attribution methods (TDA) (Zylberajch, Lertvittayakumjorn, and Toni 2021; Han and Tsvetkov 2021). We do not perform a comparison with TDA methods because (1) they focus on analyzing the training set while we are primarily interested in understanding (the model performance on) the test set. (2) TDA methods suggest proponents but leave it to the developer to discover what it is that the proponents and the test example share (unless combined with salience techniques as done in Pezeshkpour et al. 2021). Unlike that, salience-based distance is explicit about what every token contributes to the similarity.

9 Conclusion

In this work, we present a way to use aggregated salience representations to meet distinct yet common model developer needs. The proposed methods, which are based on standard input salience methods, help find prominent patterns and gain insights into single examples or entire datasets. Through a series of case studies carried out on three datasets (1 synthetically-modified, 2 academic), we show that our approach (semi-)automatically provides explanations through clustering, identifies shortcut-like patterns, uncovers patterns the model is particularly sensitive to and explains predictions through nearest neighbors. Moreover, we try out two ways of aggregating salience maps (vocabulary-based and embedding-based) and show that our method performs better than baseline representations across all use cases presented.

10 Limitations

Though the results described in this paper are promising, they are limited in the types of patterns they can recognize. Both salience-based representations (vocabulary-based and embedding-based) are order-agnostic representations. Therefore, any method using these will not be able to detect patterns that depend on word order. Additionally, they will not be able to identify patterns related to example length, token position in the example, as well as more complex patterns such as ones where only a certain part of the input is decisive for the model (e.g., hypothesis-only shortcuts in Natural Language Inference).

We run all experiments in this paper with a BERT-based model, and although we choose Grad-L2 as input salience method (because it was shown to work well for this model), any salience method is compatible with our approach, as long as it produces a single salience vector per example.

Since we focus on BERT only it is conceivable that the results in this paper may not hold for other model architectures. This, however, does not pose too large of a threat for two reasons: 1. Most current architectures are, just as BERT, based on transformers (Vaswani et al. 2017). 2. The proposed methods depend more on the faithfulness of the input salience method than on the model architecture. As long as the used input salience method is faithful to the model, the proposed methods are likely to work as well.

The representations we propose are by no means a complete list. For instance, the vocabulary representation is a sparse representation (depending on the value of top-k). One could consider applying PCA on the sparse matrix to transform it into a dense representation.

All experiments presented in this work are binary text classification tasks. In principle however, our method can be extend to multi-class, regression or text generation tasks (e.g., summarization, question answering), as long as a suitable input salience method exists for the task.

Finally, in this work, we devote our attention to better understanding a model and dataset. Our method does not directly provide robustness fixes, although identifying the model and dataset's weaknesses are a necessary first step and sometimes lead to obvious remediation techniques such as data rebalancing.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I. J.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Józefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D. G.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P. A.; Vanhoucke, V.; Vasudevan, V.; Viégas, F. B.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR*, abs/1603.04467.

Arras, L.; Horn, F.; Montavon, G.; Müller, K.; and Samek, W. 2016. Explaining Predictions of Non-Linear Classifiers in NLP. In Blunsom, P.; Cho, K.; Cohen, S. B.; Grefenstette, E.; Hermann, K. M.; Rimell, L.; Weston, J.; and Yih, S. W., eds., *Proceedings of the 1st Workshop on Representation Learning for NLP, Rep4NLP@ACL 2016, Berlin, Germany, August 11, 2016*, 1–7. Association for Computational Linguistics.

Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; and Samek, W. 2015a. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7): 1–46.

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

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bastings, J.; Ebert, S.; Zablotskaia, P.; Sandholm, A.; and Filippova, K. 2022. "Will You Find These Shortcuts?": A Protocol for Evaluating the Faithfulness of Input Salience Methods for Text Classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (to appear)*.

Bastings, J.; and Filippova, K. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 149–155. Online: Association for Computational Linguistics.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 632–642. Lisbon, Portugal: Association for Computational Linguistics.

Danilevsky, M.; Qian, K.; Aharonov, R.; Katsis, Y.; Kawas, B.; and Sen, P. 2020. A Survey of the State of Explainable AI for Natural Language Processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 447–459. Suzhou, China: Association for Computational Linguistics.

Denil, M.; Demiraj, A.; and de Freitas, N. 2015. Extraction of Salient Sentences from Labelled Documents. arXiv:1412.6815.

Devlin, J.; Chang, M.; Lee, K.; and 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 4171–4186.

DeYoung, J.; Jain, S.; Rajani, N. F.; Lehman, E.; Xiong, C.; Socher, R.; and Wallace, B. C. 2020. ERASER: A Benchmark to Evaluate Rationalized NLP Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4443–4458. Online: Association for Computational Linguistics.

Dixon, L.; Li, J.; Sorensen, J.; Thain, N.; and Vasserman, L. 2018a. Measuring and Mitigating Unintended Bias in Text Classification. In *AAAI/ACM Conference on AI, Ethics, and Society*.

Dixon, L.; Li, J.; Sorensen, J.; Thain, N.; and Vasserman, L. 2018b. Measuring and Mitigating Unintended Bias in Text Classification. In Furman, J.; Marchant, G. E.; Price, H.; and Rossi, F., eds., *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society, AIES 2018, New Orleans, LA, USA, February 02-03, 2018*, 67–73. ACM.

Eisenstein, J. 2022. Informativeness and Invariance: Two Perspectives on Spurious Correlations in Natural Language. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4326–4331. Seattle, United States: Association for Computational Linguistics.

Gardner, M.; Merrill, W.; Dodge, J.; Peters, M.; Ross, A.; Singh, S.; and Smith, N. A. 2021. Competency Problems: On Finding and Removing Artifacts in Language Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 1801–1813. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Geirhos, R.; Jacobsen, J.-H.; Michaelis, C.; Zemel, R.; Brendel, W.; Bethge, M.; and Wichmann, F. A. 2020. Shortcut Learning in Deep Neural Networks. *Nature Machine Intelligence*, 2: 665—673.

Geva, M.; Caciularu, A.; Dar, G.; Roit, P.; Sadde, S.; Shlain, M.; Tamir, B.; and Goldberg, Y. 2022. LM-Debugger: An Interactive Tool for Inspection and Intervention in Transformer-Based Language Models. *CoRR*, abs/2204.12130.

Geva, M.; Goldberg, Y.; and Berant, J. 2019. Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1161–1166. Hong Kong, China: Association for Computational Linguistics.

Gururangan, S.; Swayamdipta, S.; Levy, O.; Schwartz, R.; Bowman, S.; and Smith, N. A. 2018. Annotation Artifacts

in Natural Language Inference Data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, 107–112. New Orleans, Louisiana: Association for Computational Linguistics.

Han, X.; and Tsvetkov, Y. 2021. Influence Tuning: Demoting Spurious Correlations via Instance Attribution and Instance-Driven Updates. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 4398–4409. Punta Cana, Dominican Republic: Association for Computational Linguistics.

Hartvigsen, T.; Gabriel, S.; Palangi, H.; Sap, M.; Ray, D.; and Kamar, E. 2022. ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3309–3326. Dublin, Ireland: Association for Computational Linguistics.

Kauffmann, J.; Esders, M.; Ruff, L.; Montavon, G.; Samek, W.; and Müller, K.-R. 2022. From Clustering to Cluster Explanations via Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.

Kaushik, D.; Hovy, E. H.; and Lipton, Z. C. 2020. Learning The Difference That Makes A Difference With Counterfactually-Augmented Data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Koh, P. W.; and Liang, P. 2017. Understanding Black-box Predictions via Influence Functions. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1885–1894. PMLR.

Kokhlikyan, N.; Miglani, V.; Martin, M.; Wang, E.; Alsallakh, B.; Reynolds, J.; Melnikov, A.; Kliushkina, N.; Araya, C.; Yan, S.; and Reblitz-Richardson, O. 2020. Captum: A unified and generic model interpretability library for PyTorch. *CoRR*, abs/2009.07896.

Lapuschkin, S.; Wäldchen, S.; Binder, A.; Montavon, G.; Samek, W.; and Müller, K.-R. 2019. Unmasking Clever Hans Predictors and Assessing What Machines Really Learn. *Nature Communications*, 10: 1096.

Lertvittayakumjorn, P.; Specia, L.; and Toni, F. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 332–348. Association for Computational Linguistics.

Lertvittayakumjorn, P.; and Toni, F. 2021a. Explanation-Based Human Debugging of NLP Models: A Survey. *Transactions of the Association for Computational Linguistics*, 9: 1508–1528.

Lertvittayakumjorn, P.; and Toni, F. 2021b. Explanation-Based Human Debugging of NLP Models: A Survey. *Trans. Assoc. Comput. Linguistics*, 9: 1508–1528.

Li, J.; Chen, X.; Hovy, E.; and Jurafsky, D. 2016. Visualizing and Understanding Neural Models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 681–691. San Diego, California: Association for Computational Linguistics.

Li, J.; Monroe, W.; and Jurafsky, D. 2016. Understanding Neural Networks through Representation Erasure. arXiv:1612.08220.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

McCoy, T.; Pavlick, E.; and Linzen, T. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3428–3448. Florence, Italy: Association for Computational Linguistics.

Mitchell, M.; Wu, S.; Zaldivar, A.; Barnes, P.; Vasserman, L.; Hutchinson, B.; Spitzer, E.; Raji, I. D.; and Gebru, T. 2019. Model Cards for Model Reporting. In danah boyd; and Morgenstern, J. H., eds., *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, 220–229. ACM.

Montavon, G.; Binder, A.; Lapuschkin, S.; Samek, W.; and Müller, K. 2019. Layer-Wise Relevance Propagation: An Overview. In Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L. K.; and Müller, K., eds., *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, 193–209. Springer.

Nori, H.; Jenkins, S.; Koch, P.; and Caruana, R. 2019. InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv preprint arXiv:1909.09223*.

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

Pezeshkpour, P.; Jain, S.; Singh, S.; and Wallace, B. C. 2021. Combining Feature and Instance Attribution to Detect Artifacts. arXiv:2107.00323.

Pruthi, G.; Liu, F.; Kale, S.; and Sundararajan, M. 2020. Estimating Training Data Influence by Tracing Gradient Descent. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.-F.; and Lin, H.-T., eds., *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Pushkarna, M.; Zaldivar, A.; and Kjartansson, O. 2022. Data Cards: Purposeful and Transparent Dataset Documentation

for Responsible AI. In *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*, 1776–1826. ACM.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 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.

Ribeiro, M. T.; Wu, T.; Guestrin, C.; and Singh, S. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4902–4912. Online: Association for Computational Linguistics.

Rosenman, S.; Jacovi, A.; and Goldberg, Y. 2020. Exposing Shallow Heuristics of Relation Extraction Models with Challenge Data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3702–3710. Online: Association for Computational Linguistics.

Ross, A.; Marasovic, A.; and Peters, M. E. 2021. Explaining NLP Models via Minimal Contrastive Editing (MiCE). In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, 3840–3852. Association for Computational Linguistics.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Schoop, E.; Wedin, B.; Kapishnikov, A.; Bolukbasi, T.; and Terry, M. 2022. IMACS: Image Model Attribution Comparison Summaries.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642. Seattle, Washington, USA: Association for Computational Linguistics.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 3319–3328. PMLR.

Tenney, I.; Wexler, J.; Bastings, J.; Bolukbasi, T.; Coenen, A.; Gehrmann, S.; Jiang, E.; Pushkarna, M.; Radebaugh, C.; Reif, E.; and Yuan, A. 2020. The Language Interpretability Tool: Extensible, Interactive Visualizations and Analysis for NLP Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 107–118. Online: Association for Computational Linguistics.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need.

Wallace, E.; Tuyls, J.; Wang, J.; Subramanian, S.; Gardner, M.; and Singh, S. 2019. AllenNLP Interpret: A Framework for Explaining Predictions of NLP Models. In Padó, S.; and Huang, R., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019 - System Demonstrations*, 7–12. Association for Computational Linguistics.

Wong, K.; Paritosh, P.; and Bollacker, K. 2022. Are Ground Truth Labels Reproducible? An Empirical Study. In *ML Evaluation Standards (ICLR 2022 Workshop)*.

Wulczyn, E.; Thain, N.; and Dixon, L. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, 1391–1399. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. ISBN 9781450349130.

Yin, K.; and Neubig, G. 2022. Interpreting Language Models with Contrastive Explanations.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and Understanding Convolutional Networks. In Fleet, D.; Pajdla, T.; Schiele, B.; and Tuytelaars, T., eds., *Computer Vision – ECCV 2014*, 818–833. Cham: Springer International Publishing. ISBN 978-3-319-10590-1.

Zylberajch, H.; Lertvittayakumjorn, P.; and Toni, F. 2021. HILDIF: interactive debugging of NLI models using influence functions. 1–6. ASSOC COMPUTATIONAL LINGUISTICS-ACL.

A Appendix

A.1 Training Details

For all experiments, we use a publicly available pretrained BERT-base model and fine-tune it on the 3 datasets: synthetic SST2, IMDB and Toxicity. We use a dropout rate of 0.5, a batch size of 16 and train each model for a maximum number of 35k steps. To mitigate overfitting, we stop training if after 10k steps there is no improvement on the validation set and use the model checkpoint that had highest performance on the validation set. The models are optimized using ADAM (Kingma and Ba 2015) with weight decay of 5e-6. For varying hyper-parameters and model performance, see Tab. 7. With these hyper-parameters, the BERT models contain 109M parameters. Training and optimizing them took a total of 35 hours on TPUv2 (4-core) accelerators. Test set accuracies are in line with previously reported performances (Devlin et al. 2019; Sanh et al. 2019).

A.2 Alternative Clustering Configurations

Tab. 8 shows all results from the verification test on synthetic data (Sec. 4). Results for the task of discovering spurious correlations in IMDB (Sec. 5.1) are shown in Tab. 10.

Dataset	Max Seq. Length	Learning Rate	Test Set Acc.
Synthetic SST2	100	2e-5	.94
IMDB	500	2e-5	.93
Toxicity	128	1e-5	.93

Table 7: Training details of the BERT-base model.

Representation	Size	Prec.	Recall	top-k
B1: PMI vocab	.00	.50	.00	-
B2: avg emb	.36	.35	.62	-
B3: CLS-encoding	.20	1.00	1.00	-
S1-1: salience vocab (G-L2)	.20	1.00	1.00	5
S2-1: salience emb (G-L2)	.22	.93	1.00	5
S1-2: salience vocab (G-L2)	.20	1.00	1.00	$\|\mathcal{V}\|$
S2-2: salience emb (G-L2)	.21	.95	1.00	$\|\mathcal{V}\|$
S1-3: salience vocab (IG)	.17	1.00	.85	5
S2-3: salience emb (IG)	.16	1.00	.80	5

Table 8: Validating the Approach on Synthetic Data with 3 Clusters: *Precision* and *Recall* of the cluster that has the highest precision for the respective representation. Both are computed based on 227 synthetic examples in the data. *Size* is the data fraction of the selected cluster compared to the entire set.

Integrated Gradients (IG) Clusterings based on IG salience (S1-3, S2-3) lead to high precision but lower recall on synthetic data. On IMDB, clusters computed on IG-based representations (S1-3, S2-3) lead to results similar to the baselines.

Top-k Salient Tokens In both tasks using the entire vocabulary to compute the representation matrices ($k = |\mathcal{V}|$, S1-2, S2-2) leads to results that are similar to $k = 5$ (S1-1, S2-1; see Tab. 8 and 10).

Number of Clusters PMI (B1) does not benefit from increasing the number of clusters from 3 to 20 on the synthetic SST2 dataset. It still produces a tiny cluster (compare Tab. 8 and Tab. 9).

Increasing the number of clusters shows how sensitive the avg. embedding baseline (B2) is. The CLS (B3) and salience-based representations (all S1, S2) are rather stable and lead to similar results for both configurations.

We see similar results on the more realistic dataset IMDB, where we want to rediscover a cluster of numeric examples (Tabs. 10 and 11). The PMI avg. embeddings baselines (B1) perform poorly in both settings. The results for CLS-encoding representations (B3) vary strongly. For 20 clusters this representation creates a cluster that has high precision but is rather small. Again, the salience-based representations are rather insensitive to this hyperparameter (salience embeddings (S2) more so than salience vocab (S1)). The salience vocab representation (S1) leads to the best performance in terms of precision in both settings.

Representation	Size	Prec.	Recall	top-k
B1: PMI vocab	.00	1.00	.00	-
B2: avg emb	.00	.75	.03	-
B3: CLS-encoding	.19	1.00	.96	-
S1: salience vocab (G-L2)	.20	1.00	1.00	5
S2: salience emb (G-L2)	.20	.97	.99	5

Table 9: Validating the Approach on Synthetic Data with 20 Clusters: *Precision* and *Recall* of the cluster that has the highest precision for the respective representation. Both are computed based on 227 synthetic examples in the data. *Size* is the data fraction of the selected cluster compared to the entire set.

Representation	Size	Prec.	Recall	top-k
B1: PMI vocab	.16	.12	.19	-
B2: avg emb	.08	.16	.13	-
B3: CLS-encoding	.47	.12	.58	-
S1-1: salience vocab (G-L2)	.05	.98	.50	5
S2-1: salience emb (G-L2)	.05	.93	.53	5
S1-2: salience vocab (G-L2)	.05	.99	.52	$\|\mathcal{V}\|$
S2-2: salience emb (G-L2)	.04	.93	.48	$\|\mathcal{V}\|$
S1-3: salience vocab (IG)	.12	.13	.16	5
S2-3: salience emb (IG)	.11	.14	.16	5

Table 10: Spurious Correlations in IMDB with 5 Clusters: *Precision* and *Recall* of the cluster that has the highest precision for the respective representation. Both are computed based on 245 numeric examples in the data. *Size* is the data fraction of the selected cluster compared to the entire set.

A.3 Spurious Correlations in IMDB

Fig. 4 depicts a word cloud that was computed from the numeric cluster. Individual words are weighted by their average representation value (i.e., the mean of columns of R).

In the following, we list the regular expressions used to identify examples containing numeric expressions.

```
RATING_EXPRESSIONS = [
  # on, out of, ...
  r': (NUMBER) \( out of\b',
  r'\b(NUMBER) out of\b',
  r'\b(NUMBER)\* out of\b',
  r'\b(NUMBER) outta \d+\b',
  # Exclude, e.g., "my top 10 of".
  r'\b[^(top )](NUMBER) of \d+\b',
  r'\b(NUMBER) on \d+\b',
```

Representation	Size	Prec.	Recall	top-k
B1: PMI vocab	.05	.15	.09	-
B2: avg emb	.05	.20	.10	-
B3: CLS-encoding	.02	.94	.21	-
S1: salience vocab (G-L2)	.04	.98	.47	5
S2: salience emb (G-L2)	.02	.81	.22	5

Table 11: Spurious Correlations in IMDB with 20 Clusters: *Precision* and *Recall* of the cluster that has the highest precision for the respective representation. Both are computed based on 245 numeric examples in the data. *Size* is the data fraction of the selected cluster compared to the entire set.

Figure 4: Spurious Correlations in IMDB: Word cloud of cluster no. 3, which has tokens '10', '4', '3', '/' and '2' as top terms.

```
  r'\b(NUMBER) out \d+\b',
  # something / something
  r'\b(NUMBER) / 10\b',
  r'\b(NUMBER) / 5\b',
  # vote, ratings, ...
  r'\bvote \( (NUMBER) \)',
  r'\bvote (NUMBER)\b',
  r'\bvoted \" (NUMBER) \"',
  r'\bvote is (NUMBER)\b',
  r'\bgive this a (NUMBER)\b',
  r'\bgive it a (NUMBER)\b',
  r'\bgets a (NUMBER)\b',
  r'\brate this a (NUMBER)\b',
  r'\" (NUMBER) \" rating\b',
  # stars, points, ...
  r'\b(NUMBER) star\b',
  r'\b(NUMBER) - star\b',
  r'\b(NUMBER) stars\b',
  r'\b(NUMBER) points\b',
  # Exclude, e.g., "at 1 point".
  r'\b[^(at)] (NUMBER) point\b',
]
```

We substitute every `NUMBER` in these base expressions with the following expression:

```
zero|one|two|three|four|five|six|
seven|eight|nine|ten|\d+|\d+\.\d+|
\d+ \. \d+|\d+\+
```

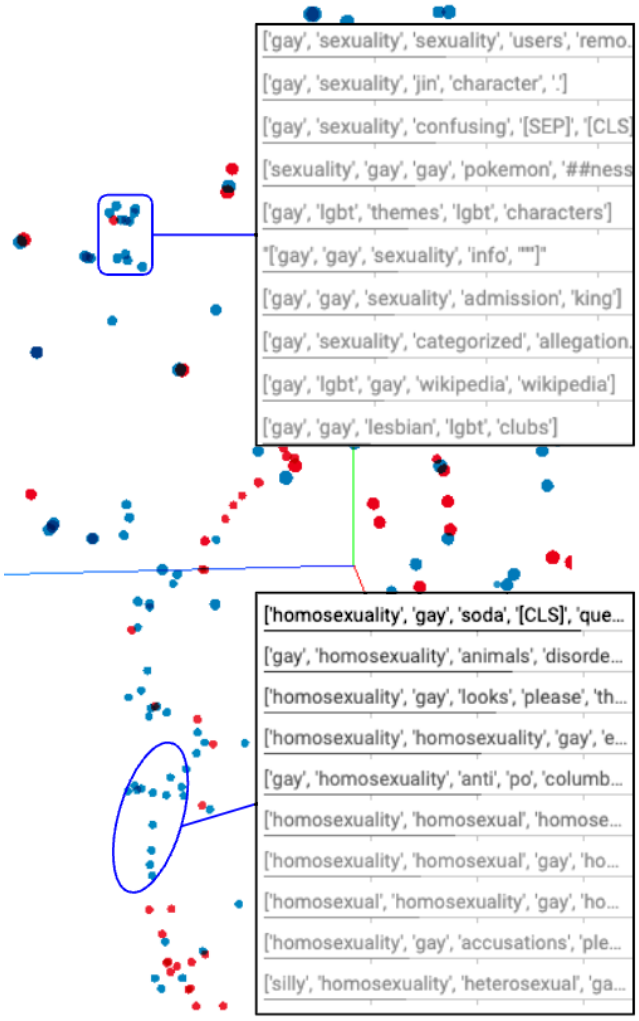

Figure 5: Patterns in Wikipedia Toxicity Subtypes: t-SNE visualization of Toxicity test examples mentioning *'gay'*. Two non-toxic clusters are selected and the 10 nearest neighbors (represented here by their most salient terms in the boxes) of the respective centrally located points are shown. All these examples have *'(homo)sexual(ity)'* or *'LGBT'* along with *'gay'* among their most salient terms.

Finally, we consider an example to contain a numeric expression if at least one of the expressions is found in the text.

A.4 Understanding Learned Patterns

Fig. 5 zooms into two non-toxic clusters computed over all the test examples mentioning the term *'gay'*.

A.5 Identifying Model Sensitivities

Tab. 12 lists all seed terms we used in the sensitivity analysis of Sec. 6.

Seed List	Seed Terms
Dixon et al. (2018b)	atheist, queer, gay, transgender, lesbian, homosexual, feminist, black, white, heterosexual, islam, muslim, bisexual
PMI(token; toxic class)	fuck, fucking, nipple, bitch, suck, sucking, lick, cock, asshole, sucker
aggregated salient terms	fuck, fucking, shit, stupid, idiot, bitch, suck, asshole, ass, dick

Table 12: Identifying Model Sensitivities: Seed lists from various sources.

A.6 Generalizing from Single Examples

Tab. 13 lists 5 examples whose predictions are not surprising. Looking at the examples' text is enough to justify the *toxic* label.

	Example	Top- Terms	Label / Prediction
1.	failure at life loser .	loser, failure, life, [SEP], [CLS]	non-toxic / toxic
2.	= = = = oh the irony = = = it just goes to show how stupid some americans really are , a scumbag mob lawyer mayor of las vegas . you couldn ' t make it up ! ! ! !	stupid, irony, mo, ##um, americans	non-toxic / toxic
3.	: oh and don ' t be a patronising ass with comments , like justin , justin . ' ' ' ' ' ' ' '	ass, patron, ##ising, comments, with	non-toxic / toxic
4.	what the f * ck is your problem ? get a life . you don ' t own any pages .	ck, f, *, problem, pages	non-toxic / toxic
5.	stop f * cking up the call - out tables please .	f, ck, *, please, tables	non-toxic / toxic

Table 13: Generalizing from Single Examples: 5 examples with their top-5 terms whose labels are not surprising.